SN Computer Science (2023) 4:417
https://doi.org/10.1007/s42979-023-01847-5

ORIGINAL RESEARCH

A Validated Performance Model for Micro-services Placement in Fog Systems

Claudia Canali[1] · Giuseppe Di Modica[2] · Riccardo Lancellotti[1] · Stefano Rossi[1] · Domenico Scotece[2]

Received: 30 August 2022 / Accepted: 20 April 2023 / Published online: 24 May 2023

Abstract

The recent evolutionary trend of modern applications is towards a development paradigm that involves the composition of multiple interconnected micro-services devoted to perform specific functions. Such applications usually rely on data collected by geographically distributed sensors or by mobile users and are often characterized by strict requirements in terms of latency and response time. These requirements may be not compatible with the traditional cloud computing approach, where the computation occurring on far-away data centers cannot always guarantee the satisfaction of latency constraints. The fog computing approach has recently received a lot of attention as a promising solution in supporting time-critical applications. Due to an intermediate layer of fog nodes located close to sensors or final users and able to process the application data, indeed, the fog systems may significantly reduce the experienced response time. In a scenario where applications are composed by a chain of multiple micro-services, however, the service placement over the nodes of the fog infrastructure represents a nontrivial issue with respect to the cloud computing context. The highly distributed and heterogeneous nature of the fog nodes requires novel solutions taking into account the different performance of the fog nodes and the network delays caused by inter-nodes connectivity. This paper proposes a performance model for the placement of application micro-services over the fog infrastructure. To face the computational complexity of the optimization model, an heuristic based on a genetic algorithm is proposed. Furthermore, the analytical model is validated by means of simulation. The performance of the proposed solution is evaluated under a wide set of scenario and parameters ranges, including a case study based on realistic micro-services characterized through a prototype implementation.

Keywords Micro-services placement · Fog computing · Genetic algorithms · Performance evaluation

Introduction

The definition of fog computing given by the OpenFog Consortium [1] is based on the concept of a geographically distributed system able to offer functions and resources in locations that are closer to the final users than the cloud computing infrastructures. The Fog computing paradigm aims to move cloud services like computation, communication and storage close to edge devices and end-users, with clear benefits in terms of enablement of low-latency applications, better support of user mobility, save on network bandwidth, enhancement of security and privacy. On the other hand, however, the nature of a fog infrastructure is profoundly different from that of a cloud data center for what concerns, e.g., the heterogeneity of available resources and computing capacity [2, 3], thus opening up to novel and complex issues in terms of infrastructure management.

This article is part of the topical collection "Cloud Computing and Services Science" guest edited by Donald Ferguson, Markus Helfert and Claus Pahl.

✉ Riccardo Lancellotti
riccardo.lancellotti@unimore.it

Claudia Canali
claudia.canali@unimore.it

Giuseppe Di Modica
giuseppe.dimodica@unibo.it

Stefano Rossi
stefano.rossi@unimore.it

Domenico Scotece
domenico.scotece@unibo.it

[1] Department of Engineering "Enzo Ferrari", University of Modena and Reggio Emilia, Modena, Italy

[2] Department of Engineering and Computer Science, University of Bologna, Bologna, Italy

Compared with the cloud, an infrastructure of fog nodes relies on a relatively small pool of resources which makes the fog unfit to deal with huge workloads that, instead, the cloud can easily handle by virtue of an almost unlimited availability of computing power. Furthermore, the computing nodes of a fog may exhibit heterogeneous features (in terms of, e.g., CPU, memory) which exacerbates the problem of scheduling resources in an effective and efficient way. Consequently, novel and more complex resource management and service allocation algorithms are required to meet the demand of quality of service in the fog. One more issue to face in this scenario is the highly distributed nature of the fog nodes, which may cause non negligible and sometimes unpredictable network delays. These delays have a certain impact on both fog-to-cloud and fog-to-fog communications, and need to be taken into great consideration when composite and distributed applications demand fog resources.

In this paper, the focus is on the realistic case of smart IoT applications based on a micro-service architecture, i.e., applications natively designed as a chain of multiple interconnected micro-services. As reported in these recent surveys [4, 5], the micro-service architectural paradigm has become very popular in business contexts for the clear advantages it brings in terms of modularity, maintainability, flexibility and scalability. It is worth to note that the problem of transforming a monolithic application into a chain of micro-services is out of the scope of the paper and is assume to be already carried out before considering the deployment schemes of the micro-services chain. In such a composite scenario, assuming that each micro-service can be deployed in any of the available fog nodes, a critical issue to guarantee the QoS fulfillment to the final application consumers is represented by the correct placement of the micro-services over the nodes of the fog infrastructure. The heterogeneity of the available resources and the presence of network-related delays makes the micro-service placement a very complex task.

This paper presents some innovative contributions that are summarized as follows:

- An analytical model for the placement of micro-service chains over the nodes of a geographically-distributed fog infrastructure;
- A heuristic based on Genetic Algorithms (GA) to solve in finite time the non-linear problem of minimizing the service chains' response time;
- Validation of the proposal by means of a simulation approach, which considers the cases of both simple applications and more complex ones with long chains of multiple interconnected micro-services;
- Assessment of the viability and evaluation of the performance of the proposed strategy in some case studies envisioning typical real micro-services of a smart city scenario.

The rest of the paper is organized in the following way. Section "Related Work" presents some related work. Section "Motivating Scenario" describes the motivating scenario for the micro-service placement over the nodes of a fog infrastructure. Section "Performance Model" introduces the theoretical performance model, describes the optimization problem, illustrates the heuristic based on genetic algorithms and presents the characterization of two micro-services implemented on a real prototype. Section "Model validation" presents the validation of the model through simulation. Section "Experimental Results" describes the results of the experimental evaluation. Finally, Section" Conclusions" concludes the paper with some final remarks.

Related Work

The issue of process allocations and load balancing in distributed systems and cloud computing infrastructures has received a lot of attention over the past years, as testified by several existing surveys on the topic [6–9]. In the context of Cloud computing, the issues more investigated by researchers and practitioners have been related to optimal placement of virtual machines (VMs) over the physical servers of a cloud datacenter [10–12] and to efficient task scheduling to avoid overloading/underloading cloud virtual machines [13, 14]. In these studies, intelligent and nature inspired meta-heuristics emerged as viable and promising solutions to address performance issues related to resource distributions in cloud distributed systems. In Singh et al. [13] most of the literature concentrates on Genetic Algorithms (GAs), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO) based scheduling techniques for task and work-flow scheduling in cloud systems. In particular, Genetic Algorithms represent the base of the principally investigated heuristic in [15], that aims to optimize task scheduling based on multiple cloud properties and resources, such as speed, capacity, task size, number of tasks, number of virtual machines, and throughput. Similarly, in [16] the authors propose a modified genetic algorithm combined with a greedy strategy for cloud task scheduling optimization, showing the feasibility of applying this kind of heuristic to cloud computing distributed environments.

In the context of fog computing systems, many studies focus on the service placement issue, but they mainly rely on a simplifying assumption based on the fact that an application (typically IoT or smart city application) consists of a single service requested by clients. Among these studies, Yu et al. [17] present a model to jointly optimize the placement of the IoT applications and the routing of related data.

In [18], a placement strategy based on the characteristics of IoT services, including their requirements in terms of Quality of Service (QoS) is proposed: they model the system in terms of fog colonies and propose a solution based on Integer Linear Programming. Finally, in [19] the authors present a first proposal of service placement for fog computing architectures exploiting an heuristic based on genetic algorithms, showing the suitability and good performance of the solution in fog systems through experimental results. However, all these studies do not take into account that, in real scenarios, the applications requiring a fog infrastructure to be supported typically involve a chain of multiple micro-services that may have dependencies and constraints, thus generating a placement problem much more complex to be addressed.

Far less attention in literature has been devoted to the problem of allocating micro-services over the geographically distributed nodes of a fog computing system. Indeed, few studies start from the assumption of modeling complex applications as chains of micro-services and address the issue of placing them over the geographically distributed nodes of the fog layer. In this context, solutions can be divided in completely distributed [20, 21] and centralized [22, 23] solutions. The study in [20] presents an approximation method based on game-theory aiming at optimizing the energy consumption and the costs related to communications among the nodes of the infrastructure. The solution proposed in [21] is based on the cooperation between the fog nodes to define the workload that has to be assigned to other nodes with the final aim to reduce the final response time for the end users. Among the centralized solutions, it is worth to mention the studies in [22], where the micro-service placement is defined by a centralized controller, and in [23], proposing a mixed-integer linear programming formulation for the orchestration of Virtual Reality services in fog infrastructures.

In this paper, an heuristic based on Genetic Algorithms is presented to address the non linear nature of the optimization problem considered for modeling the micro-service placement and minimize the global response time of the requesting applications. The heuristic was originally proposed by the same authors in [24]. This paper significantly extends the previous work by adding a validation of the proposed model based on simulation and covering both cases of simple applications and more complex scenarios with long chains of multiple micro-services. Moreover, the present study proposes a characterization of the execution times of micro-services typically involved in smart cities applications and the analysis of a realistic case study.

Finally, some studies consider the service placement problem in the context of fog-to-cloud infrastructures [25–27]. For example, in [25] authors present a novel mechanism able to determine, based on the requirements of the services and on the characteristics of the system resources, the optimal execution offloading. The studies in [26, 27] propose to assign to fog nodes the services characterized by low latency, while sending the other services to the cloud. On the other hand, in this paper a solution for the placement of all the micro-services belonging to an application chain over the nodes of the fog layer is proposed, under the assumption that the fog infrastructure can process every kind of request, with a consequent improvement of the user experience.

Motivating Scenario

Due to the large availability of computing capacity worldwide, the amount of resources that one single application can obtain from the Cloud is pretty much unbounded. Unfortunately, the distance between data sources and the Cloud providers' premises (often covered by geographic network links) makes the Cloud paradigm unfit to serve delay-sensitive applications in typical IoT scenarios. The fog computing promises to shorten the distance by bringing computing resources close to data. Unfortunately, the benefit that the fog computing provides in terms of decreased network delay is counterbalanced by the reduced availability of processing power, as in that regard the fog computing facilities are not even comparable to the cloud ones. A further drawback of the fog is the heterogeneity of the provided resources, since fog nodes typically exhibit non-uniform capacities and sometimes even imbalanced interlinks. Such an imbalance represents a problem in scenarios where fog resources must be efficiently allocated to multiple requesting applications. Also, considering that most business applications are composed of multiple micro-services [4, 5], each deployable in the fog infrastructure independently of one other, the problem to face is even more complex. If, on the one hand, the modularity of applications increases the chance of maximizing the utility of available resources, on the other one it calls for smarter resource scheduling strategies.

This paper addresses the depicted scenario and tackles the problem of smartly and efficiently allocating fog resources (*nodes*, from now on) to requesting microservice-based applications. In the literature, this is known as a *service placement* problem. In the considered scenario, the fog infrastructure comprises a set of nodes of known and variable capacity. Nodes are networked via high-speed links. The network topology modeling was intentionally left out of the scope of the study, though this will be part of the future work. The node-to-node network delay is a parameter of the fog infrastructure that may impact on the quality of the provided service; therefore, it is taken into consideration when formulating the service placement problem.

On the service providers side, a set of micro-service-based applications demand computing resources from the

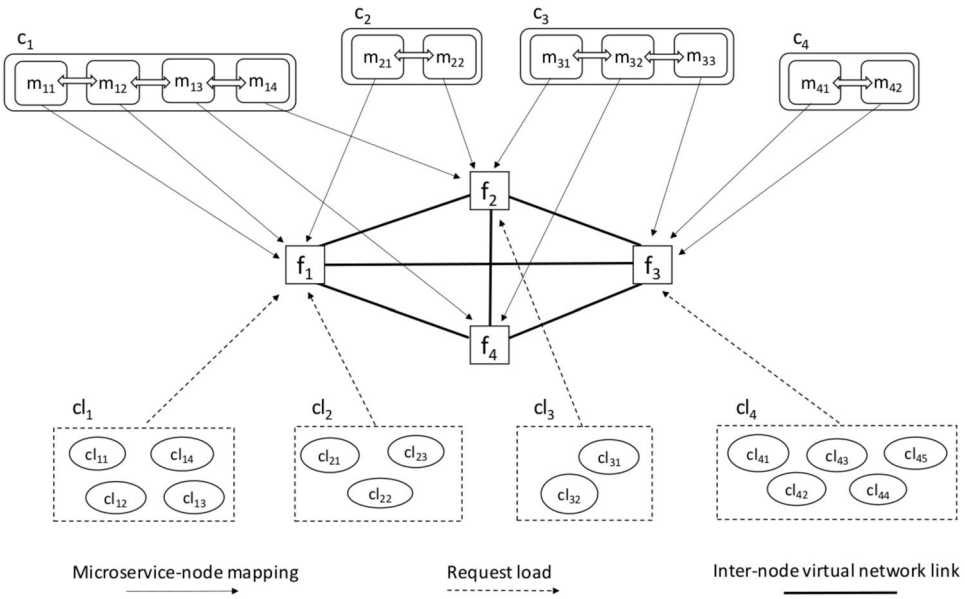


Fig. 1 Service placement scenario

fog infrastructure to execute. Customers generate service requests towards the deployed applications according to *a-priori* known workloads. The aim is to develop a service placement strategy that, leveraging the mentioned boundary conditions, devises an optimal resource allocation scheme that minimizes applications' response time. Also, recalling the assumption made earlier, concerning the fact that all considered applications are natively decomposed into smaller micro-services, when seeking for optimal micro-service-to-node placement, metrics such as the average number of nodes spanned by applications, the network delay experienced by requests and the overall load balance are taken into consideration.

Figure 1, depicts a simple service placement scenario where four applications must be deployed on a fog infrastructure composed of four nodes. The application is modeled as a *micro-service chain*, i.e., a set of micro-services (two, at least) that cooperate to fulfill clients requests. In the practice, micro-services of an application can cooperate with one another to form different topologies. Without loss of generality, it is possible to assume that a micro-service chain c_i is actually implemented by wiring (i.e., putting one after one) a sequence of micro-services $m_{ij}, j \in 1, 2, ...n$. The micro-service occupying the first position in the chain (m_{i1}) will take c_i input, make its elaboration and invoke the next micro-service in the chain. The cascade of invocations goes on until the last micro-service is reached (m_{in}), which will return back the result of its elaboration through the chain. Eventually, m_{i1} will respond to the client's request. The "length" of a micro-service chain refers to the number of micro-services composing the chain.

The goal of the service placement scenario is to place micro-services of each chain into one or multiple nodes. An obvious constraint is that the number of nodes hosting one chain may not exceed the chain length. In its turn, a fog node may host micro-services belonging to different micro-service chains. The figure depicts one exemplary placement scheme. According to the scheme, the three micro-services composing the chain c_3 are hosted by three distinct nodes (f_2, f_3 and f_4 respectively), while the entire micro-service chain c_4 is placed in one fog node (f_3). A generic node f_k is the recipient of all client requests addressed to the micro-service chain(s) having their first micro-service hosted by f_k. By way of example, fog node f_1 will receive all requests addressed to chain c_1 and c_2 (represented in the figure by request loads cl_1 and cl_2, respectively), since f_1 is hosting m_{11} and m_{21} which are the first micro-services in the chains c_1 and c_2, respectively. It is worth to remind that fog nodes have different computing capacity and are interconnected with each other via heterogeneous high-speed links.

The goal is to devise and implement a micro-service placement strategy that strives to minimize the average *service response time*, i.e., the time taken by the application (the chain, in the considered case) to reply to a client request. The strategy will account for the following boundary conditions: applications request loads, (ii) average service time of micro-services composing the chains, and (iii) computing capacity of fog nodes that will host the micro-services. In a scenario where multiple applications must be served, the strategy will pursue the minimization of all applications response time.

Performance Model

This section introduces the model that is the basis for the micro-services placement problem. Next, an heuristic, based on Genetic algorithms that can solve the problem is presented. The model and the algorithm refer to the scenario described in Sect. "Motivating Scenario", where *service chains* consisting of several *micro-services* must be deployed over a set of *fog nodes*. Chains are activated by data sources that can be either sensing devices of mobile users, that are called simply *clients*.

Performance Metrics

Throughout the analysis the response time is considered as the main performance metric. Response time is defined as the time between the data is sent by a client and the moment the service chain completes the processing of such data (that is when the last micro-service of the chain ends). For the sake of clarity it is worth to point out, that response time is different from the service time: the response time contains also the contribution of network delays and queuing for accessing resources on fog nodes, while service time is just the time required to execute the micro-service, without any data transfer or queuing. Optimizing response time implies the optimization of several other more specialized metrics. For example, an unbalanced load on the infrastructure, with some fog nodes that are close to overload would automatically cause an unacceptably high response time. To reduce response time load should be evenly distributed over the infrastructure. Similarly, if the placement of a service chain spreads the micro-services over many fog nodes, the network-related contribution of the response time would explode. Hence a poor performance for a locality-based performance metric implies a high response time, while a good score for a locality-based metric wold automatically have a positive impact on response time.

Furthermore, the analysis takes into account the notion of Service Level Agreement (SLA) in the form of a maximum acceptable response time for every service chain, being response time one of the most common and challenging requirements for cloud continuum contexts [28].

Table 1 contains a summary for the notation used in the model. The symbols are basically he same used in Sect. "Motivating Scenario", with the main difference that a micro-service is identified with the symbol m, dropping the reference of the service chain, for the sake of brevity.

The initial focus is on the performance of a single micro-service m. The service time of such micro-service is described by a generic probability distribution with an average value of S_m and with a standard deviation σ_m. Service time is measured at server-side (hence it does not include network-related delays) and considers just the time spent executing the micro-service (hence it does not include waiting time related to the processing of other, concurrent, micro-services).

Table 1 Notation and parameters for the proposed model

Model parameters	
$\mathcal{M}$	Set of micro-services
$\mathcal{F}$	Set of fog nodes
$\mathcal{C}$	Set of service chains
λ_m	Incoming request rate to micro-service m
λ_f	Incoming request rate to fog node f
λ_c	Incoming request rate to service chain c
Λ	Incoming global request rate
S_m	Average service time for micro-service m
σ_m	Standard deviation of S_m
P_f	Computational power of fog node f
W_f	Average waiting time on fog node f
S_f	Average service time on fog node f
σ_f	Standard deviation of S_f
R_c	Average response time for service chain c
T_c^{SLA}	SLA of service chain c
o_{m_1,m_2}	order of execution of micro-services in a chain
δ_{f_1,f_2}	network delay between nodes f_1 and f_2
Model indices	
f	A fog node
c	A service chain
m	A micro-service
Decision variables	
$x_{m,f}$	Allocation of micro-service m to fog f

To model the performance of the fog infrastructure, the following assumptions are made: (1) the system is in a steady-state condition, avoiding transient times; (2) no jobs (that are requests to process incoming data for a micro-service) are dropped between two consecutive micro-services. This means that, once a chain is invoked it is executed completely without external interruptions. This assumption is consistent with the goal of the study that is to provide a mechanism for efficient deployment of fog applications, without overload or errors. From a model point of view this assumption implies that, for each micro-service in a chain, the incoming rate equals the outgoing rate (that is the incoming rate for the subsequent micro-service). This assumption can be formalized as $\forall c \in \mathcal{C}$, and $\forall m_1, m_2 \in c$, $\lambda_{m_1} = \lambda_{m_2}$.

For the sake of clarity, it is worth to anticipate that the decision variable for the optimization problem is a matrix of Boolean variables $X = \{x_{m,f}, m \in \mathcal{M}, f \in \mathcal{F}\}$ such that $x_{m,f} = 1 \iff$ micro-service m runs on fog node f.

For the nature of the decision variable, a micro-service cannot be split over multiple fog nodes. Furthermore, every micro-service must run on one and only one fog node.

A fog node is modeled according to queuing theory where jobs can wait in a queue before being processed by the fog node. As multiple micro-services can be present on the same fog node, each node behaves as a multi-class server.

Let node f be a generic fog node. Its multi-class workload will present a class for every micro-service m hosted on f. The resulting multi-class system will be described with a service time that is a mixture of distributions where each component is the distribution of the service time for a micro-service.

In the proposed model, fog nodes can be heterogeneous, meaning that the computational power of each node can be different. As a result, there can be a speedup (or a slowdown) with respect to the expected service time depending on the computational power P_f of fog node f.

The formulation for the service time S_f and of its standard deviation σ_f for a fog node f can be summarized as follows:

$$S_f = \frac{1}{P_f} \cdot \sum_{m \in \mathcal{M}} x_{m,f} \frac{\lambda_m}{\lambda_f} S_m \tag{1}$$

$$\sigma_f^2 = \left(\frac{1}{P_f^2} \cdot \sum_{m \in \mathcal{M}} x_{m,f} \frac{\lambda_m}{\lambda_f} \left(S_m^2 + \sigma_m^2 \right) \right) - S_f^2, \tag{2}$$

where $\lambda_f = \sum_{m \in \mathcal{M}} x_{m,f} \lambda_m$ is the total incoming load on node f.

The expected waiting time W_f for the fog node can be derived using the Pollaczek Khinchin equation:

$$W_f = \frac{S_f^2 + \sigma_f^2}{2} \cdot \frac{\lambda_f}{1 - \lambda_f S_f}. \tag{3}$$

It is worth to note that Eq. (3) assumes that the whole queuing system describing the fog infrastructure can be expressed in a product form. Unfortunately, this is not always true. When the service time of each micro-service does not follows an exponential distribution, the arrival process in subsequent fog nodes cannot be considered as a exponential distribution. Hence the M/G/1 model used in Eq. (3) should become a G/G/1 model. However, no closed-form solution is available to describe the inter-arrival time of a queuing server that is placed next to a G/G/1 server. The approximation of a M/G/1 model can be used when many micro-services are located on a fog node or when the standard deviation of the service time of the micro-services is close to the mean value. As the actual system characteristics drift away from these assumptions the performance model become inaccurate. Section "Model Validation" provides a validation that the used approximation holds for the scenarios considered in the experiments.

Under the above limitation, Eq. (3) can be used to define the response time for a service chain c as the sum of waiting time on the fog nodes, plus the sum of service times plus the network delay due to transferring data between two subsequent micro-services in the service chain.

$$R_c = \sum_{m \in c} \sum_{f \in \mathcal{F}} x_{m,f} \cdot W_f + \sum_{m \in c} S_m + \sum_{m_1, m_2 \in c} \sum_{f_1, f_2 \in \mathcal{F}} o_{m_1,m_2} \cdot x_{m_1,f_1} \cdot x_{m_2,f_2} \cdot \delta_{f_1,f_2} \tag{4}$$

where o_{m_1,m_2} is the order of execution of micro-services in c. In particular $o_{m_1,m_2} = 1 \iff m_1 \prec m_2$, meaning that service m_1 s invoked just before m_2 in the service chain.

Optimization Problem

The focus of this subsection is the optimization problem for allocating micro-services over the infrastructure of fog nodes. To this aim, the definitions of Sect. "Performance Metrics" and the notation in Table 1 is used.

$$\min obj(X) = \sum_{c \in \mathcal{C}} w_c R_c \tag{5}$$

subject to:

$$\sum_{f \in \mathcal{F}} x_{m,f} = 1 \quad \forall m \in \mathcal{M}, \tag{6}$$

$$\lambda_f S_f < 1 \quad \forall f \in \mathcal{F}, \tag{7}$$

$$R_c < T_c^{SLA} \quad \forall c \in \mathcal{C}, \tag{8}$$

$$x_{m,f} = \{0, 1\}, \quad \forall m \in \mathcal{M}, f \in \mathcal{F}, \tag{9}$$

Objective function in Eq. (5) is a weighted sum of the response time for very considered service chain, based on Eq. (4). The weights $w_c, \forall c \in \mathcal{C}$ are chosen such that $\sum_c w_c = 1$. A possible approach, used in this paper, is to consider the weight of every service chain proportional to its activation frequency λ_c (that is $w_c = \lambda_c / \Lambda$, with $\Lambda = \sum_c \lambda_c$), although other solutions can be adopted without altering the general framework of the model.

In the optimization problem three constraints are imposed. Equation (6) means that every micro-service is allocated on one and just one fog node. Equation (7) forces every fog node not to be in an overload condition ($\lambda_f S_f$ is the load on fog node f). The last model constraint, Eq. (8),

forces the respect of the SLA of each service chain. This study considers $T_{SLA} = K \cdot \sum_{m \in c} S_m$, with $K = 10$, that is a common approach on cloud and fog-based systems [29]. The last constraint, Eq. (9), captures the Boolean nature of the decision variable $x_{m,f}$.

Genetic Algorithm

Having formulated a problem for the placement of applications that are designed as a chain of micro-services, there is now the need to find away to solve this problem. The objective function determines the non-linear nature of the optimization problem. For this reason, finding a simple heuristic or solving the problem is not a trivial task. To overcome this problem, evolutionary meta-heuristics that have been used in literature to solve similar problems [30, 31] can be used. This study considers Genetic Algorithms (GAs) as a promising approach to tackle the problem. However, this is just a way to solve the problem, that retains a general validity. Other heuristics can still be designed an applied to solve the formalization proposed in Sect. "Optimization Problem".

In a genetic algorithm, the solution is encoded as a sequence of symbols called *chromosome*, with each symbol, namely a *gene*, representing a parameter of the problem solution. A *population* is initialized using (randomly generated) *individuals*, with each individual being a candidate solutions, and the population evolves through a finite number of *generations*.

Considering the micro-service allocation problem described in Sect. "Optimization Problem", a chromosome is set of $M = \|\mathcal{M}\|$ genes, where M is the number of considered micro-services. The value of each gene is an an integer number $\in [1, F]$, with $F = \|\mathcal{F}\|$, being the number of fog nodes. To map the decision variable $x_{m,f}$ on the genes the m^{th} gene can be defined as as $g_m = \{f : x_{m,f} = 1\}$. Due to constraint (6) the microservice m will be allocated on just one fog node. Hence, the proposed chromosome definition automatically satisfies Eqs. (6) and (9).

To guide the evolution process, each individual has a *fitness score* based on the objective function defined in Eq. (5). It is worth to note that not every solution encoded as a chromosome is acceptable as the constraints (7) and (8) on the fog node overload and on SLA respect are not guaranteed to be met. However, the population evolves including also non-acceptable solution to fully exploit the ability of GAs to explore large space of configurations. Violated constraints are still taken into account adding two penalty factor to the objective function that is re-defined as:

$$obj^*(X) = \sum_{c \in \mathcal{C}} w_c R_c + P_{ol} + P_{\text{SLA}}, \tag{10}$$

where $P_{ol} = S_f^*(1 + \lambda_f - 1 * S_f)$ if overload occurs, with S_f^* being an arbitrary high value of the service time for a fog node. In a similar way, a penalty for SLA violations as $P_{\text{SLA}} = S_c^*(S_c / T_{\text{SLA}})$ is defined, with S_c^* being an arbitrary high value of the service time of a service chain. Both penalties are proportional to the level of constraint violation, aiming to prune from the genetic pool individual that violates (hardly) the two constraints. The proportionality is useful when the whole genetic pool contains a significant fraction of unacceptable individuals. In this case the most unacceptable individuals will be removed first, preserving a beneficial level of genetic diversity.

As the population evolves through the generations a set of genetic operators such as *mutation*, *crossover* and, *selection* are used to increase the fitness score of the individuals. In the experiments a *random mutation* operator alters a single random gene in the chromosome. This operator enables the exploration of new areas in the solution space. The proposed algorithm also uses a *uniform crossover* that takes two parent individuals and merges them in two offspring individuals aiming to spread successful genes over the population. Finally, a *tournament selection* selects the best individuals that should be passed from the current to the next generation.

The algorithm is implemented using the DEAP[1] library. The parameters of the genetic algorithm are tunes using preliminary tests. In particular the mutation probability is set to $P_{\text{mut}} = 0.8\%$ and the crossover probability to $P_{cx} = 0.8\%$. Furthermore the population is composed of 600 individuals and the number of generations is 600. The parameters are consistent with results carried out on similar problems [19].

Micro-service Characterization

This subsection, introduces some examples of typical and realistic micro-services that could be used as building blocks to compose service chains in a Fog environment. It also provides a characterization of such micro-services in terms of features that are useful for the proposed model. To this end, the features of interests are the average service execution time and its standard deviation. In particular, the **average service execution time** is defined as the average time between when the request is issued by the user and when the server has provided a response.

In this work, the considered micro-services fits a typical Smart City scenario. Two categories of micro-services are taken into account, of which one is computationally intensive while the other is very low demanding. As examples of computing-intensive service category, a video recognition micro-service and an image recognition micro-service are

[1] DEAP: Distributed Evolutionary Algorithms in Python— https://deap.readthedocs.io/en/master/.

Table 2 Micro-services characterization

Service name	Service ID	Avg. time (s)	St. Dev (s)
Video recognition: high resolution	VREC_H	7.897	0.1649
Video recognition: low resolution	VREC_L	6.787	0.1021
Image recognition	IMREC	1.649	0.557
Service discovery	SDISC	0.009	0.0015

analyzed and profiled. For the other category, a very simple service discovery application is taken into account.

Video recognition This micro-service emulates a video object detection and tracking service from an input video. In particular, it produces a matrix with the positions of the tracked objects in the video. Moreover, two different versions of the same micro-service are considered by changing the quality and resolution of the video (high-res and low-res, respectively). More in detail, the application is written in Python and leverages the *ffmpeg* Linux library to transform the video into a numerical matrix.

Image recognition This micro-service offers objects classification functionalities from input images. The service leverages a pre-trained ML model for image classification. The application is written in Python and uses *Flask* as a Python Web engine. When the micro-service receives an image as an input, the latter is translated into a numeric array for further processing. On that array the ML model will perform the recognition.

Service Discovery This micro-service supports users in finding the needed services. From the implementation viewpoint, the service makes use of a Redis database that stores data in a key-value fashion. When a user issues a query, this service will reply with the details of matching services.

The four mentioned micro-services are benchmarked on an off-the-shelf PC. To get robust statistics, each micro-service was tested with 50 requests. The following Table 2 shows the characterization of the discussed micro-services.

Model Validation

The performance model previously proposed, is now validated by means of simulation.

Simulator Setup

For the simulations the Omnet++ framework[2] is used, enriched with additional modules to capture the nature of the considered problem. Specifically, new models introduced for the analysis are a traffic generator that can describe a request comprising multiple invocations to small micro-services, delay centers to model network delays. Furthermore, the fog nodes are modeled in the simulator to extract the relevant information on the micro-service to be executed and implements a queue where the jobs are stored while the processor is busy (thus mimicking the behavior of the queuing network model). Additional simulations (not reported for space reasons) confirm the validity of the proposed solution also in the case of processor-sharing setups, where job executions are interleaved. A set of routing modules sends the jobs from one fog node to the next based on the allocation of micro-services over the fog infrastructure.

In the tests both the execution times of the micro-services and the network delays are modeled according to log-normal distributions, with different standard deviation and average values, depending on the specific scenario.

Fog Node Performance

The first scenario considered in the tests is a simple scenario with a single fog node and two co-located micro-services. This scenario aims to validate the model for the fog node service time described in Eqs. (1) and (2).

Considering two micro-services, namely m_1 and m_2. The two micro-services are characterized by service times equal to $S_1 = 0.1$ s and $S_2 = 0.05$ s and by standard deviation equal to $\sigma_1 = 0.02$ s and $\sigma_2 = 0.01$ s, respectively. The two micro-services have an incoming flow of requests that follows an exponential distribution for the inter-arrival times, with an average λ_1 and λ_2.

Throughout the experiment the arrival rate of the two micro-services changes in the range $\lambda_1 \in [0.5, 4.5]$ req/s and $\lambda_2 \in [1, 9]$ req/s.

Figure 2 presents a comparison between simulation and model results for the average service time and for a measure of the service time variance as a function of the two parameters λ_1 and λ_2.

Figure 2a shows the service time of the two combined micro-services. The simulation (grid of purple lines with square points) matches very closely the theoretical model (grid of cyan lines), thus validating Eq. (1). In a similar way, Fig. 2b shows the coefficient of variation (CoV) for the service time, that is the ration between the standard deviation and the average value. The comparison between

[2] https://omnetpp.org/.

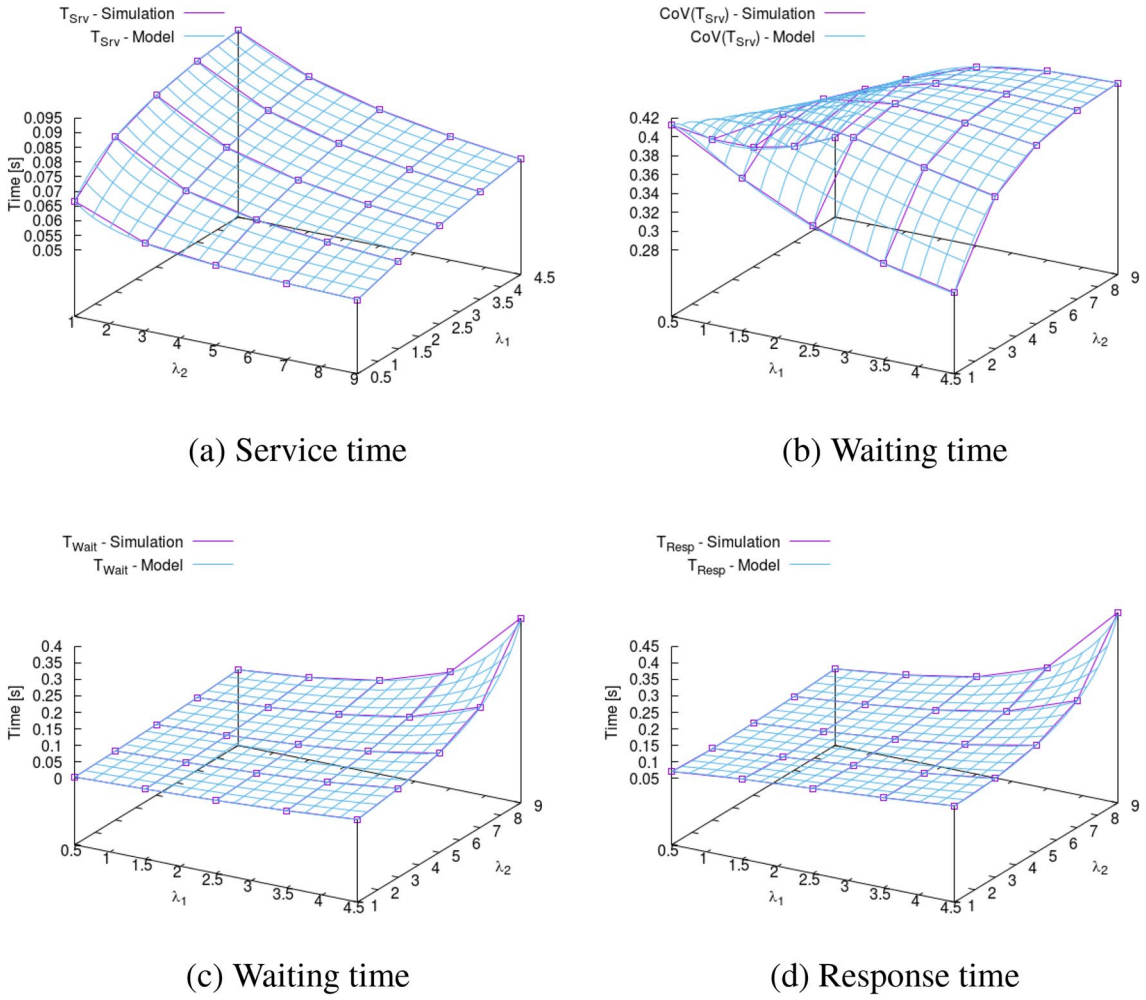


Fig. 2 Validation for two heterogeneous micro-services

the simulation results and the theoretical model provides a validation of Eq. (2).

After the validation of the execution of heterogeneous micro-services in a multi-class fog node, the focus shifts towards waiting and response times. Figure 2c shows the time spent waiting in a queue (modeled in Eq. (3)) while Fig. 2d shows the overall response time for the fog. For both figures it is possible to observe a very close matching between the model prediction and the simulator output, thus confirming the validity of the model.

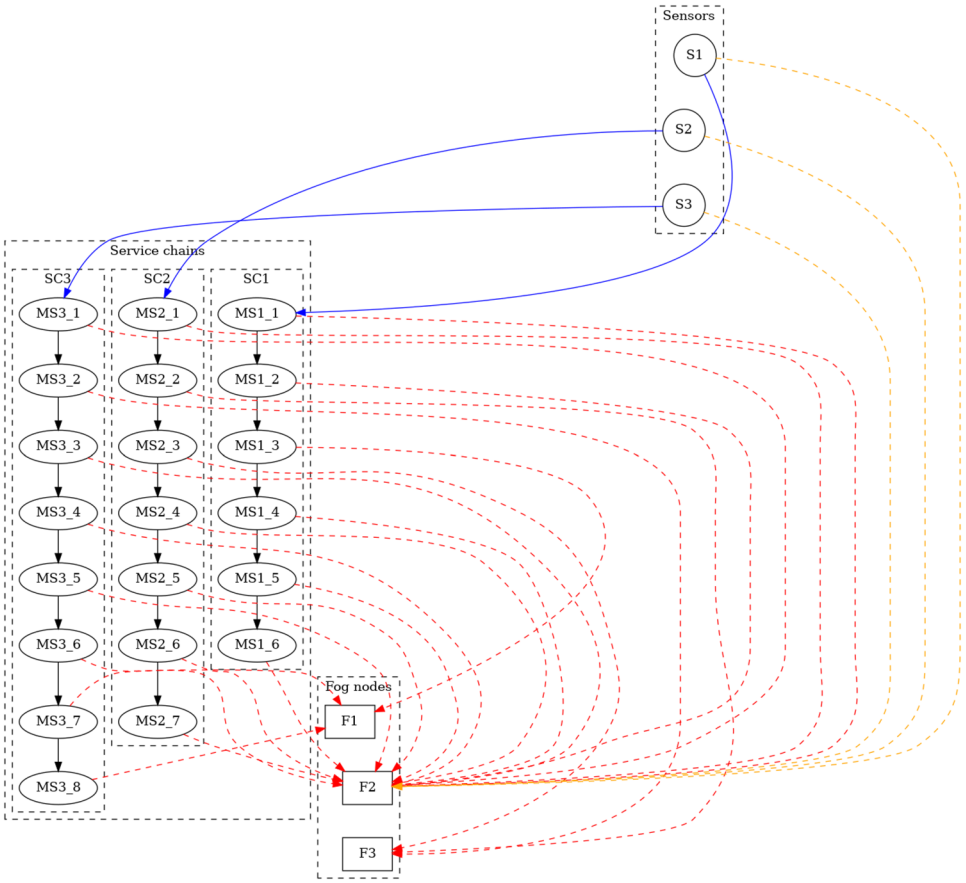
Service Chain Performance

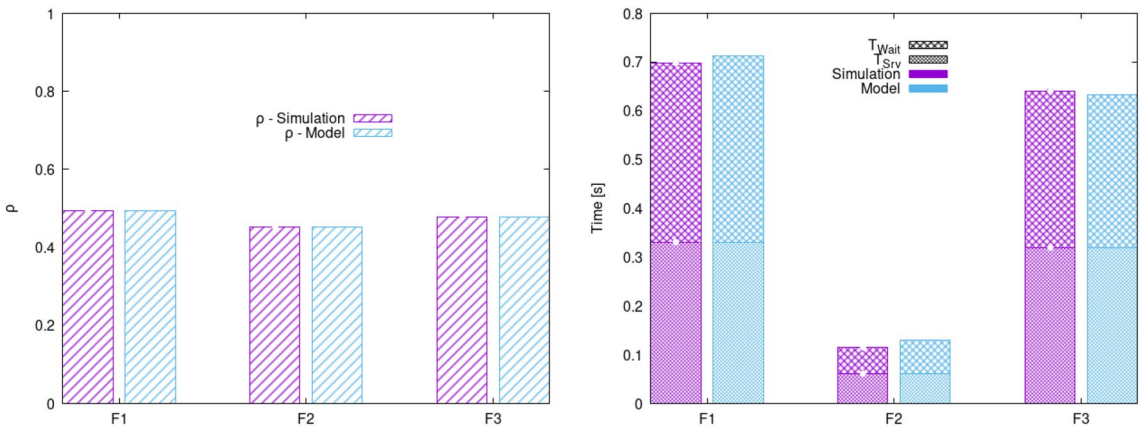
After a validation of the basic building blocks of the model, the validation of the whole model relies on a more complex scenario where multiple service chains, each composed by several micro-services, are distributed across an infrastructure with several fog nodes.

Figure 3 presents an example of a problem used for this validation along with its deployment scheme. The problem is composed by 3 service chains each composed by 8 micro-services. The data flows in the logical organization of the micro-services are represented by solid lines. The data flows from client to micro-services are depicted in blue, the data exchanged by micro-services in a service chain in black. The mapping of logical elements over the infrastructure is represented in dashed lines: yellow for links from client to the fog nodes, red for the deployment of micro-services over the fog nodes.

The model considers multiple service chains of different lengths (in the example ranging from 6 to 8 micro-services) and with a different aggregated service time. Each micro-service service time has a standard deviation close to 1 and the mapping ensures that each fog node hosts multiple micro-services. Under these conditions, it is possible to compare the performance predicted by the performance model with the simulator output.

Fig. 3 Service chains mapping





(a) Fog node utilization (b) Fog node times

Fig. 4 Fog node parameters

Figure 4 evaluates the expected utilization ρ of the fog nodes (Fig. 4a) as well as the service and waiting times (Fig. 4b). In both figures the first column (purple color) is related to the simulation, while the second column (azure color) refers to the theoretical model. For the times, the lower part of the columns (fine mesh) is the service time, while the upper part (coarse mesh) is the waiting time. The figure shows that the fog node utilization predicted by the model is accurate, as well as the service time. For the waiting time, a small discrepancy (below 10%) can be observed due to the approximation caused by the use of a M/G/1 model.

Evaluating the micro-service chains, the prediction of the service time and of the network delay times is quite accurate.

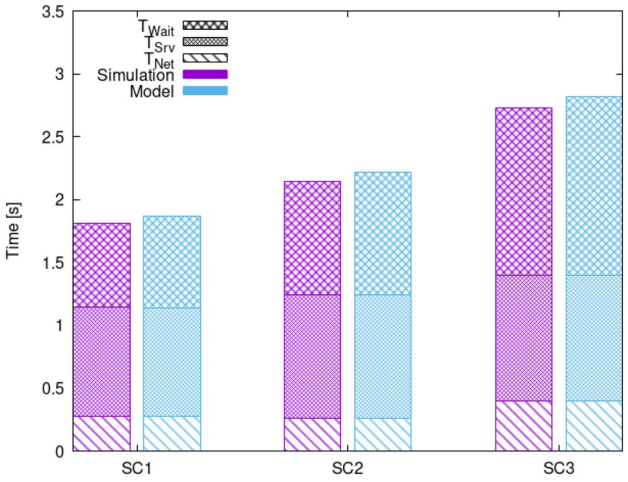


Fig. 5 Micro-service chain response time

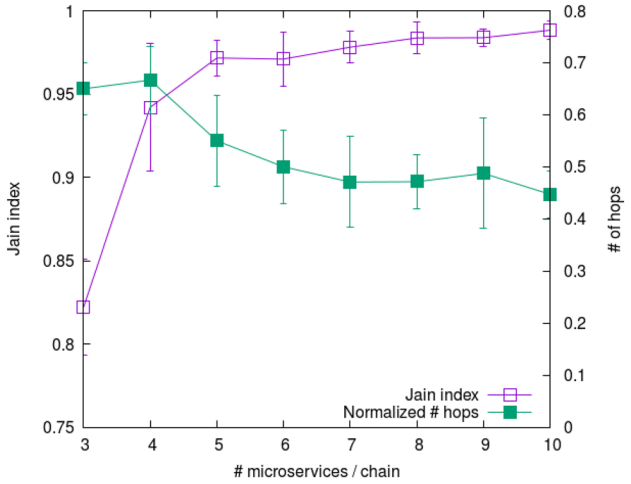


Fig. 6 Load balancing and hops *vs.* chain length L_c

Again, it is possible to observe an approximation in the waiting time component of the service chain response time, with the model tending to over-estimate the response time. However, it is worth to note that the error introduced by the M/G/1 model remains below 10% (see Fig. 5).

Experimental Results

Experimental Setup

This section aims to prove the suitability of the proposed GA-based approach to the micro-service placement problem. Specifically, experiments are aimed at assessing the quality of the solution suggested by the proposed heuristic for several random problems.

A placement problem is defined in terms of:

- Service chain length L_c, i.e., the number of micro-services composing a chain;
- Service time of a service chain S_c;
- Average network delay δ between two fog nodes;
- Overall infrastructure load ρ;
- Problem size, i.e., the number of fog nodes and of service chains.

Firstly, the behavior of the solver is studied in the case of equally long chains, that is $L_c = \|\{m \in c\}\|$ is constant $\forall c \in \mathcal{C}$. To that end, the considered chains are composed of 5 micro-services. The impact of the variability of L_c is discussed in Sect. "Sensitivity to Service Chain Length".

For all the experiments, the setting for the service request load is such that the average utilization of fog nodes is in the order of 60%. The overall load is defined as $\rho = \sum_c \lambda_c L_c \cdot \sum_c S_c / \sum_f P_f$, which means that $\rho \in [0, 1]$. The system's sensitivity to this parameter is analyzed in Sect. "Sensitivity to System Load", while for all other experiments $\rho = 0.6$, which corresponds to a medium utilization of the infrastructure.

For what concerns the problem size, the number of nodes is identified by $\|\mathcal{F}\|$, while that of chains is $\|\mathcal{C}\|$. As default values, a set of 10 fog nodes supporting 4 service chains is considered. The same values hold for all experiments except for the one aimed at testing the scalability of the proposed approach, which is discussed in Sect. "Scalability Analysis".

Furthermore, for each service the SLA is 10× the service time of the whole chain: such assumption is in line with typical SLA settings of cloud applications [29]. In the experiments, the SLA is automatically satisfied as long as no overload is incurred. This motivates the choice of not doing a specific analysis to study the impact of this parameter.

Significant KPIs in the analyses are the *response time* of service chains and the *average number of hops* in the chain deployments, normalized against the chain length (ranging in [0, 1]). Another critical yet useful performance metric adopted in the present study is the *Jain index*, a measure of fairness that quantifies the ability of the genetic algorithm to achieve load balancing over the fog infrastructure. The Jain index is defined as $J = 1/(1 + \mathrm{CoV}(\rho_f)^2)$, where ρ_f is the utilization of each node $f \in \mathcal{F}$ and $\mathrm{CoV}(\cdot)$ is the coefficient of variation (i.e., the ratio between standard deviation and mean) computed over all fog nodes. An index of 1 means a perfect balance, whilst values of the Jain index closer to 0 reveal that the load is unevenly distributed among the fog nodes.

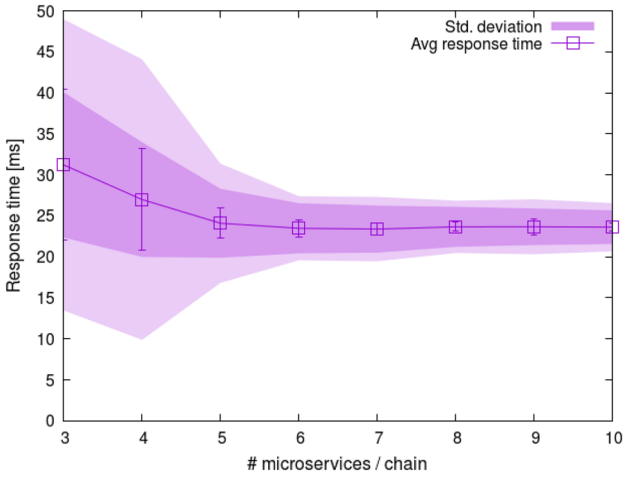


Fig. 7 Response times *vs.* chain length L_c

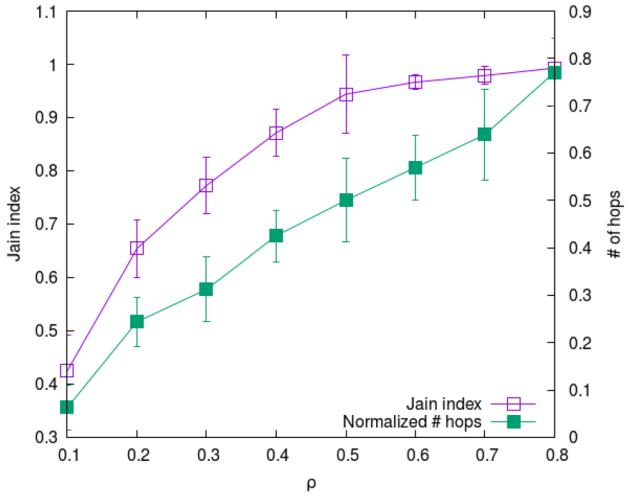


Fig. 8 Load balancing and hops vs. load ρ

Sensitivity to Service Chain Length

In this section the sensitivity of the optimization algorithm to the service chain length is assessed.

Figure 6, presents two curves representing the trend of the Jain index (purple line with empty squares) and the trend of the average number of hops (green line with filled squares) as a function of the service chain length L_c. It is worth to note that that, on a low number of micro-services in a chain, achieving the load balancing is trickier as there are a few, computing-intensive micro-services that alone could possibly saturate the processing power of a fog node. On the other hand, in the case of multiple lighter micro-services (that is, when L_c is higher), the ability of the genetic algorithm to find a good load balancing is proved by the Jain index value being close to 1. At the same time, due to the finer-grain placement options, also the average number of hops for each service is reduced by roughly 35%, confirming the ability of the proposed algorithm to reduce the impact of network delays. Being the normalized number of hops close to 0.5, on average, there is one hop every two micro-services.

Figure 7 shows the trend of the average response time when the service chains' length increases. The poor load balancing observed for low values of L_c causes an increase of the average response time, as local near-overload condition may arise on some fog nodes. The purple aura provides a measure of the variance of response times within each problem (dark purple) and between different problems (light purple). Due to the coarse-grained placement when $L_c < 5$, it is possible to observe a response time up to 30% higher than longer service chains and an increase of the variance of the samples by a factor of 4. It is worth to note that this behavior is due to problem becoming ill-conditioned, due to the presence of micro-services that, alone, can bring a fog node to close to overload. Under these circumstances, the GA heuristic must cope with a set of viable solution that are limited and often quite sparse, thus hindering the ability to converge towards an optimal solution.

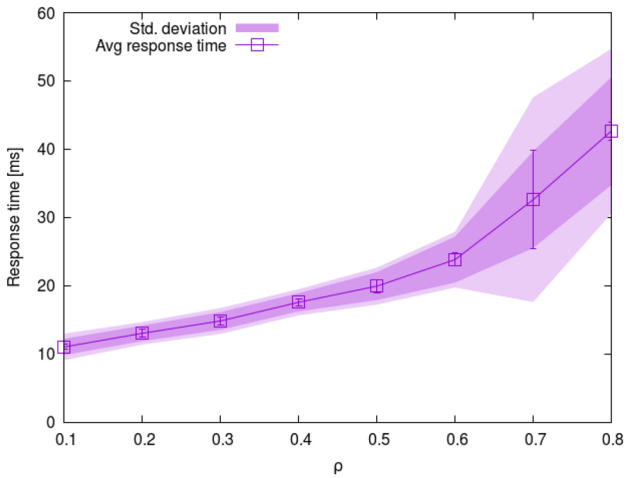


Fig. 9 Response times *vs.* load ρ

Sensitivity to System Load

Figure 8 shows the trend of the load balancing and the network delay as the overall infrastructure load ρ grows. For low values of load (e.g., $\rho \leq 0.2$) the response time is mostly contributed by the network delays. For this reason, it is possible to observe a poor load balancing ($J \leq 0.66$), because the fog nodes, despite are unbalanced, are still far from an overload condition. At the same time, the average number of hops between two consecutive micro-services is low, meaning that the probability of incurring in a network delay

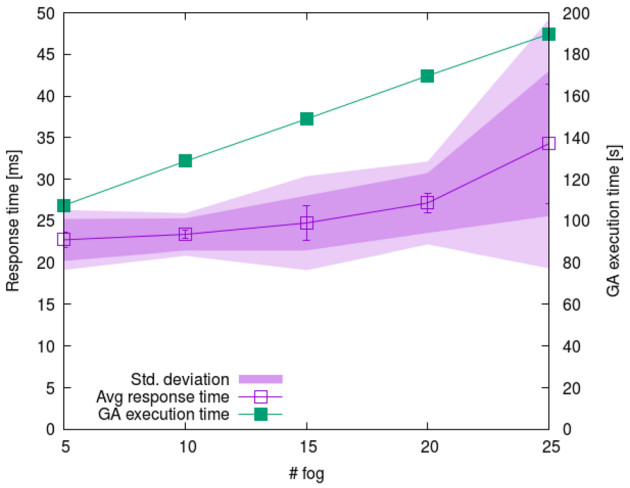


Fig. 10 Response and execution time *vs.* $\|\mathcal{F}\|$

is below 30%. As the load increases, the impact on load balancing becomes more evident, up to the case of $\rho = 0.8$ when the load balancing is nearly perfect ($J > 0.99$). Also, the probability of having two micro-services deployed in different nodes is close to 80%, which is 2.5× more than the low-load case. This result confirms that the GA algorithm can find solutions that cope correctly with the characteristics of the scenario considered in each experiment.

Figure 9 depicts the response time of the service chains as the load grows. When the load is close to 0 (e.g., $\rho = 0.1$), the response time close to 10 ms, that is the sum of the service times for the service chains. The response time is nearly doubled for $\rho = 0.5$ and is quadrupled for $\rho = 0.8$. The increase in the response time is driven by both the growth of the requests' waiting time in the servers queues and by the higher network delay caused by the increased number of hops. Along with the growth of the response time, an increase in the response time variance can be observed. This reason is that when the load on the fog nodes is high, even a small unbalancing can determine a significant impact on the response time, thus justifying the 6× increase in the standard deviation of the response time as ρ grows from 0.1 to 0.8. As for the service chain length, this is a characteristic of the problem, that becomes ill-conditioned for high loads.

Scalability Analysis

This section presents the results of the experiments to test the scalability of the algorithm as the number of fog nodes $\|\mathcal{F}\|$ grows by 5× from 5 to 25 nodes. The number of service chains $\|\mathcal{C}\|$ is increased proportionally from 2 to 10, with each chain hosting 5 micro-services.

Figure 10 shows that, as the configuration space to explore widens, the genetic algorithm presents a steady growth in the execution time (green line with filled squares). This result can be explained by considering that the chromosome length corresponds to the number of micro-services to place. As the chromosome grows, the cost of most genetic operators (from the computation of the objective function to mutation and crossover) increases, thus causing the growth in the execution time that is nearly doubled as the problem size increases by 5×. Also, the larger the search space the less effective the genetic algorithm is in identifying the best solutions. This explains the growth of the response time (54%) as well as its standard deviation (more than 4×), suggesting that for extremely large problems the algorithm may provide lower quality solutions.

A Realistic Case Study

To conclude the analysis, a final experimental scenario that is based on the set of realistic micro-services described in Sect. "Micro-service Characterization". The analysis has two goals. The first is to demonstrate that even in a realistic scenario the model prediction is accurate. The second is to demonstrate that the solution found by the proposed heuristic outperforms more naïve placement solutions Fig. 11.

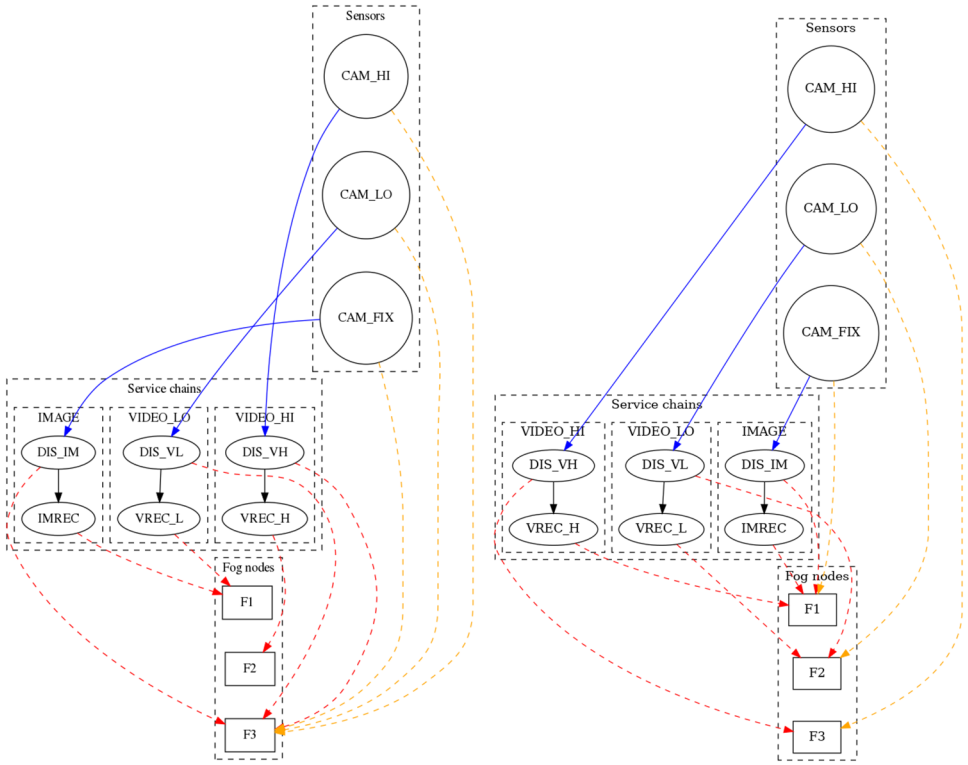
In the considered scenario, three different clients produce data. The clients are represented by a set of cameras: `CAM_HI` can take high-resolution videos, `CAM_LO` can operate only in low resolution and `CAM_FIX` can take only snapshots and cannot produce videos.

An infrastructure composed of three fog nodes is considered, two powerful processing elements (with an 8x speedup), that are `F1` and `F2` and a more ordinary edge node with no speedup that is `F3`.

As shows in Fig. 11, the micro-services are organized in three chains, each suitable to process the input of a client. Specifically, the service discovery is used to assign each data sample to the correct micro-service. To this aim, the service discovery step appears three times, one for each service depending on the type of expected output. This means that the original service discovery `SDISC` described in Sect. "Micro-service Characterization" is replicated as `DIS_IM` if the input is an image, `DIS_VH` for high-quality videos and `DIS_VL` for low-quality videos.

Figure 12 provides a comparison of the performance expected for the three micro-service chains for the placement based on the Genetic Algorithm in Fig. 11a. It is worth to note that the model can predict rather faithfully the output of the simulation.

The comparison concerns the performance of the Genetic algorithm placement (in Fig. 11a) compared to a naive alternative (Fig. 11b). To achieve this result, a random placement scheme is used. A purely random placement is not a viable option because it has no guarantee that the placement is feasible (that is no overload occurs on the nodes). In the considered example, this means that the two most



(a) Genetic Algorithm placement (b) Naïve random placement

Fig. 11 Placement of realistic micro-service chain

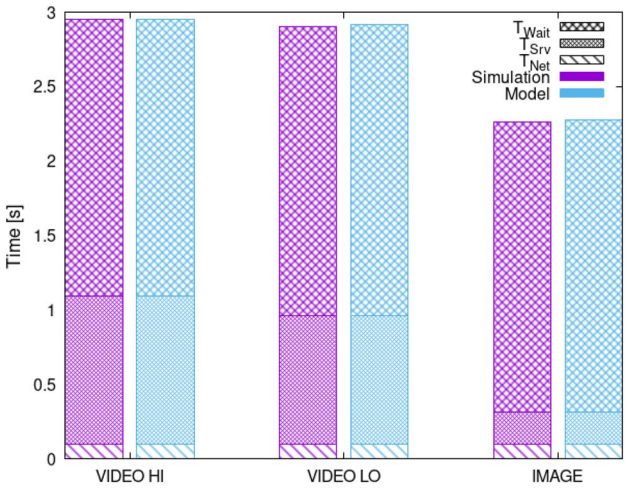


Fig. 12 Service chain response times

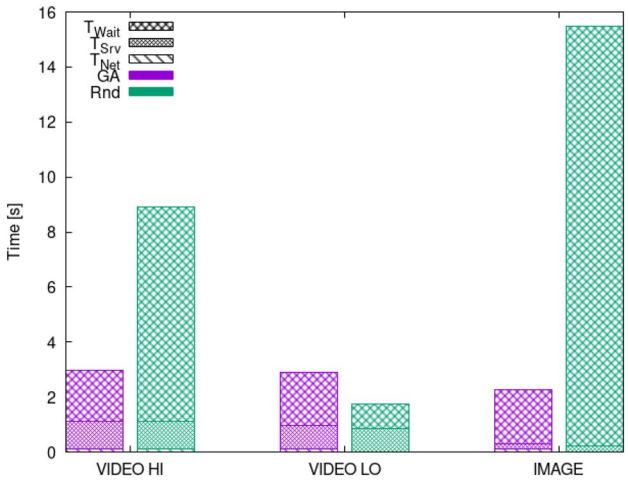


Fig. 13 Response times of GA and random service placement

computationally-demanding services must be placed on the fog nodes with more computational capacity `F1` and `F2`.

However, the comparison in Fig. 13 shows that the placement of the much faster discovery service on the same node performing heavy computations has a detrimental effect, due to the high waiting time of the jobs even for this lightweight service. furthermore, it is possible to observe how a random placement determines a significant increase in the variance of the response times of the different service chains, due to the lack of a centralized vision that aims to provide a global benefit for the whole system.

This simple example demonstrates how the proposed model and the resulting optimization problem can be applied to achieve a significant reduction of the response time experienced by the users of the system.

Conclusions

The infrastructures of fog computing have a great potential to support modern applications consisting of chains of micro-services. However, the heterogeneous and highly distributed nature of such infrastructures makes the resources allocation and management a challenging issue. This paper presents a novel optimization model for the placement of microservice-based applications over a heterogeneous infrastructure of geographically-distributed fog nodes. To support the model validation, real prototypical implementation of micro-services of a typical smart city scenario were benchmarked in the aim of building application profiles to be fed to the optimization model. Finally, to prove the viability of the ideas discussed in the paper, the GA-based placement algorithm is compared to a baseline approach enforcing a random assignment of resources to microservices. Specifically, the paper contributes to the literature with an analytical framework to model the service placement problem, a simulation-driven validation of the model, an heuristic based on genetic algorithms and its evaluation to assess the suitability of the solution under different scenarios, the use of real prototypes of micro-services. Experiments proved the effectiveness of the proposed solution, with the simulation results within 10% of the theoretical model and the optimization heuristic clearly outperforming naïve solutions. Moreover, a sensitivity analysis was conducted to assess the performance with respect to a varying length of the service chains, systems load and size of the fog architecture. In the future, a small-scaled testbed will be developed in the aim of assessing the quality of the micro-service placement schemes produced by the algorithm in a real fog-like execution environment.

Data availability All data in the paper (including both numerical results and code) in this paper can be requested to the authors.

Declarations

Conflict of Interest All authors declare that they have no conflict of interest.

Ethical Approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Byers C, Swanson R. Openfog consortium openfog reference architecture for fog computing. OpenFog Consortium Archit. Working Group, Fremont, CA, USA, Tech. Rep. OPFRA001 20817, 2017.
2. Hao Z, Novak E, Yi S, Li Q. Challenges and software architecture for fog computing. IEEE Internet Comput. 2017;21(2):44–53. https://doi.org/10.1109/MIC.2017.26.
3. Hu P, Dhelim S, Ning H, Qiu T. Survey on fog computing: architecture, key technologies, applications and open issues. J Netw Comput Appl. 2017;98:27–42. https://doi.org/10.1016/j.jnca.2017.09.002.
4. O'reilly. Microservices adoption in 2020. 2020. https://www.oreilly.com/radar/microservices-adoption-in-2020/. Accessed 12 May 2023.
5. Statista. Microservices adoption level in organizations worldwide 2021. 2021. https://www.statista.com/statistics/1233937/microservices-adoption-level-organization/. Accessed 12 May 2023.
6. Jafarnejad Ghomi E, Masoud Rahmani A, Nasih Qader N. Load-balancing algorithms in cloud computing: a survey. J Netw Comput Appl. 2017;88:50–71. https://doi.org/10.1016/j.jnca.2017.04.007.
7. Kumar P, Kumar R. Issues and challenges of load balancing techniques in cloud computing: a survey. ACM Comput Surv. 2019. https://doi.org/10.1145/3281010.
8. Kumar M, Sharma SC, Goel A, Singh SP. A comprehensive survey for scheduling techniques in cloud computing. J Netw Comput Appl. 2019;143:1–33. https://doi.org/10.1016/j.jnca.2019.06.006.
9. Singh S, Chana I. A survey on resource scheduling in cloud computing: Issues and challenges. J Grid Comput. 2016;14(2):217–64. https://doi.org/10.1007/s10723-015-9359-2.
10. Mann ZA. Allocation of virtual machines in cloud data centers-a survey of problem models and optimization algorithms. ACM Computing Surveys 2015;48(1). https://doi.org/10.1145/2797211
11. Canali C, Lancellotti R. Scalable and automatic virtual machines placement based on behavioral similarities. Computing. 2017;99(6):575–95. https://doi.org/10.1007/s00607-016-0498-5.
12. Canali C, Lancellotti R. Exploiting ensemble techniques for automatic virtual machine clustering in cloud systems. Automated Software Engineering, 2014;1–26. https://doi.org/10.1007/s10515-013-0134-y. Available online
13. Singh P, Dutta M, Aggarwal N. A review of task scheduling based on meta-heuristics approach in cloud computing. Knowl Inf Syst. 2017;52(1):1–51. https://doi.org/10.1007/s10115-017-1044-2.
14. Zhao J, Yang K, Wei X, Ding Y, Hu L, Xu G. A heuristic clustering-based task deployment approach for load balancing using Bayes theorem in cloud environment. IEEE Trans Parallel Distrib Syst. 2016;27(2):305–16. https://doi.org/10.1109/TPDS.2015.2402655.
15. Abualigah L, Alkhrabsheh M. Amended hybrid multi-verse optimizer with genetic algorithm for solving task scheduling problem in cloud computing. J Supercomput. 2022;78(1):740–65. https://doi.org/10.1007/s11227-021-03915-0.
16. Zhou Z, Li F, Zhu H, Xie H, Abawajy JH, Chowdhury MU. An improved genetic algorithm using greedy strategy toward task scheduling optimization in cloud environments. Neural Comput Appl. 2019;32:1531–41.
17. Yu R, Xue G, Zhang X. Application provisioning in FOG computing-enabled Internet-of-Things: a network perspective. In: IEEE INFOCOM 2018—IEEE Conference on computer communications, 2018; p. 783–791. https://doi.org/10.1109/INFOCOM.2018.8486269.
18. Skarlat O, Nardelli M, Schulte S, Dustdar S. Towards qos-aware fog service placement. In: 2017 IEEE 1st International Conference on fog and edge computing (ICFEC), 2017; p. 89–96. https://doi.org/10.1109/ICFEC.2017.12.
19. Canali C, Lancellotti R. A fog computing service placement for smart cities based on genetic algorithms. In: Proc. of International Conference on cloud computing and services science (CLOSER 2019), Heraklion, Greece 2019.

20. Kayal P, Liebeherr J. Distributed service placement in fog computing: an iterative combinatorial auction approach. In: 2019 IEEE 39th International Conference on distributed computing systems (ICDCS), 2019; p. 2145–2156. https://doi.org/10.1109/ICDCS.2019.00211.
21. Xiao Y, Krunz M. Qoe and power efficiency tradeoff for fog computing networks with fog node cooperation. In: IEEE INFOCOM 2017—IEEE Conference on computer communications, 2017; p. 1–9. https://doi.org/10.1109/INFOCOM.2017.8057196.
22. Santos J, Wauters T, Volckaert B, De Turck F. Towards delay-aware container-based service function chaining in fog computing. In: NOMS 2020—2020 IEEE/IFIP Network Operations and Management Symposium, 2020; p. 1–9. https://doi.org/10.1109/NOMS47738.2020.9110376.
23. Santos J, van der Hooft J, Vega MT, Wauters T, Volckaert B, De Turck F. Efficient orchestration of service chains in fog computing for immersive media. In: 2021 17th International Conference on network and service management (CNSM), 2021; p. 139–145. https://doi.org/10.23919/CNSM52442.2021.9615539.
24. Canali C, Di Modica G, Lancellotti R, Scotece D. Optimal placement of micro-services chains in a fog infrastructure. In: 12th International Conference on cloud computing and services science (CLOSER), 2022.
25. Souza VB, Masip-Bruin X, Marín-Tordera E, Sànchez-López S, Garcia J, Ren GJ, Jukan A, Juan Ferrer A. Towards a proper service placement in combined Fog-to-Cloud (F2C) architectures. Futur Gener Comput Syst. 2018;87:1–15. https://doi.org/10.1016/j.future.2018.04.042.
26. Gupta H, Vahid Dastjerdi A, Ghosh SK, Buyya R. ifogsim: a toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments. Softw Pract Exp. 2017;47(9):1275–96. https://doi.org/10.1002/spe.2509.
27. Yousefpour A, Ishigaki G, Jue JP. Fog computing: Towards minimizing delay in the internet of things. In: 2017 IEEE International Conference on edge computing (EDGE), 2017; p. 17–24. https://doi.org/10.1109/IEEE.EDGE.2017.12.
28. Aslanpour MS, Gill SS, Toosi AN. Performance evaluation metrics for cloud, fog and edge computing: a review, taxonomy, benchmarks and standards for future research. Internet of Things. 2020;12: 100273. https://doi.org/10.1016/j.iot.2020.100273.
29. Ardagna D, Ciavotta M, Lancellotti R, Guerriero M. A hierarchical receding horizon algorithm for qos-driven control of multi-iaas applications. IEEE Trans Cloud Comput. 2018. https://doi.org/10.1109/TCC.2018.2875443.
30. Binitha S, Sathya SS, et al. A survey of bio inspired optimization algorithms. International Journal of Soft Computing and Engineering. 2012;2(2):137–51.
31. Yusoh ZIM, Tang M. A penalty-based genetic algorithm for the composite saas placement problem in the cloud. In: IEEE Congress on Evolutionary Computation, 2010; p. 1–8. https://doi.org/10.1109/CEC.2010.5586151