SAIoT: Scalable Anomaly-Aware Services Composition in CloudIoT Environments

Mohammadreza Razian, Mohammad Fathian, Huaming Wu, *Member, IEEE*, Ahmad Akbari, *Member, IEEE*, and Rajkumar Buyya, *Fellow, IEEE*

***Abstract*—Among the novel IT paradigms, cloud computing and the Internet of Things (CloudIoT) are two complementary areas designed to support the creation of smart cities and application services. The CloudIoT not only presents ubiquitous services through IoT nodes but it also provides virtually unlimited resources through services composition. The services composition problem aims to find a set of services among functionally equivalent services with different Quality of Service (QoS) concerning users' constraints. To this aim, previous studies calculate QoS values through service logs without considering the presence of anomalies in the existing QoS values; however, the dynamicity of distributed service environments and communication networks in CloudIoT environments causes anomalies in the QoS values. Therefore, existing approaches fail to model QoS values accurately that leads to service-level agreement (SLA) violation and penalties for service broker. To address this challenge, we propose a scalable anomaly-aware approach (SAIoT) including two main components: the first component models QoS values based on a machine learning anomaly detection technique, to remove the existing abnormal QoS records, and the second component finds a near-optimal composition by using an effective and efficient metaheuristic algorithm. The experimental results based on real-world data sets show that our approach achieves 30.64% of the average improvement in the QoS value of a composite plan with equal or even less price compared to the previous works, such as information theory-based and advertised QoS-based methods.**

***Index Terms*—Anomaly detection, cloud computing, Internet of Things (IoT), optimization, scalability, services composition.**

Manuscript received May 11, 2020; revised August 13, 2020; accepted September 6, 2020. Date of publication September 15, 2020; date of current version February 19, 2021. This work was supported in part by the Melbourne-Chindia Cloud Computing Research Network, the University of Melbourne, Australia; in part by the National Key Research and Development Program of China under Grant 2018YFC0809800; and in part by the National Natural Science Foundation of China under Grant 61801325. *(Corresponding author: Mohammad Fathian.)*

Mohammadreza Razian is with the Cloud Computing and Distributed Systems Laboratory, School of Computing and Information Systems, University of Melbourne, Melbourne, VIC 3010, Australia, and also with the School of Industrial Engineering, Iran University of Science and Technology, Tehran 16844, Iran (e-mail: razian.mr@gmail.com).

Mohammad Fathian is with the School of Industrial Engineering, Iran University of Science and Technology, Tehran 16844, Iran (e-mail: fathian@iust.ac.ir).

Huaming Wu is with the Center for Applied Mathematics, Tianjin University, Tianjin 300072, China (e-mail: whming@tju.edu.cn).

Ahmad Akbari is with the School of Computer Engineering, Iran University of Science and Technology, Tehran 16844, Iran (e-mail: akbari@iust.ac.ir).

Rajkumar Buyya is with the Cloud Computing and Distributed Systems Laboratory, School of Computing and Information Systems, University of Melbourne, Melbourne, VIC 3010, Australia (e-mail: rbuyya@unimelb.edu.au).

Digital Object Identifier 10.1109/JIOT.2020.3023938

I. Introduction

The convergence of the Internet of Things (IoT), cloud, and data analytics has created a great opportunity for software vendors and system integrators to develop more value-added composite plans. Although cloud services are able to provide users with virtually unlimited resources, they are limited in scope. On the other hand, IoT devices are limited in computing resources, such as storage and processing [1], while they are pervasive in scope, i.e., they are distributed in many locations (in the vicinity of end users). Consequently, in the novel IT paradigm, cloud computing and IoT play a complementary role, which is referred to as cloud computing and the IoT (CloudIoT) [2].

Recently, the microservices architecture (MSA), a variant of the traditional service-oriented architecture, has become more popular than other software architectures through the *composition* of fine-grained and loosely-coupled CloudIoT *services* [3], [4]. In MSA, every single service is recognized by its function and Quality-of-Service (QoS) attributes. The QoS attributes describe the characteristics of a given service in terms of availability, reputation, response time, etc. Because a service is limited to a single function, an isolated service cannot perform the entire workflow; therefore, the services composition problem (SCP) is raised. The SCP aims to find a set of services among functionally equivalent CloudIoT services but different in QoS, concerning users' constraints/preferences and objective(s).

Many researchers have addressed the QoS-aware SCP [5]–[8]. However, there are three major limitations associated with the current approaches. First, most of the previous works model QoS values by using the service provider's advertised QoS values. In addition, they assume that the advertised QoS values remain constant over time. However, due to the inherent dynamicity of distributed services, the QoS values may not rely on predefined constant values and change in the real-world environments; therefore, modeling QoS attributes of services based on provider's advertised values results in inaccurate composition and service-level agreement (SLA) violation. For example, unmanned aerial vehicle (UAV) swarms are latency critical and QoS aware since they have to make real-time decisions to avoid collisions and obstacles [9], [10]. Second, current services composition approaches directly calculate QoS values through service logs and ignore the presence of anomalies in the historical QoS records [11]–[14]. Clearly, these approaches will fail in modeling QoS attributes of CloudIoT scenarios, where the factors, such as intermittent connections and sporadic access [15], cause anomalies in performance indicators of distributed services [16]. Third, the majority of previous studies have been devoted to services composition where services are deployed in static repositories (data centers). However, CloudIoT environments are highly dynamic and change continuously due to joining/leaving new/deprecated services [17], which require an adaptive data structure and composition algorithm to manage candidate services.

These limitations pose two interesting challenges. First, to achieve an accurate composition, anomalies in historical QoS records should be detected and removed before QoS modeling. The detection of anomalies helps the system models and calculates QoS values more accurately. In our proposed approach, we constructed a data analytic model by using an isolation forest (iForest) algorithm to detect and remove abnormal historical records before the calculation of QoS values. Second, an effective and efficient algorithm needs to be developed in order to not only manage the changes in candidate services in a timely manner but also select services for a given workflow (near-)optimally. To this aim, we propose the scalable anomaly-aware approach (SAIoT) architecture, a scalable anomaly-aware services composition in the CloudIoT environment to provide a high-quality composite plan with minimum cost (price) satisfying user's constraints. To the best of our knowledge, this is the first effort to apply the anomaly detection in services composition. Numerical results obtained by experiments based on real-world data sets demonstrate that the proposed architecture improves the aggregated QoS by almost 30.64%. The key contributions of this article are summarized as follows.

1) A data analytic model to find anomalies in historical QoS records to provide a more precise QoS modeling.
2) A mathematical formulation for the CloudIoT services composition problem so that both the objective functions and cost measurements are clearly defined.
3) An adaptive structure to model a given workflow and candidate services dynamically and efficiently.
4) A fast optimization algorithm that selects CloudIoT services among a large number of candidate services to minimize the cost.
5) Real-world data sets are taken into account to validate that the proposed algorithm is efficient enough to find a (near-)optimal composite plan in a reasonable amount of time.

The remainder of this article is structured as follows. Section II reviews the related work along with the conclusion on limitations of previous studies. Section III introduces and formulates the services composition problem for CloudIoT application. In Section IV, we demonstrate our proposed SAIoT architecture, as well as adopted algorithms and anomaly detection technique. The performance evaluation of the proposed approach in comparison with existing approaches has been included in Section V. Finally, the conclusion and future work are presented in Section VI.

II. Related Work

In order to achieve an end-to-end *optimal* QoS-aware services composition, [18]–[22] utilize integer programming and mixed-integer programming to solve the global optimization problem under the assumption that the QoS values remain constant over time. As an example, Ardagna and Pernici [20] considered range (min–max) values for some of QoS attributes. The services with QoS values fell into $\mu \pm 3\sigma$ are kept for entering into the service selection phase. Although this approach can overtake the problem of considering a constant value for QoS attributes, it still faced with the problem of constant range. Wada *et al.* [23] introduced a multiobjective approach based on a genetic algorithm to find heuristically Pareto solutions. To tackle the multicloud scenario, Yu *et al.* [24] applied the ant colony algorithm to find the minimum number of clouds in a multicloud environment. Jian *et al.* [25] targeted QoS-based service scheduling in the edge cloud computing environment to reduce the total execution time using a modified version of birds swarm optimization algorithm. Although it is important to find a composite plan in an acceptable time, falling into the local optimum solutions is the main concern for the *validity* of metaheuristic algorithms. All these studies depend on the advertised QoS values. However, practically the providers' advertised QoS values may not reflect the real-world QoS values. In other words, unlike traditional Web and cloud services composition, in the CloudIoT environments, services are distributed across the real-world *intelligent nodes*, and therefore, the QoS values may change during time.

To address the problem of estimation of QoS values, researchers utilized historical QoS records and users' ranking (on services) to model QoS attributes [11]. Wang *et al.* [33] incorporated information theory concepts into the service selection phase. Their proposed approach, first, prunes the unreliable services that are those with higher variance and entropy. The values of variance and entropy come from historical QoS records. Then, by using a mathematical optimization method, they find services satisfying users' preferences. Karimi *et al.* [13] and Khanouche *et al.* [14] applied the *K-means* clustering algorithm to speed up the process of services composition. However, the efficiency of this method is highly dependent on the veracity of historical QoS records. *Fuzzy logic*-based QoS optimization mechanism has also been applied in services composition [26]. Jian *et al.* [28] utilized historical records to model QoS attributes using an interval-based fuzzy ranking approach. Ye *et al.* [29] estimated the QoS values using multivariate time-series analysis by using service logs. Elhabbash *et al.* [30] proposed a time-awareness approach for dynamic knowledge management in volunteer computing using Chebyshev's inequality for the estimation of distribution. In addition, *recommendation systems* have been adopted in service computing for finding the user's required service. Recommender systems try to predict unknown QoS values by using other service users' experiences [27]. White *et al.* [32] proposed a recommendation-based QoS modeling by using Pearson's correlation coefficient (PCC) for finding the similarity between users/services. Recently, Wang *et al.* [34] proposed a novel QoS modeling method based on cultural distance in cyber–physical–social systems (workflows that interconnect the resources in physical, cyber, and social worlds in real time). Users in a social

TABLE I
RELATED WORK AND COMPARISON TO OUR PROPOSED SAIoT

Parameters	**Related Work**												**SAIoT**
	[11]	**[26]**	**[27]**	**[13]**	**[14]**	**[28]**	**[29]**	**[30]**	**[31]**	**[32]**	**[33]**	**[34]**	
QoS estimation	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Real dataset	✓	✗	✓	✓	✗	✓	✓	✗	✓	✓	✓	✓	✓
Scalable composition	✗	✗	✗	✓	✓	✓	✗	✓	✗	✗	✗	✗	✓
(near-)Optimality	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
Anomaly detection	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
Adaptive structure	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
CloudIoT architecture	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓

TABLE II
SUMMARY OF NOTATIONS

#	**Notation**	**Description**
1	T	Set of tasks (t_i) included in a workflow
2	Υ	A typical service formed from 2-tuple $\langle \chi, \psi \rangle$ which χ and ψ are inputs and outputs of a service
3	n	Total number of tasks in a workflow
4	ζ_i	Number of candidate services for task t_i
5	Z	The set of number of corresponding candidate services for each t_i
6	Q	Set of QoS parameters
7	CS_i	Set of candidate services for ith task
8	cs_i^j	The jth candidate service for ith task
9	B	Maximum or minimum possible aggregated QoS values for composite plan
10	b_{RTime}	Maximum possible aggregated response time value for composite plan
11	b_{Avail}	Minimum possible aggregated availability value for composite plan
12	b_{Reput}	Minimum possible aggregated reputation value for composite plan
13	W	Weight of QoS parameter declared by composite plan requester
14	ω_{cost}	Requester weight for cost parameter
15	ω_{RTime}	Requester weight for response time parameter
16	ω_{Avail}	Requester weight for availability parameter
17	ω_{Reput}	Requester weight for reputation parameter
18	$Cost(cs_i^j)$	Function for getting cost value of cs_i^j
19	$RTime(cs_i^j)$	Function for getting response time value of cs_i^j
20	$Avail(cs_i^j)$	Function for getting availability probability value of cs_i^j
21	$Reput(cs_i^j)$	Function for getting reputation average value of cs_i^j
22	k	Number of cycles in a loop structure
23	$U(Q)$	Utility function for QoS value normalization
24	CP	Composite plan (it is also referred to as composite service)
25	$s_i^{\xi_i}$	Selected candidate service in CP for t_i
26	x_{ij}	A binary variable indicating selection of a candidate service
27	$\eta(u, v)$	Heuristic information value
28	α	Intensification degree
29	β	Diversification degree
30	ρ	Evaporation rate
31	$\tau(r, s)$	Amount of pheromone currently on the path
32	$p_k(u, v)$	probability that kth ant will choose the candidate service for next task
33	$allowed_k$	Set of all remaining candidate services that should be investigated for t_{i+1}

system can advertise their observations about a service. To find a QoS value for each service, they calculate the average of users-advertised QoS values (users' rating) for each service.

Table I provides a theoretical comparison of the proposed SAIoT with other QoS-estimation studies. The main criteria used for comparison are: *QoS estimation*, *real data set*, *scalable composition*, *(near-)optimality*, *anomaly detection*, *adaptive structure*, and *CloudIoT architecture*. Considering the discussed QoS-estimation studies and comparison results in Table I, we can summarize that: 1) all of these approaches simply estimate the QoS values over service logs and they entirely ignore the presence of anomalies [35] in historical QoS records; 2) all the selection and composition approaches do not take into account an adaptive structure for candidate services encoding and use a fixed structure in the QoS modeling and workflow encoding process. However, the adaptive structure is necessary to support changes in candidate services pool and workflow; and 3) none of the previous studies propose a CloudIoT architecture to cope with the dynamicity of service environments. CloudIoT architecture helps industries to develop their software using a composition of isolated, independent, and fine-grained IoT services in a dynamic environment.

III. PROBLEM FORMULATION

Here, we propose a formal representation of the QoS-aware services composition problem. This formulation is in high-level abstraction, without considering a particular application domain. Furthermore, at the end of this section, we introduce a motivation scenario that comes from the healthcare domain to explicitly present the mechanism of CloudIoT services composition.

A. Services Composition

The main purpose of services composition is choosing a set of best fitted atomic services from a variety of candidate services according to the user' constraint on QoS values. Table II summarizes a brief description of the notations used in this article.

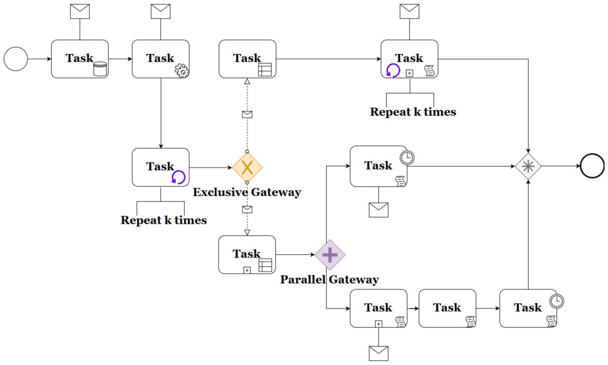


Fig. 1. Given workflow, including sequence, loop, selection, and parallel structures.

1) Workflow: Nowadays, companies and organizations only implement their primitive business functionalities and outsource other application services over trusted third parties [36]. A workflow is a collection of tasks that originated from a business process, such as authentication, payment, search/recommend a movie/hotel, navigation, etc. The set $T = \{t_1, t_2, \ldots, t_n\}$ presents a workflow within n tasks, in which n is a total number of tasks included in the workflow (we further discuss the workflow in Section III-B). Fig. 1 presents well-known structures of a workflow, including sequence, loop, selection, and parallel with the business process model and notation.

2) Service and Candidate Service: Service is a single function, loosely coupled, and highly maintainable with well-defined interfaces and operations organized around business capabilities. We define a typical *service* Υ as a 2-tuple $\langle \chi, \psi \rangle$, in which χ and ψ are inputs and outputs of a service, respectively. Let $CS_i = \{cs_i^1, cs_i^2, \ldots, cs_i^{\zeta_i}\}$ denote the candidate services that are able to perform t_i; $Z = \{\zeta_1, \zeta_2, \ldots, \zeta_n\}$ holds the number of candidate services for each t_i, i.e., a given ζ_i presents the number of candidate services corresponding to t_i. Also, cs_i^j denotes the jth candidate service for performing the ith task. We assume that the CloudIoT environment consists of multiple service providers that offer various candidate services to perform a given task at different QoS.

3) QoS Parameters: The set $Q = \{\text{cost}, \text{responseTime}, \text{availability}, \text{reputation}\}$ denotes the set of QoS parameters. For further argumentation, we defined the functions named $\text{Cost}(cs_i^j)$, $\text{RTime}(cs_i^j)$, $\text{Avail}(cs_i^j)$, and $\text{Reput}(cs_i^j)$ that return the value of the QoS parameters for a given candidate service $cs_i^j \in CS_i$. In this article, the sequential structure is taken into account while the other workflow structures, such as loop, parallel, and conditional can be converted to the sequential composition model through the methods mentioned in [37]. Thus, $t_i \in T$ is the ith task in a sequential structure and n is the total number of tasks within a workflow. QoS normalization is designed to eliminate the influence of scores in different domains, where several high scores QoS parameters reduce the distinction of those low scores on some other QoS parameters within the same operation [38]. We define the utility function $U(Q)$ as follows [18]:

$$U(Q) = \frac{Q^{\max} - Q}{Q^{\max} - Q^{\min}} \tag{1}$$

where $U(Q)$ assigns the normalized QoS values to candidate services $cs_i^j \in CS_i$, for the negative QoS attributes such as response time (longer time and lower quality), and

$$U(Q) = \frac{Q - Q^{\min}}{Q^{\max} - Q^{\min}} \tag{2}$$

and for the positive QoS attributes such as reputation (more reputation and higher quality). The terms $Q^{\max}$ and $Q^{\min}$ are the maximum and minimum values of the corresponding QoS attribute that can be obtained from service pool. We consider $U(Q) = 1$, if $Q^{\max} - Q^{\min} = 0$.

4) User's Constraints: Let $B = \{b_{\text{RTime}}, b_{\text{Avail}}, b_{\text{Reput}}\}$ denote user' constraints on response time, availability, and reputation, respectively. The composite plan must satisfy these constraints such as $\sum_{i=1}^{n} \text{RTime}(cs_i^j) \leq b_{\text{RTime}}$. The objective function minimizes the cost according to these three constraints.

5) QoS Weights: The set $W = \{\omega_{\text{RTime}}, \omega_{\text{Avail}}, \omega_{\text{Reput}}\}$ defines the weight of each QoS parameter, where $\omega_{\text{RTime}} + \omega_{\text{Avail}} + \omega_{\text{Reput}} = 1$. The user determines his/her desire weights according to the business domain of activities. For example, in a time-sensitive application such as healthcare, the cost parameter has less weight than the response time.

6) QoS-Aware Services Composition: By using the above notation, the services composition problem can be formally defined as follows. For a given workflow T, including n tasks and ζ_i candidate services for each t_i, find a composite plan (it is also referred to as composite service) $CP = \langle s_1^{\xi_1}, s_2^{\xi_2}, \ldots, s_n^{\xi_n} \rangle$ for T, where $s_i^{\xi_i} \in S_i$ represents the selected candidate service.

We have modeled the SCP as a mathematical optimization model according to the aforementioned notations. Equation (3) defines the objective function, which is to select those candidate services that maximize the aggregated utilities. In this article, we use the simple additive weighting (SAW) technique for the aggregated utility function. We consider (4)–(6) to enforce the model to satisfy user's constraints. In addition, we assume that there are several candidate services that can be invoked to address each task. Therefore, (7) defines a binary decision variable x_{ij} with the interpretation that $x_{ij} = 1$ if and only if cs_i^j is selected for task t_i. Note that x_{ij} must satisfy (8) to guarantee that the solver assigns just an exclusive candidate service for each task

$$\begin{aligned} \max \sum_{1 \leq i \leq n} \sum_{j \in Z} & x_{ij} * \omega_{\text{cost}} * U\Big(\text{Cost}\Big(cs_i^j\Big)\Big) \\ & + x_{ij} * \omega_{\text{RTime}} * U\Big(\text{RTime}\Big(cs_i^j\Big)\Big) \\ & + x_{ij} * \omega_{\text{Avail}} * U\Big(\text{Avail}\Big(cs_i^j\Big)\Big) \\ & + x_{ij} * \omega_{\text{Reput}} * U\Big(\text{Reput}\Big(cs_i^j\Big)\Big) \end{aligned} \tag{3}$$

$$\text{s.t.} \quad \sum_{1 \leq i \leq n} \sum_{j \in Z} U\Big(\text{RTime}\Big(cs_i^j\Big)\Big) * x_{ij} \leq b_{\text{RTime}} \quad \forall j \tag{4}$$

$$\prod_{1 \leq i \leq n} \sum_{j \in Z} U\Big(\text{Avail}\Big(cs_i^j\Big)\Big) * x_{ij} \geq b_{\text{Avail}} \quad \forall j \tag{5}$$

$$\frac{1}{n} * \sum_{1 \leq i \leq n} \sum_{j \in Z} U\Big(\text{Reput}\Big(cs_i^j\Big)\Big) * x_{ij} \geq b_{\text{Reput}} \quad \forall j \tag{6}$$

$$\sum_{1\le i\le n} x_{ij} = 1 \quad \forall j \tag{7}$$

$$x_{ij} \in \{0, 1\} \quad \forall i, j \tag{8}$$

$$1 \le j \le \zeta_i,\ \ \zeta_i \in Z,\ \ 1 \le i \le n \tag{9}$$

$$\omega_{\text{cost}} + \omega_{\text{RTime}} + \omega_{\text{Avail}} + \omega_{\text{Reput}} = 1. \tag{10}$$

B. CloudIoT Service Scenario

We consider a software company *A* developing a healthcare software application. The company *A* needs to consider a wide variety of CloudIoT services to combine them into the development of an integrated health system with the power of real-time medical care and predictive analytics. More precisely, the company *A* requires the following services for the underlying healthcare application.

1) *Sensing Service:* To acquire the desired data, such as location, body temperature, and blood pressure.
2) *Navigation Service:* To offer users the nearest clinic or hospital and suggest the best route to the destination.
3) *Storage Service:* To safe and reliable storing of collected data (because of the limitation in IoT devices)
4) *Analytic Service:* To analyze the acquired data for identification, prediction, and clinical decision support services.
5) *Translation Service:* To support different locales to present customized reports for the users.
6) *Payment Service:* To provide an online payment method for insurance/medical fees.

It is difficult for the company *A* to find and select the best services in terms of QoS parameters. This is because there exist lots of combinations among candidate services (i.e., services that are able to invoke for performing each aforementioned task). Furthermore, relying on the providers' advertised QoS values may not reflect the actual performance of services. Therefore, the company *A* applies a composition request to a *service broker* to find the best *composite plan*. The service broker tries to model QoS values and find the best composition in a reasonable amount of time. It is notable that the application of our proposed approach is not limited to this motivation scenario.

IV. SAIoT: Scalable Anomaly-Aware Services Composition

In order to overcome the problem of service composition in CloudIoT environments, we propose the *SAIoT* architecture shown in Fig. 2. The SAIoT is designed to ensure the successful composition using: 1) an adaptive structure to cope with the dynamicity of CloudIoT services; 2) anomaly-aware QoS modeling to reduce the effect of outliers in historical QoS records; and 3) a (near-)optimal service selection algorithm to form the composite plan in a timely manner. More precisely, there are three main components in SAIoT architecture.

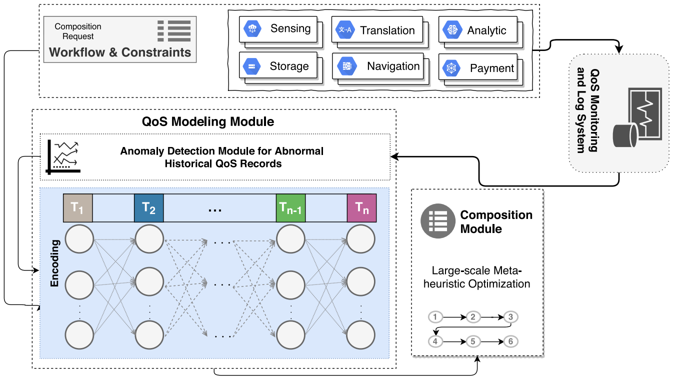


Fig. 2. Architecture of the proposed SAIoT (scalable anomaly-aware service composition in CloudIoT).

1) *Workflow and constraints* receive composition requests, as well as advertised services and their QoS values. Typically, a composition request includes a set of tasks (workflow) along with the user's constraints/preferences. Besides, Internet companies advertise their services to service brokers. The service broker provides users with an SLA and also monitors (by using a *QoS monitoring and log system*) the compliance to SLA during the service operation.
2) *QoS modeling* calculates the utility of each candidate service based on the corresponding QoS values. This module itself utilizes isolation forest (IF), a machine learning anomaly detection technique, to remove the existing abnormal QoS records (we further discuss QoS anomaly detection in Section IV-A). The *QoS modeling module* also adaptively encodes the required candidate services according to the given workflow using the proposed anomaly-aware QoS modeling and workflow encoding (AMWE) Algorithm 1 (more details are provided in Section IV-B).
3) *Composition* pursues the selection of a (near-)optimal set of services in terms of QoS attributes concerning user's constraints using the proposed (**ACFS**) Algorithm 2 (we further discuss scalable QoS-aware service selection in Section IV-C).

It is worth mentioning that the proposed SAIoT architecture is general and can be applied to different types of applications.

A. Anomaly-Aware QoS Modeling

Services on the Internet may be affected by heavy system workload, temporary machine down, and network failure [12], which all cause anomalies in QoS records. Anomalies that are also known as outliers are deviant or unusual data points. Therefore, to construct an accurate QoS model, it is essential to analyze the historical QoS records to remove anomalies. The anomaly detection is a well-researched area and there is a sufficient amount of literature that covers it in statistical and data science. We adopted IF [39], an unsupervised anomaly detection method to deal with anomalies. IF builds an ensemble of random trees for a given data set, the anomalies are points with the shortest average path length on the isolation tree [39]. We exploited the IF anomaly detection system because it is an unsupervised algorithm, which means it does not need labels to identify the anomalies in the historical QoS records. Besides, it is a lightweight anomaly detection method

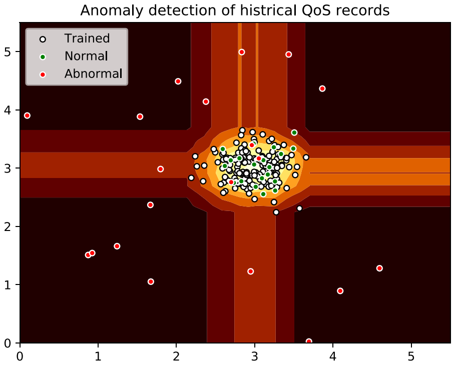


Fig. 3. Anomaly score contour of IF for a Gaussian distribution of data points.

than others that calculate distance or density [40]. In addition, the linear time complexity and a low memory requirement are best fitted for the large-scale historical data set of distributed CloudIoT services [35]. Last but not the least, parameter tuning of the IF algorithm is based on two straightforward input parameters, i.e., subsampling size and the number of trees. Liu *et al.* [39] suggested the default value of 256 for subsample and 100 trees.

An IF algorithm follows the following steps. Random and recursive partition of QoS values is performed, which is represented as a random tree. This is the training stage, where the user defines the parameters of the subsample and the number of trees. The tree construction is ended when the recursive partition of data is finished. This random partitioning produces noticeable shorter paths for anomalies. In other words, it is expected that the distance taken to reach the outliers is farther than that for the normal data (hence, they are highly likely to be anomalies). The distance of the path has been averaged and normalized to calculate the anomaly score.

As shown in Fig. 3, an anomaly score of 1 is considered as an outlier, values close to 0 is considered normal. The decision on the anomaly point is made based on this score; hence, there is no need for a label.

B. Encoding of Workflow and Services

In the next phase, we arrange the candidate services according to the workflow in a lightweight graph structure. This lightweight structure, therefore, is able to be adaptively updated according to the service pool and QoS values in a timely manner. To this aim, we encode the candidate services of a given workflow into a directed acyclic graph according to task dependency. A directed graph, or digraph, is a graph with directions assigned to its edges and denoted by (V, E), where V and E present the set of vertices and edges, respectively. The vertices represent candidate services for each task. An edge from candidate service $s_i^{\xi_i}$ to $s_{i+1}^{\xi_{i+1}}$ is connected if the execution of t_{i+1} is dependent on the execution of t_i in the workflow. Algorithm 1, the AMWE, summarizes AMWE. It is notable that the weight on edge connecting $s_i^{\xi_i}$ to $s_{i+1}^{\xi_{i+1}}$ represents the utility of service $s_{i+1}^{\xi_{i+1}}$ based on (1) and (2), which are adopted from anomaly removed historical QoS records.

Algorithm 1: AMWE

```
Input  : $T = (t_1, t_2, \ldots, t_n)$
         $CS_i = \{cs_1^i, cs_2^i, \ldots, cs_i^{\zeta_i}\}$
         $Q = \{cost,\ responseTime,\ availability,\ reputation\}$
         $W = \{\omega_{\text{cost}}, \omega_{\text{RTime}}, \omega_{\text{Avail}}, \omega_{\text{Reput}}\}$
Output : $CSGraph$: Candidate services and their QoS
         values structured as a DAG
$Q \leftarrow$ AnomalyDetectionAndFiltering($Q$) /* Find
    the average value from
    anomaly-removed historical QoS */
foreach $cs_i^j \in CS_i$ do
    $ucs_i^j \leftarrow \omega_{cost} * U(Cost(cs_i^j)) + \omega_{RTime} * U(RTime(cs_i^j)) +$
        $\omega_{Avail} * U(Avail(cs_i^j)) + \omega_{Reput} * U(Reput(cs_i^j))$
        /* Aggregated QoS values using
        SAW */
end
$startNode \leftarrow true$, $endNode \leftarrow true$
while $task\ t_i\ in\ T$ do
    if $startNode$ then
        foreach $candidate\ service\ s_1^j\ in\ CS_1$ do
            $(start, cs_1^j) \leftarrow ucs(cs_1^j)$ /* the edge
                between start node to
                candidate services for first
                task */
            $append(CSGraph, (start, cs_1^j))$
        end
        $startNode \leftarrow false$
    end
    if $endNode$ then
        foreach $candidate\ service\ cs_n^j\ in\ CS_n$ do
            $(cs_n^j, end) \leftarrow \epsilon$ /* the edge between
                candidate services for final
                task to end node */
            $append(CSGraph, (start, cs_1^j))$
        end
        $endNode \leftarrow false$
    end
    foreach $candidate\ service\ cs_i^j\ in\ CS_i$ do
        foreach $candidate\ service\ cs_i^j\ in\ CS_{i+1}$ do
            $(cs_i^j, cs_{i+1}^j) \leftarrow ucs(cs_{i+1}^j)$
            $append(CSGraph, (cs_i^j, cs_{i+1}^j))$
        end
    end
end
Set the utility value 0 to all other edges in $CSGraph$;
return ($CSGraph$)
```

C. Scalable Composition Algorithm

We developed an ant colony-based algorithm for CloudIoT services composition named ACFS in order to solve the mathematical optimization model of (3). The ant colony is an optimization algorithm inspired by swarm intelligence of natural ants when discovering the shortest path in navigation from the nest to a food source with pheromone trails [41]. Each ant moves at random and deposits pheromone on the path. The deposition of pheromone is the way that ants communicate with each other. Ants detect lead ant's path and tend to follow. As pheromone on a route increases, the selection probability of that route increases. We employed an ant colony-based algorithm because it is widely used and its proficiency in service composition has been proved [24], [42]–[44].

In proposed ACFS, artificial ants travel on the structure *CSGraph* (output of Algorithm 1) to evaluate the different feasible composite plans. In *CSGraph*, nodes present advertised candidate services and the weights on edges state the utility of candidate service regarding QoS values. Each ant is placed at a random node. The ant decides where to go based on probabilities calculated from pheromone strengths and heuristic information. The value of $\tau(u, v)$ gives the amount of pheromone that is currently on the path from a given node u to given node v. The amount of pheromone determines the level of historical fitness of that candidate service, which is investigated by other ants in previous iterations of the algorithm. The value of $\eta(u, v)$ presents the heuristic information value of the edge. The heuristic information is the score of utility for candidate services that is computed using (11), which means the more utility value a candidate service, the higher the heuristic value it obtains

$$\eta(u, v) = \omega_{\text{cost}} \cdot U\big(\text{Cost}\big(cs_v^{\xi_i}\big)\big) + \omega_{\text{RTime}} \cdot U\big(\text{RTime}\big(cs_v^{\xi_i}\big)\big) + \omega_{\text{Avail}} \cdot U\big(\text{Avail}\big(cs_v^{\xi_i}\big)\big) + \omega_{\text{Reput}} \cdot U\big(\text{Reput}\big(cs_v^{\xi_i}\big)\big) \tag{11}$$

$$p_{u,v}^k(\theta) = \begin{cases} \frac{[\tau_{u,v}(\theta)]^\alpha \cdot [\eta_{u,v}]^\beta}{\sum_{s \in \text{allowed}_k} \big[\tau_{u,s}(\theta)\big]^\alpha \cdot \big[\eta_{u,s}\big]^\beta}, & v \in \text{allowed}_k \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

Each ant endeavors to find a composite plan by finding the best candidate service. When an ant finds a best candidate service for task t_i, it should find the best candidate in CS_{i+1} for the task $t + 1$. Therefore, in (12), the term $p_{u,v}^k(\theta)$ is the probability that the kth ant chooses the candidate service for the next task when the ant is at candidate service u of task t_i, and tries to find next the candidate service v, for task t_{i+1} at time θ. Because each candidate service can be assigned exactly to a unique task [based on (7)], the set of all remaining candidate services that should be investigated for t_{i+1} is denoted as allowed$_k$. The parameters α and β have a fixed value at the beginning of a run and determine the relative importance of pheromone strengths and heuristic information, respectively. By using α and β, our proposed ACFS is able to adjust the degree of intensification (exploitation), i.e., using the latest best composite plan according to pheromone length on edges, and the degree of diversification (exploration), i.e., finding a new composite plan according to the candidate services pool.

The framework of the our ACFS algorithm is as depicted in Algorithm 2: when all ants chosed the desired candidate services for all tasks in the workflow and constructed a composite plan, the global pheromone updating takes happen according to (13). This means the current pheromone levels on all links are reduced (i.e., pheromone levels decay over time). Pheromone is lain (belatedly) by each ant as follows: it places pheromone on all links of its composite plan, with the special strength depending on the fitness of composite plan

$$\tau_{u,v}(\theta + 1) = (1 - \rho) * \tau_{u,v}(\theta) + \sum_{s=1}^{nAnt} \Delta\tau_{u,v}^k(\theta) \quad \forall(u, v) \tag{13}$$

$$\Delta\tau_{u,v}(\theta)^k = \begin{cases} \frac{1}{au^k(\theta)}, & \text{if path } (u, v) \text{ is used by ant } k \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

where $\rho \in [0, 1]$ is a parameter that controls the rate of evaporation, and $au^k(\theta)$ is the composite plan utility obtained by the kth ant. As described in Algorithm 2, the whole process will be repeated until the ACFS reaches a termination condition.

Algorithm 2: ACFS Algorithm

```
Input     : CSGraph: Candidate services and their
            (Anomaly-removed) QoS values structured in
            a DAG using AMWE algorithm
Parameter: maxIter: maximum number of iterations,
            nAnt: number of ants, $\alpha$, $\beta$: relative
            importance between global and heuristic
            information, $\rho$: the evaporation rate
Output    : BCP: best composite plan
Initialize using CSGraph
while maxIter do
    Randomly position nAnt artificial ants on some nodes
        /* Each node presents a typical
        candidate service                    */
    foreach ant = 1 to nAnt do
        ant_i Builds a composite plan in CSGraph with
            respect to $\alpha$ and $\beta$ /* select
            candidate services one after
            the other for each task in a
            workflow with probability $p_k(u, v)$
            */
    end
    Decay pheromone levels over time with respect to $\rho$

    Pheromone is lain with strength depending on how
        much the composite plan is good using
        $\tau_{u,v}(\theta + 1)$ /* apply the global
        pheromone updating rule              */
    BCP = the best composite plan obtained so far
end
return(BCP)
```

V. Performance Evaluation

The proposed SAIoT architecture is evaluated in several scenarios of services composition. The composite plan considered in the simulation scenarios is based on the sequential structure discussed in Section III-B; since any other workflow structures, such as loop, parallel, and condition can be converted to the sequential structure through the methods mentioned in [45] and [46]. For anomaly detection, we used the IF algorithm from the *scikit-learn* machine learning library in

Python [47]. IF was introduced in 2008 and became available in the scikit-learn v0.21.3 in 2016. All the measurements and experiments have been performed on an Intel Core i7-6650U 2.21-GHz processor with 16-GB RAM. The machine is running under Windows 10 and MATLAB R2018b.

To evaluate the SAIoT composition framework and its main components (QoS modeling, anomaly detection, and service selection), we first introduce the metrics and baselines defined for the evaluation of our framework. After that, we assess the *quality of composition* using multiple scenarios. The results show that our approach achieves 30.64% of the average improvement in the QoS value of a composite plan with equal or even less price compared to the previous works. Then, we show the presence of anomalies in a real data set in *QoS anomaly detection*, and finally, we present the *scalability* as well as the optimality of our composition mechanism. The evaluation proves that our proposed algorithms obtain a (near-)optimal composition in a timely manner.

A. Performance Metrics and Baselines for Comparison

To evaluate the performance of the SAIoT architecture, a series of experiments is conducted to compare our *anomaly-aware* approach with the recent attempts targeting fluctuation and variability of QoS values. We utilize the following baselines for comparison.

1) *AdQoS Based [34]:* In this approach, users of a social system advertise their observation about a service. To find a QoS value for each service, the average users-advertised QoS values (users' rating) for each service are calculated. This approach is selected because it tries to *estimate QoS values* based on users' rating. Furthermore, this approach is proposed to compose cyber–physical services that are similar to the CloudIoT environment.
2) *infoTherory Based [33]:* In this approach, unreliable candidate services are pruned before the service selection phase. The unreliable candidate services are those services with higher variance and entropy in their QoS values. According to [33], we chose 1/5 candidate services with lower variance. This approach is selected because it concerns the problem of the variability of QoS values. Furthermore, this approach has had an acceptable performance in comparison with well-known approaches, such as the skyline method [19] and global approach [11].

The following performance metrics are defined to evaluate the efficiency and effectiveness of our proposed SAIoT architecture.

1) *Quality of Composition:* Based on the concept of reliable composition used in [33] and the well-know constraint-based utility concept applied in [19], the *quality of composition* is defined as the maximum aggregated QoS for a given composition request that an approach can result with a minimum cost (price). It is worth mentioning that to provide a fair evaluation of the *quality of composition*, in the experiments, we provide decision makers with the both price of the composite plan and its aggregated QoS, simultaneously. Also, based on [33] and [34], to find the exact impact of each approach on the *quality of composition*, we implemented all approaches using 0–1 mixed-integer programming. Finally, we used the response time parameter, as the most used QoS attribute in the IoT literature [8].
2) *QoS Anomaly Detection:* In light of evidence from the study [48], as the IoT systems are getting popular, they are increasingly being used in industries over the world. However, we found no studies targeting explicitly anomaly detection in the services composition. Therefore, using the *QoS anomaly detection* criteria defined in [16], we show the presence of anomalies in the real data set and its effect on SLA violation.
3) *Scalability and Optimality:* To evaluate the execution time of the ACFS algorithm and the optimality of its solution, we compare the running time of ACFS to 0–1 mixed-integer programming (which is used by other *AdQoS-based* and *infoTherory-based* approaches. This metric proves the performance of ACFS especially for a large number of tasks or candidate services, in terms of *execution time* and *optimality of composition*. Importantly, we study the required parameters of ACFS to make sure the proposed algorithm is efficient enough to find a (near)-optimal composite plan in a reasonable amount of time.

B. Quality of Composition

Our anomaly-based QoS modeling has been tested on a wide set of experiments. The data set used in these experiments is based on the QoS values of Planetweb reported in the WS-Dream project [49], which consists of 1 974 675 real-world Web service invocations by 339 service users from 30 countries on 5825 real-world Web services in 73 countries. A number of compute nodes from the PlanetLab[1] are employed to serve as service users. PlanetLab is a global research network that supports the development of new network services and consists of 1353 nodes at 717 sites. The data set includes information of 339 service users comprising user ID, IP address, country, autonomous system (AS) number, latitude, longitude, region, and city. Moreover, information of 5.825 Web services, including service ID, WSDL address, service provider, IP address, country, AS, latitude, longitude, region, and city are included in this data set. We generate the cost price values synthetically as a function of the response time values according to [22].

In the first experiment, we have evaluated the *quality of composition* of our proposed *anomaly-aware* approach with different workflow sizes (the number of tasks). For each test case, the number of tasks in the workflow ($|T|$) has been varied between 5 and 50. The number of candidate services ($|CS_i|$) for each task has been set to 10 for all test cases. Fig. 4 shows the quality of composition, i.e., the response time of the obtained composite plan, and the price of the composite plan, simultaneously. As shown in Fig. 4, our proposed *anomaly-aware approach* not only improves the quality of composition, but it also presents a composite plan with equal or less price

[1] https://www.planet-lab.org/

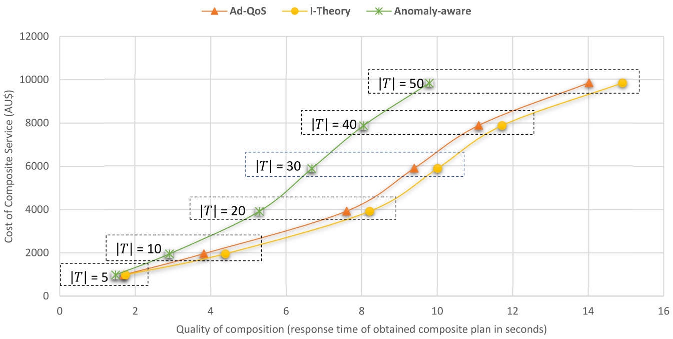


Fig. 4. Quality of composition resulted in our proposed *anomaly-aware* approach compared to other approaches based on the increment in the number of tasks in the workflow.

than *AdQoS-based* and *infoTherory-based* approaches. This is because neither *AdQoS based* nor *infoTherory based* considers the presence of anomalies in their QoS modeling phase.

More precisely, the *infoTherory-based* approach detects unreliable services and removes them from the service pool before assessing their quality and price in the selection phase. Although this service pruning helps the algorithm finds the optimal composition in a shorter time, it leads to inaccurate QoS modeling. Unlike *infoTherory-based* approach, ACFS only removes abnormal historical records rather than removing the service. Also, as the results show, in compare with *AdQoS-based*, ACFS always find a better composition with higher quality and lower price. This is because the *AdQoS-based* approach simply calculates the average QoS each service based on user observation (rating). However, users-item matrix (the collection of users' rating on the services) includes some anomalies which impact on the calculation of the utility of the candidate services $U(Q(cs_{ij}))$. On the other hand, our proposed approach first detects anomalies in recorded QoS values and therefore, is able to find a composite plan with a higher QoS with equal or even less price.

In the second experiment, we have evaluated the *quality of composition* of our proposed *anomaly-aware* approach with the increment in the number of candidate services ($|CS_i|$). In CloudIoT environments, the service broker faces with several candidates to assess and select. Therefore, for each test case, the number of candidate services has been varied between 10 and 50 with step 10. For all test cases, the number of tasks in the workflow ($|T|$) has been set to 10. Fig. 5 indicates the quality of composition (response time of the obtained composite plan) and the price of the composite plan, simultaneously. From Fig. 5, our proposed *anomaly-aware* approach not only improves the quality of composition but it also presents a composite plan with equal or less price than *AdQoS-based* and *infoTherory-based* approaches. In fact, our *anomaly-aware* approach uses a machine-learning anomaly detection system to remove the existing outliers when considering all candidate services in the selection phase. This finding can be explained by the fact that the *infoTherory-based* approach filters the unreliable candidate services from the candidate services pool and it means it takes into account only those candidate services that provide the lower variance. However, this filtering helps the system to decrease the size of the search space, it increases the probability of finding solutions with less quality of composition in comparison with the other approaches. As shown in Fig. 5, although the QoS values are observed based on users' ratings in *AdQoS based*, intermittent connections and sporadic access [15] still cause anomalies in recorded QoS values, leading to overestimation or underestimation in the QoS modeling phase.

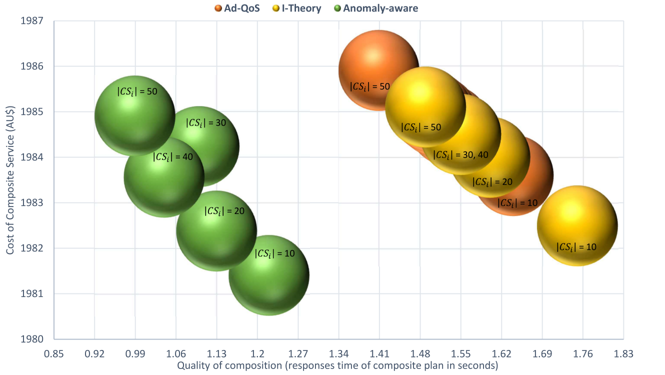
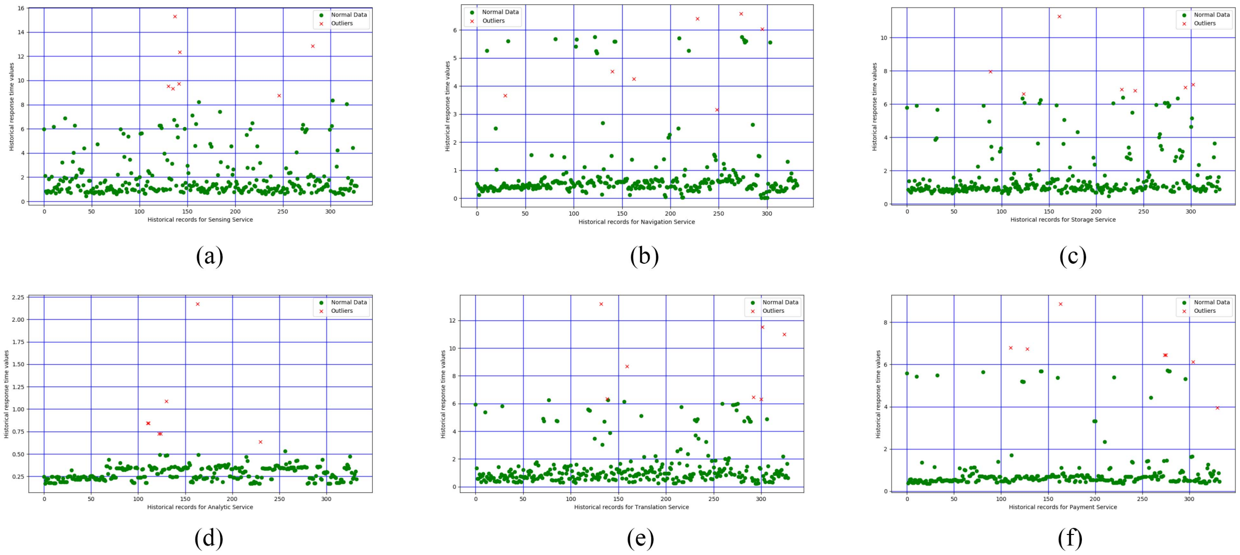


Fig. 5. Quality of composition resulted in our proposed anomaly-aware approach compared to other approaches based on the increment in the number of candidate services.

C. QoS Anomaly Detection

To show the presence of anomalies in historical records of real-world services, we detect the anomalies of six services based on the motivation scenario discussed in Section III-B. The historical records come from the aforementioned reported QoS values of Planetweb introduce in the WS-Dream project [49], which consists of 1 974 675 real-world Web service invocations by 339 service users from 30 countries on 5825 real-world Web services in 73 countries. Fig. 6 shows the anomalies detected in 320 historical QoS records of each service. As we can see, IF is able to detect anomalies in historical QoS records effectively. This is because anomalies in CloudIoT services come with two properties: the first characteristic is that the small portion of QoS instances is anomaly (see Fig. 6) and the second one is that the anomalies in historical QoS records are *few and different*. These properties make IF an ideal system for detecting anomalies [16], [50]. Using these properties, IF can effectively consider the susceptible QoS values (which are rare instances) as isolation than normal QoS instances [39].

However, one may ask why *threshold-based* approaches such as [20] have not been used for removing anomalies. It is worth mentioning that although setting a threshold value is simple and straightforward, it cannot reflect the real-world behavior of CloudIoT environments, where the dynamicity of IoT nodes and cloud infrastructure (like VM consolidation [51] and Multitenancy [52] causes anomalies in QoS values. As Fig. 6 shows, it is not possible to set a predefined value as a threshold. While the simple threshold-based technique is not able to adapt itself with abnormal changes in QoS values (which leads to inaccurate QoS modeling), our anomaly detection subsystem *adaptively* can estimate the abnormal QoS records. As a numerical comparison, the QoS values of the



(a) (b) (c)

(d) (e) (f)

Fig. 6. Anomalies detected in historical QoS records by our anomaly detection module. (a) Sensing service. (b) Navigation service. (c) Storage service. (d) Analytic service. (e) Translation service. (f) Payment service.

sensing service depicted in Fig. 6(a) with and without anomalies are 2.12 and 1.93, respectively. Interestingly, as we can see in Fig. 6(b), the abnormal QOS records of *navigation service* are not limited to a predefined range or threshold, i.e., there exist some QoS values that are abnormal and they are still in the range of normal QoS values; but they are abnormal because they are *few* and *different*. Unlike threshold-based approaches, our anomaly-aware approach does not rely on a threshold or bound.

D. Scalability and Optimality

The metaheuristic approaches such as ACO do not warranty to find an optimal solution and may fall into local optimum. Besides, another aspect of these approaches is the ability of "convergence" [53]. Having this ability means the proposed algorithm can find the optimal solution eventually. There are many attempts targeted the theoretical analysis and prove the convergence of ACO [53]–[55]. To validate that our proposed algorithm not only composes services in a timely manner but it also produces almost optimal solutions, we compared the optimality of ACFS with an optimal approach, namely, exact or *optimal* solution finder (ESF). It is notable that both *AdQoS-based* and *infoTherory-based* approaches use this approach, i.e., mathematical optimization of 0–1 mixed-integer programming to find the optimal composition. We consider two cases of experiments depending on the workflow size and the number of candidate services by using a total 40 test experiments. All experiments are executed 30 times and the average value is reported. The results show that the cost of the ACFS's composite plan is near optimum as compared to the solution obtained from ESF. Thus, ACFS is capable to compose near-optimally service set with respect to QoS parameters.

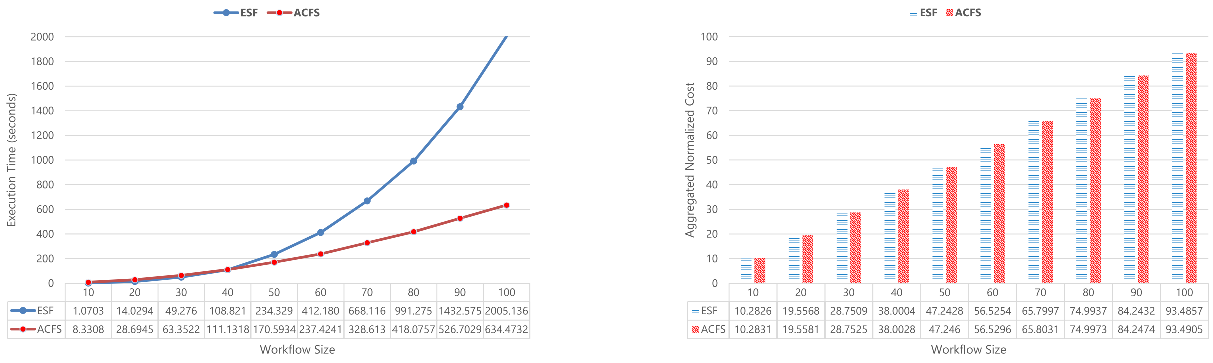
The purpose of the following experiments is to evaluate the time complexity and optimality of the proposed ACFS algorithm than other approaches. To this aim, we generated QoS values synthetically using the QWS data set collected by Al-Masri *et al.* [56].

1) Case 1: We show the performance of ACFS according to the different workflow sizes. We have taken 50 for the number of candidate services for all scenarios. Each scenario includes different workflow sizes ranging between 10 and 100 by the step of 10. Fig. 7(a) shows the execution time of ACFS and ESF. If we have a closer look at these results, we can see when the size of a workflow becomes more than 40, the ESF approach grows exponentially to compose service while our proposed ACFS algorithm grows linearly. Demonstratively, as shown in Fig. 7(b), ACFS is able to present near-optimal composition when the size of the workflow grows. Notably, when the size of the workflow becomes more than 40, ESF consumes exponential time to solve the composition problems, whereas ACFS presents a composite plan in acceptable time with high accuracy.

2) Case 2: We show the performance of ACFS according to the different numbers of candidate services. We vary the number of candidate services, ranging from 50 to 500 by the step of 50. Fig. 8(a) depicts the impact of the number of candidate services on the execution time. Notably, when the number of candidate services becomes more than 200, the execution time of ESF grows exponentially while ACFS increases linearly. These results show that mathematical optimization methods are best suited for *small-scale* scenarios. However, they take more time in real-world CloudIoT environments, where the number of tasks in a workflow and/or the number of candidate services grows increasingly. As shown in Fig. 8(b), ACFS is able to present almost optimal composition when the number of candidate services is increased.

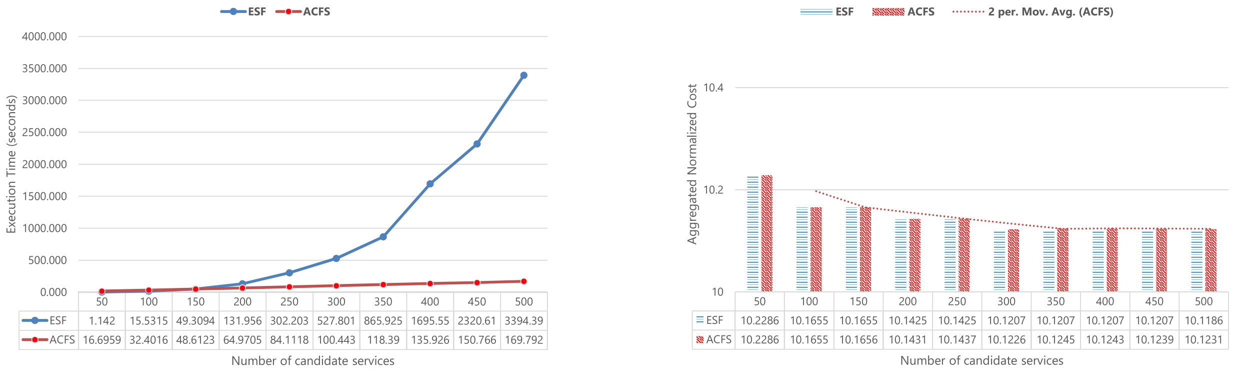
E. Discussion

We evaluate the impact of weight on heuristic information on the optimality of the solution. ACFS is able to adjust the degree of intensification and diversification in a fine-grained manner using α and β. The intensification (or exploitation) degree considers the history of the latest best composite plan derived from all ants in their previous iterations, whereas the



(a) (b)

Fig. 7. Comparison between execution times of ACFS and ESF with the increment in workflow size. (a) Different workflow sizes. (b) Different workflow sizes.



(a) (b)

Fig. 8. Comparison between execution times of ACFS and ESF with increment in the number of candidate services. (a) Different numbers of candidate services (b) Different numbers of candidate services.

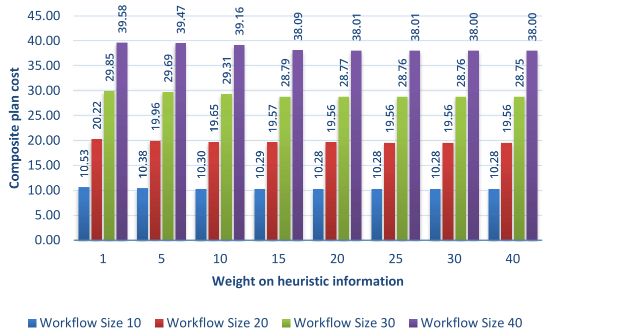


Fig. 9. Effect of heuristic information on cost of the composite plan with different workflow sizes.

degree of diversification (or exploration) guides the algorithm to explore the whole candidate services pool to find a new composite plan using heuristic information [introduced in (11)]. The first set of experiments aims to measure the influence of *heuristic information weight* on the aggregated cost of the composite plan. We consider a total of 32 experiments in which the size of workflow increases from 10 to 40 by step of 10. In all experiments, α, ρ, the number of ants, and the number of candidate services are set to 2, 0.02, 200, and 50, respectively. As shown in Fig. 9, ACFS can find a high-quality composite plan according to different workflow sizes when the weight of heuristic information is set to 40.

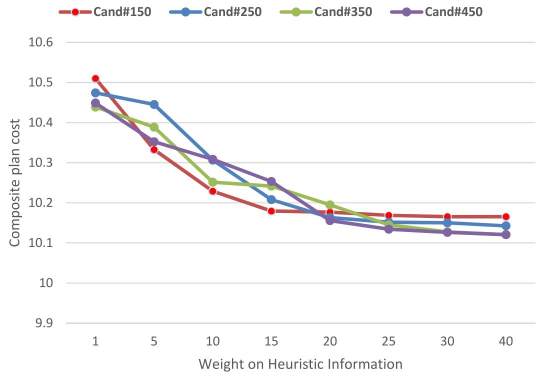


Fig. 10. Effect of heuristic information on cost of composite plan with different numbers of candidate services.

The second set of experiments is designed to show how the weight of heuristic information impacts on the cost of the composite plan with respect to the number of candidate services. Similar experiments are conducted for the increasing number of candidate services. We consider a total of 32 experiments in which the number of candidate services increases from 150 to 450 by a step of 100. In all experiments, α, ρ, the number of ants, and the workflow size are set to 2, 0.02, 1000, and 10, respectively. As shown in Fig. 10, ACFS can find a high-quality composite plan when the weight of heuristic information is set to 40 with an increment candidate service.

As a result, from Figs. 9 and 10, we can see that the more weight ACFS considers for heuristic information, the better the composite plan it finds by comparing with different workflow sizes and candidate services.

VI. Conclusion

This article introduces a novel architecture, named SAIoT (scalable anomaly-aware services composition in CloudIoT environments), to solve the problem of service composition in an integrated environment of cloud and IoT (CloudIoT). An important advantage of using the proposed architecture is that it considers both phases of QoS-modeling and composition, simultaneously. It is the first time that the QoS-modeling module is empowered by an anomaly-aware system to accurately and adaptively calculate the QoS values. Furthermore, we developed an effective and efficient algorithm to select candidate services for a given workflow based on an ant colony optimization algorithm, named ACFS. We conducted a series of experiments on the real-world data set to evaluate the proficiency of the SAIoT architecture. The results show that our approach achieves 30.64% of the average improvement in QoS value of a composite plan with equal or even less price compared to the previous works, such as *information theory-based* and *advertised QoS-based* methods.

This study can be extended in several directions. First, the method for detecting anomalies in historical QoS records can be extended to other anomaly detection systems. Second, the time complexity of our proposed algorithm can be improved by adjusting context-aware parameters in CloudIoT, e.g., the number of ants, the maximum iteration, and the weight of heuristic information. Furthermore, for the fog or edge environment, it is still an open research problem to develop a more efficient, dynamic, and anomaly-aware service composition method with polynomial-time complexity and high-quality composition.

References

[1] H. Wu, W. Knottenbelt, and K. Wolter, "An efficient application partitioning algorithm in mobile environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 7, pp. 1464–1480, Jul. 2019.

[2] A. Botta, W. De Donato, V. Persico, and A. Pescapé, "Integration of cloud computing and Internet of Things: A survey," *Future Gener. Comput. Syst.*, vol. 56, pp. 684–700, Mar. 2016.

[3] A. Balalaie, A. Heydarnoori, and P. Jamshidi, "Microservices architecture enables DevOps: Migration to a cloud-native architecture," *IEEE Softw.*, vol. 33, no. 3, pp. 42–52, May/Jun. 2016.

[4] F. Khomh and S. A. Abtahizadeh, "Understanding the impact of cloud patterns on performance and energy consumption," *J. Syst. Softw.*, vol. 141, pp. 151–170, Jul. 2018.

[5] T. H. Tan, M. Chen, É. André, J. Sun, Y. Liu, and J. S. Dong, "Automated runtime recovery for QoS-based service composition," in *Proc. 23rd Int. Conf. World Wide Web*, 2014, pp. 563–574.

[6] P. Rodriguez-Mier, C. Pedrinaci, M. Lama, and M. Mucientes, "An integrated semantic Web service discovery and composition framework," *IEEE Trans. Services Comput.*, vol. 9, no. 4, pp. 537–550, Jul./Aug. 2016.

[7] S. Deng, Z. Xiang, J. Yin, J. Taheri, and A. Y. Zomaya, "Composition-driven IoT service provisioning in distributed edges," *IEEE Access*, vol. 6, pp. 54258–54269, 2018.

[8] P. Asghari, A. M. Rahmani, and H. H. S. Javadi, "Service composition approaches in IoT: A systematic review," *J. Netw. Comput. Appl.*, vol. 120, pp. 61–77, Oct. 2018.

[9] Q. Zhang, M. Jiang, Z. Feng, W. Li, W. Zhang, and M. Pan, "IoT enabled UAV: Network architecture and routing algorithm," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3727–3742, Apr. 2019.

[10] W. Chen, B. Liu, H. Huang, S. Guo, and Z. Zheng, "When UAV swarm meets edge-cloud computing: The QoS perspective," *IEEE Netw.*, vol. 33, no. 2, pp. 36–43, Mar./Apr. 2019.

[11] M. Alrifai, D. Skoutas, and T. Risse, "Selecting skyline services for QoS-based Web service composition," in *Proc. 19th Int. Conf. World Wide Web*, 2010, pp. 11–20.

[12] H. Kil, R. Cha, and W. Nam, "Transaction history-based Web service composition for uncertain QoS," *Int. J. Web Grid Serv.*, vol. 12, no. 1, pp. 42–62, 2016.

[13] M. B. Karimi, A. Isazadeh, and A. M. Rahmani, "QoS-aware service composition in cloud computing using data mining techniques and genetic algorithm," *J. Supercomput.*, vol. 73, no. 4, pp. 1387–1415, 2017.

[14] M. E. Khanouche, F. Attal, Y. Amirat, A. Chibani, and M. Kerkar, "Clustering-based and QoS-aware services composition algorithm for ambient intelligence," *Inf. Sci.*, vol. 482, pp. 419–439, May 2019.

[15] H. Wu, Y. Sun, and K. Wolter, "Energy-efficient decision making for mobile cloud offloading," *IEEE Trans. Cloud Comput.*, vol. 8, no. 2, pp. 570–584, Apr.–Jun. 2020.

[16] S. Kardani-Moghaddam, R. Buyya, and K. Ramamohanarao, "Performance anomaly detection using isolation-trees in heterogeneous workloads of Web applications in computing clouds," *Concurrency Comput. Pract. Exp.*, vol. 31, no. 2, p. e5306, 2019.

[17] R. Ramacher and L. Mönch, "Dynamic service selection with end-to-end constrained uncertain QoS attributes," in *Proc. Int. Conf. Serv. Orient. Comput.*, 2012, pp. 237–251.

[18] L. Zeng, B. Benatallah, A. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoS-aware middleware for Web services composition," *IEEE Trans. Softw. Eng.*, vol. 30, no. 5, pp. 311–327, May 2004.

[19] T. Yu, Y. Zhang, and K.-J. Lin, "Efficient algorithms for Web services selection with end-to-end QoS constraints," *ACM Trans. Web*, vol. 1, no. 1, p. 6, 2007.

[20] D. Ardagna and B. Pernici, "Adaptive service composition in flexible processes," *IEEE Trans. Softw. Eng.*, vol. 33, no. 6, pp. 369–384, Jun. 2007.

[21] M. Alrifai and T. Risse, "Combining global optimization with local selection for efficient QoS-aware service composition," in *Proc. 18th Int. Conf. World Wide Web*, 2009, pp. 881–890.

[22] D. Schuller, U. Lampe, J. Eckert, R. Steinmetz, and S. Schulte, "Cost-driven optimization of complex service-based workflows for stochastic QoS parameters," in *Proc. IEEE Int. Conf. Web Serv. (ICWS)*, Honolulu, HI, USA, 2012, pp. 66–73.

[23] H. Wada, J. Suzuki, Y. Yamano, and K. Oba, "E^3: A multiobjective optimization framework for SLA-aware service composition," *IEEE Trans. Services Comput.*, vol. 5, no. 3, pp. 358–372, 3rd Quart., 2012.

[24] Q. Yu, L. Chen, and B. Li, "Ant colony optimization applied to Web service compositions in cloud computing," *Comput. Elect. Eng.*, vol. 41, pp. 18–27, Jan. 2015.

[25] C. Jian, M. Li, and X. Kuang, "Edge cloud computing service composition based on modified bird swarm optimization in the Internet of Things," *Clust. Comput.*, vol. 22, pp. 8079–8087, Jul. 2019.

[26] S. de Gyvés Avila and K. Djemame, "Fuzzy logic based QoS optimization mechanism for service composition," in *Proc. Int. Symp. Serv. Orient. Syst. Eng.*, Redwood City, CA, USA, 2013, pp. 182–191.

[27] A. A. P. Kazem, H. Pedram, and H. Abolhassani, "BNQM: A Bayesian network based QoS model for grid service composition," *Expert Syst. Appl.*, vol. 42, no. 20, pp. 6828–6843, 2015.

[28] X. Jian, Q. Zhu, and Y. Xia, "An interval-based fuzzy ranking approach for QoS uncertainty-aware service composition," *Optik Int. J. Light Electron Opt.*, vol. 127, no. 4, pp. 2102–2110, 2016.

[29] Z. Ye, S. Mistry, A. Bouguettaya, and H. Dong, "Long-term QoS-aware cloud service composition using multivariate time series analysis," *IEEE Trans. Services Comput.*, vol. 9, no. 3, pp. 382–393, May/Jun. 2016.

[30] A. Elhabbash, R. Bahsoon, and P. Tino, "Self-awareness for dynamic knowledge management in self-adaptive volunteer services," in *Proc. IEEE Int. Conf. Web Serv.*, Honolulu, HI, USA, 2017, pp. 180–187.

[31] A. Urbieta, A. González-Beltrán, S. B. Mokhtar, M. A. Hossain, and L. Capra, "Adaptive and context-aware service composition for IoT-based smart cities," *Future Gener. Comput. Syst.*, vol. 76, pp. 262–274, Nov. 2017.

[32] G. White, A. Palade, C. Cabrera, and S. Clarke, "IoTPredict: Collaborative QoS prediction in IoT," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. (PerCom)*, Athens, Greece, 2018, pp. 1–10.

[33] S. Wang, L. Huang, L. Sun, C.-H. Hsu, and F. Yang, "Efficient and reliable service selection for heterogeneous distributed software systems," *Future Gener. Comput. Syst.*, vol. 74, pp. 158–167, Sep. 2017.

[34] S. Wang, Y. Guo, Y. Li, and C.-H. Hsu, "Cultural distance for service composition in cyber–physical–social systems," *Future Gener. Comput. Syst.*, vol. 108, pp. 1049–1057, Jul. 2020.

[35] S. Kardani Moghaddam, R. Buyya, and K. Ramamohanarao, “ACAS: An anomaly-based cause aware auto-scaling framework for clouds,” *J. Parallel Distrib. Comput.*, vol. 126, pp. 107–120, Apr. 2019.

[36] R. Buyya *et al.*, “A manifesto for future generation cloud computing: Research directions for the next decade,” *ACM Comput. Surveys*, vol. 51, no. 5, p. 105, 2018.

[37] D. Ardagna and B. Pernici, “Global and local QoS guarantee in Web service selection,” in *Proc. Int. Conf. Bus. Process Manag.*, 2005, pp. 32–46.

[38] G. Zou, Q. Lu, Y. Chen, R. Huang, Y. Xu, and Y. Xiang, “QoS-aware dynamic composition of Web services using numerical temporal planning,” *IEEE Trans. Services Comput.*, vol. 7, no. 1, pp. 18–31, Jan.–Mar. 2014.

[39] F. T. Liu, K. M. Ting, and Z.-H. Zhou, “Isolation forest,” in *Proc. 8th IEEE Int. Conf. Data Min.*, Pisa, Italy, 2008, pp. 413–422.

[40] F. T. Liu, K. M. Ting, and Z.-H. Zhou, “Isolation-based anomaly detection,” *ACM Trans. Knowl. Discover. Data*, vol. 6, no. 1, p. 3, 2012.

[41] H. Lloyd and M. Amos, “Analysis of independent roulette selection in parallel ant colony optimization,” in *Proc. Genet. Evol. Comput. Conf.*, 2017, pp. 19–26.

[42] W. Zhang, C. K. Chang, T. Feng, and H.-Y. Jiang, “QoS-based dynamic Web service composition with ant colony optimization,” in *Proc. IEEE 34th Annu. Comput. Softw. Appl. Conf.*, Seoul, South Korea, 2010, pp. 493–502.

[43] Q. Wu and Q. Zhu, “Transactional and QoS-aware dynamic service composition based on ant colony optimization,” *Future Gener. Comput. Syst.*, vol. 29, no. 5, pp. 1112–1119, 2013.

[44] Y. Yang, B. Yang, S. Wang, F. Liu, Y. Wang, and X. Shu, “A dynamic ant-colony genetic algorithm for cloud service composition optimization,” *Int. J. Adv. Manuf. Technol.*, vol. 102, nos. 1–4, pp. 355–368, 2019.

[45] W. Dou, X. Zhang, J. Liu, and J. Chen, “HireSome-II: Towards privacy-aware cross-cloud service composition for big data applications,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 2, pp. 455–466, Feb. 2015.

[46] H. Zheng, W. Zhao, J. Yang, and A. Bouguettaya, “QoS analysis for Web service compositions with complex structures,” *IEEE Trans. Services Comput.*, vol. 6, no. 3, pp. 373–386, Jul.–Sep. 2013.

[47] *Isolation Forest Implementation*. Accessed: Jul. 19, 2019. [Online]. Available: https://scikit-learn.org/stable/auto_examples/ensemble/plot_isolation_forest.html

[48] L. D. Xu, E. L. Xu, and L. Li, “Industry 4.0: State of the art and future trends,” *Int. J. Prod. Res.*, vol. 56, no. 8, pp. 2941–2962, 2018.

[49] Z. Zheng, Y. Zhang, and M. R. Lyu, “Investigating QoS of real-world Web services,” *IEEE Trans. Services Comput.*, vol. 7, no. 1, pp. 32–39, Jan.–Mar. 2014.

[50] Z. Zou, Y. Xie, K. Huang, G. Xu, D. Feng, and D. Long, “A docker container anomaly monitoring system based on optimized isolation forest,” *IEEE Trans. Cloud Comput.*, early access, Aug. 20, 2019, doi: 10.1109/TCC.2019.2935724.

[51] I. Mohiuddin and A. Almogren, “Workload aware VM consolidation method in edge/cloud computing for IoT applications,” *J. Parallel Distrib. Comput.*, vol. 123, pp. 204–214, Jan. 2019.

[52] B. P. Rimal and M. Maier, “Workflow scheduling in multi-tenant cloud computing environments,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 1, pp. 290–304, Jan. 2017.

[53] W. J. Gutjahr, “On the finite-time dynamics of ant colony optimization,” *Methodol. Comput. Appl. Probab.*, vol. 8, no. 1, pp. 105–133, 2006.

[54] T. Stützle and M. Dorigo, “A short convergence proof for a class of ant colony optimization algorithms,” *IEEE Trans. Evol. Comput.*, vol. 6, no. 4, pp. 358–365, Aug. 2002.

[55] M. Dorigo, M. Birattari, and T. Stutzle, “Ant colony optimization,” *IEEE Comput. Intell. Mag.*, vol. 1, no. 4, pp. 28–39, Nov. 2006.

[56] E. Al-Masri and Q. H. Mahmoud, “Discovering the best Web service: A neural network-based solution,” in *Proc. IEEE Int. Conf. Syst. Man Cybern. (SMC)*, San Antonio, TX, USA, 2009, pp. 4250–4255.

Mohammadreza Razian received the master’s degree in information technology from Sharif University of Technology, Tehran, Iran, in 2014. He is currently pursuing the Ph.D. degree with the Faculty of Industrial Engineering, Iran University of Science and Technology, Tehran.

He currently joined with the Cloud Computing and Distributed Systems Laboratory, University of Melbourne, Melbourne, VIC, Australia. His research interests include the Internet of Things, cloud computing, privacy and security, and data analysis.

Mohammad Fathian received the M.S. and Ph.D. degrees in industrial engineering from Iran University of Science and Technology, Tehran, Iran, in 1997 and 2002, respectively.

He is a Professor with the School of Industrial Engineering, Iran University of Science and Technology. He is working in the areas of information technology and industrial engineering. He has more than 60 journal papers and five books in the areas of industrial engineering and information technology.

Huaming Wu (Member, IEEE) received the B.E. and M.S. degrees in electrical engineering from Harbin Institute of Technology, Harbin, China, in 2009 and 2011, respectively, and the Ph.D. degree (with Highest Hons.) in computer science from Freie Universität Berlin, Berlin, Germany, in 2015.

He is currently an Associate Professor with the Center for Applied Mathematics, Tianjin University, Tianjin, China. His research interests include mobile cloud computing, edge computing, fog computing, Internet of Things, and deep learning.

Ahmad Akbari (Member, IEEE) received the B.Sc. degree in electronics engineering and the M.Sc. degree in communications engineering from IUT, Isfahan, Iran, in 1987 and 1989, respectively, and the Ph.D. degree in electrical engineering from Rennes 1 University Rennes, Rennes, France, in 1995.

He is an Associate Professor with the School of Computer Engineering, Iran University of Science and Technology, Tehran, Iran. He has been the Dean of the IUST School of Computer Engineering since 2013. His research interests include computer networks, data communications, and signal processing applications.

Rajkumar Buyya (Fellow, IEEE) received the B.E. degree in computer science and engineering from Mysore University, Mysore, India, in 1992, the M.E. degree in computer science and engineering from Bangalore University, Bengaluru, India, in 1995, and the Doctor of Philosophy (Ph.D.) degree in computer science and software engineering from Monash University, Melbourne, VIC, Australia, in 2002.

He is a Redmond Barry Distinguished Professor and the Director of the Cloud Computing and Distributed Systems Laboratory, University of Melbourne, Melbourne. He is also serving as the Founding CEO of Manjrasoft, Melbourne, a spin-off company of the University, commercializing its innovations in Cloud Computing. He served as a Future Fellow of the Australian Research Council from 2012 to 2016. He has authored over 625 publications and seven text books including “Mastering Cloud Computing” (McGraw Hill, China Machine Press, and Morgan Kaufmann for Indian), Chinese and international markets, respectively. He also edited several books including “Cloud Computing: Principles and Paradigms” (Wiley Press, USA, February 2011). He is one of the highly cited authors in computer science and software engineering worldwide (h-index = 137, g-index = 304, and over 101 300 citations).

Dr. Buyya was recognized as a “Web of Science Highly Cited Researcher” from 2016 to 2018, by Thomson Reuters, and Scopus Researcher of the Year 2017 with Excellence in Innovative Research Award by Elsevier for his outstanding contributions to Cloud computing. Microsoft Academic Search Index ranked him #1 authored in the world from 2005 to 2016, for both field rating and citations evaluations in the area of Distributed and Parallel Computing. “A Scientometric Analysis of Cloud Computing Literature” by German scientists ranked him as the World’s Top-Cited (#1) Authored and the World’s Most-Productive (#1) Authored in Cloud Computing. He is currently serving as an Editor-in-Chief for *Journal of Software: Practice and Experience*.