SAIoT: Scalable Anomaly-aware Services Composition in CloudIoT Environments

Mohammadreza Razian, Mohammad Fathian, Huaming Wu, *Member, IEEE,* Ahmad Akbari, 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. 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 cause anomalies in the QoS values. Therefore, existing approaches fail to model QoS values accurately which 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 meta-heuristic algorithm. The experimental results based on real-world datasets 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.**

***Index Terms*—IoT, Anomaly Detection, Cloud Computing, Services Composition, Scalability, Optimization**

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 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 service composition problem (SCP) is raised. The SCP aims to find a set of services among functionally equivalent CloudIoT services but different in Quality of Service (QoS), concerning users' constraints/preferences and objective(s).

Many researchers have addressed the QoS-aware service composition problem [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' advertised values results in inaccurate composition and SLA (Service Level Agreement) 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 service 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 like 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 service composition where services 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

M. Razian is with Cloud Computing and Distributed Systems (CLOUDS) Laboratory, School of Computing and Information Systems, The University of Melbourne, Australia and with School of Industrial Engineering, Iran University of Science and Technology, Tehran, Iran. E-mail: razian.mr@gmail.com

M. Fathian is with School of Industrial Engineering, Iran University of Science and Technology, Tehran, Iran. E-mail: fathian@iust.ac.ir

H. Wu is with Center for Applied Mathematics, Tianjin University, Tianjin 300072, China. E-mail: whming@tju.edu.cn

A. Akbari is with School of Computer Engineering, Iran University of Science and Technology, Tehran, Iran. E-mail: akbari@iust.ac.ir

R. Buyya is with Cloud Computing and Distributed Systems (CLOUDS) Laboratory, School of Computing and Information Systems, The University of Melbourne, Australia. E-mail: rbuyya@unimelb.edu.au

(Corresponding author: Mohammad Fathian)

QoS records should be detected and removed before QoS modeling. 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 Isolation Forest (iForest) algorithm to detect and remove the 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 **SAIoT** architecture, a **S**calable **A**nomaly-aware Services Composition in Cloud**IoT** 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 datasets demonstrate that the proposed architecture improves the aggregated quality of service by almost 30.64%. The key contributions of this paper 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 datasets 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 rest of the paper is structured as follows: Section II reviews the related work along with the conclusion on the 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, 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] consider a range (min-max) values for some of QoS attributes. 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 multi-objective approach based on a genetic algorithm to find heuristically Pareto solutions. To tackle the multi-cloud scenario, Yu et al. [24] applied the Ant Colony algorithm to find the minimum number of clouds in a multi-cloud 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 meta-heuristic 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 the QoS attributes [11]. Wang et al. [33] incorporate information theory concepts into the service selection phase. Their proposed approach, first, prunes the unreliable services which 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. Researchers in [13], [14] applied the *K-means* clustering algorithm to speed up the process of service 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 applied in services composition [26]. Jian et al. [28] utilize historical records to model QoS attributes using an interval-based fuzzy ranking approach. Ye et al. [29] estimate the QoS values using multi-variate time series analysis by using service logs. Elhabbash et al. [30] propose a time-awareness approach for dynamic knowledge management in Volunteer Computing using Chebyshev's inequality for 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 the unknown QoS values by using other service users' experiences [27]. White et al. [32] propose a recommendation-based QoS modeling by using PCC (Pearson's correlation coefficient) 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 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 proposed SAIoT with other QoS-estimation studies. The main criteria used for comparison are: *QoS estimation*, *Real dataset*, *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

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	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓

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. (3) None of previous studies propose a CloudIoT architecture to cope with 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 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 which comes from health-care domain to explicitly present the mechanism of CloudIoT services composition.

A. Services Composition

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

a) 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 $T = \{t_1, t_2, \cdots, t_n\}$ 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.

b) 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 $\langle \chi, \psi \rangle$ which χ and ψ are inputs and outputs of a service, respectively. Let $CS_i = \{cs_i^1, cs_i^2, \cdots, cs_i^{\zeta_i}\}$ denotes the candidate services which are able to perform t_i; $Z = \{\zeta_1, \zeta_2, \cdots, \zeta_n\}$ holds

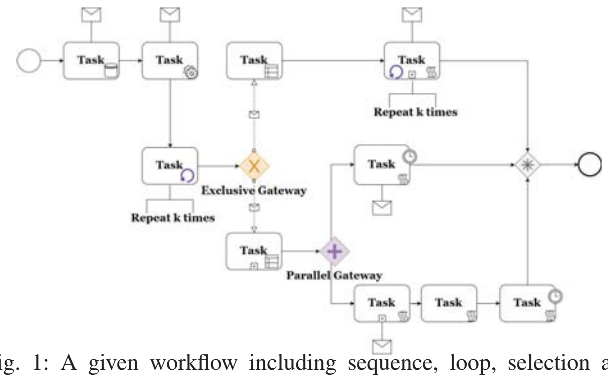


Fig. 1: A given workflow including sequence, loop, selection and parallel structures

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 jth candidate service for performing ith task. We assume that the CloudIoT environment consists of multiple service providers which offer the various candidate services to perform a given task at different quality of service.

c) QoS parameters: The set $Q = \{$*cost, responseTime, availability, reputation*$\}$ denotes the set of quality of service (QoS) parameters. For further argumentation, we defined the functions named $Cost(cs_i^j)$, $RTime(cs_i^j)$, $Avail(cs_i^j)$, and $Reput(cs_i^j)$ which return the value of the QoS parameters for a given candidate service $cs_i^j \in CS_i$. In this paper, 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 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 like response time (longer time, lower quality), and

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

and for the positive QoS attributes like reputation (more reputation, higher quality). The terms Q^{max} and Q^{min} are

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}

the maximum and minimum values of the corresponding QoS attribute which can be obtained from service pool. We consider $U(Q) = 1$, if $Q^{max} - Q^{min} = 0$.

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

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

f) 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) $CP = \langle s_1^{\xi_1}, s_2^{\xi_2}, \cdots, s_n^{\xi_n} \rangle$ for T, where $s_i^{\xi_i} \in S_i$ represents selected candidate service.

We have modeled the SCP as a mathematical optimization model according to the aforementioned notations. Eq. 3 defines the objective function which is to select those candidate services that maximize the aggregated utilities. In this paper, we use the simple additive weighting (SAW) technique for the aggregated utility function. We consider Eqs. 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, Eq. 7 defines a binary decision variable x_{ij} with the interpretation that $x_{ij} = 1$ if and only if the cs_i^j is selected for task t_i. Note that x_{ij} must satisfy Eq. 8 to guarantee that the solver assigns just an exclusive candidate service for each task.

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

s.t.

$$\sum_{1 \leq i \leq n} \sum_{j \in Z} U\big(RTime(cs_i^j)\big) * x_{ij} \leq b_{RTime} \ \forall j \tag{4}$$

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

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

$$\sum_{1 \leq i \leq n} x_{ij} = 1, \ \forall j \tag{7}$$

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

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

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

B. CloudIoT Service Scenario

We consider a software company A developing a health-care software application. The company A needs to consider a wide variety of CloudIoT services to combine them into development of an integrated health system with the power

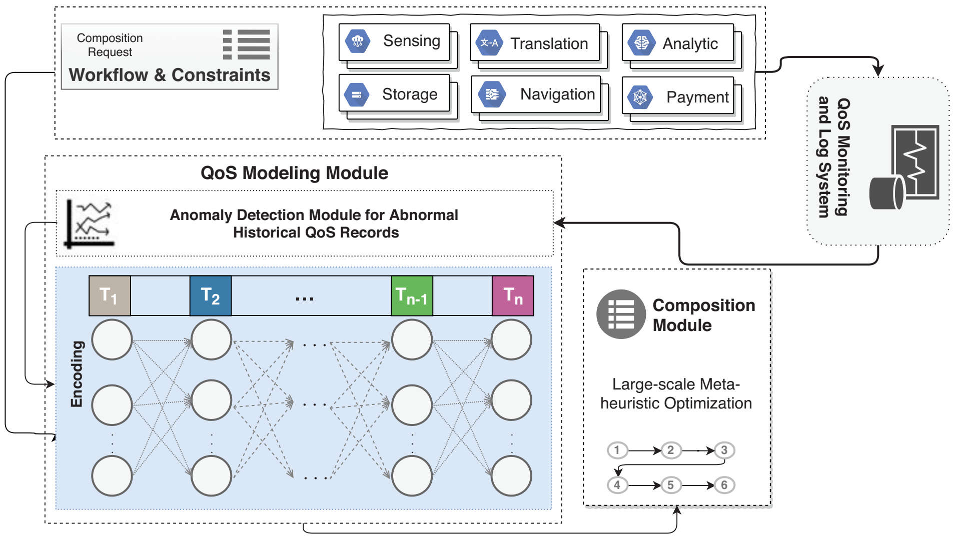


Fig. 2: Architecture of proposed SAIoT (scalable anomaly-aware service composition in CloudIoT)

of real-time medical care and predictive analytics. More precisely, the company A requires the following services for the underlying health-care application:

- *Sensing Service*: to acquire desired data like location, body temperature, and blood pressure.
- *Navigation Service*: to offer users the nearest clinic or hospital and suggest the best route to the destination.
- *Storage Service*: to safe and reliable storing of collected data (because of limitation in IoT devices)
- *Analytic Service*: to analyze the acquired data for identification, prediction, and clinical decision support services.
- *Translation Service*: to support different locale to present customized reports for the users.
- *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 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 the SAIoT architecture:

- *Workflow and Constraints* receives composition requests as well as advertised services and their QoS values. Typically, a composition request includes a set of tasks (workflow) along with user's constraints/preferences. Besides, Internet companies advertise their services to the service brokers. Service broker provides users with an SLA and also monitors (by using a *QoS Monitoring and Log System*) the compliance to the SLA during the service operation.
- *QoS Modeling* calculates the utility of each candidate service based on corresponding QoS values. This module itself utilizes Isolation Forest, 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 proposed (**AMWE**) algorithm 1 (more details are provided in Section IV-B).
- *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 which 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 Isolation Forest (IF) [39], an unsupervised anomaly detection method to deal with anomalies. Isolation Forest (IF) builds an ensemble of random trees for a given dataset, the anomalies are points with the shortest average path length on the Isolation Tree [39]. We exploited the Isolation Forest 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 light-weight anomaly detection method than others that calculate distance or density [40]. In addition, the linear time complexity and a low memory requirement is best-fitted for the large-scale historical dataset of distributed CloudIoT services [35]. Last but not the least, parameter tuning of the Isolation Forest algorithm is based on two straight-forward input parameters, i.e., sub-sampling size and number of trees. The authors in [39] suggest the default value of 256 for sub-sample and 100 trees.

Isolation Forest algorithm follows the below 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.

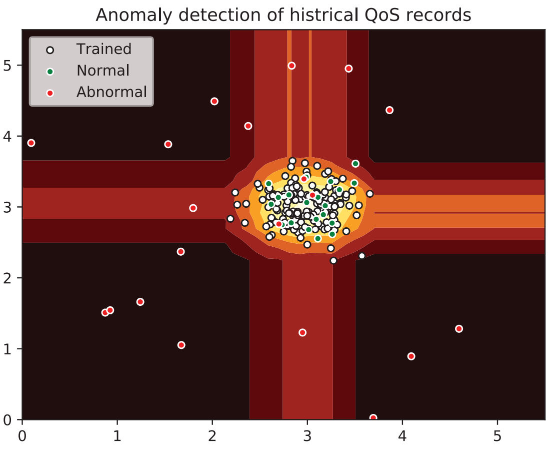


Fig. 3: Anomaly score contour of IF for a Gaussian distribution of data points

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 light-weight graph structure. This light-weight 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 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 Anomaly-aware QoS Modeling and Workflow Encoding (AMWE), summaries the anomaly-aware QoS modeling and workflow encoding. 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 equations 1 and 2 which are adopted from anomaly-removed historical QoS records.

C. Scalable Composition Algorithm

We developed an **A**nt **C**olony-based algorithm **f**or CloudIoT **S**ervices composition named **ACFS** in order to solve the mathematical optimization model of Eq. 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 plan. 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 which 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 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 which is computed using Eq. 11, which means the more utility value a candidate service, the higher the heuristic value it obtains.

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

$$p_{u,v}^k(\theta) = \begin{cases} \dfrac{\big[\tau_{u,v}(\theta)\big]^\alpha * \big[\eta_{u,v}\big]^\beta}{\sum_{s \in allowed_k} \big[\tau_{u,s}(\theta)\big]^\alpha * \big[\eta_{u,s}\big]^\beta}, v \in allowed_k \\ 0, \quad otherwise \end{cases} \tag{12}$$

Each ant endeavors to find a composite plan through 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 Eq. 12, the term $p_{u,v}^k(\theta)$ is the probability that kth ant chooses the candidate service for

Algorithm 1: Anomaly-aware QoS Modeling and Workflow Encoding (AMWE)

```
Input     : $T = (t_1, t_2, \cdots, t_n)$
            $CS_i = \{cs_1^i, cs_2^i, \cdots, cs_i^{\zeta_i}\}$
            $Q = \{cost,\ responseTime,\ availability,\ reputation\}$
            $W = \{\omega_{cost},\ \omega_{Rtime},\ \omega_{Avail},\ \omega_{Reput}\}$
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 ($CSGraph$)
```

next task when the ant is at candidate service u of task t_i, and tries to find next 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 Eq. 7), the set of all remaining candidate services that should be investigated for t_{i+1} 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 degree of diversification (exploration), i.e., finding a new composite plan according to the candidate services pool.

Algorithm 2: The 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 $CSGraph$ w.r.t $\alpha$ 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 w.r.t $\rho$

    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($BCP$)
```

The framework of the our ACFS algorithm is as depicted in Algorithm 2: when all ants chose the desire candidate services for all tasks in the workflow and constructed a composite plan, the global pheromone updating takes happen according to Eq. 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.

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

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

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 kth ant. As described in Algorithm 2, the whole process will be repeated until the ACFS reaches a termination condition.

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], [46]. For anomaly detection, we used the Isolation Forest algorithm from the *scikit-learn* machine learning library in Python [47]. Isolation Forest 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(R) Core(TM) 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 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 dataset 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 are 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:

- *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 is 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 which are similar to the CloudIoT environment.
- *infoTherory-based* [33]: In this approach, unreliable candidate services are pruned before 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 variability of QoS values. Furthermore, this approach has had an acceptable performance in comparison with well-known approaches like 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:

- 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 quality of service for a given composition request which 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], [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].
- QoS Anomaly Detection: In light of evidence from 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 dataset and its effect on SLA violation.
- Scalability and Optimality: To evaluate the execution time of 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 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 dataset 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 5,825 real-world web services in 73 countries. A number of compute nodes from the PlanetLab[1] are employed to serve

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

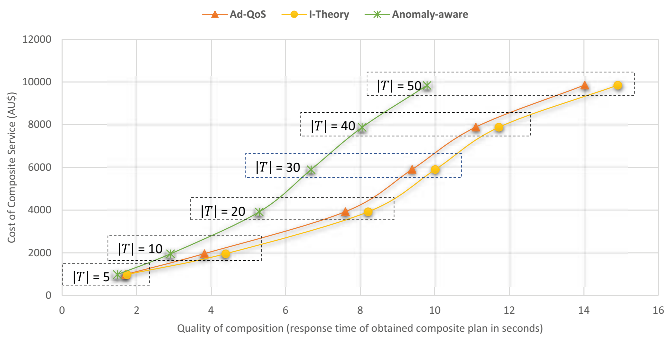


Fig. 4: The quality of composition resulted in our proposed *anomaly-aware* approach compared to other approaches based on the increment in number of tasks in the workflow

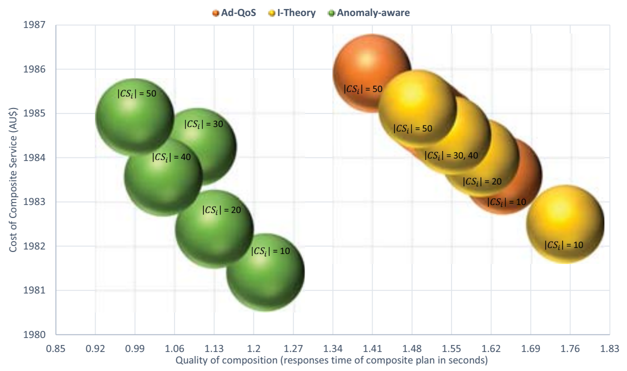


Fig. 5: The quality of composition resulted in our proposed anomaly-aware approach compared to other approaches based on the increment in the number of candidate services

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 dataset 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, city are included in this dataset. We generate the cost price values synthetically as a function of the response time values according to [22].

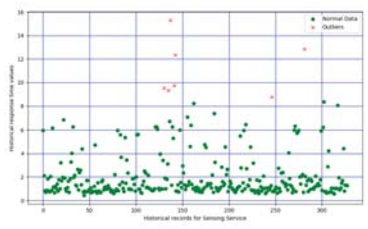
In the first experiment, we have evaluated the *Quality of composition* of our proposed *anomaly-aware* approach with different workflow sizes (number of tasks). For each test case, the number of tasks in the workflow ($|T|$) has been varied between 5 and 50. The number of candidate services ($|CS_i|$) 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 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 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 quality of service with equal or even less price.

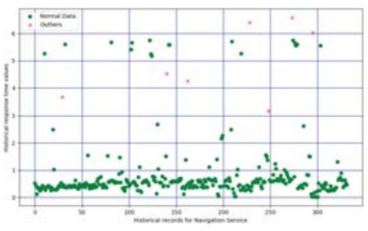
In the second experiment, we have evaluated the *Quality of composition* of our proposed *anomaly-aware* approach with the increment in the number of candidate services ($|CS_i|$). In CloudIoT environments, the service broker faces with several candidates to assess and select. Therefore, for each test 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 obtained composite plan) and the price of the composite plan, simultaneously. Refer to 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 *infoTherory-based* approach filters the unreliable candidate services from the candidate services pool and it means it takes into account only those candidate services which 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.

C. QoS Anomaly Detection

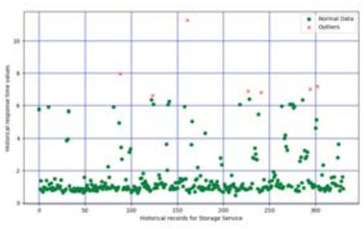
To show the presence of anomalies in historical records of real-world services, we detect the anomalies of six services based on the motivation scenario discussed in Section III-B. The historical records come from aforementioned reported QoS values of Planetweb introduce in the WS-Dream project [49] which consists of 1,974,675 real-world web service invocations by 339 service users from 30 countries on 5,825 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, Isolation Forest is able to detect anomalies in historical QoS records effectively. This is because



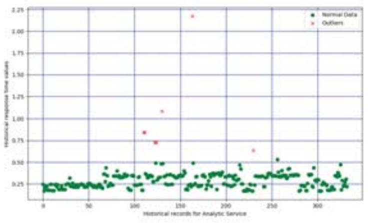
(a) Sensing Service



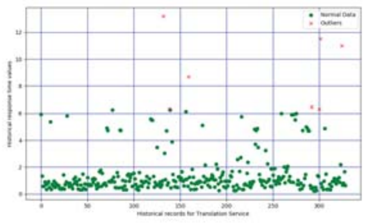
(b) Navigation Service



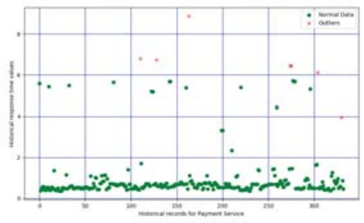
(c) Storage Service



(d) Analytic Service

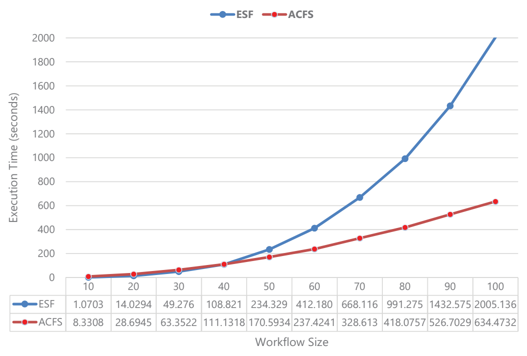


(e) Translation Service



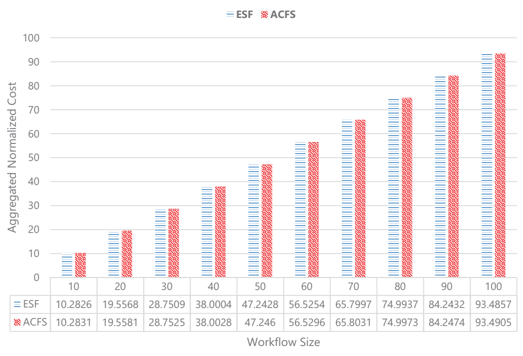
(f) Payment Service

Fig. 6: The anomalies detected in historical QoS records by our Anomaly Detection Module



	10	20	30	40	50	60	70	80	90	100
ESF	1.0703	14.0294	49.276	108.821	234.329	412.180	668.116	991.275	1432.575	2005.136
ACFS	8.3308	28.6945	63.3522	111.1318	170.5934	237.4241	328.613	418.0757	526.7029	634.4732

(a) Different workflow sizes



	10	20	30	40	50	60	70	80	90	100
ESF	10.2826	19.5568	28.7509	38.0004	47.2428	56.5254	65.7997	74.9937	84.2432	93.4857
ACFS	10.2831	19.5581	28.7525	38.0028	47.246	56.5296	65.8031	74.9973	84.2474	93.4905

(b) Different workflow sizes

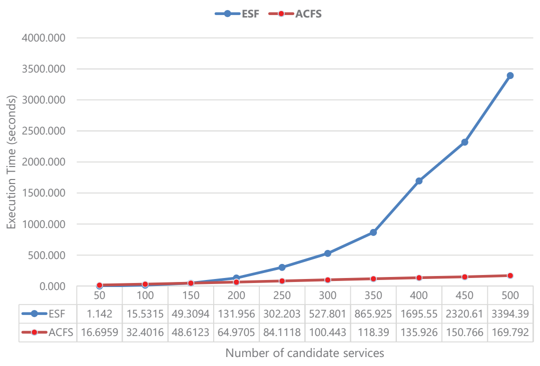
Fig. 7: Comparison between Execution times of ACFS and ESF with the increment in workflow size

anomalies in CloudIoT services come with two properties: the first characteristic is that the small portion of QoS instances are anomaly (see Fig. 6) and the second one is that the anomalies in historical QoS records are *few and different*. These properties make Isolation Forest an ideal system for detecting anomalies [16], [50]. Using these properties, Isolation Forest 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 like [20] has 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 Multi-tenancy [52] cause 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 *Sensing Service* depicted in Fig. 6a, with and without anomalies is 2.12 and 1.93, respectively. Interestingly, as we can see in Fig. 6b, the abnormal QOS records of *Navigation Service* are not limited to a predefined range or threshold, i.e., there exist some QoS values which 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.

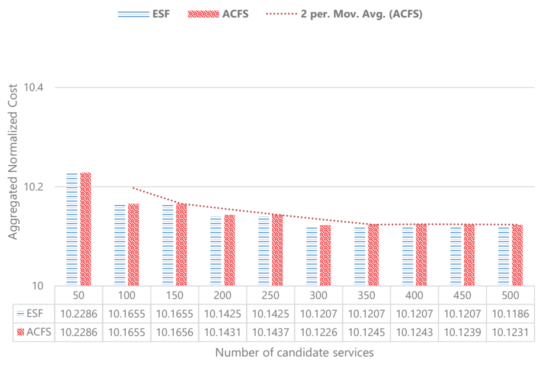
D. Scalability and Optimality

The meta-heuristic approaches like 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 proof 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 **E**xact or *optimal* **S**olution **F**inder (**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 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



	50	100	150	200	250	300	350	400	450	500
ESF	1.142	15.5315	49.3094	131.956	302.203	527.801	865.925	1695.55	2320.61	3394.39
ACFS	16.6959	32.4016	48.6123	64.9705	84.1118	100.443	118.39	135.926	150.766	169.792

(a) Different number of candidate services



	50	100	150	200	250	300	350	400	450	500
ESF	10.2286	10.1655	10.1655	10.1425	10.1425	10.1207	10.1207	10.1207	10.1207	10.1186
ACFS	10.2286	10.1655	10.1656	10.1431	10.1437	10.1226	10.1245	10.1243	10.1239	10.1231

(b) Different number of candidate services

Fig. 8: Comparison between execution times of ACFS and ESF with increment in number of candidate services

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 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 dataset collected by Al-Masri et al. [56].

1) Case 1: We show the performance of the 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 to 100 by step of 10. Fig. 7a 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. 7b, 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 the ACFS according to the different number of candidate services. We vary the number of candidate services, ranging from 50 to 500 by the step of 50. Fig. 8a 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. 8b, 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 latest best composite plan derived from all ants in their previous iterations, whereas degree of diversification (or exploration) guides algorithm to explore the whole candidate services pool to find a new composite plan using heuristic information (introduced in Eq. 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.

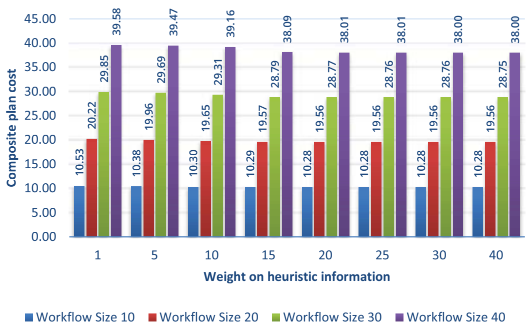


Fig. 9: The effect of heuristic information on cost of the composite plan with different workflow sizes

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 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 services.

As a result, from Figs. 9-10, 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.

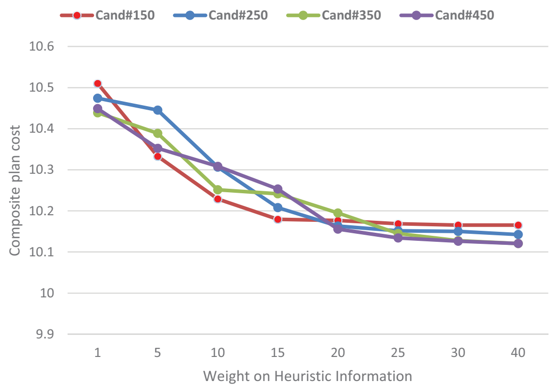


Fig. 10: The effect of heuristic information on cost of composite plan with different number of candidate services

VI. Conclusions and Future Work

This paper introduces a novel architecture, named SAIoT (**S**calable **A**nomaly-aware Services Composition in Cloud**IoT** 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 dataset to evaluate the proficiency of 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 Transactions on Parallel and Distributed Systems*, vol. 30, no. 7, pp. 1464–1480, 2019.

[2] A. Botta, W. De Donato, V. Persico, and A. Pescapé, "Integration of cloud computing and internet of things: a survey," *Future Generation Computer Systems*, vol. 56, pp. 684–700, 2016.

[3] A. Balalaie, A. Heydarnoori, and P. Jamshidi, "Microservices architecture enables devops: Migration to a cloud-native architecture," *Ieee Software*, vol. 33, no. 3, pp. 42–52, 2016.

[4] F. Khomh and S. A. Abtahizadeh, "Understanding the impact of cloud patterns on performance and energy consumption," *Journal of Systems and Software*, vol. 141, pp. 151–170, 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 *Proceedings of the 23rd international conference on World wide web*, pp. 563–574, ACM, 2014.

[6] P. Rodriguez-Mier, C. Pedrinaci, M. Lama, and M. Mucientes, "An integrated semantic web service discovery and composition framework," *IEEE transactions on services computing*, vol. 9, no. 4, pp. 537–550, 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," *Journal of Network and Computer Applications*, vol. 120, pp. 61–77, 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 of Things Journal*, vol. 6, no. 2, pp. 3727–3742, 2019.

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

[11] M. Alrifai, D. Skoutas, and T. Risse, "Selecting skyline services for qos-based web service composition," in *Proceedings of the 19th international conference on World wide web*, pp. 11–20, 2010.

[12] H. Kil, R. Cha, and W. Nam, "Transaction history-based web service composition for uncertain qos," *International Journal of Web and Grid Services*, 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," *The Journal of Supercomputing*, 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," *Information Sciences*, vol. 482, pp. 419–439, 2019.

[15] H. Wu, Y. Sun, and K. Wolter, "Energy-efficient decision making for mobile cloud offloading," *IEEE Transactions on Cloud Computing*, 2018.

[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 and Computation: Practice and Experience*, p. e5306, 2019.

[17] R. Ramacher and L. Mönch, "Dynamic service selection with end-to-end constrained uncertain qos attributes," in *International Conference on Service-Oriented Computing*, pp. 237–251, Springer, 2012.

[18] L. Zeng, B. Benatallah, A. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "Qos-aware middleware for web services composition," *IEEE Transactions on software engineering*, vol. 30, no. 5, pp. 311–327, 2004.

[19] T. Yu, Y. Zhang, and K.-J. Lin, "Efficient algorithms for web services selection with end-to-end qos constraints," *ACM Transactions on the Web (TWEB)*, vol. 1, no. 1, pp. 6–es, 2007.

[20] D. Ardagna and B. Pernici, "Adaptive service composition in flexible processes," *IEEE Transactions on software engineering*, vol. 33, no. 6, pp. 369–384, 2007.

[21] M. Alrifai and T. Risse, "Combining global optimization with local selection for efficient qos-aware service composition," in *Proceedings of the 18th international conference on World wide web*, pp. 881–890, ACM, 2009.

[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 *Web Services (ICWS), 2012 IEEE 19th International Conference on*, pp. 66–73, IEEE, 2012.

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

[24] Q. Yu, L. Chen, and B. Li, "Ant colony optimization applied to web service compositions in cloud computing," *Computers & Electrical Engineering*, vol. 41, pp. 18–27, 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," *Cluster Computing*, pp. 1–9, 2018.

[26] S. de Gyvés Avila and K. Djemame, "Fuzzy logic based qos optimization mechanism for service composition," in *International Symposium on Service-Oriented System Engineering*, pp. 182–191, IEEE, 2013.

[27] A. A. P. Kazem, H. Pedram, and H. Abolhassani, "Bnqm: a bayesian network based qos model for grid service composition," *Expert Systems with Applications*, 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-International Journal for Light and Electron Optics*, 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 Transactions on Services Computing*, vol. 9, no. 3, pp. 382–393, 2016.

[30] A. Elhabbash, R. Bahsoon, and P. Tino, “Self-awareness for dynamic knowledge management in self-adaptive volunteer services,” in *International Conference on Web Services*, pp. 180–187, IEEE, 2017.

[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 Generation Computer Systems*, vol. 76, pp. 262–274, 2017.

[32] G. White, A. Palade, C. Cabrera, and S. Clarke, “Iotpredict: collaborative qos prediction in iot,” in *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pp. 1–10, IEEE, 2018.

[33] S. Wang, L. Huang, L. Sun, C.-H. Hsu, and F. Yang, “Efficient and reliable service selection for heterogeneous distributed software systems,” *Future Generation Computer Systems*, vol. 74, pp. 158–167, 2017.

[34] S. Wang, Y. Guo, Y. Li, and C.-H. Hsu, “Cultural distance for service composition in cyber–physical–social systems,” *Future Generation Computer Systems*, 2018.

[35] S. Kardani Moghaddam, R. Buyya, and K. Ramamohanarao, “Acas: An anomaly-based cause aware auto-scaling framework for clouds,” *Journal of Parallel and Distributed Computing*, vol. 126, pp. 107–120, 2019.

[36] R. Buyya, S. N. Srirama, G. Casale, R. Calheiros, Y. Simmhan, B. Varghese, E. Gelenbe, B. Javadi, L. M. Vaquero, M. A. Netto, *et al.*, “A manifesto for future generation cloud computing: Research directions for the next decade,” *ACM Computing Surveys (CSUR)*, vol. 51, no. 5, p. 105, 2018.

[37] D. Ardagna and B. Pernici, “Global and local qos guarantee in web service selection,” in *International Conference on Business Process Management*, pp. 32–46, Springer, 2005.

[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 Transactions on Services Computing*, vol. 7, no. 1, pp. 18–31, 2014.

[39] F. T. Liu, K. M. Ting, and Z.-H. Zhou, “Isolation forest,” in *2008 Eighth IEEE International Conference on Data Mining*, pp. 413–422, IEEE, 2008.

[40] F. T. Liu, K. M. Ting, and Z.-H. Zhou, “Isolation-based anomaly detection,” *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 6, no. 1, p. 3, 2012.

[41] H. Lloyd and M. Amos, “Analysis of independent roulette selection in parallel ant colony optimization,” in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 19–26, ACM, 2017.

[42] W. Zhang, C. K. Chang, T. Feng, and H.-y. Jiang, “Qos-based dynamic web service composition with ant colony optimization,” in *2010 IEEE 34th Annual Computer Software and Applications Conference*, pp. 493–502, IEEE, 2010.

[43] Q. Wu and Q. Zhu, “Transactional and qos-aware dynamic service composition based on ant colony optimization,” *Future Generation Computer Systems*, 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,” *The International Journal of Advanced Manufacturing Technology*, vol. 102, no. 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 Transactions on Parallel and Distributed Systems*, vol. 26, no. 2, pp. 455–466, 2015.

[46] H. Zheng, W. Zhao, J. Yang, and A. Bouguettaya, “Qos analysis for web service compositions with complex structures,” *IEEE Transactions on Services Computing*, vol. 6, no. 3, pp. 373–386, 2013.

[47] “Isolation forest implementation.” https://scikit-learn.org/stable/auto_examples/ensemble/plot_isolation_forest.html. Accessed: 2019-07-19.

[48] L. D. Xu, E. L. Xu, and L. Li, “Industry 4.0: state of the art and future trends,” *International Journal of Production Research*, 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 transactions on services computing*, vol. 7, no. 1, pp. 32–39, 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 Transactions on Cloud Computing*, 2019.

[51] I. Mohiuddin and A. Almogren, “Workload aware vm consolidation method in edge/cloud computing for iot applications,” *Journal of Parallel and Distributed Computing*, vol. 123, pp. 204–214, 2019.

[52] B. P. Rimal and M. Maier, “Workflow scheduling in multi-tenant cloud computing environments,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 1, pp. 290–304, 2016.

[53] W. J. Gutjahr, “On the finite-time dynamics of ant colony optimization,” *Methodology and Computing in Applied Probability*, vol. 8, no. 1, pp. 105–133, 2006.

[54] T. Stützle, M. Dorigo, *et al.*, “A short convergence proof for a class of ant colony optimization algorithms,” *IEEE Transactions on evolutionary computation*, vol. 6, no. 4, pp. 358–365, 2002.

[55] M. Dorigo, M. Birattari, and T. Stutzle, “Ant colony optimization,” *IEEE computational intelligence magazine*, vol. 1, no. 4, pp. 28–39, 2006.

[56] E. Al-Masri and Q. H. Mahmoud, “Discovering the best web service: A neural network-based solution,” in *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, pp. 4250–4255, IEEE, 2009.

Mohammadreza Razian recently joined to the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne, Australia. He received his Master’s (2014) degrees in Information Technology from Sharif University of Technology, Tehran, Iran. He is currently working towards the Ph.D. degree at the Faculty of Industrial Engineering, Iran University of Science and Technology, Tehran, Iran. His research interests include the Internet of Things, cloud computing, privacy and security, and data analysis.

Mohammad Fathian is professor of the school of Industrial Engineering of Iran University of Science and Technology, Tehran. He received his MS and Ph.D. degrees in Industrial Engineering from the same university. Dr. Fathian 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 received the B.E. and M.S. degrees from Harbin Institute of Technology, China in 2009 and 2011, respectively, both in electrical engineering. He received the Ph.D. degree with the highest honor in computer science at Freie Universität Berlin, Germany in 2015. He is currently an associate professor in the Center for Applied Mathematics, Tianjin University. His research interests include mobile cloud computing, edge computing, fog computing, internet of things (IoTs), and deep learning.

Ahmad Akbari is an associate professor at school of computer engineering, Iran university of Science and Technology in Tehran, Iran. He has a BSc and MSc in telecommunication engineering from IUT, Esfahan, Iran and a PhD in signal processing and telecommunications from Rennes 1 university Rennes, France. His research interests include computer networks, data communications and signal processing applications. He is the dean of IUST school of computer engineering since 2013.

Rajkumar Buyya is a Redmond Barry Distinguished Professor and Director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne, Australia. He is also serving as the founding CEO of Manjrasoft, a spin-off company of the University, commercializing its innovations in Cloud Computing. He served as a Future Fellow of the Australian Research Council during 2012-2016. He has authored over 625 publications and seven text books including "Mastering Cloud Computing" published by 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, Feb 2011). He is one of the highly cited authors in computer science and software engineering worldwide (h-index=127, g-index=280, 84,900+ citations). Microsoft Academic Search Index ranked Dr. Buyya as #1 author in the world (2005-2016) 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 Dr. Buyya as the World's Top-Cited (#1) Author and the World's Most-Productive (#1) Author in Cloud Computing. Recently, Dr. Buyya is recognized as a "Web of Science Highly Cited Researcher" in 2016, 2017, and 2018 by Thomson Reuters, a Fellow of IEEE, and Scopus Researcher of the Year 2017 with Excellence in Innovative Research Award by Elsevier for his outstanding contributions to Cloud computing. He is currently serving as Co-Editor-in-Chief of Journal of Software: Practice and Experience, which was established over 45 years ago.