The Journal of Systems and Software 164 (2020) 110557

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

ARC: Anomaly-aware Robust Cloud-integrated IoT service composition based on uncertainty in advertised quality of service values

Mohammadreza Razian[a,b], Mohammad Fathian[b,*], Rajkumar Buyya[a]

[a] Cloud Computing and Distributed Systems (CLOUDS) Laboratory, School of Computing and Information Systems, The University of Melbourne, VIC 3010, Australia
[b] School of Industrial Engineering, Iran University of Science and Technology, Tehran, Iran

ARTICLE INFO

Article history:
Received 1 June 2019
Revised 16 January 2020
Accepted 20 February 2020
Available online 21 February 2020

Keywords:
Service composition
Uncertainty
Robust optimization
Anomaly detection
IoT
Cloud Computing

ABSTRACT

From the IoT perspective, each intelligent device can be considered as a potential source of service. Since several services perform the same function, albeit with different quality of service (QoS) parameters, service composition becomes a crucial problem to find an optimal set of services to automate a typical business process. The majority of prior research has investigated the service composition problem with the assumption that advertised QoS values are deterministic and do not change over time. However, factors like sensors failure and network topology changes cause uncertainty in the advertised QoS values. To address this challenge, we propose a novel Anomaly-aware Robust service Composition (ARC) to deal with the problem of uncertainty of QoS values in a dynamic environment of Cloud and IoT. The proposed approach uses *Bertsimas and Sim* mathematical robust optimization method, which is independent of the statistical distribution of QoS values, to compose services. Moreover, our approach exploits a machine learning-based anomaly detection technique to improve the stability of the solution with a fine-grained identification of abnormal QoS records. The results demonstrate that our approach achieves 14.55% of the average improvement in finding optimal solutions compared to the previous works, such as information theory-based and clustering-based methods.

1. Introduction

Internet of things (IoT) and Cloud computing are changing the way industries and enterprises do their businesses with lower capital expenditure (CAPEX) (Xu, 2012; Goudarzi et al., 2017; Buyya et al., 2018). Currently, Kubernetes Kubernetes, Docker Swarm swarm, and Apache Mesos mesos have become modern choices for container and data center orchestration to deliver applications and services at a high velocity to industries (Zhang et al., 2019). Although the Cloud services provide virtually unlimited resources, they are limited in terms of scope (i.e., located in data centers). On the other hand, IoT devices are limited in computational resources, but they are becoming increasingly ubiquitous and pervasive. Therefore, the integration of Cloud and IoT (Botta et al., 2016), namely Cloud-integrated IoT, has provided an unprecedented opportunity for developers to develop more value-added software services.

Every *intelligent device* in IoT can be considered as a potential service provider offering a **micro-service** using network-based APIs (Urbieta et al., 2017). Microservices architecture (MSA) is a variant of the traditional service-oriented architecture which develops an application by composing a suite of independent services (Balalaie et al., 2016) with a fine-grained functionality. The term *fine-grained* means each service performs a specific and predefined task like reporting temperature by a sensor or stock inventory by an RFID tag. Furthermore, there are some quality of service (QoS) parameters describing the performance of a service in terms of availability, security, reliability, cost, and response time. Since several services can perform the same function, albeit with different QoS, service composition becomes a crucial problem to find an optimal set of Cloud and IoT services to automate workflow in a typical business process.

A large number of works have been devoted to addressing the service composition problem *(SCP)* with the assumption that the advertised QoS values are deterministic (i.e., they assume that the advertised QoS values of service providers do not change over the time) (Jula et al., 2015; Zhou and Yao, 2017; Jatoth et al., 2018). However, in reality, the factors like workstation load, multi-tenancy, sensors failure, network topology changes, network con-

* Corresponding author.
E-mail addresses: mrazian@unimelb.edu.au (M. Razian), fathian@iust.ac.ir (M. Fathian), rbuyya@unimelb.edu.au (R. Buyya).

https://doi.org/10.1016/j.jss.2020.110557
0164-1212/© 2020 Elsevier Inc. All rights reserved.

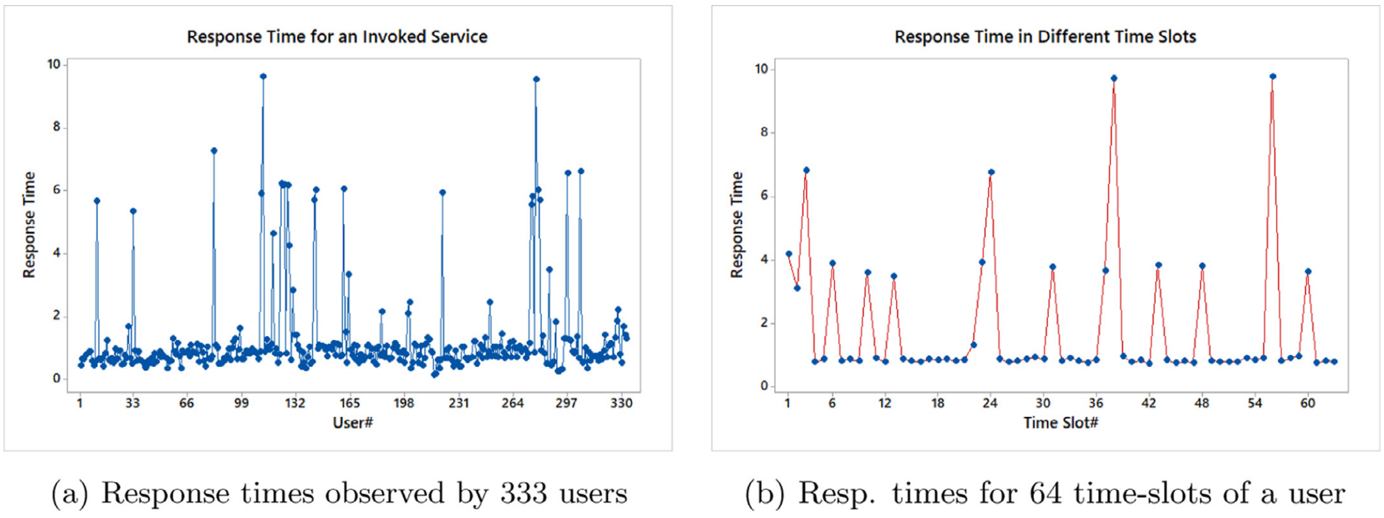


Fig. 1. The severity of fluctuation in response time values (seconds) for a typical service. The historical data have been collected by Zheng et al. (2014, 2010).

gestion, and economic policies cause uncertainty in the QoS values (White et al., 2017b; Bronsted et al., 2010; Raychoudhury et al., 2013). Traditionally, a service broker who is responsible for service composition offers a *composite service* to the users based on the advertised QoS values and user's constraints. However, due to the uncertainty of QoS values, the aggregated QoS values of this composite service may violate the user's constraint. In this situation, the service broker will be charged according to the service level agreement (Schuller et al., 2012). Fig. 1a shows the real response time values of an invoked service observed by 333 users (Zheng et al., 2014; 2010). The figure indicates that different users can receive different response times for a unique service. Moreover, Fig. 1b shows that even for a unique user, the values of response time may fluctuate in different time slots (here, the values have been collected for 64 time-slots).

Recently, some studies have concentrated on the problem of service composition for uncertain QoS values. However, there are three major limitations associated with the current service composition approaches:

1. They assume that there exist sufficient and reliable historical records of QoS values for all services; however, in the dynamic environment of IoT, service nodes come from different providers and join or leave the network frequently. Thus, the broker has not sufficient and reliable historical QoS values about a new service which is recently joined the network. As a result, cold start and data sparsity are two crucial problems that degrade the performance of these approaches.
2. They suppose that the QoS values follow a constant or well-known statistical distribution in long-term; practically, QoS values may not rely on a constant probability distribution function precisely (Zheng et al., 2016);
3. They do not consider the dynamicity of a Cloud-integrated IoT environment where intermittent network connection and sporadic access are common causes of *anomalies* in monitored QoS values (Moghaddam et al., 2018).

Clearly, these approaches will fail in the dynamic environment of Cloud-integrated IoT. In this paper, we propose an **A**nomaly-aware **R**obust **C**loud-integrated IoT service composition **(ARC)** considering uncertainty in advertised QoS values. ARC provides a high-level composition algorithm to eliminate much of these limitations and makes the composition process less error-prone. The ARC algorithm is based on an abstract service model representing candidate services, user's tasks in terms of workflow, user's preferences in terms of constraints, QoS values under uncertainty, a robust model for providing a composite service, and an adaption mechanism using anomaly detection algorithm for managing environment dynamicity. The key contributions of this paper are summarized as follows:

- First, we analyze and demonstrate the uncertainty of QoS values by analysis of a real dataset.
- Second, we propose a mathematical robust optimization model to deal with the uncertainty of QoS values under users' constraints to minimize the cost.
- Third, an innovative fine-grained approach is proposed to identify the amount of uncertainty around services using an unsupervised Isolation-Trees based approach.
- Fourth, a flexible parameter namely *protection degree* is introduced which allows the decision-makers to control the trade-off between robustness and optimality
- Fifth, we conduct a comprehensive set of experiments on the real dataset and compare our approach with existing information theory-based and clustering-based methods.

The rest of the paper is structured as follows: Section 2 presents a motivation example and reviews the related work along with a conclusion on the pros and cons of previous studies. In Section 3, we define the service composition problem using notations and mathematical optimization modeling. The proposed ARC algorithm is presented in Section 4. In Section 5, the efficiency and effectiveness of ARC are evaluated in a comparison of known approaches in the literature. Finally, in Section 6, we conclude the paper and propose future work.

2. Background and related work

2.1. Motivation example

In this section, a motivation example is used to clarify the problem of Cloud-integrated IoT QoS-aware service composition under uncertainty. The basic background of this example is derived from the health-care domain (Raychoudhury et al., 2013; Avila et al., 2017). IoT plays an important role in the future of health-care and presents new opportunities to detect, prevent, and treat disease. Let us assume *Company A* that develops health-care software applications based on *MSA* needs a collection of services (i.e., a composite service). *Company A* prepares a document including a workflow (i.e., required tasks and their execution sequence) and QoS

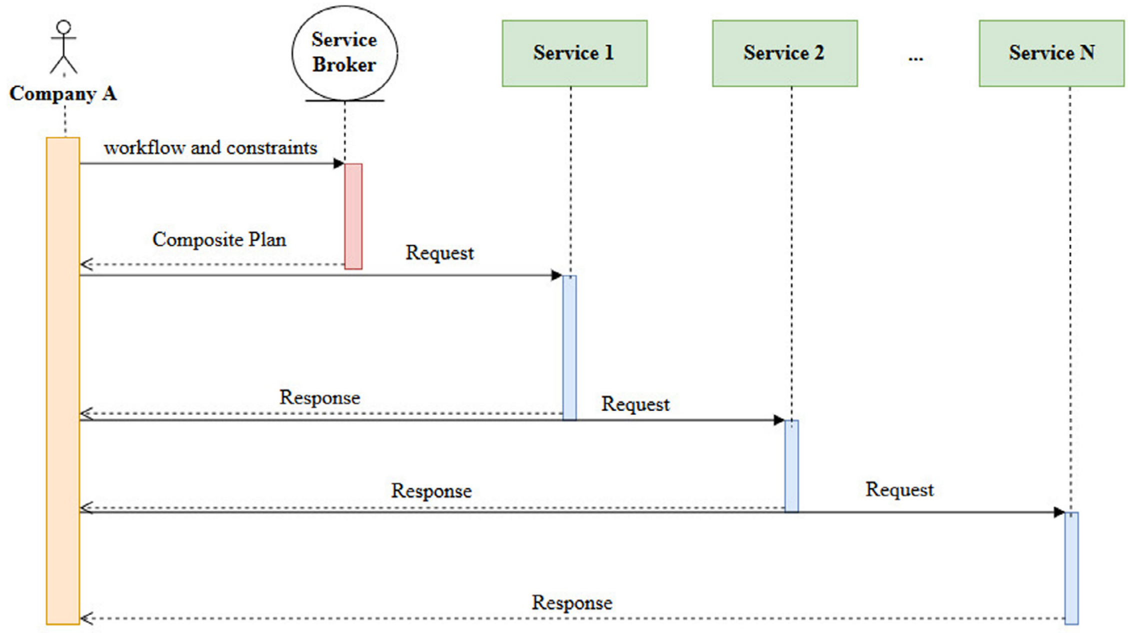


Fig. 2. Sequence diagram of service composition problem.

constraints. Consider *Company A* decides to provide the following set of services to the users:

- **Sensing** A group of metrics, including heart rate, blood pressure, and glucose rate is acquired by using IoT services with various QoS attributes, which are provided by multiple vendors.
- **Navigation** *Company A* needs a navigation service to provide the best route to a public clinic or hospital to users in the emergency conditions by using intelligent IoT sensors for finding a location and estimating the real-time traffic information (Lin et al., 2019).
- **Data Warehouse** It is crucial for *Company A* to store the users' electronic health records in a reliable, highly-available, and secure (Guan et al., 2019) data warehouse. A lot of Cloud-based storage systems such as Oracle Cloud platform offer storage services with different QoS attributes.
- **Translation** In order to localize the *user interface* of the application, *Company A* requires a translator service to change the application language according to users' preference.
- **Analytics** *Company A* also applies an analytics service to find users' behavior and activities to propose a custom health program. For instance, Amazon AWS AWS provides a wide variety of services like *AWS IoT Analytics* that collects, pre-processes, enriches, stores, and analyzes IoT device data.
- **Representation** The graphical representation of health records is another requirement of *Company A*. Data visualization and charting services help users to find out their health status, especially when the volume of monitored data grows increasingly.

A wide variety of service providers advertise their services and corresponding QoS values to the service broker. The broker decides which services are appropriate for performing the tasks in the workflow according to *Company A* document and advertised QoS values of service providers. Fig. 2 shows the whole process of service composition using *UML* sequence diagram.

Table 1 shows the candidate services of *Company A*'s tasks and QoS values for the response time and cost parameters. To adhere to the motivation scenario, we assigned response time values to the services mentioned above. These values come from reports (Zheng et al., 2014; 2010). These reports have measured the observed response time of online services like *online dictionary for slang words and phrases* and *Navigator Online* from geographically-distributed users. The reports collected response time values for a given service observed by all users. For the aforementioned motivation scenario, we calculated the average values of the reported times for each service. Suppose that there are two candidate services for performing the task t_1 along with five candidate services for task t_2, three candidate services for task t_3, four candidate services for task t_4, two candidate services for task t_5, and five candidate services for task t_6.

Table 1
The corresponding candidate services of *Company A*' workflow

Task	Cand. service	Resp. time	Cost
t_1	s_{11}	2.12	988
	s_{21}	0.33	1167
t_2	s_{12}	0.37	1163
	s_{22}	0.33	1167
	s_{32}	0.32	1168
	s_{42}	0.91	1109
	s_{52}	1.22	1078
t_3	s_{13}	1.03	1097
	s_{23}	0.55	1145
	s_{33}	1.31	1069
t_4	s_{14}	1.04	1096
	s_{24}	1.21	1079
	s_{34}	3.43	857
	s_{44}	1.77	1023
t_5	s_{15}	1.30	1070
	s_{25}	1.70	1030
t_6	s_{16}	1.67	1033
	s_{26}	1.11	1089
	s_{36}	0.29	1171
	s_{46}	0.31	1169
	s_{56}	1.37	1063

If *Company A* does not consider any restrictions on the aggregated value of response time, the broker suggests the minimum cost plan that is 6055 for the composite service $\langle s_{11}, s_{52}, s_{33}, s_{34}, s_{25}, s_{16}\rangle$. Although, *Company A* considers some constraints on the aggregated QoS values, such as the response time parameter. Hence, the service broker finds the optimal plan according to these constraints. Fig. 3 represents the effect of response time's

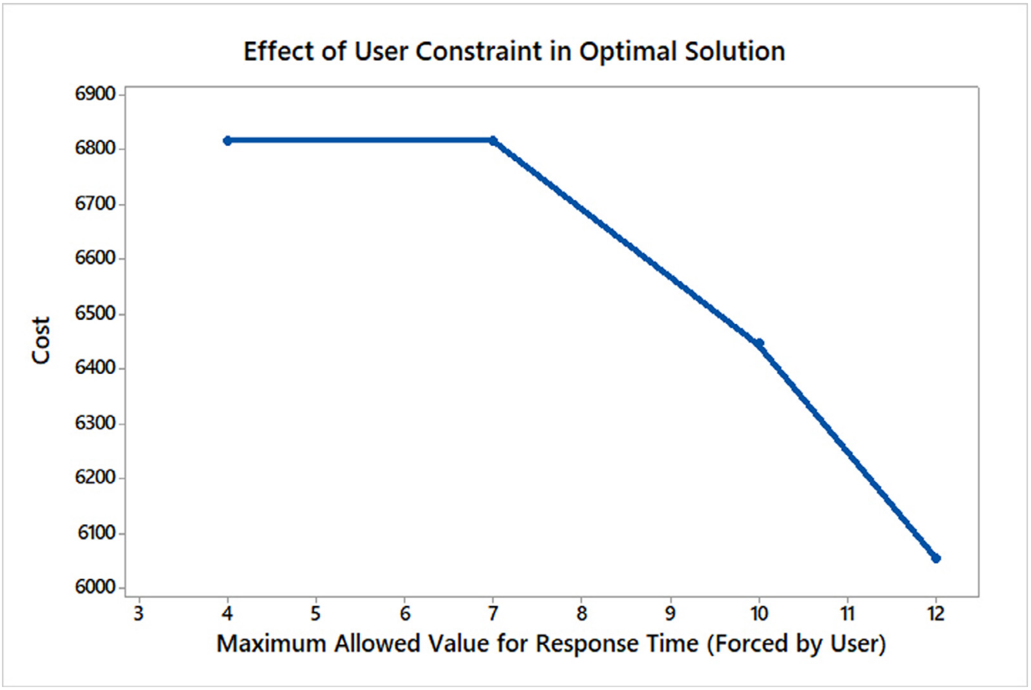


Fig. 3. Impact of user's constraint on the composite service cost.

constraint on the cost of the composite service. These results show that when *Company A* increases the constraint on the aggregated response time value (i.e., a service set with lower execution and communication time), the cost of composite service also increases.

If the advertised QoS values fluctuate, the broker will be charged with the extra penalty fees because of SLA violation. For example, consider *Company A* waits 4 seconds to receive the result of a composite service. If the aggregated QoS violates from this constraint, the penalty will be applied for the broker. It is notable that, in the above example, when we set the maximum acceptable value for aggregated response time to 4 seconds, the selected services become $\langle s_{21}, s_{32}, s_{23}, s_{14}, s_{15}, s_{36}\rangle$, which are different from the previous ones. Through an extensive literature review, we found that a lot of previous studies assume that QoS values rely on normal distribution (Wang et al., 2010; Schuller et al., 2012; Ramacher and Mönch, 2012; Ye et al., 2014; Wang et al., 2016; Mostafa and Zhang, 2015). However, measurement of real services like YouTube (Zheng et al., 2016), shows that for example, the response time cannot be fitted to well-known statistical distributions. More information about the QoS values statistics and distributions can be found in Zheng et al. (2016). Because of sensor failure, hardware power loss, intermittent network connections, and sporadic access, the QoS fluctuations are intensified when *IoT nodes* supply services. Therefore, to decrease the penalty for violating the users' constraints, the broker needs to select services robustly and adapts the selection procedure concerning changes in the environment.

2.2. Related work

Service composition problem (SCP) is a well-researched area as many works have been devoted to solving it (Asghari et al., 2018). In order to model the SCP, formal methods such as Petri net (Tan et al., 2009) and Process algebra (Tu et al., 2010) have been employed. However, these methods are difficult to implement in real-world scenarios (Oh et al., 2006). Jaeger et al. (2004) discuss different QoS parameters and basic structures for workflow construction. Ardagna and Pernici (2007) formalize the service composition problem as a mixed-integer programming (MIP) problem for the various workflow patterns such as sequence and loop as well as their corresponding QoS aggregation function. Also, agent-based architectures were proposed to facilitate the broker-customer negotiations and agreement (Chhetri et al., 2006; Gutierrez-Garcia and Sim, 2010).

Many heuristic (Liu et al., 2009; Li et al., 2010; Luo et al., 2011) and meta-heuristic (Yang et al., 2010; Jatoth et al., 2018) methods have been applied to SCP to find the (near-)optimal set of services (Jatoth et al., 2017). Yu et al. (2007) developed an algorithm for services composition with the QoS constraints for the whole workflow. They mapped the SCP to the multi-dimension multi-choice 0-1 knapsack problem. A* (Rodriguez-Mier et al., 2011) and Hill climbing (Klein et al., 2011) are some famous algorithms that were used to decrease the processing time of SCP solving. Nevertheless, these approaches do not guarantee to find the global optimum solution and may get trapped in a local optimum. In order to overcome this problem, many meta-heuristic algorithms such as genetic algorithm (GA) (Amiri and Serajzadeh, 2010; Yilmaz and Karagoz, 2014), multi-objective genetic algorithm (Sharifara et al., 2014; Wada et al., 2012), particle swarm optimization (PSO) (Tao et al., 2008; Wang et al., 2013), ant colony optimization (ACO) (Zhang et al., 2010; Yu et al., 2015), artificial bee colony (ABC) (Zhou and Yao, 2017; Lartigau et al., 2015), fruit fly optimization (Zhang et al., 2015; Seghir and Khababa, 2016), and Cuckoo search (Chifu et al., 2011) have been considered to solve the SCP. However, parameter tuning for intensification and diversification is the main concern in meta-heuristic approaches. Although meta-heuristic algorithms can find solutions faster by decreasing the search space, typically, they find a near-optimal solution rather than an exact optimum solution.

All the aforementioned approaches assume that the QoS values are deterministic. As discussed in Section 2.1, in reality, the QoS values are not deterministic, and there are some perturbations in the values due to inherent dynamicity in Cloud-integrated IoT environments. In order to encounter the QoS perturbation, Zeng et al. (2004) presented a local (for each task) and global (total workflow) optimization algorithm. They observed that the execution duration of a service is not deterministic. Furthermore, they considered a random variable as a Normal distribution to model the execution duration. Hwang et al. (2004, 2007) proposed a probabilistic model using a discrete random variable with a probability mass function (PMF). In order to decrease the sample size of the random variable, they chose a value for the same domains. They used both dynamic programming and greedy method to find the best fitting domain value. Albeit, this grouping (i.e., the selection of a single value on behalf of all values in the same domain) leads to aggregation error. Wang et al. (2007) discuss on a fuzzy rule-based system to select a service from a service file repository. They propose a service selection method using objective information (i.e., QoS parameters) and subjective information (i.e., users' observation and satisfaction) to match user's preferences to existing services. Moreover, they use a genetic algorithm for adjusting fuzzy rules to eliminate the errors in users' satisfaction scores. These scores have been acquired using the questionnaire in the Likert five-point rating.

Rosario et al. (2008) proposed a soft contract concept based on probability distribution to decrease the impact of uncertain QoS values on the SLA violation. In order to generate different contracts, they deployed TOrQuE (Tool for Orchestration simulation and Quality of service Evaluation) based on Monte Carlo simulation. The aim of authors in Wiesemann et al. (2008), is to minimize the risk that originated from stochastic programming. The average value-at-risk (AVaR) measure is applied to calculate the worst-case risk function for the time and cost parameters. The worst-case decision results are more conservative choices than the expected value approach. Yu and Bouguettaya (2010) concentrate on Cloud provider selection and explore how users can flexibility select the provider. They propose a p-dominant service skyline to find the most preferred service provider. In order to deal with the uncertainty of QoS values, a *p-R-tree*, an indexing data structure, is used, and a dual-pruning process is employed to prune the uncertain providers that are dominated by other providers. Violation of the user's constraints will result in an additional penalty cost for the service broker who presents the composite service. To avoid this penalty, Schuller et al. (2012) adopted a greedy algorithm to minimize this penalty with the assumption that the probabilistic features of QoS parameters are known (for example, they consider response time values as a Normal distribution). Also, they replace the greedy algorithm with a genetic adaption algorithm in Schuller et al. (2014). In order to provide a better search space, authors in Hwang et al. (2015) identify the local optimum services by decomposing global constraints (i.e., workflow level constraints) to some local constraints (i.e., task level). Then, they gradually improve the initial services assignment (in a time-intensive manner) to obtain a better composition. It is worth mentioning that, decomposing global constraints to local constraints using historical QoS values causes the improper outcome. In Mostafa and Zhang (2015), the authors have designed a multi-objective model for SCP using a multi-objective partially observable Markov Decision. Reinforcement learning is adopted to improve the solution periodically. They evaluate their proposed method using the synthetically generated dataset (based on Normal statistical distribution) for QoS values. Decreasing the total number of service invocations has investigated in Chattopadhyay and Banerjee (2016). They proposed an A^* algorithm to find the minimum service invocation required for performing a workflow. Albeit, in many scenarios, the user needs to execute all of the tasks in the workflow and does not permit to reduce the tasks included in a corresponding workflow. They use the *Tchebysheff's inequity* for the QoS values, which follow an unknown distribution to estimate the largest possible population variance (worst-case scenario).

In Zheng et al. (2016), Zheng et al. applied a multivariate time-series to predict the future for long-term service composition by using the historical data; although, the cold-start problem threats their procedure. Chen et al. (2016b) propose a robust technique to defend against the uncertainty by considering a fixed interval value. In essence, determining a fixed interval value in a highly dynamic circumstance like Cloud-integrated IoT leads to inaccurate outcomes with a high penalty for the service broker. Based on the Cloud model (Wang et al., 2011), Wang et al. (2017) measure the uncertainty of candidate services according to Entropy and Variance values of monitored services. They remove uncertain service in the pre-composition phase and find the composite service using mixed inter programming. Recently, Khanouche et al. (2019) propose a clustering-based services composition algorithm for the IoT environment. They categorize candidate services according to the QoS level to three clusters, i.e., High-QoS, Middle-QoS, and Low-QoS.

Additionally, in real-time literature, the concept of probabilistic Worst-Case Execution Time *pWCET* has been introduced. There are two approaches in pWCET to estimate the worst-case value: static analysis and measurement. The former tries to find out processor behaviors like caching and using this, calculate the worst-case path of a program through syntax tree representation of the program. However, this approach may fail when analyzing data-intensive programs and/or employing today's advanced CPU features (with acceleration features like cache, pipelines, branch prediction buffers and out of order execution) (Bernat et al., 2003). The latter, i.e., the measurement approach, tries to observe the real system. However, they may fail to capture the worst-case using pre-defined test cases. Besides, there exist some hybrid approaches which adopt both static analysis and measurement (Bernat et al., 2003) to obtain the probability distributions of the individual execution time of the program's blocks. However, running under many test scenarios to find the probability is a time-consuming process that does not fit in a dynamic IoT environment where the on-the-fly services join/leave network. Furthermore, pWCET has been introduced to ascertain whether software programs execute within the time bounds assigned to them in terms of execution time and response time (Cazorla et al., 2019). However, as mentioned in White et al. (2017a), according to the quality model ISO/IEC 25010 (Standardization, 2016) in service composition, there is broad coverage of QoS attributes like availability, reliability, and usability. Therefore, it is needed an approach to be able to support time-based and other QoS attributes. Also, WCET of the program engages with the execution time of a program in an embedded system and does not consider the communication network aspects of a Cloud-integrated IoT environment where intermittent network connection and sporadic access are common causes anomaly in captured QoS values. Therefore, WCET approaches would not be proficient due to a lack of ability for calculation of infrastructure delay and communication between IoT node and Cloud data centers.

However, there are several limitations in these approaches including 1) They assume that there exists a full list of useful historical records related to QoS values for all services are available; however, in the dynamic environment of IoT, service nodes join or leave the network. Furthermore, cold start and data sparsity are two crucial problems that degrade the performance, notably when the broker has not sufficient and reliable historical QoS values about a new service which is recently joined to network; 2) They assume that the historical data are fitted to a well-known or constant statistical distribution in the long-term. Practically, QoS values may not rely on a constant probability distribution function precisely (Zheng et al., 2016); 3) They do not consider the dynamicity of a Cloud-integrated IoT environment where intermittent network connection and sporadic access are common causes of *anomaly* in monitored QoS values (Moghaddam et al., 2018).

Clearly, these approaches will fail in a dynamic environment of Cloud-integrated IoT.

Table 2 summarizes the *Uncertainty-aware* approaches using a following qualitative criteria: some studies generate QoS values randomly or use a specific statistical distribution. It is worth to note that *Applying real dataset* is necessary to deal with uncertainty. *Protection degree* which allows the decision-makers to control the trade-off between robustness and optimality. In order to support a Cloud-integrated IoT scenario, it is important to focus on *environment adaption* to reach an effective and efficient composite service. The factors like sensor failures or corruption in network infrastructure among smart devices result in the abnormal data in the historical monitored QoS. Dealing with anomalies (*Anomaly detection*) becomes an important aspect of service composition problem under uncertainty, particularly in the Cloud-integrated IoT environments. To the best of our knowledge, this work is a pioneer in proposing service composition *Architecture for Cloud-integrated IoT*. This architecture helps developers to deploy their software using a composition of isolated, independent and fine-grained IoT services in an uncertain environment.

3. Problem definition

In this section, we formalize the QoS-aware service composition problem. Table 3 summaries the notations used in this paper with a brief description. Suppose T is a workflow including a set of tasks t_i which in $T = \{t_1, t_2, ..., t_N\}$, N is the number of tasks included in T, and $1 \leq i \leq N$. Consider the set $M = \{m_1, m_2, ..., m_N\}$ so that $m_i \in M$ represents the number of candidate services for performing the task t_i and $m_i \geq 1$. Let S_i be a set of candidate services for performing t_i where $S_i = \{s_i^1, s_i^2, ..., s_i^{m_i}\}, 1 \leq i \leq N$. Therefore, $s_i^j \in S_i$ is jth candidate service which is potentially able to be invoked for performing the ith task. The set $Q = \{q_1, q_2, ..., q_L\}$ defines QoS parameters so that L is the number of QoS parameters and $q_i \in Q$ presents a QoS parameter (e.g. $q_1 = RTime$). The request of user for a composite service usually is accompanied with a set of constraints on QoS parameters. The set $B = \{b_1, b_2, ..., b_L\}$ defines constraints b_i, $1 \leq i \leq L$ for QoS attributes q_i. Table 4 describes the QoS parameters used in this paper and their definitions. We emphasize that our work is general and does not depend on a specific QoS parameter. Using these preferences, service broker finds a composite service with the minimum *Cost* (the price of using a service which is typically stated in terms of per *hour/user/byte*).

A workflow is constructed by a set of tasks with the assumption that each task has its candidate services. The different structures that can be considered for a workflow are sequence (successive tasks), loop, condition, and parallel (concurrent tasks). For the sake of simplicity, we consider the sequential structure; other composition structures such as loop, parallel, and condition can be converted to the sequential structure through the methods mentioned in Dou et al. (2015); Zheng et al. (2013). In Eq. (1), the QoS-aware service composition problem is modeled in the form of objective function Z:

$$Z = MIN \sum_{1 \leq i \leq N} \sum_{j \in M} x_{ij} * Cost(s_i^j) \tag{1}$$

Subjected to

$$\sum_{1 \leq i \leq N} \sum_{j \in M} RTime(s_i^j) * x_{ij} \leq b_{RTime} \tag{2}$$

$$\prod_{1 \leq i \leq N} \sum_{j \in M} Avail(s_i^j) * x_{ij} \geq b_{Avail} \tag{3}$$

$$\frac{1}{N} * \sum_{1 \leq i \leq N} \sum_{j \in M} Reput(s_i^j) * x_{ij} \geq b_{Reupt} \tag{4}$$

Table 2
Related work and comparison to our proposed ARC.

Parameters	Related Work														ARC
	Zeng et al. (2004)	Hwang et al. (2007)	Wang et al. (2007)	Rosario et al. (2008)	Wiesemann et al. (2008)	Yu and Bouguettaya (2010)	Wang et al. (2017)	Schuller et al. (2014)	Hwang et al. (2015)	Mostafa and Zhang (2015)	Chattopadhyay and Banerjee (2016)	Zheng et al. (2016)	Chen et al. (2016b)	Khanouche et al. (2019)	
Uncertainty-aware	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Real dataset	✗	✗	✓	✗	✗	✗	✓	✗	✓	✗	✗	✓	✓	✓	✓
Protection degree	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
Anomaly detection	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
Adaption	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
CloudIoT Arc.	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓

Table 3
Summary of notations

Notation	Description
t_i	ith task in a given workflow
T	Set of tasks t_i included in a given workflow
N	Total number of tasks in a workflow
m_i	Number of corresponding candidate services for each t_i
M	Set of the number of candidate services (m_i)
S_i	Set of candidate services for ith task
s_i^j	The jth candidate service for ith task
Q	Set of QoS parameters
L	Number of QoS parameters
B	Set of user's constraints for QoS attributes
b_{RTime}	Maximum allowed response time value for the composite service
b_{Avail}	Minimum allowed availability value for the composite service
b_{Reput}	Minimum allowed reputation value for the composite service
$Cost(s_i^j)$	Retrieving the price of s_i^j
$RTime(s_i^j)$	Retrieving the response time of s_i^j
$Avail(s_i^j)$	Retrieving the availability probability of s_i^j
$Reput(s_i^j)$	Retrieving the reputation score of s_i^j
x_{ij}	A binary variable indicting selection of a candidate service s_i^j
Γ	Protection degree
χ	A particular data point in Isolation Forest tree
$h(\chi)$	The number of edges in a tree for a particular data point χ
$E(h(\chi))$	The average of $h(\chi)$ from a set of isolation trees
$nc(n)$	Normalization constant for a dataset of size n
a_{ij}	The advertised QoS value (or nominal value) for s_i^j
$\hat{a_{ij}}$	The amount of perturbation for s_i^j
J_i	Set of coefficients which are subject to uncertain parameters
Z	Objective function (for minimizing cost)
c	The coefficient of the objective function (i.e. service's cost)
ζ	A dual variable used in *Bertsimas and Sim* formulation
p_{ij}	A dual variables used in *Bertsimas and Sim* formulation
y_j	Used to represent absolute variable $\lvert x^{*}{}_j \rvert$ as $-y_j \le x_j \le y_j$
Δ	The time period between adaption phases
Υ	Contamination rate (the proportion of outliers in data)
μ_i^j	Mean value of the historical records of s_i^j after removing the anomalies
σ_i^j	Standard deviation of the historical records of s_i^j after removing the anomalies

Table 4
QoS parameters name and definition

Name	Definition
Response time	The time between sending a request and receiving the reply is considered as a response time. More precisely, the summation of the processing time and the transmission duration is considered as a response time which is measured in seconds.
Availability	The total number of times that a service has been accessible to a total number of invocation (a probability value between [0, 1]).
Reputation	Reputation or fidelity is a measure of trustworthiness about a service from the users' perspective (an average score between for example 0 to 10).

$$\sum_{1\le i\le N} x_{ij} = 1, \forall j \tag{5}$$

$$x_{ij} \in \{0,1\}, \quad \forall i, j \quad 1 \le j \le m_i \quad \forall i, \quad 1 \le i \le N \tag{6}$$

An optimization model includes three main components: 1) The objective function in Equation 1 minimizes the cost of composite service; 2) A set of constraints in Eqs. (2) to (4) control the value of the objective function; and 3) Binary variable x_{ij} which determines whether a service s_j^i is selected or not. The Eq. (5) enforces the model to select exactly one candidate service per task t_i. In next section, we improve this model to obtain an adaptive robust QoS-aware service composition.

4. ARC: Anomaly-aware Robust Service Composition

In the heterogeneous and distributed Cloud-integrated IoT environment, the QoS values of services change over time. These changes impact on constraints such as Eq. (2). In this section, we propose a service composition framework, as shown in Fig. 4, to address the problem of service composition under QoS uncertainty. There are four distinct components in our framework:

- *Abstract Composition Request* formulates a business workflow as a collection of abstract services. An abstract service can be executed by invoking a candidate service to perform task t_i in a workflow. Furthermore, for each abstract service, the user specifies the QoS constraints. As a result, this component prepares a document including the required services and the corresponding QoS constraints.
- *Candidate services with Advertised QoS* is responsible for finding the concrete services developed by third-party software companies for each abstract service. Indeed, the third-parties advertise several Cloud and IoT services with same functionality and different QoS values. The output of this component is a list of candidate services which is prepared for each abstract service.
- *Robust Service Composition* is responsible for modeling SCP based on *Bertsimas and Sim* robust optimization model. This central component itself uses *amount of perturbation* and *protection degree* sub-components for setting $\hat{a}_{ij}$ and Γ parameters, respectively. The details are provided in 4.1.
- *Computing QoS Uncertainty* finds the amount of perturbation of QoS values. This component utilizes the collected QoS information from monitoring systems and exploits Isolation Forest, a

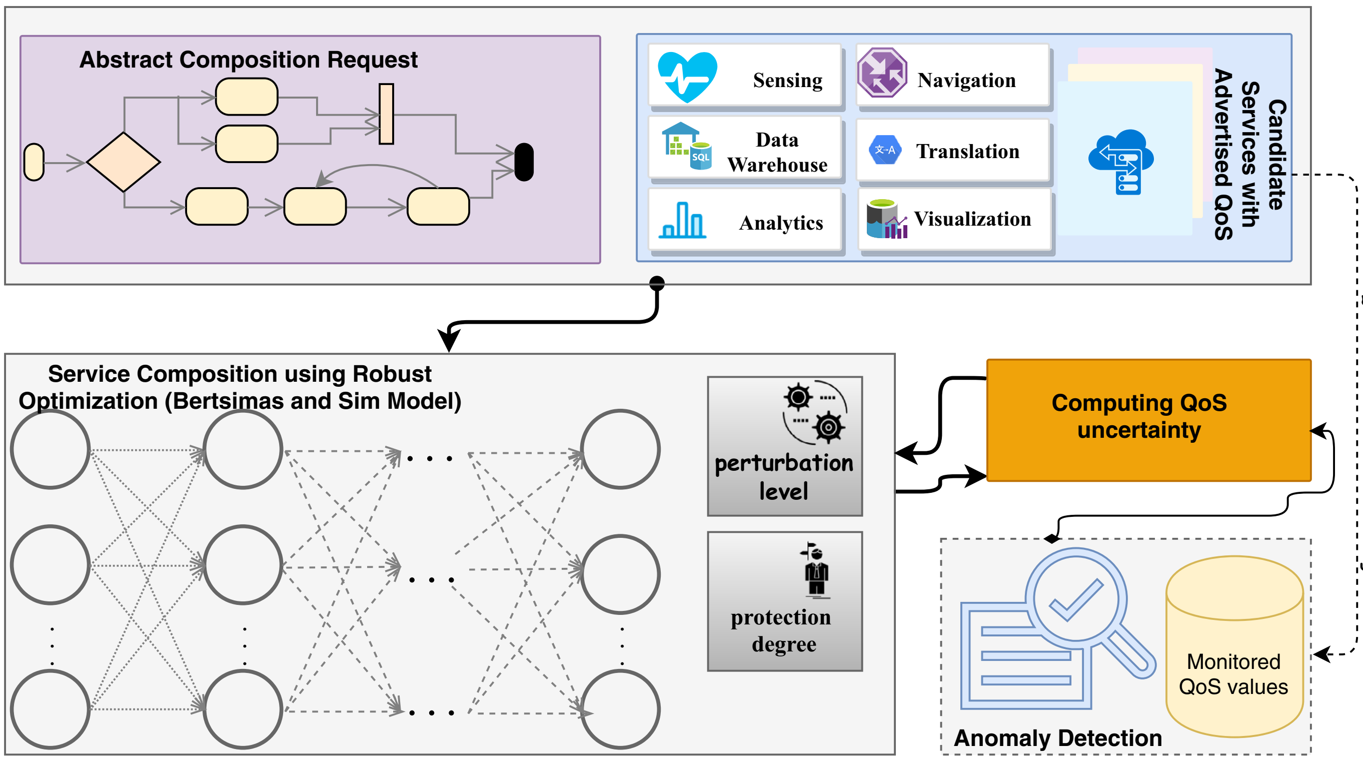

Abstract Composition Request
Sensing
Navigation
Data Warehouse
Translation
Analytics
Visualization
Candidate Services with Advertised QoS
QoS Service Monitoring
Service Composition using Robust Optimization (Bertsimas and Sim Model)
perturbation level
protection degree
Computing QoS uncertainty
Monitored QoS values
Anomaly Detection

Fig. 4. The Architecture of proposed ARC.

machine learning anomaly detection technique. The details are provided in 4.2.

It is notable that our proposed framework is general and can be applied to different types of applications.

4.1. *Dealing with uncertainty*

In order to overcome the problem of uncertainty in QoS values, we exploit the *Bertsimas and Sim* (Bertsimas and Sim, 2004) robust optimization approach for the service composition problem. *Robust optimization* is a mathematical modeling technique to handle the optimization problems when the data are uncertain (Ben-Tal and Nemirovski, 2002). We chose this approach because in contrary to the probabilistic approach, in the absence of reliable and complete historical data, it is still feasible to find a robust composite service. *Bertsimas and Sim*-based approach relies on a single $\hat{a}$ parameter which can be determined by an expert even with existing incomplete and unreliable historical data. Another advantage of robust optimization is its independency to the specific assumptions (like a unique probability distribution) on the QoS uncertainty (Poss, 2014). Furthermore, this approach attempts to make a trade-off between optimality and robustness (Agra et al., 2013) through a flexible adjustment of the level of conservatism of the robust solutions through a parameter named *Protection* degree. Eq. (7), which comes from Bertsimas and Sim (2004), formulates the *Bertsimas and Sim*-based robust optimization method.

$$\begin{aligned}
&\textit{minimize} \quad c^T x \\
&\textit{subject to} \quad \sum_j a_{ij}x_j + \zeta_i\Gamma_i + \sum_{j\in J_i} p_{ij} \le B_i \quad \forall i \\
&\qquad \zeta_i + p_{ij} \ge \hat{a}_{ij}y_j \quad \forall i,\ j\in J_i \qquad (7)\\
&\qquad -y_j \le x_j \le y_j \\
&\qquad \zeta_i,\ p_{ij},\ y_j \ge 0
\end{aligned}$$

In this equation, c is the coefficient of the objective function (i.e., service's cost), B is the user's constraints vector for parameters (i.e., constraint on aggregated QoS values) and x is a binary variable to determine which service is selected. Let x^* be the optimal solution of Equation 7. At optimality, clearly, $y_j = |x*_j|$; therefore, in the equation, the term $|x^*_j|$ is formulated as $-y_j \le x_j \le y_j$. Also, the terms ζ and p_{ij} are the dual variables used in *Bertsimas and Sim* formulation which are not dependent on an application scenario. More information about the *Bertsimas and Sim* robust optimization approach can be found in Bertsimas and Sim (2004). The J_i is the set of coefficients a_{ij}, $j \in J_i$ (i.e., QoS values) which are subjected to uncertain parameters. This means that only the parameters included in this set are allowed to change and take their worst-case value. These uncertain parameters are allowed to take values in interval $\left[a_{ij} - \hat{a}_{ij}, a_{ij} + \hat{a}_{ij}\right]$ where a_{ij} is the advertised QoS value (or nominal value) and $\hat{a}_{ij}$ is the amount of perturbation. As an example, consider the set $J_i = \{1, 4, 5, 7\}$, $|J_i| = 4$ which means that the first, fourth, fifth, and the seventh parameters of ith row of coefficient matrix are uncertain parameters. The main idea of this approach is the ability to control the protection degree. The protection degree allows decision-makers to select throughout a range between hazardous or a conservative decision (i.e., worst-case scenario). This feature is derived by using an uncertainty parameter named Γ. The parameter Γ_i is not necessarily integer, and it takes value in the interval $[0, |J_i|]$. This parameter helps decision-makers to determine how many uncertain parameters in the J_i must be treated as their worst-case value. For example if $\Gamma = 2.6$, it means that two parameters in the set J_i are allowed to take their worst-case value and another parameter (let us call it a_{it_i}) changes by $[(\Gamma_i - \lfloor\Gamma_i\rfloor) * \hat{a}_{it_i}]$, i.e., $0.6\hat{a}_{it_i}$. The value of $\hat{a}_{ij}$ can be obtained empirically from existing historical QoS records (which are not necessarily sufficient and reliable). Obviously, if $\Gamma = 0$, the model is converted to a deterministic service composition problem (non of parameters take their worst-case value). Moreover, the $\Gamma = |J_i|$ means that all QoS values have to take their worst-case value.

Based on Eq. (7), the **robust QoS-aware service composition model** for N tasks, $m_i \in M$ candidate services and uncertainty around the response time values is defined in Eq. (8). By solving the Eq. (8) using mixed-integer programming (MIP) techniques, a (near-)optimal composite service subjected to the constraints is obtained. It is notable that in this equation, the parameter Γ_i leverages the protection degree (i.e., the degree of risk around the composite service). By using this parameter, the decision-maker can control the trade-off between robustness and optimality. In Section 5.5, we show how changes in this parameter impacts on the optimality.

$$
\begin{aligned}
&Z = min \sum_{0\le i\le N} \sum_{j\in M} x_{ij} * Cost(s_i^j)\\
&Subjected\ to\\
&\sum_{0\le i\le N} \sum_{j\in M} RTime(s_i^j) * x_{ij} + \zeta * \Gamma + \sum_{i\in I} \sum_{j\in J_i} p_{ij} \le b_{RTime}\\
&\zeta + p_{ij} \ge \hat{a}_{ij} * x_{ij} \ \ \forall i, j \in J_i\\
&\prod_{0\le i\le N} \sum_{j\in M} Avail(s_i^j) * x_{ij} \ge b_{Avail}\\
&\frac{1}{n} * \sum_{0\le i\le N} \sum_{j\in M} Reput(s_i^j) * x_{ij} \ge b_{Reupt}\\
&\sum_{0\le i\le N} x_{ij} = 1 \ \ \forall j \in M\\
&x_{ij} \in \{0, 1\}\\
&0 \le i \le N, \qquad 0 \le j \le m_i\\
&p_{ij} \ge 0, \qquad \zeta \ge 0
\end{aligned}
\tag{8}
$$

The proposed solution tries to provide an "acceptable" performance under most realizations of the uncertain parameters with no distribution assumption on uncertain parameters. To this aim, the bounds of the perturbation range of uncertain parameters are defined by using existing data. Robust optimization only requires the maximum and minimum of existing values of uncertain parameters to model uncertainty, which usually is accessible. It is worth mentioning that while probabilistic approaches require reliable and sufficient historical data to fit distribution for QoS modeling (Pishvaee et al., 2011; Bertsimas and Sim, 2004; Bertsimas and Thiele, 2006), our robust optimization approach only needs a bound (maximum and minimum value) for a given QoS attribute which can be easily obtained even from insufficient historical data. Furthermore, contrary to simple worst-case QoS estimation and modeling which results in solutions that are too conservative, the robust optimization model addresses the issue of over-conservatism using the following considerations:

- In our model, we have a set named J_i: It is the set of coefficients a_{ij}, $j \in J_i$, which are subject to uncertain parameters (uncertain QoS values). This means among all parameters, only the parameters included in this set are allowed to change and take the worst-case value (Bertsimas et al., 2011).
- And, there is a parameter named Γ_i. Using this parameter, the robust optimization approach provides decision-makers with flexibility in determining the level of conservativeness (Bertsimas et al., 2011). This feature helps decision-makers to avoid over-protection. In other words, this parameter allows the decision-makers to control a trade-off between robustness and optimality.

A simple robust optimization works with a constant perturbation rate, which is determined in advance and remains constant without any adaption in response to changes in the operational environment. However, without incorporating the environment changes in finding a proficient perturbation rate, the model may result in a high cost of robustness (over-conservatism). Therefore, in this manuscript, as the system continues, we estimate the perturbation rate from abnormal-removed monitored data. In the following, we explain how our model finds the amount of perturbation.

4.2. Finding amount of perturbation

One crucial question is how the system determines the value of perturbation in Eq. (8). The proposed model in Eq. (8) considers the worst-case QoS values according to the amount of reported perturbation. Considering the motivation scenario discussed in Section 2.1, suppose that $\langle 1.5, 1.6, 0.9, 2, 2.2, 10, 1.2\rangle$ are the reported historical response time values captured from the invocation of the *Navigation Recommender* service. According to the robustness interval $[a_{ij} - \hat{a}_{ij}, a_{ij} + \hat{a}_{ij}]$, the broker must consider an updated value for $\hat{a}_{ij}$ throughout the time especially in Cloud-integrated IoT dynamic environment where sensor failures, intermittent network connections, and sporadic access are the factors of abnormal perturbation in QoS values. However, one can show that the value of 10 seconds for the response time of the *Navigation Recommender* service is abnormal. If the broker considers abnormal values in the calculation of perturbation rate, overestimation takes place. Therefore, the broker is not able to provide an efficient composite service for the requester. Analysis of historical data presented in Section 5.4.1 proves that there are abnormal data points in the response time which are called as *anomaly* (Liu et al., 2012). In order to eliminate the effects of anomalies in the proposed robust optimization model, we use Isolation-Tree based anomaly detection technique (Liu et al., 2008) in our environment adaption phase. The Isolation Forest is an ensemble regressor that discovers the anomalies in data. It builds an ensemble of random trees as a base estimator for a given dataset; anomalies are the points with the shortest average path length from the tree root to the leaf. Isolation Forest calculates an *anomaly score* $= 2^{-\frac{E(h(\chi))}{nc(n)}}$, where $h(\chi)$ is the number of edges in a tree for a particular data point χ, $E(h(\chi))$ is the average of $h(\chi)$ from a set of isolation trees, and $c(n)$ is a normalization constant for a dataset of size n.

$$As\ \ E(h(\chi)) \to 0, anomalyscore \to 1 \tag{9}$$

As the average number of edges in a tree goes to zero, the score shows a higher degree of abnormality, as shown in Eq. (9). The nodes of the tree are built by splitting instances based on randomly chosen attributes with randomly chosen split points. Fig. 5 shows the result of anomalies that are detected by iForest.

There are several reasons that we choose the iForest algorithm. Firstly, the basic assumption of IForest is that anomalies are few and different (Moghaddam et al., 2018) and more susceptible to isolation; therefore, it is best-fitted for the highly skewed data Kardani-Moghaddam et al.. Secondly, compared with other algorithms, iForest has a linear time complexity with a low memory requirement (Liu et al., 2008). Moreover, iForest has the ability to scale up to find anomalies in extremely large historical records related to a large number of services in a dynamic environment of IoT (Liu et al., 2012). Finally, iForest can easily (using less engagement with parameters) find abnormal historical records of services without relying on any distance or density measure (which decreases the computational cost). More discussions on time analysis of iForest can be found in Sections 5.4.2 and 5.6.

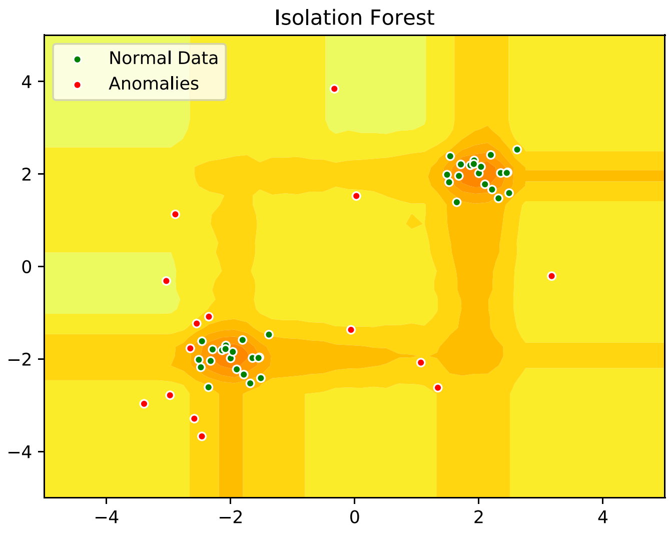


Fig. 5. A sample for iForest anomaly detection result.

4.3. ARC algorithm

While probabilistic approaches start by assuming that the uncertainty has a probabilistic description, in the robust optimization approach, the decision-maker forms a solution that is feasible for any realization of the uncertainty in a given set with a specified bound (Bertsimas et al., 2011). The advantage of robust optimization-based service composition is two folds: first, it does not start by assuming that the uncertainty has a probabilistic description like a certain statistical distribution. Second, a robust optimization approach provides decision-makers with flexibility in determining the level of conservativeness (Bertsimas et al., 2011). Robust optimization (RO) will be applied in situations where a model should reflect experts' opinion, while cannot collect sufficiently large statistical data to apply a probability theory-based approach (because RO only needs the bound, i.e., the maximum and minimum QoS values, which can be extracted from incomplete historical records). Refer to motivation example in Section 2.1, suppose that for each service in Table 1, there are a few and incomplete QoS historical records. Thus, the broker (refer to Fig. 2) cannot fit any distribution on these services but still is able to obtain a bound of QoS changes (maximum and minimum of historical QoS records). Algorithm 1 demonstrates our proposed adaptive robust service composition mechanism. Our algorithm consists of four different parts:

- Inputs: The input part of the algorithm consists the tasks in a given workflow T, the user's constraint B; pool of candidate services S_i, QoS values Q, set of coefficients which are subject to uncertain parameters named J_i, and maximum allowed perturbation amount $\hat{a_{ij}}$ for each s_i^j which are identified by decision-maker (experts' opinion).
- Parameters: The next part of the algorithm provides the parameters: the protection degree Γ, the time period Δ between adaption phases, and the contamination rate Υ, i.e., the proportion of outliers in the monitored QoS values used in the anomaly detection phase.
- Output: The third part of the proposed algorithm introduces a variable *RCS*, which retains the robust (near-)optimal composite service.
- Operations: The last part contains various operations, including *Bertsimas and Sim* approach for model construction, service selection for composition, and anomaly detection for system adaption.

Algorithm 1: An Overview of **ARC**.

```
input  : $T = (t_1, t_2, ..., t_N)$:$T$ is a workflow including $N$ tasks
         $B = (b_{RTime}, b_{Avail}, b_{Reput})$:$B$ is user's constraints
         $S_i = \{s_i^1, s_i^2, ..., s_i^{m_i}\}$:$s_i^j \in S_i$ presents $j$th candidate
         service for performing the $i$th task
         $Q = \{q_1, q_2, ..., q_K\}$: $Q$ defines the QoS values for $s_i^j$
         $J_i$: Set of coefficients which are subject to uncertain
         parameters
         $\hat{a_{ij}}$: Maximum allowed amount of perturbation for
         each $s_i^j$ identified by decision maker (experts'
         opinion)
param  : $\Gamma$: Protection degree,
         $\Delta$: The time period between adaption phases,
         $\Upsilon$: Contamination rate, i.e., the proportion of outliers
         in monitored QoS values
output : $RCS$: The resulted robust composite service

1  $MQV \leftarrow \varnothing$ /* At the start point, we do not have any
       monitored QoS values */
2  $aHat_{ij} \leftarrow \hat{a_{ij}}$ /* As the system continues working, the
       value of the state variable $aHat$ will be updated
       gradually */
3  $compositionRequestQueue.enqueue(\text{Broker}.accept())$
4  while $compositionRequestQueue\ != Empty$ do
5      $compositionRequestQueue.dequeue()$ /* Picking up a
           composition request and extracting required
           inputs and parameters from it */
6      $coefMat \leftarrow$ Encoding tasks of $T$ and services of $S_i$ into
       coefficient matrix using QoS values of $Q$
7      $BertSim \leftarrow$ Formulate the parameters: uncertainty set of $J_i$,
       protection degree of $\Gamma$, and perturbation rate of $aHat$,
       which come from the $BertsimasandSim$ approach
8      $Model \leftarrow$ Construct the robust optimization model based
       on the $coefMat$ and $BertSim$
9      Initialize $RCS$ using a set of arbitrary services to form a
       composite service
10     foreach $model \in Model$ do
11         if $Cost(model) < Cost(RCS)$ then
12             $RCS \leftarrow model$ /* Evaluate the the solution based
                   on objective function of $Cost(S_i)$ and
                   constraint $B$ */
13         end
14     end
15     $Print\ (RCS)$ /* Resulted (near-)optimal robust
           composite service based on user' constraint */
16     if $\Delta$ is elapsed then
17         $MQV \leftarrow QoSMonitoring()$ /* Utilize historical data
               from the monitoring subsystem */
18         $anomalyRemovedMQV \leftarrow IsolatedForest(MQV, \Upsilon)$
               /* Remove the anomalies using the Isolated
               Forest with specified contamination rate $\Upsilon$ */
19         $aHat \leftarrow$
               $aHatEstimator(anomalyRemovedMQV)$ /* Update
               the state variable of $aHat$ using the
               anomaly-removed QoS values
               ($anomalyRemovedMQV$) */
20     end
21 end
```

The first step of the algorithm tries to declare a variable named *MQV* to store the monitored QoS values. We can see that the value of this variable is set to Null. In other words, our proposed ARC algorithm can start its operation without requiring historical data. The value of state variable *aHat* is initialized to $\hat{a_{ij}}$, which is the maximum allowed perturbation amount for each candidate service identified by decision-maker. The value of state variable *aHat* is further updated in the main while loop. It is worth mentioning that while probabilistic approaches start by assuming that the uncertainty has a probabilistic description or lies on specific statistical distribution, in robust optimization decision maker forms a solution that is feasible for any realization of the uncertainty in a given set with specified bound (Bertsimas et al., 2011). The statement *Broker.accept()* (at line 3) accepts the incoming composition requests continuously and adds them to the queue of *compositionRequestQueue*.

The main *while* loop (lines 4 to 21) will be repeated until no composition request exists. In each iteration, three main stages are operating: 1) *Bertsimas and Sim*-based robust optimization model construction (lines 5 to 8); 2) Composite service selection (lines 9 to 15); 3) Anomaly detection and system adaption (lines 16 to 20). In the first stage of the *while* loop (line 5), the algorithm picks up a composition request from the queue and extracts required inputs (like tasks T and candidate services S_i) and parameter Γ from the request. Encoding tasks of T and services of S_i into the coefficient matrix using QoS values of Q is the next operation. Our algorithm formulates the components *uncertainty set* of J_i, *protection degree* of Γ, and *perturbation rate* of *aHat* which come from the *BertsimasandSim* approach in Eq. (8). Finally (line 8), the algorithm constructs the robust optimization model based on the output of steps 6 and 7.

Refer to Fig. 2, when a user submits a composition request (namely R), the system 1) retrieves the available services from the service repository, 2) creates an arbitrary initial composite service and assigns to *RCS*, 3) computes the aggregated cost of the composite service, and 4) searches for a better composite service that satisfies R (according to user's constraints). More precisely, the *foreach* loop is executed (lines 10 to 14) for each composition request. This inner loop refers to the iterative evaluation of all possible composite services. It makes a composite service as an input of function *Cost* and returns the aggregated cost of the current composite service. It is worth mentioning that the best composite service is obtained by assessing the aggregated cost of **whole** workflow. This is because, in our system, the task-to-service map is generated for the whole workflow, i.e., the best composite service is calculated based on all tasks in a given workflow (T). At line 15, the robust (near-)optimal composite service is outputted.

The last important part of the algorithm tries to deal with system adaption (lines 16-20). After a composition request is resolved, the algorithm checks whether it is the time to adapt or not by checking a timer and comparing it with Δ. In this situation, if the adaption time arrives, first, the system refers to *monitoring subsystem* (line 17) to utilize the existing historical data (in our algorithm, the container *MQV* retains these data). In the next step, the abnormal QoS values are removed from the *MQV* container by using the subsystem *IsolatedForest(MQV,* Υ*)* with specified contamination rate Υ. The parameter Υ determines the proportion of outliers that are about to be removed. The anomaly-removed QoS values are set to a new container *anomalyRemovedMQD* (line 18). For adaption phase, the value of the state variable *aHat* is updated by invoking *aHatEstimator (anomalyRemovedMQV)* (line 19). This adaption adjusts the system settings effectively and efficiently. This algorithm not only faces uncertainty using the *Bertsimas and Sim* robust optimization approach, but it also adjusts the amount of perturbation periodically according to the time interval of Δ. If the accuracy is most important in a system, this parameter can be set to lower value, which increases the number of invocations of the anomaly detection subsystem.

5. Performance evaluation

5.1. Simulation configuration

The ARC algorithm has been evaluated in different scenarios of service composition. Due to the dynamic nature of Cloud-integrated IoT service environments, working on real QoS values of candidate services is crucial for an accurate QoS estimation. This is because neither random generated QoS values (based on specific probabilistic distribution) nor synthetic QoS values (using simulation packages like NS3) do not reflect the real-world uncertainties. According to the literature, many researchers (Wang et al., 2010; Schuller et al., 2012; Ye et al., 2014; Mostafa and Zhang, 2015) used randomly/synthetic generated QoS values (response time, etc.). Although using these generated QoS values is easy to access and straightforward, it does not reflect the real-world behavior of QoS. In other words, for effective uncertainty-aware service composition, it is crucial to deal with a real dataset. As an example, when an end-user invokes a service, four factors cause a delay in the communication networks: transmission delay, processing delay, queuing delay, and propagation delay. Hence, by using a real-world dataset, we can also reflect the impact of uncertainty of communication networks in the QoS values.

The experiments are conducted on a real-world QoS dataset that 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 reported by Zheng et al. (2014). A number of computing nodes from the PlanetLab[1] have been 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 dataset includes information of 339 service users comprising user ID, IP address, country, AS (Autonomous System) 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 values synthetically as a function of the response time values according to Schuller et al. (2012). The composite service considered in the simulation scenarios has a sequential structure since any other composition structures such as loop, parallel, and condition can be transformed into a sequential structure (Zheng et al., 2011; Alrifai et al., 2012; Dou et al., 2013) through the methods mentioned in Alrifai et al. (2012); Zheng et al. (2013); Dou et al. (2015). For example, a loop structure is a specific number of repetitions of sequence structure.

The proposed robust optimization problem of Eq. (8) is modeled as mixed-integer programming and solved by IBM ILOG CPLEX Optimizer. For anomaly detection, we used the Isolation Forest algorithm from the scikit-learn machine learning library in Python sci. Isolation Forest is introduced in 2008 and became available in scikit-learn v0.18 in 2016. The experiments are conducted on a machine with Intel(R) Core(TM) i7-6650U 2.21 GHz processor and 16GB RAM. The machine is running under Windows 10.

5.2. Performance metrics and baselines for comparison

The performance of the ARC algorithm is compared to the following approaches proposed by existing works:

- Information Theory-based unreliable service selection (*iTheory-based*) (Wang et al., 2017): In the iTheory-based approach, un-

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

reliable services are filtered in *uncertain service filtering* phase of service composition based on two well-known criteria of Entropy and Variance. We select this approach since the authors have achieved an impressive performance in comparison with other approaches like the Skyline approach. The candidate services with higher variance and entropy are the most probable services for ignoring. The entropy value is useful when the variance of two candidate services are the same.

- QoS-aware Clustering-based service selection for ambient intelligence (*cluster-based*) (Khanouche et al., 2019): This new approach has been selected for comparison because it presents a good performance in terms of composition optimality and it performs better than other clustering approaches like (Mabrouk et al., 2009). Furthermore, this approach has been proposed for ambient intelligence, which is very close to our work in terms of considering the functionality of smart objects like sensors as a software service. They cluster candidate services into three categories: High-QoS, Middle-QoS, and Low-QoS based on QoS.
- Deterministic service selection for service composition (*dSelection*)(Zeng et al., 2004): This approach proposes a goal-driven service composition in mobile and pervasive computing to avoid composition failure. However, it does not consider the uncertainty in advertised QoS values in the process of service composition. Therefore, using an optimization algorithm, this approach leads to an optimal composition. We have considered this approach as a baseline to draw a comparison between our proposed approach and the aforementioned approaches in terms of optimality.

In order to prove the effectiveness of the ARC algorithm compared to the approaches mentioned above, we used the following metrics:

- Optimality of the composition: According to the definition in Wang et al. (2017) and Khanouche et al. (2019), this metric illustrates the ratio between obtained fitness and an ideal (optimum) fitness for a given composite service. Formally:

$$Optimality = \frac{F_{ARC}}{F_{optimal}} \tag{10}$$

- Anomaly awareness: To assess the role of anomaly awareness in our proposed approach, we have defined the *anomaly awareness* metric. Anomaly-awareness can be performed using the analysis of historical monitored QoS values. Making a correct decision to the inclusion or exclusion of a typical candidate service in the composition phase is dependent on fine-grained uncertain service identification.
- Protection degree: Protection degree, which allows the decision-makers to control a trade-off between robustness and optimality. This metric is defined as an ability of the system to adjust robustness by using a subset of variables as the uncertain parameters. A decision-maker specifies the protection degree by changing the parameter Γ in (8).
- Time Complexity: In order to have a better understanding of the performance of the proposed method, we analyze the time taken by the procedure of **dealing with uncertainty**, using asymptotic notation. For a fair comparison, we used the same algorithm for the composition phase (the service selection is performed by using 0-1 integer programming to find and select services according to user preferences and QoS constraints) (Wang et al., 2017). In other words, *dealing with uncertainty* is taken into account as a core part of each approach.

5.3. Optimality of composition

In the first experiment, the impact of the number of candidate services on the ARC performance is evaluated. The number of candidate services for performing each task varies from 5 to 50. Based on our motivation scenario discussed in Section 2.1, we set the number of tasks to 6. Many researchers in literature have evaluated their proposed approach by applying it to their motivation scenario. For example, in Ye et al. (2014), the authors use the trip planning scenario containing six tasks as their testing environment. Also, the authors in Hwang et al. (2014) consider five tasks in their scenario, i.e., Electronic Product Purchase composite service, Chen et al. (2016a) used ten service classes (tasks) in their mobile and pervasive computing scenario. Other studies like Zhao et al. (2015); Zheng et al. (2016); Liu et al. (2016); Deng et al. (2015) employed 10, 7, 9, 4 tasks for the evaluation, respectively. Importantly, *iTheory-based* (Wang et al., 2017) and *cluster-based* (Khanouche et al., 2019) approaches set the number of tasks in a workflow to 5 and 3, respectively.

The protection degree Γ and perturbation amount are set to 6 and the value of *standard deviation* of QoS historical record, respectively. For a fair comparison, the value of protection degree is considered as 6 for maximum protection degree (minimum level of optimality). According to Wang et al. (2017) and Khanouche et al. (2019), we chose 1/5 candidate services with lower variance and 10% of candidate services belong to the High-QoS cluster for *iTheory-based* and *cluster-based*, respectively.

Fig. 6 demonstrates the optimality of composition obtained with the ARC algorithm and other algorithms with the increment of candidate services number. The ARC algorithm uses a mathematical robust optimization programming that is based on considering all candidate services in the process of selection. This finding can be explained by the fact that iTheory-based and cluster-based algorithms filter the unreliable candidate services from search space, and it means they take into account only those candidate services which provide the lower variance and higher fitness, respectively. Although this filtering helps the system to decrease the size of the search space, it increases the probability of finding solutions with less optimality in comparison with the optimal approach. As shown in Fig. 7, although the selection of reliable candidate services provides extra cost for the user (requester of the composite service), the ARC algorithm proposes the better composite service in comparison with other approaches in terms of cost (price).

In the second experiment, in order to investigate the generality of the proposed method, we have evaluated the performance of the ARC algorithm with different workflow sizes (number of tasks). The number of tasks in each workflow varies from 5 to 50. The number of candidate services for each task is set to 5. The protection degree Γ value and perturbation amount are set to N and the value of the standard deviation of QoS historical record, respectively. For a fair comparison, we increase the value of protection degree according to the N to obtain a maximum protection degree that results in finding a near-optimal solution rather than an optimal one. According to Wang et al. (2017) and Khanouche et al. (2019), we chose 1/5 candidate services with lower variance and 10% of candidate services belong to the high-QoS cluster for *iTheory-based* and *cluster-based*, respectively. Fig. 8 indicates the optimality of composition obtained with the ARC algorithm and other algorithms with the increment of workflow size. We found that the optimality obtained by our approach is always higher than *iTheory-based* and *cluster-based* approaches with an increasing number of tasks. These experimental results prove the generality of the ARC algorithm in finding a near-optimal solution for different workflow sizes.

However, removing or discarding the candidate services before the composition phase (in *iTheory-based* and *cluster-based* approaches) reduces the search space, it leads to a costly composition. Fig. 9 indicates that in terms of composition cost, the ARC algorithm is able to narrow the *near optimal*-optimal gap effectively

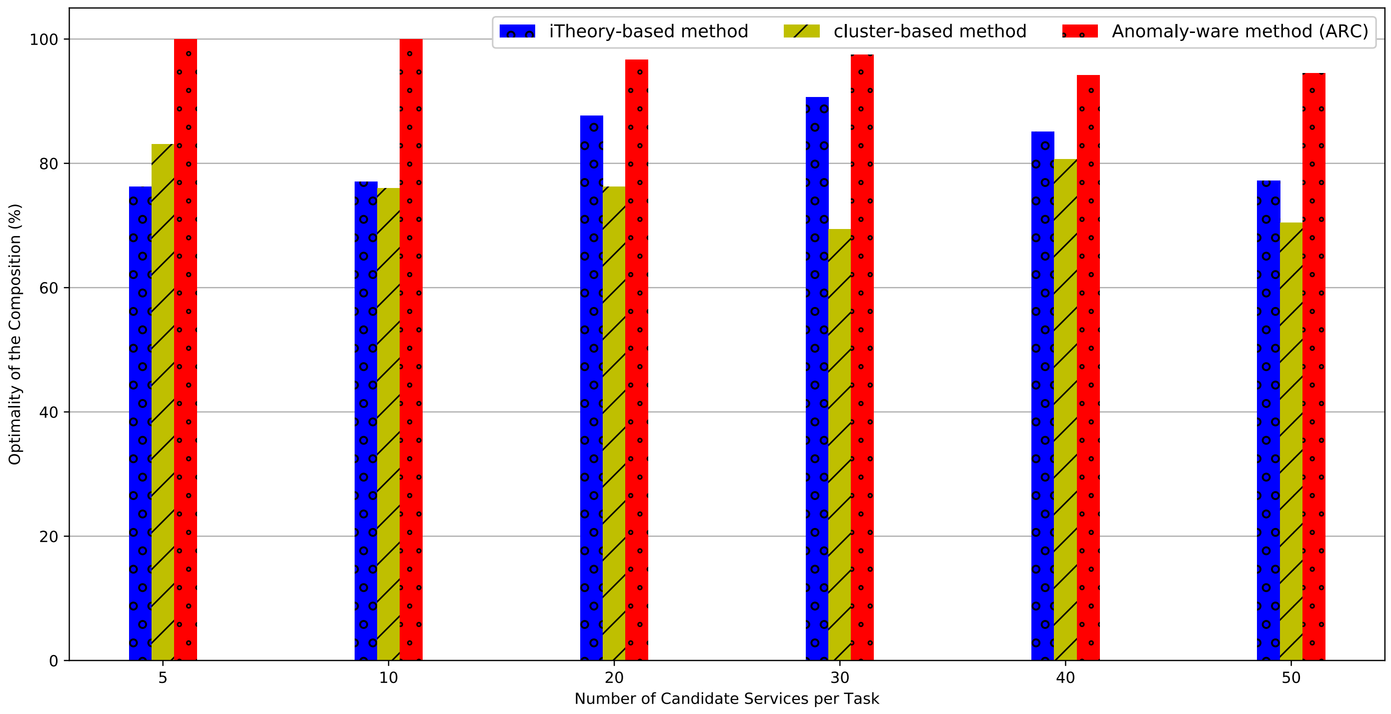


Fig. 6. Optimality of composition versus the number of candidate services.

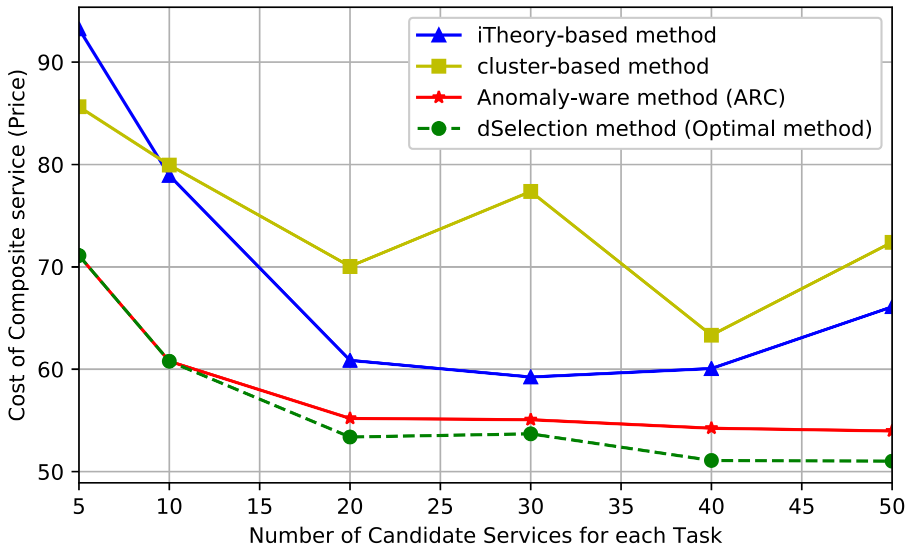


Fig. 7. Cost (price) of composite service versus the number of candidate services.

compared with the optimal solution that is obtained in the *dSelection* approach.

5.4. Anomaly awareness

Both *cluster-based* and *iTheory-based* methods define the unreliable candidate services using the historical records. However, the *cluster-based* approach determines clusters of service in terms of High/Medium/Low QoS level without investigation of the variability and anomalies in historical records. To utilize the historical records, *iTheory-based* method proposes a novel reliable service selection based on variance and entropy of candidate services. The first and second phases of this approach are devoted to QoS uncertainty computing and uncertain service pruning, respectively. However, removing the uncertain services reduces the search space in the pre-composition phase, it affects on obtaining an optimal composition. Especially when a candidate service that may be part of the optimal composition is removed, the optimality of composition also will be lost. These limitations are the major drawbacks of the *iTheory-based* and *cluster-based* approaches.

While simple robust service composition is not able to adapt itself with changes in QoS values (which leads to wide perturbation ranges, i.e., over-conservatism), our proposed adaptive robust service composition can estimate the bounds of the perturbation rate of uncertain parameters periodically. Fig. 10 shows the impact of the protection degree on the cost of the composite service (i.e., optimality) according to the different number of candidate services. This figure shows that ARC achieves better performance (composite service with lower cost) than simple robust optimization. This is because in simple robust optimization, the amount of perturbation

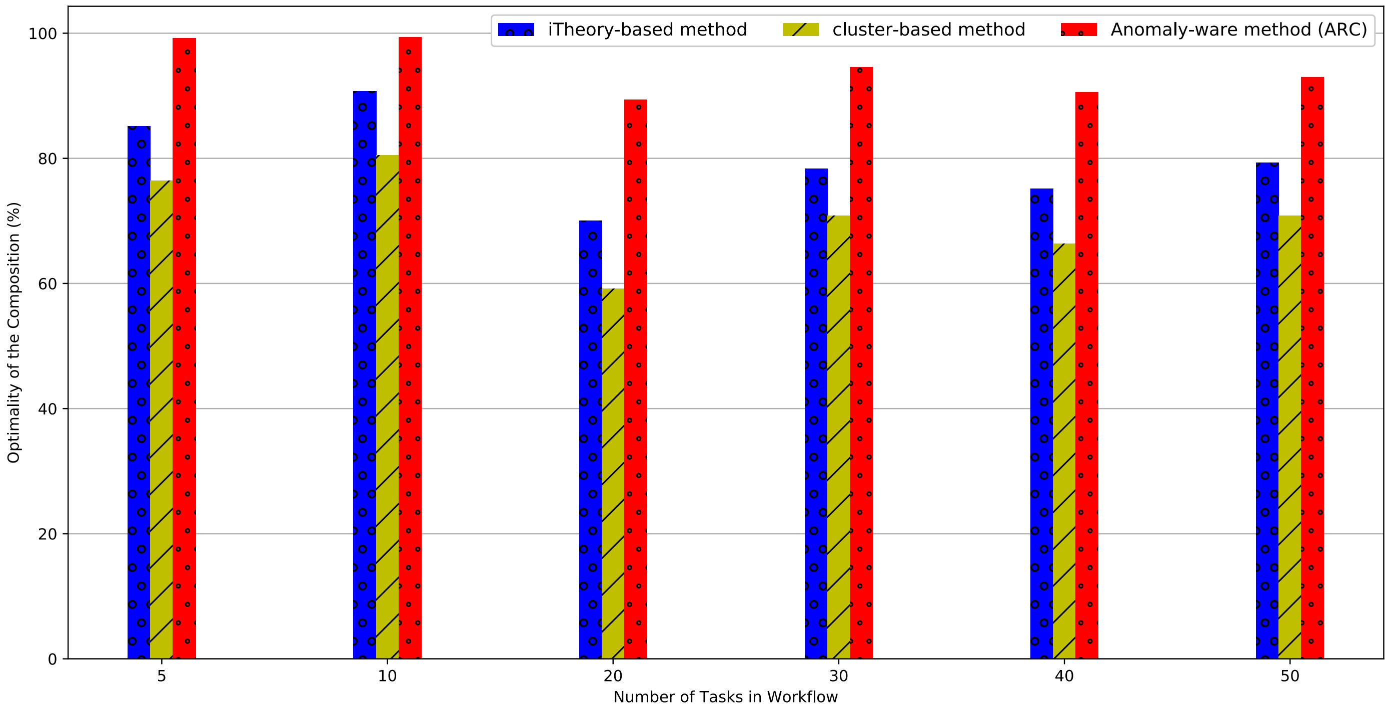


Fig. 8. Optimality of composition versus the workflow size.

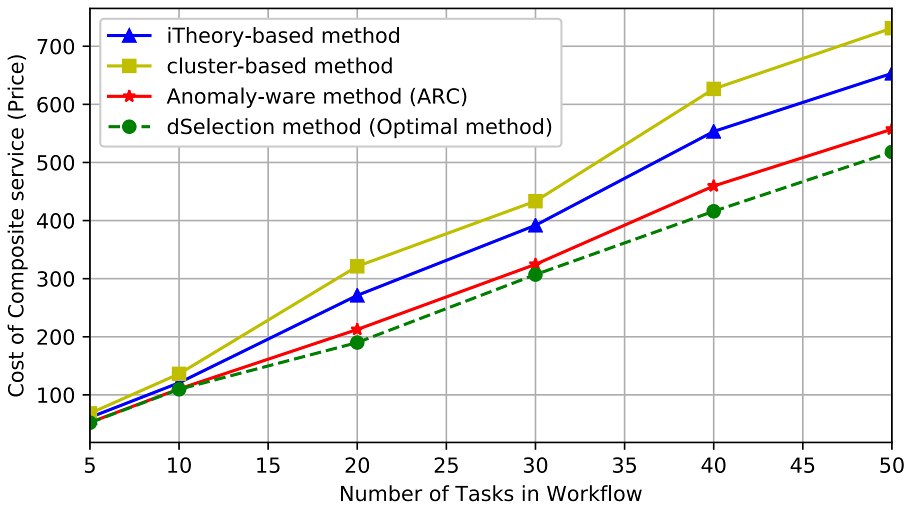


Fig. 9. Cost (price) of composite service versus the workflow size.

for uncertain QoS parameters is considered as a fixed value, which leads to over-conservatism in the process of candidate services selection. Nevertheless, in ARC, as the system continues working, it can invoke the *monitoring* subsystem to access the recorded QoS values to update the perturbation amount. From the dataset, in this experiment, we used 320 monitored QoS values (called as transactions) for each service in a composite service. Fig. 11 shows the uncertainties (changes in QoS values) along with existing anomalies which are adopted from transaction logs of two services of the evaluated workflow. These changes come from the inherent uncertainty of the execution of services and communication networks in the real-world. To achieve an accurate composition, it is essential to remove existing anomalies before calculating the amount of perturbation from transaction logs.

To validate the proficiency of ARC, in the next experiment, we evaluate *Anomaly-aware* robust service composition with a *simple* robust service composition. To this end, the aggregated cost of the composite service is adopted as the main comparison criterion of our experiments. We conduct a totally of 75 experiments in the five categories. In each category, user's constraint (b_{RTime} in Eq. (8)) is set to 5, 10, 20, 40 and 60 seconds accordingly. The number of tasks is set to 10, whereas the number of candidate services for each task varies between 10 and 50. Additionally, we set the contamination ratio of Isolation Forest to 0.02. It means that 0.02 of monitored QoS data will be treated as outliers. From the dataset, for each service, we consider 320 historical records of monitored response time values. Fig. 10 shows the cost of a composite service regarding the number of candidate services for different categories. In all categories, ARC offers a cheaper composite service without any violation of the user's constraint. It means that setting a fixed amount of perturbation for QoS parameters, which are used in simple robust service composition, leads to underestimation or overestimation in the process of candidate services selection.

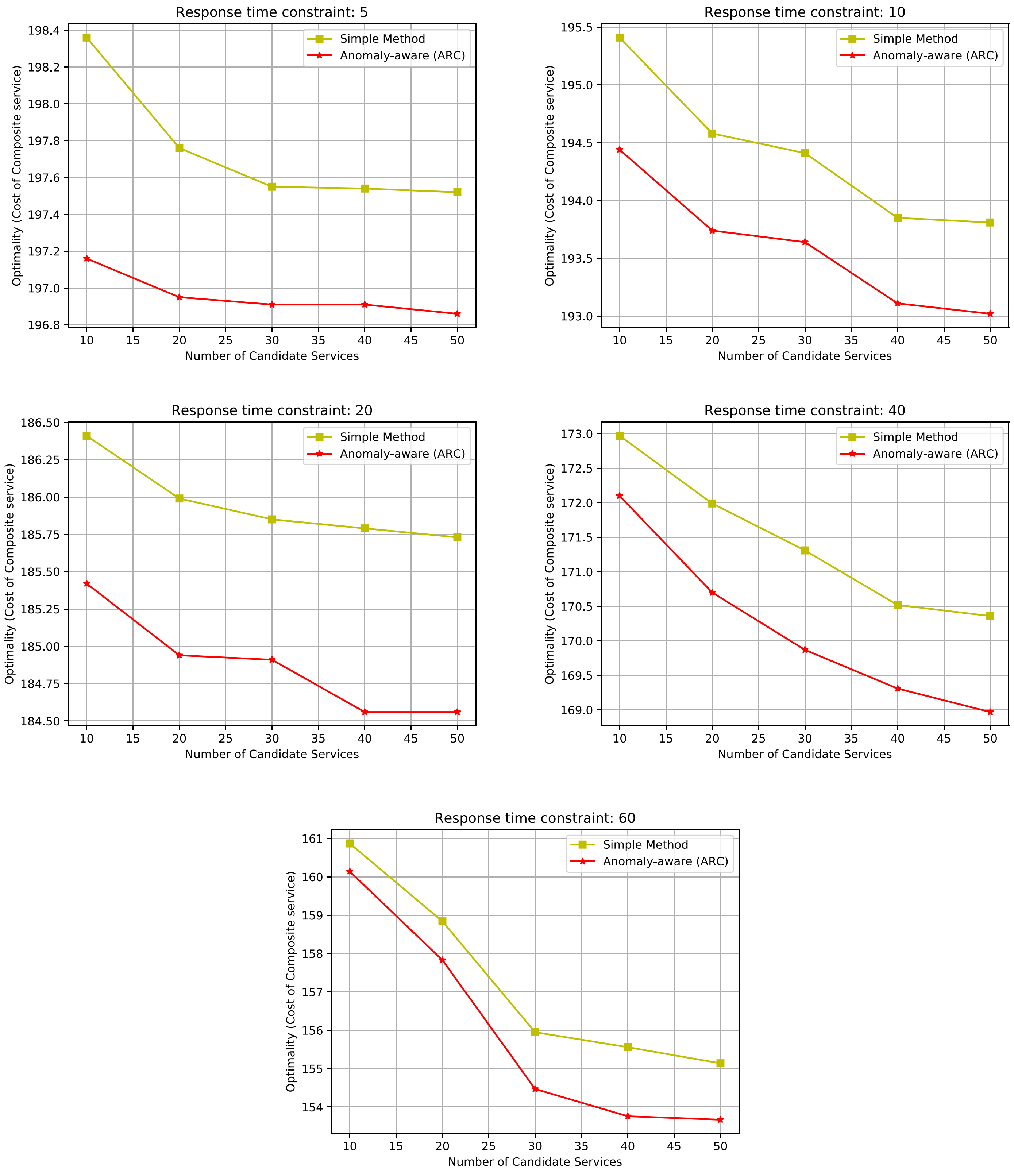
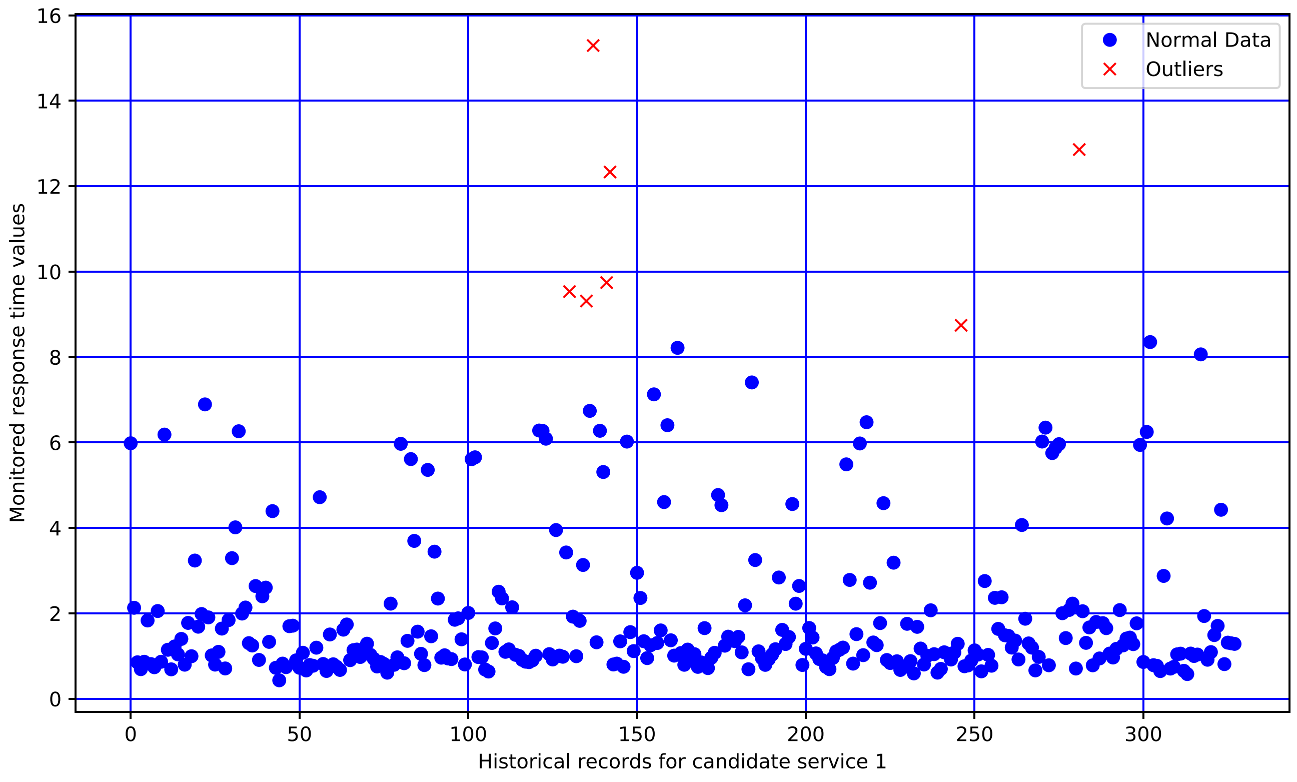


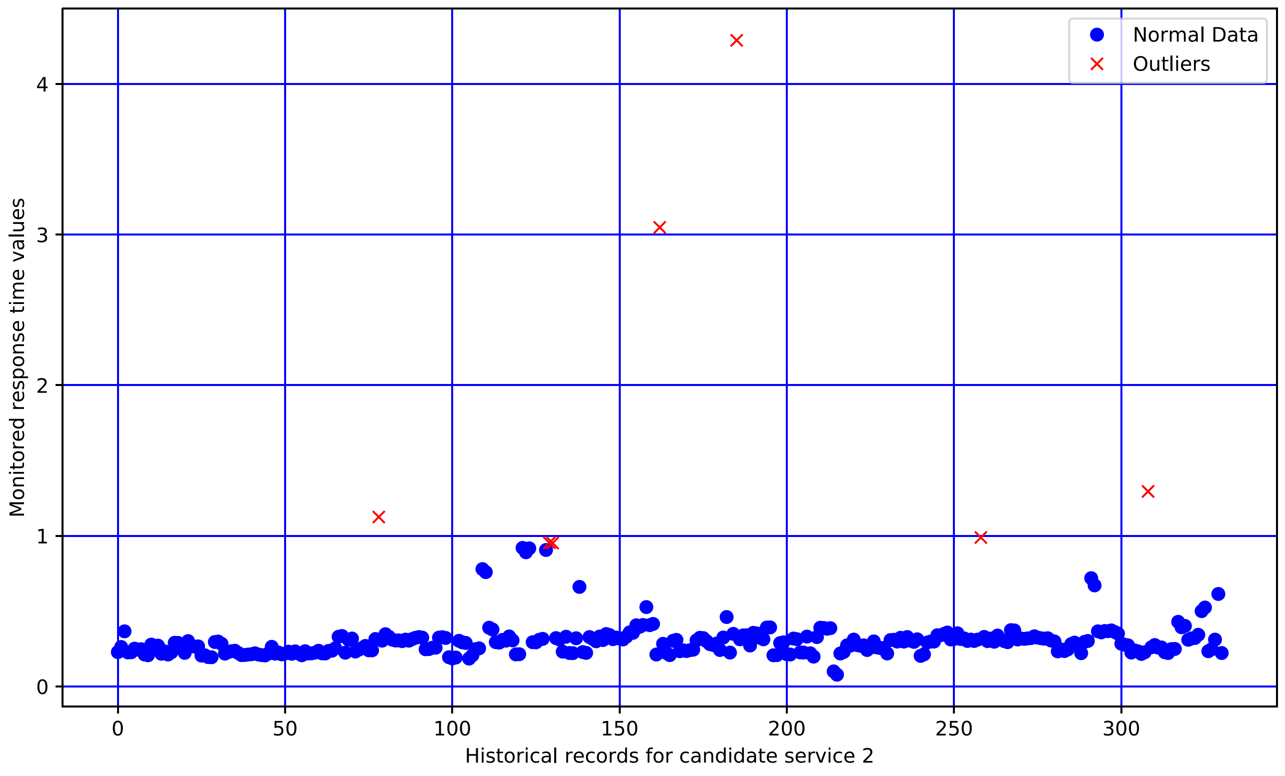
Fig. 10. Impact of the protection degree on the cost of the composite service with different number of candidate services

5.4.1. Discussion

The ARC algorithm calculates the amount of perturbation (parameter $\hat{a_{ij}}$ in Eq. (8)) for standard service to determine the level of uncertainty for that service. The ARC algorithm repeats this procedure based on monitored historical QoS records for all candidate services. For each candidate service s_i^j, the utility value is set to μ_i^j, and the perturbation amount is set to σ_i^j, where μ_i^j and σ_i^j represent the mean and standard deviation of the historical records after removing the anomalies, respectively. As an example, we found that the iTheory-based approach removes the services shown in



(a) Identification of abnormal QoS records for Service 1



(b) Identification of abnormal QoS records for Service 2

Fig. 11. Finding abnormal historical records using an unsupervised Isolation Trees-based (iForest) approach.

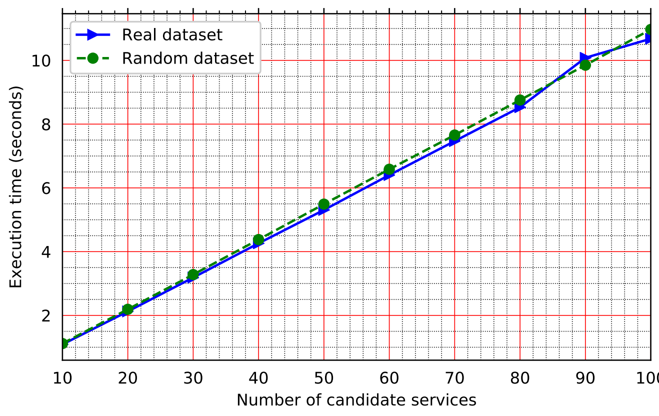


Fig. 12. Execution time of procedure of abnormal records identification (320 historical records per candidate service).

Fig. 11, because of their higher variance. However, using a fine-grained analysis, we found out that the reason for the high variance is some anomalies in historical records, which is prevalent in Cloud and IoT based services. An in-depth analysis of types of performance anomalies, including CPU and memory bottlenecks, is performed in Moghaddam et al. (2018). Therefore, with the knowledge of anomaly detection analysis, we calculate the amount of perturbation for these services instead of pruning them. Meanwhile, a question that may arise is whether simple threshold-based filtering is usable or not? Due to the dynamicity of Cloud and IoT environments, calculation of a predefined and fixed threshold is not possible. The experiments indicate that the level of this threshold for a service such as *Service 1* in Fig. 11a can be different from another one like *Service 2* in Fig. 11b. Furthermore, for the sake of simplicity, we adopt response time as target QoS parameters in this experiment. However, in real-world scenarios, the number of QoS attributes are not limited to one or two attributes; indeed, the calculation of threshold for each attribute in the threshold-based approach is not feasible.

5.4.2. *Time of anomaly detection subsystem*

To asses the efficiency of anomaly detection procedure, we have evaluated *computation time* required for abnormal record identification using two types of datasets. The first is the real-world QoS dataset introduced in Section 5.1. We also performed experiments with a randomly generated dataset with the uniform distribution that contains the QoS value of candidate services. Each experiment consisted of several candidate services in a range of 10 to 500 by increment 20. The results indicate that the procedure of abnormal records identification increases linearly according to the number of candidate services. As shown in Fig. 12, the required times for finding abnormal records in the real dataset and random dataset are nearly the same when the number of candidate services grows. The results also show that in a real environment with a high number of candidate services, it takes 10.679 s when the number of candidate services is 100, and the total number of historical QoS values is 32K records. Note that, this fine-grained procedure can be launched *periodically* according to the system configuration (refer to Algorithm 1, lines 16 to 20) to identify the abnormal records and determine the amount of perturbation.

5.5. *Impact of protection degree*

In order to verify the influence of the parameter of protection degree (i.e., Γ) on the aggregated cost (optimality) of the composition, we consider two cases of experiment depending on the number of candidate services and the workflow size. This parameter allows the decision-makers to control a trade-off between robustness and optimality. Neither the iTheory-based approach nor the cluster-based method considers this feature for the decision-maker. In our proposed robust optimization method, the parameter Γ leverages the protection degree (i.e., the degree of risk around the composite service). In the following experiments, we gradually increase the value of Γ, to verify the correctness of this parameter.

Case 1 In this case, we prove the correctness of ARC depending on the different number of candidate services. In the experiments, the value of Γ is considered as 0.5, 0.75, 1, 1.5, and 2 to 12 *by the increment 1*, and the size of workflow is set to 10. The real response time values are extracted from the aforementioned real dataset to assign the candidate services. We consider 10 and 40 candidate services for each task and set the user's constraint (on aggregated cost) to 10 s. As shown in Fig. 13, the aggregated cost of composite service increases when the protection degree (i.e., the value of Γ) increases. According to the results, we can see that when the value of Γ is 0.5, the composition cost is 191.54 and 191.09 for each experiment, respectively. Moreover, in the maximum protection degree (i.e., $\Gamma = 10$), the cost of the composition has been grown 1.51% (194.44) and 1.05% (193.11). The main reason behind this growth is that when the protection degree increases, the number of abstract services that must be taken their worst-case value (according to Eq. (8)) also increases. As an example, consider that the advertised value of a typical service is 2.5 ms, and the amount of perturbation has been calculated as 0.2 ms by our anomaly-aware system. In the worst-case scenario, the value of response time for this service will be considered as 2.2 ms. Notably, considering 2.2 for this service may result in a violation from the user's constraint (See Eq. (2)); thus, the ARC algorithm has to search for another service with lower response time, which is more expensive than the previous one. Also, when the number of candidate services increases, the solution space grows and ARC can find better (cheaper) composite service (See Fig. 13 for the *Cand=10* vs *Cand=40*). From the results, ARC effectively allows the decision-makers to control the trade-off between robustness and optimality for Cloud-integrated IoT environments.

Case 2 This experiment is to prove the correctness of the ARC algorithm concerning the workflow size. In the experiments, the value of Γ is considered as 1, 5, 10, 20, 30, 40 and the number of candidate services is fixed to 5 for all scenarios. Each scenario consists of different workflow sizes ranging between 10 to 40 by increment 10. As shown in Fig. 14, the aggregated cost of composite service increased when the protection degree increased. The reason is that when the number of candidate services that are allowed to take their worst-case value increases, the ARC algorithm selects the services with a lower response time to satisfy the user's constraint. The reason for choosing services with lower response time value (that are more expensive) is that the ARC algorithm adds the amount of perturbation to the QoS value. This increases the left-hand side value of the constraint in Eq. (2). Therefore, in order to satisfy the user's constraint, the broker selects services with lower response times, which are more expensive than the other services. This feature enables the decision-maker to make a flexible decision between a risky decision ($\Gamma = 0$) and a decision with the highest protection level ($\Gamma = 10$).

5.6. *Time complexity analysis*

To evaluate the efficiency of our proposed approach, we analyze the order of growth of the running time of each approach using asymptotic notation. Basically, calculating entropy on Integer numbers (like response time as discussed in *iTheory-based* approach) involves categorizing the transactions (historical QoS records) into a finite number of intervals. Presumably, the time complexity of the entropy calculation applied to data of length n, and B bins are

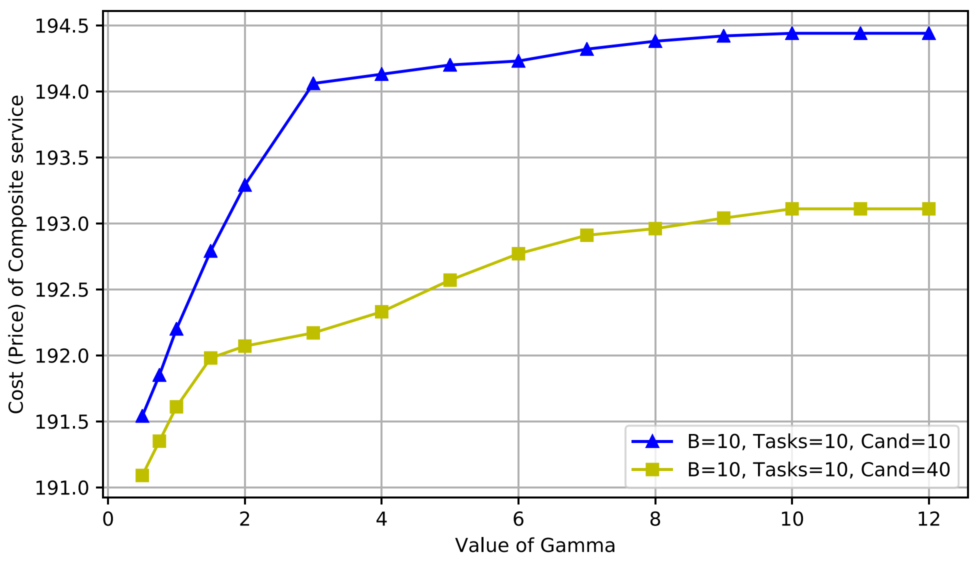


Fig. 13. Impact of the protection degree on the cost of composite service w.r.t number of candidate services.

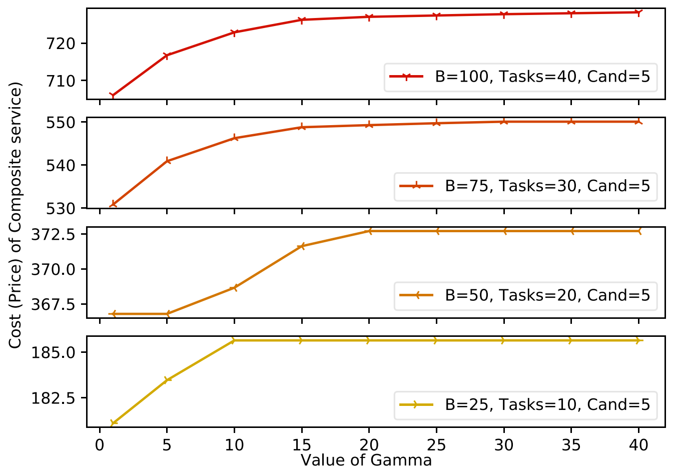


Fig. 14. Impact of the protection degree on the cost of composite service w.r.t workflow size.

$O(n.B)$ (the exact definition of B depends on the implementation of uncertain service filtering method). Considering a composition of N abstract services in *iTheory-based* approach, the time complexity of the uncertain service filtering based on the entropy and variance is then $O(N.B.n) + O(N.m_i.n) + O(N.m_i.n)$ where n represents the number of the transactions, m_i is the number of candidate services for ith task, and B is the number of intervals. Considering a composition of N abstract services in *cluster-based* approach, the time complexity of the candidate services filtering based on the utility value is then $O(N.m_i.k.q.t) + O(N.m_i)$ where m_i is the number of candidate services for ith task, k is the number of clusters, q is the number of QoS attributes, and t refers to the number of iterations before the convergence of the method. There are two input parameters in the iForest algorithm: the sub-sampling size φ and the number of trees t. Authors in Liu et al. (2012) found $\varphi = 256$ and $t = 100$ empirically. Therefore, the training time complexity is constant when the subsampling size and ensemble size are fixed. Considering a composition of N abstract services in our approach, the time complexity of the finding anomalies and amount of perturbation is then $O(N.m_i.n) + O(N.m_i)$ where n represents the number of the transactions, and m_i is the number of candidate services for ith task. It is worth mentioning that while a large k in *cluster-based* and large B in *iTheory-based* increase the computation time substantially (Liu et al., 2012), ARC can be scaled up to handle large and high-dimensional datasets. Finally, unlike clustering-based algorithms such as *K-means*, the use of the iForest method does not impose specific assumptions on the data to be partitioned.

6. Conclusions and future work

In this paper, an **A**nomaly-aware **R**obust service **C**omposition (ARC) architecture is proposed to address the problem of service composition under uncertainty around QoS parameters for the Cloud-integrated IoT environment where the heterogeneous smart *things* are connected to Cloud. Traditional approaches for dealing with uncertainty filter the uncertain services in the pre-composition phase, which leads to a non-optimal solution. We applied a *mathematical robust optimization* method to concern with an uncertainty of the advertised QoS values. This means our proposed approach is able to encounter the perturbation in the QoS values without depending on a specific statistical distribution. One of the crucial features of ARC is a numerical simplicity-controlled parameter, namely (*protection degree*), which allows the decision-makers to control a trade-off between robustness and optimality. Furthermore, to enhance the composition optimality, we extended the proposed architecture with an adaption phase to adjust the required parameters of our robust model. Adaption phase exploits a machine learning *anomaly detection* system to deal with uncertain services in a fine-grained manner by the identification of abnormal QoS records instead of filtering the service. Most notably, we observed that the adaption phase for the service composition problem is critical in the Cloud-integrated IoT dynamic environment where sensor failures, intermittent network connections, and sporadic access cause some anomalies in QoS monitored values. Moreover, we applied a series of experiments using a real dataset to verify and validate the performance of our proposed ARC. The experiment results show that the proposed approach significantly outperforms existing solutions and achieves *14.55% of the average improvement* in the finding optimal solution than previous works like information theory-based and the clustering-based method.

This study can be extended in several directions. First, the method used to solve the robust optimization model is based on mixed-integer programming. We will continue to investigate an improved method by applying the meta-heuristic algorithm to find the optimum service set. Second, other QoS parameters like energy consumption also can be investigated as an uncertain QoS parameter in future work. In consequence, it is still an open research problem to develop a more efficient dynamic QoS-aware service composition method with polynomial time complexity and a high-quality composite service for new dynamic *Internet Computing* paradigms like fog and edge computing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Mohammadreza Razian: Conceptualization, Methodology, Writing - original draft, Writing - review & editing, Data curation, Software, Validation, Visualization, Investigation. **Mohammad Fathian:** Conceptualization, Methodology, Supervision, Writing - review & editing, Project administration, Validation. **Rajkumar Buyya:** Methodology, Writing - review & editing, Supervision, Validation.

References

Agra, A., Christiansen, M., Figueiredo, R., Hvattum, L.M., Poss, M., Requejo, C., 2013. The robust vehicle routing problem with time windows. Comput. Oper. Res. 40 (3), 856–866.

Alrifai, M., Risse, T., Nejdl, W., 2012. A hybrid approach for efficient web service composition with end-to-end qos constraints. ACM Trans. Web (TWEB) 6 (2), 7.

Amiri, M.A., Serajzadeh, H., 2010. Qos aware web service composition based on genetic algorithm. In: 5th International Symposium on Telecommunications (IST), 2010. IEEE, pp. 502–507.

Ardagna, D., Pernici, B., 2007. Adaptive service composition in flexible processes. IEEE Trans. Softw. Eng. 33 (6).

Asghari, P., Rahmani, A.M., Javadi, H.H.S., 2018. Service composition approaches in iot: a systematic review. J. Netw. Comput. Appl. 120, 61–77.

Avila, K., Sanmartin, P., Jabba, D., Jimeno, M., 2017. Applications based on service-oriented architecture (soa) in the field of home healthcare. Sensors 17 (8), 1703.

Aws iot analytics. https://aws.amazon.com/iot-analytics/pricing/?p=ps, accessed: 2019-05-01.

Balalaie, A., Heydarnoori, A., Jamshidi, P., 2016. Microservices architecture enables devops: Migration to a cloud-native architecture. IEEE Softw. 33 (3), 42–52.

Ben-Tal, A., Nemirovski, A., 2002. Robust optimization–methodology and applications. Math. Program. 92 (3), 453–480.

Bernat, G., Colin, A., Petters, S., 2003. pwcet: a Tool for Probabilistic Worst-Case Execution Time Analysis of Real-Time Systems. University of York, Department of Computer Science.

Bertsimas, D., Brown, D.B., Caramanis, C., 2011. Theory and applications of robust optimization. SIAM Rev. 53 (3), 464–501.

Bertsimas, D., Sim, M., 2004. The price of robustness. Oper. Res. 52 (1), 35–53.

Bertsimas, D., Thiele, A., 2006. Robust and data-driven optimization: modern decision making under uncertainty. In: Models, Methods, and Applications for Innovative Decision Making, INFORMS, pp. 95–122.

Botta, A., De Donato, W., Persico, V., Pescapé, A., 2016. Integration of cloud computing and internet of things: a survey. Future Gen. Comput. Syst. 56, 684–700.

Bronsted, J., Hansen, K.M., Ingstrup, M., 2010. Service composition issues in pervasive computing. IEEE Pervasive Comput. 9 (1).

Buyya, R., Srirama, S.N., Casale, G., Calheiros, R., Simmhan, Y., Varghese, B., Gelenbe, E., Javadi, B., Vaquero, L.M., Netto, M.A., et al., 2018. A manifesto for future generation cloud computing: Research directions for the next decade. ACM Comput. Surv. (CSUR) 51 (5), 105.

Cazorla, F.J., Kosmidis, L., Mezzetti, E., Hernandez, C., Abella, J., Vardanega, T., 2019. Probabilistic worst-case timing analysis: Taxonomy and comprehensive survey. ACM Comput. Surv. (CSUR) 52 (1), 14.

Chattopadhyay, S., Banerjee, A., 2016. Qscas: Qos aware web service composition algorithms with stochastic parameters. In: 2016 IEEE International Conference on Web Services (ICWS). IEEE, pp. 388–395.

Chen, N., Cardozo, N., Clarke, S., 2016. Goal-driven service composition in mobile and pervasive computing. IEEE Trans. Serv. Comput. 11 (1), 49–62.

Chen, Y., Jiang, L., Zhang, J., Dong, X., 2016. A robust service selection method based on uncertain qos. Math. Prob. Eng..

Chhetri, M.B., Lin, J., Goh, S., Yan, J., Zhang, J.Y., Kowalczyk, R., 2006. A coordinated architecture for the agent-based service level agreement negotiation of web service composition. In: Australian Software Engineering Conference, 2006.. IEEE, pp. 10–pp.

Chifu, V.R., Pop, C.B., Salomie, I., Suia, D.S., Niculici, A.N., 2011. Optimizing the semantic web service composition process using cuckoo search. In: Intelligent Distributed Computing V. Springer, pp. 93–102.

Deng, S., Huang, L., Li, Y., Zhou, H., Wu, Z., Cao, X., Kataev, M.Y., Li, L., 2015. Toward risk reduction for mobile service composition. IEEE Trans. Cybern. 46 (8), 1807–1816.

Dou, W., Zhang, X., Liu, J., Chen, J., 2013. Hiresome-ii: towards privacy-aware cross–cloud service composition for big data applications. IEEE Trans. Parallel Distrib.Syst. 26 (2), 455–466.

Dou, W., Zhang, X., Liu, J., Chen, J., 2015. Hiresome-ii: Towards privacy-aware cross–cloud service composition for big data applications. IEEE Trans. ParallelDistrib.Syst. 26 (2), 455–466.

Goudarzi, M., Zamani, M., Haghighat, A.T., 2017. A fast hybrid multi-site computation offloading for mobile cloud computing. J. Netw. Comput. Appl. 80, 219–231.

Guan, Z., Zhang, Y., Wu, L., Wu, J., Li, J., Ma, Y., Hu, J., 2019. Appa: an anonymous and privacy preserving data aggregation scheme for fog-enhanced iot. J. Netw. Comput. Appl. 125, 82–92.

Gutierrez-Garcia, J.O., Sim, K.M., 2010. Agent-based service composition in cloud computing. In: Grid and distributed computing, control and automation. Springer, pp. 1–10.

Hwang, S.-Y., Hsu, C.-C., Lee, C.H., 2014. Service selection for web services with probabilistic qos. IEEE Trans. Serv. Comput. 8 (3), 467–480.

Hwang, S.-Y., Hsu, C.-C., Lee, C.H., 2015. Service selection for web services with probabilistic qos. IEEE Transactions on Services Computing (1). 1–1

Hwang, S.-Y., Wang, H., Srivastava, J., Paul, R.A., 2004. A probabilistic qos model and computation framework for web services-based workflows. In: International Conference on Conceptual Modeling. Springer, pp. 596–609.

Hwang, S.-Y., Wang, H., Tang, J., Srivastava, J., 2007. A probabilistic approach to modeling and estimating the qos of web-services-based workflows. Inf. Sci. 177 (23), 5484–5503.

Jaeger, M.C., Rojec-Goldmann, G., Muhl, G., 2004. Qos aggregation for web service composition using workflow patterns. In: IEEE International Enterprise Distributed Object Computing Conference. IEEE, pp. 149–159.

Jatoth, C., Gangadharan, G., Buyya, R., 2017. Computational intelligence based qos-aware web service composition: a systematic literature review. IEEE Trans. Serv. Comput. 10 (3), 475–492.

Jatoth, C., Gangadharan, G., Fiore, U., Buyya, R., 2018. Qos-aware big service composition using mapreduce based evolutionary algorithm with guided mutation. Future Gen. Comput. Syst. 86, 1008–1018.

Jula, A., Othman, Z., Sundararajan, E., 2015. Imperialist competitive algorithm with proclus classifier for service time optimization in cloud computing service composition. Expert Syst. Appl. 42 (1), 135–145.

Kardani-Moghaddam, S., Buyya, R., Ramamohanarao, K.,. Performance anomaly detection using isolation-trees in heterogeneous workloads of web applications in computing clouds. In: Concurrency and Computation: Practice and Experience e5306.

Khanouche, M.E., Attal, F., Amirat, Y., Chibani, A., Kerkar, M., 2019. Clustering-based and qos-aware services composition algorithm for ambient intelligence. Inf. Sci. 482, 419–439.

Klein, A., Ishikawa, F., Honiden, S., 2011. Efficient heuristic approach with improved time complexity for qos-aware service composition. In: IEEE International Conference on Web Services (ICWS). IEEE, pp. 436–443.

Kubernetes,. https://kubernetes.io/, accessed: 2019-05-01.

Lartigau, J., Xu, X., Nie, L., Zhan, D., 2015. Cloud manufacturing service composition based on qos with geo-perspective transportation using an improved artificial bee colony optimisation algorithm. Int. J. Prod. Res. 53 (14), 4380–4404.

Li, J., Zhao, Y., Liu, M., Sun, H., Ma, D., 2010. An adaptive heuristic approach for distributed qos-based service composition. In: IEEE Symposium on Computers and Communications (ISCC). IEEE, pp. 687–694.

Lin, J., Niu, J., Li, H., Atiquzzaman, M., 2019. A secure and efficient location-based service scheme for smart transportation. Future Gen. Comput. Syst. 92, 694–704.

Liu, D., Shao, Z., Yu, C., Fan, G., 2009. A heuristic qos-aware service selection approach to web service composition. In: IEEE/ACIS International Conference on Computer and Information Science. IEEE, pp. 1184–1189.

Liu, F.T., Ting, K.M., Zhou, Z.H., 2008. Isolation forest. In: 2008 Eighth IEEE International Conference on Data Mining. IEEE, pp. 413–422.

Liu, F.T., Ting, K.M., Zhou, Z.H., 2012. Isolation-based anomaly detection. ACM Trans. Knowl. Discov. Data (TKDD) 6 (1), 3.

Liu, Z.-Z., Chu, D.-H., Jia, Z.-P., Shen, J.-Q., Wang, L., 2016. Two-stage approach for reliable dynamic web service composition. Knowl.-Based Syst. 97, 123–143.

Luo, Y., Qi, Y., Hou, D., Shen, L., Chen, Y., Zhong, X., 2011. A novel heuristic algorithm for qos-aware end-to-end service composition. Comput. Commun. 34 (9), 1137–1144.

Mabrouk, N.B., Beauche, S., Kuznetsova, E., Georgantas, N., Issarny, V., 2009. Qos-aware service composition in dynamic service oriented environments. In: ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing. Springer, pp. 123–142.

Machine learning library in python, https://scikit-learn.org/stable/, accessed: 2019-05-01.

mesos, A.. http://mesos.apache.org/, accessed: 2019-05-01.

Moghaddam, S.K., Buyya, R., Ramamohanarao, K., 2018. Acas: An anomaly-based cause aware auto-scaling framework for clouds. J. Parallel Distrib. Comput..

Mostafa, A., Zhang, M., 2015. Multi-objective service composition in uncertain environments. IEEE Trans. Serv. Comput..

Oh, S.-C., Lee, D., Kumara, S.R., 2006. A comparative illustration of ai planning-based web services composition. ACM SIGecom Exchanges 5 (5), 1–10.

Pishvaee, M.S., Rabbani, M., Torabi, S.A., 2011. A robust optimization approach to closed-loop supply chain network design under uncertainty. Appl. Math. Model. 35 (2), 637–649.

Poss, M., 2014. Robust combinatorial optimization with variable cost uncertainty. Eur. J. Oper. Res. 237 (3), 836–845.

Ramacher, R., Mönch, L., 2012. Dynamic service selection with end-to-end constrained uncertain qos attributes. In: International Conference on Service-Oriented Computing. Springer, pp. 237–251.
Raychoudhury, V., Cao, J., Kumar, M., Zhang, D., 2013. Middleware for pervasive computing: a survey. Pervasive Mob. Comput. 9 (2), 177–200.
Rodriguez-Mier, P., Mucientes, M., Lama, M., 2011. Automatic web service composition with a heuristic-based search algorithm. In: 2011 IEEE International Conference on Web Services. IEEE, pp. 81–88.
Rosario, S., Benveniste, A., Haar, S., Jard, C., 2008. Probabilistic qos and soft contracts for transaction-based web services orchestrations. IEEE Trans. Serv. Comput. 1 (4), 187–200.
Schuller, D., Lampe, U., Eckert, J., Steinmetz, R., Schulte, S., 2012. Cost-driven optimization of complex service-based workflows for stochastic qos parameters. In: IEEE International Conference on Web Services (ICWS), IEEE, pp. 66–73.
Schuller, D., Siebenhaar, M., Hans, R., Wenge, O., Steinmetz, R., Schulte, S., 2014. Towards heuristic optimization of complex service-based workflows for stochastic qos attributes. In: IEEE International Conference on Web Services (ICWS). IEEE, pp. 361–368.
Seghir, F., Khababa, A., 2016. A hybrid approach using genetic and fruit fly optimization algorithms for qos-aware cloud service composition. J. Intell. Manuf. 1–20.
Sharifara, P., Yari, A., Kashani, M.M.R., 2014. An evolutionary algorithmic based web service composition with quality of service. In: 7th International Symposium on Telecommunications (IST), 2014. IEEE, pp. 61–65.
Standardization, I.O.f., 2016. Systems and Software Engineering: Systems and Software Quality Requirements and Evaluation (SQuaRE): Measurement of System and Software Product Quality. ISO.
swarm, D.,. https://docs.docker.com/engine/swarm/, accessed: 2019-05-01.
Tan, W., Fan, Y., Zhou, M., 2009. A petri net-based method for compatibility analysis and composition of web services in business process execution language. IEEE Trans. Autom. Sci.Eng. 6 (1), 94–106.
Tao, F., Zhao, D., Hu, Y., Zhou, Z., 2008. Resource service composition and its optimal-selection based on particle swarm optimization in manufacturing grid system. IEEE Trans. Ind. Inform. 4 (4), 315–327.
Tu, L., Xiao, F., Huang, Z., 2010. Modeling service composition using priced probabilistic process algebra. In: IEEE International Symposium on Service Oriented System Engineering. IEEE, pp. 35–38.
Urbieta, A., González-Beltrán, A., Mokhtar, S.B., Hossain, M.A., Capra, L., 2017. Adaptive and context-aware service composition for iot-based smart cities. Future Gen. Comput. Syst. 76, 262–274.
Wada, H., Suzuki, J., Yamano, Y., Oba, K., 2012. E^3: a multiobjective optimization framework for sla-aware service composition. IEEE Trans. Serv. Comput. 5 (3), 358–372.
Wang, H., Zhang, X., Yu, Q., 2016. Integrating pomdp and sarsa λ for service composition with incomplete information. In: International Conference on Service-Oriented Computing. Springer, pp. 677–684.
Wang, H., Zhou, X., Zhou, X., Liu, W., Li, W., Bouguettaya, A., 2010. Adaptive service composition based on reinforcement learning. In: International Conference on Service-Oriented Computing. Springer, pp. 92–107.
Wang, H.-C., Lee, C.-S., Ho, T.H., 2007. Combining subjective and objective qos factors for personalized web service selection. Expert Syst. Appl. 32 (2), 571–584.
Wang, S., Huang, L., Sun, L., Hsu, C.-H., Yang, F., 2017. Efficient and reliable service selection for heterogeneous distributed software systems. Future Gen. Comput. Syst.s 74, 158–167.
Wang, S., Sun, Q., Zou, H., Yang, F., 2013. Particle swarm optimization with skyline operator for fast cloud-based web service composition. Mob. Netw. Appl. 18 (1), 116–121.
Wang, S., Zheng, Z., Sun, Q., Zou, H., Yang, F., 2011. Cloud model for service selection. In: 2011 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). IEEE, pp. 666–671.
White, G., Nallur, V., Clarke, S., 2017. Quality of service approaches in iot: a systematic mapping. J. Syst. Softw. 132, 186–203.
White, G., Palade, A., Clarke, S., 2017. Qos prediction for reliable service composition in iot. In: International Conference on Service-Oriented Computing, Springer, pp. 149–160.
Wiesemann, W., Hochreiter, R., Kuhn, D., 2008. A stochastic programming approach for qos-aware service composition. In: 8th IEEE International Symposium on Cluster Computing and the Grid. IEEE, pp. 226–233.
Xu, X., 2012. From cloud computing to cloud manufacturing. Robot. Comput.-Integr. Manuf. 28 (1), 75–86.
Yang, Z., Shang, C., Liu, Q., Zhao, C., 2010. A dynamic web services composition algorithm based on the combination of ant colony algorithm and genetic algorithm. J. Comput. Inf. Syst. 6 (8), 2617–2622.
Ye, Z., Bouguettaya, A., Zhou, X., 2014. Economic model-driven cloud service composition. ACM Trans. Internet Technol. (TOIT) 14 (2–3), 20.
Yilmaz, A.E., Karagoz, P., 2014. Improved genetic algorithm based approach for qos aware web service composition. In: IEEE International Conference on Web Services (ICWS). IEEE, pp. 463–470.
Yu, Q., Bouguettaya, A., 2010. Computing service skyline from uncertain qows. IEEE Trans. Serv. Comput. 3 (1), 16–29.
Yu, Q., Chen, L., Li, B., 2015. Ant colony optimization applied to web service compositions in cloud computing. Comput. Electr. Eng. 41, 18–27.
Yu, T., Zhang, Y., Lin, K.J., 2007. Efficient algorithms for web services selection with end-to-end qos constraints. ACM Trans. Web (TWEB) 1 (1), 6.
Zeng, L., Benatallah, B., Ngu, A.H., Dumas, M., Kalagnanam, J., Chang, H., 2004. Qos-aware middleware for web services composition. IEEE Trans. Softw. Eng. 30 (5), 311–327.
Zhang, W., Chang, C.K., Feng, T., Jiang, H.y., 2010. Qos-based dynamic web service composition with ant colony optimization. In: IEEE Computer Software and Applications Conference. IEEE, pp. 493–502.
Zhang, X., Wu, T., Chen, M., Wei, T., Zhou, J., Hu, S., Buyya, R., 2019. Energy-aware virtual machine allocation for cloud with resource reservation. J. Syst. Softw. 147, 147–161.
Zhang, Y., Cui, G., Wang, Y., Guo, X., Zhao, S., 2015. An optimization algorithm for service composition based on an improved foa. Tsinghua Sci. Technol. 20 (1), 90–99.
Zhao, X., Shen, L., Peng, X., Zhao, W., 2015. Toward sla-constrained service composition: an approach based on a fuzzy linguistic preference model and an evolutionary algorithm. Inf. Sci. 316, 370–396.
Zheng, H., Yang, J., Zhao, W., 2016. Probabilistic qos aggregations for service composition. ACM Trans. Web (TWEB) 10 (2), 12.
Zheng, H., Yang, J., Zhao, W., Bouguettaya, A., 2011. Qos analysis for web service compositions based on probabilistic qos. In: International Conference on Service-Oriented Computing. Springer, pp. 47–61.
Zheng, H., Zhao, W., Yang, J., Bouguettaya, A., 2013. Qos analysis for web service compositions with complex structures. IEEE Trans. Serv. Comput. 6 (3), 373–386.
Zheng, Z., Zhang, Y., Lyu, M.R., 2010. Distributed qos evaluation for real-world web services. In: IEEE International Conference on Web Services. IEEE, pp. 83–90.
Zheng, Z., Zhang, Y., Lyu, M.R., 2014. Investigating qos of real-world web services. IEEE Trans. Serv. Comput. 7 (1), 32–39.
Zhou, J., Yao, X., 2017. A hybrid artificial bee colony algorithm for optimal selection of qos-based cloud manufacturing service composition. Int. J. Adv. Manuf.Technol. 88 (9–12), 3371–3387.

Mohammadreza Razian recently has been 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, big data analysis. Email: razian.mr@gmail.com.

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. Email: fathian@iust.ac.ir.

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=123, g-index=271, 78,000+ 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. For further information on Dr.Buyya, please visit his cyberhome: www.buyya.com.