Future Generation Computer Systems 94 (2019) 185–198

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Optimal Fitness Aware Cloud Service Composition using an Adaptive Genotypes Evolution based Genetic Algorithm

Chandrashekar Jatoth [a], G.R. Gangadharan [b,*], Rajkumar Buyya [c]

[a] Sreenidhi Institute of Science and Technology, Hyderabad, India
[b] National Institute of Technology, Tiruchirappalli, India
[c] University of Melbourne, Australia

HIGHLIGHTS

- A novel service composition method using an evolution based genetic algorithm.
- Describes service composition dealing with multiple QoS parameters.
- Considers the connectivity constraints between service composition.

ARTICLE INFO

Article history:
Received 8 April 2018
Received in revised form 29 September 2018
Accepted 14 November 2018
Available online 26 November 2018

Keywords:
Cloud service composition
Genetic algorithm
Adaptive Genotypes Evolution
Quality of Service (QoS)

ABSTRACT

With the seamless proliferation of cloud services, it becomes challenging to select and compose cloud services that satisfy the requirements of users. A service may be connected with another service(s) for satisfying a workflow/function in a service composition. Further, the service assessment based on one or two QoS parameters is not accurate enough to achieve the desired optimality in a cloud service composition. Most of the existing methods in the literature consider either a single QoS parameter or two QoS parameters for QoS-aware composition and do not consider the balancing of QoS parameters and/or the connectivity constraints between two compositions. In this paper, we present an Optimal Fitness Aware Cloud Service Composition (OFASC) using an Adaptive Genotype Evolution based Genetic Algorithm (AGEGA) dealing with multiple QoS parameters and providing the solutions that satisfy the balancing QoS parameters and connectivity constraints of service composition. Experimental results show that our approach enhances the efficiency of cloud service composition by converging quickly and obtains better composition when compared to other approaches.

1. Introduction

Cloud computing has been widely recognized as one of the most influential information technologies because of its unprecedented advantages [1]. Without the ownership of technology infrastructure, organizations can access technology-enabled services from the cloud via the Internet. The cloud offers several benefits to businesses by providing different services at reduced cost (the businesses pay only for what they use) [2]. Cloud services are proliferating nowadays considering these advantages in the adoption of cloud technologies to the organizations.

Cloud service composition aims to fulfill the functional demands of the business logic. Business processes comprised of several tasks of varying functionalities require a composition strategy of cloud services based on Quality of Service (QoS) parameters and an execution priority connectivity constraints based on the context of composition [3,4]. The resulting composite service is characterized by these QoS parameters, which helps users to assess the quality of service composition. A connectivity constraint refers to the connection between any two constraints that depend on each other [5]. Between two service compositions, a cloud service can be functionally dependent on another cloud service or several cloud services can be executed in parallel. Further, the same task of a cloud service may be reinitiated by another cloud service. There may be a dependency upon the intermediate execution of several cloud services for completing the task of a cloud service.

Nowadays, several cloud services with comparable functionality are offered to consumers at varying QoS level. As cloud services

* Corresponding author.
E-mail addresses: chandrashekar.jatoth@gmail.com (C. Jatoth), ganga@nitt.edu (G.R. Gangadharan), rbuyya@unimelb.edu.au (R. Buyya).

https://doi.org/10.1016/j.future.2018.11.022
0167-739X/

that are offered have different QoS levels for a specific task, a QoS-based ranking of these services helps the user in selecting services [6]. A service can be ranked as the best service under a particular QoS parameter but may be ranked as the worst under another QoS parameter. The performance of a service can be ranked differently according to various QoS parameters. The priorities of QoS parameters vary from one cloud service composition context to another. This requires equal consideration of all QoS parameters (called as balancing of QoS parameters) in selecting a cloud service without neglecting the influence of a primary QoS parameter in a cloud service composition.

In general, the solutions for QoS-aware cloud service composition [7–10] address either a single QoS parameter or two QoS parameters and do not consider the balancing of QoS parameters and/ or satisfying of the connectivity constraints between two compositions. In this paper, we present an Optimal Fitness Aware Cloud Service Composition (OFASC) using an Adaptive Genotype Evolution based Genetic Algorithm (AGEGA) dealing with multiple QoS parameters and providing the solutions that satisfy the balancing QoS parameters and connectivity constraints of service composition. Our approach uses the adaptive genotype progressive evolution strategy for restricting the evolution iterations. This strategy considers those offspring in the new generation that have better fitness than any of their parents. The salient contributions of this paper are as follows:

- A novel methodology to assess cloud service fitness and composition fitness by using Discrete Uniform Rank Distribution (DURD) and Discrete Uniform Service Rank Distribution (DUSRD).
- An Optimal Fitness Aware Cloud Service Composition (OFACSC) using an Adaptive Genotypes Evolution based Genetic Algorithm (AGEGA) to improve the convergence rate and computational complexity.

The rest of the paper is organized as follows. Section 2 discusses the related works in QoS aware cloud services composition. Section 3 presents our novel Optimal Fitness Aware Cloud Service Composition (OFASC) using an Adaptive Genotype Evolution based Genetic Algorithm (AGEGA) for providing the solutions that satisfy the balancing QoS parameters and connectivity constraints of service composition. Section 4 discusses the results of experiments followed by conclusions in Section 5.

2. Related work

Genetic algorithms effectively address the complications in the optimization process of composition of services. Canfora et al. [11] proposed a genetic algorithm (GA) based method to solve QoS-ware web service composition. Yilmaz et al. [12] proposed improved versions of genetic algorithms combined with Simulated Annealing and Harmony search for QoS-aware service composition. Karimi et al. [13] proposed a genetic algorithm based approach for service composition using association rules and clustering. Seghir et al. [7] developed a hybrid algorithm for QoS-aware cloud service composition, where GA is used for global search and fruit fly optimization is used for local search. Mistry et al. [14] proposed a dynamic metaheuristic optimization approach to compose an optimal set of IaaS service requests incorporating the factors of dynamic pricing and operation cost modeling of the service requests in cloud computing. Ding et al. [15] presented a GA-based approach to achieve transaction and QoS aware optimal service selection. However, if there are large number of services in the repository, then the GA based methods result in poor readability of the chromosomes and fail to predict the information related to the semantics of services.

2.1. Applications of genetic algorithms in cloud, IoT, and other related domains

Using a genetic algorithm to tackle resource scheduling and management in cloud computing has received increasing attention in recent years [16,17]. These genetic algorithms offer an NP-hard problem global solution acceptable in a time frame proportional to the number of variables. Thiruvenkadam et al. [18] proposed a hybrid genetic algorithm for scheduling and optimizing virtual machines in cloud environment. The algorithm minimizes the number of migrations when balancing the virtual machines, considering more attention to the variable loads of hosts and dynamicity of virtual machine allocations. Anastasi et al. [19] developed a cloud brokering approach using a genetic algorithm to match services and cloud resources and to aim at finding IaaS resources for satisfying the QoS requirements of cloud applications in multicloud environment. Mennes et al. [20] presented a distributed genetic algorithm for placing a service on a hybrid cloud with multiple individual smaller clouds and different capabilities. Guerrero et al. [21] proposed a genetic algorithm approach using the Non-dominated Sorting Genetic Algorithm-II, to optimize container allocation and elasticity management in cloud architectures.

Genetic algorithms are widely applied in Internet of Things (IoT) device selection and placement, thus connecting the real world and cyberspace via physical objects that embed with various types of intelligent sensors. Li et al. [22] proposed a decentralized semantics-based service discovery framework, to locate trustworthy services based on requester's demands and changing context requirements. Cuka et al. [23] proposed an integrated intelligent system for IoT device selection and placement in opportunistic networks using Fuzzy Logic and Genetic Algorithm. Several resource provisioning and optimal service placement algorithms in IoT and fog environment are being developed by various researchers considering time, cost, or energy efficiency optimization [24–26]. Li et al. [27] proposed a multi-population genetic algorithm to solve the multi-criteria objective programming model and help to keep the variety of population in the domain of IoT. Skarlat et al. [28] modeled the service placement problem for IoT applications over fog resources as an optimization problem, which explicitly considers the heterogeneity of applications and resources in terms of Quality of Service attributes. Kalsi et al. [29] developed a method for key generation based on the theory of natural selection using Genetic Algorithm with Needleman–Wunsch algorithm and a method for implementation of encryption and decryption based on DNA computing.

2.2. Achieving higher QoS in cloud, IoT, and related services

This paper intends to provide a solution to service consumers for selecting the best cloud service composition achieving the optimal QoS values, satisfying the connectivity constraints and balancing of QoS parameters. Similarly, different methods aiming for achieving excellent QoS in offering services to consumers exist in IoT and Smart Grid related fields. Wang et al. [30] proposed a dependable time series analytics framework for IoT-based smart grid, capable of providing a dependable data transforming from cyber physical systems to the target database. Cao et al. [31] proposed a QoS-aware service recommendation based on relational topic model and factorization machines for IoT Mashup applications allowing developers to compose existing Web APIs to create value-added composite Web services. Jatoth et al. [32] developed a novel MapReduce based evolutionary algorithm with guided mutation to solve QoS-aware Big service composition across virtual and physical domains. Faheem et al. [33] proposed a novel dynamic clustering based energy efficient and quality-of-service (QoS)-aware routing algorithm, inspired by the real behavior of the

bird mating optimization, to balance the data traffic and energy consumption load evenly among clusters in the smart grid. Wu et al. [34] developed a novel deviation-based neighborhood models for QoS prediction by taking advantages of crowd intelligence to obtain personalized quality of cloud/IoT services and assist users selecting appropriate services. Tomovic et al. [35] presented a software defined networking based architecture for virtualization of IoT networks, allowing virtualization of IoT resources in order to provide to users the intended service at the requested level of quality.

3. Optimal fitness aware cloud service composition using an adaptive genotypes evolution based genetic algorithm (OFASC-AGEGA)

3.1. Adaptive genotype evolution using genetic algorithm

Adaptive genotype evolution is a progressive evolution strategy [36] in which the evolution iterations are restricted to a minimal number. A progressive evolution strategy considers the new generation's offspring that have better fitness than any of their parents. The connectivity of QoS constraints and the balancing of QoS parameters for cloud service composition motivated us to define a genetic algorithm approach based on adaptive genotype evolution that discovers a set of best-fit QoS-aware cloud service compositions from among all possible compositions. It finds the fitness of the resultant compositions of the GA evolution iterations as described in Sections 3.2 and 3.1.1, and 3.3. Further compositions are ranked according to their composition fitness value $cf\upsilon$, and the top n compositions are selected from the sorted list. These selected compositions are sorted in descending order of their associability fitness value $af\upsilon$ and are used in the same sort to finalize services for further evolution or to recommend best compositions after completion of the evolution.

3.1.1. Evaluating discrete uniform rank distribution as service fitness

Let P be a set of QoS parameters $\{p_1, p_2, \ldots, p_q\}$ of each service in the given cloud service set $ST_i = \{s_{i1}, \ldots, s_{ix}\}$. Let the QoS parameter P_{opt} be the anchor parameter for ranking the cloud services. The QoS parameters are classified into positive parameters (having higher values for representing higher utility level (e.g., availability and reliability)) and negative parameters (having higher values for representing lower utility level (e.g., cost and response time) [37]. These positive and negative parameter values are normalized as follows:

$$m_k = \begin{cases} \dfrac{p_k - \min_k p_k}{\max_k p_k - \min_k p_k} & \text{for positive parameter} \\ \dfrac{\max_k p_k - p_k}{\max_k p_k - \min_k p_k} & \text{for negative parameter} \end{cases} \tag{3.1}$$

Further, the cloud services related to a specific task are ranked in descending order of the values of these QoS parameters. Each cloud service is ranked in different positions under different QoS parameters, which are used for measuring the discrete uniform rank distribution (DURD). For each cloud service ST_i of task T, the discrete uniform distribution scope [38] of the QoS parameter ranks can be measured using the following three parameters:

- Mean $\mu R(Ts_i)$ of QoS parameter ranks for the cloud service.
- Variance $\upsilon QV(Ts_i)$ of correlations between discrete parts of the QoS parameter rank set for cloud service s_i of task T. The number of discrete partitions of the rank set is q.
- Deviation $\sigma QV(Ts_i)$ of correlations between discrete parts of the QoS parameter rank set for cloud service s_i of a task T.

Measurement of mean of QoS parameter ranks for a service. Let a rank set for a cloud service $\left[p_j \in ST_i \wedge ST_i \in CS\right]$ be $r(s_i) = [r(p_1), r(p_2), \ldots, r(p_n)]$. The service fitness value ($sf\upsilon$) of this cloud service s_i is explored in the following step. The mean $\mu R(ts_i)$ of the ranks of all QoS parameters P is measured as follows:

$$\mu R(Ts_i) = \left[\frac{\sum_{i=1}^{|P|} r(p_j \in P)}{|P|}\right] \tag{3.2}$$

Measurement of correlations between discrete partition of the QoS parameter rank set. We partition the rank set $r(s_i)$ into q partitions, which can be referred to as $Q1_{r(s_i)}, Q2_{r(s_i)}, Q3_{r(s_i)}, \ldots, Qq_{r(s_i)}$. Then, we find the correlation of each partition with all other partitions and the covariance of these resultant correlations as follows:

Algorithm 1: Correlations for each partition

```
υ(Q_{s_i}) ← φ; {initializing a vector}
for each k:1,2,.....,q do
    ρ(Q_k) ← φ ; {initializing a vector}
    for each j:1,2,.....,q do
        ρ(Q_k) ← cor(Qk_{r(s_i)}, Qj_{r(s_i)}) ;
    end
    υ(Q_{s_i}) ← covυ(ρQ_k) ; {adding covariance of the values of vector
    ρ(Q_k) to υ(Q_{s_i}).}
end
```

In Algorithm 1, $\rho(Q_k)$ is a vector of size q, and each entry of this vector is the correlation between partition $Qk_{r(s_i)}$ and one of the partitions $Qj_{r(s_i)}$. $\upsilon(Q_{s_i})$ is a vector of size q, and each entry of this vector is the covariance of the vector $\rho(Q_k)$ entries (correlations observed between a part $Qk_{r(s_i)}$ and all parts of $r(s_i)$).

Further, we find the mean of all entries in the vector $\upsilon(Q_{Ts_i})$ for each service s_i of a specific task T as follows:

$$\upsilon QV(Ts_i) = \frac{\sum_{j=1}^{|\upsilon(Q_{Ts_i})|} \upsilon(Q_{Ts_i})_j}{|\upsilon(Q_{Ts_i})|} \tag{3.3}$$

where $\upsilon(Q_{Ts_i})$ is a vector of size q and each entry of this vector is the covariance of $\rho(Q_k)$. $\rho(Q_k)$ is a vector of correlations observed between a part $Qk_{rs(Ts_i)}$ and all parts of the $rs(Ts_i)$ entries.

3.1.2. Measurement of deviation of correlations between discrete partitions of the QoS parameters rank set

We find the deviation $\sigma QV(Ts_i)$ of service s_i of a task T as follows:

$$\sigma QV(Ts_i) = \sqrt{\frac{\sum_{k=1}^{|\upsilon(Q_{Ts_i})|}((\mu(\upsilon(Q_{Ts_i})) - \upsilon(Q_{Ts_i}))_k)^2}{|\upsilon(Q_{Ts_i})|}} \tag{3.4}$$

where $\upsilon(Q_{Ts_i})$ is a vector of size q, each entry of which is the covariance of the vector $\rho(Q_k)$ entries (the correlations observed between a part $Qk_{rs_{Ts_i}}$ and all parts of rs_{Ts_i}), and $\mu QV(Ts_i)$ is the mean of the vector $\upsilon(Q_{Ts_i})$. The pseudocode representation of DURD is shown as Algorithm 2.

3.1.3. Service selection by DURD

We apply the processes explored in this section 3.1.1 for all the available cloud services for a task T. We sort all the cloud services by their μR values in ascending order and discard the cloud services with a μR value greater than the given threshold (generally the average of the μR values for all services). Then, we sort the remaining cloud services in ascending order of their υQV values and discard cloud services with an υQV value greater than the given threshold (generally the average of the υQV values for

Algorithm 2: Discrete Uniform Rank Distribution (DURD)

Ranks of cloud service: a set that contains ranks for each QoS parameter of a cloud service.

Steps applied to all possible cloud services to a task:

for *each task* $(T_k, \forall k \in 0, 1, 2, \ldots, m)$ **do**
  for *each service* $(s_i, \forall i \in 0, 1, 2, \ldots, n)$ **do**
    $\mu R(s_i) \leftarrow \frac{\sum_{j=1}^{|r(s_i)|}(r(p_j) \in r(s_i))}{|r(s_i)|}$; {Mean of Ranks}
    $\mu R_{T_k} \leftarrow \mu R(s_i)$; {Move $\mu R(s_i)$ as an entry to vector μR_{T_k}}
    Partition the ranks set $r(T_k)$ into q sets ;
    for *each partition* $(Q_j \forall j \in 1, 2, 3, \ldots, q)$ **do**
      for *each partition* $(Q_l \forall l \in 1, 2, 3, \ldots, q)$ **do**
        $\upsilon(Q_{s_l}) \leftarrow$ *Correlation*(Q_j, Q_l) ; {correlation of ranks}
        $\rho_{s_l}(Q_l) \leftarrow \upsilon(Q_{s_l})$;
      end
      $\upsilon(Q_{s_l}) \leftarrow$ *variance*$(\rho_{s_l}(Q_l))$; {Find variance $\rho_{s_l}(Q_l)$}
    end
    $\upsilon VQ(T_k) \leftarrow$ *mean* $(\upsilon(Q_{s_l}))$; {Find mean $\upsilon(Q_{T_k s_i})$}
    $\sigma Q(t_k) \leftarrow$ *deviation* $(\upsilon(Q_{s_j}))$; {Find deviation $\upsilon(Q_{s_j})$}
  end
  Sort (μR_{T_k}); {Sort the services Ascending Order} **for** *each service* $s_i, \forall i \in 0, 1, 2, \ldots, n$ **do**
    if $(\mu R_(s_i) <$ *Threshold*$)$ **then**
      service_variance.add(s_i);
    end
  end
  Sort (*service_deviation*); {Sort service deviation} **for** *each service* $s_d \forall d \in 0, 1, 2, \ldots, m$ **do**
    if $(\sigma Q(s_d) <$ *Threshold*$)$ **then**
      select_services.add(s_d);
    end
  end
  best_rank (select_ services); {Select best services based on their average rank of QoS parameters}.
end

all cloud services). Afterwards, the remaining cloud services are sorted in ascending order of their σQV values, and cloud services with σQV values greater than the given threshold (generally the average of the σQV values for all cloud services) will be discarded. The cloud services surviving this DURD filtering process are those that exhibit the best QoS fitness. These cloud services are then sorted in ascending order of their μR values. An illustration of the DURD model is presented in Appendix A.

3.2. Evaluation of associability fitness of the composition

Let us consider a set of possible service compositions C be $\{c_1, c_2, c_3, \ldots, c_n\}$. The connection associability score (cas) of composition c_i refers to the number of connections having associability against the number of connections requiring associability (see [5]). The associability fitness value $af\upsilon(c_i)$ is determined as follows:

$$af\upsilon(c_i) = \frac{1}{1 + \mu(cas(c_i), AC)} \tag{3.5}$$

where $\mu(cas(c_i), AC)$ is the mean of $cas(c_i)$ and AC. AC is the associability count, representing the total number of edges between tasks that require associability in the target application of the service composition. $af\upsilon(c_i)$ is the associability fitness of composition c_i, measured by normalizing the standard deviation observed from $cas(c_i)$ and AC to $0 \leq af\upsilon(c_i) \leq 1$. The resultant standard deviation is increased by 1 in order to avoid a divide-by-zero error.

3.3. Evaluation of overall fitness of the composition

The overall composition fitness of a composition is measured by finding the service fitness values, the fitness score of the composition, and the composition fitness values and associability fitness value.

3.3.1. Finding service fitness

The fitness values of the cloud services involved in a composition is measured as follows:

If

$$\begin{cases} \mu R(T_j s_k) \leq \mu(\sum_{i=1}^{n} \mu R(T_j s_i)) \text{and} \\ \mu QV(T_j s_k) \leq \mu(\sum_{i=1}^{n} \mu QV(T_j s_i)) \text{and} \\ \sigma QV(T_j s_k) \leq \mu(\sum_{i=1}^{n} \sigma QV(T_j s_i)) \end{cases} \tag{3.6}$$

then $T_j s_k$ is considered to be fit.

3.3.2. Finding fitness score of the composition

In a given composition c such that $c = T_i S_j \exists j \in [1, 2, 3, \ldots] \forall i \in [1, 2, 3, \ldots, |c|]$ (c being a composition of cloud services related to all tasks in a sequence), if a cloud service of the composition is fit, then the fitness score of the composition $fc(c)$ will be incremented by 1. Then we select all compositions having a fitness score less than a given threshold as an input for assessing the discrete uniform service rank distribution.

3.3.3. Sorting by composition fitness and associability fitness values

The resultant compositions are sorted in descending order of their composition fitness values $cf\upsilon$, and the one having the maximum best service compositions ratio ($mbscr$) is selected from the best compositions $mbsc$ among the resultant ordered compositions. Then, the compositions $mbsc$ are sorted in descending order of their associability fitness $af\upsilon$, and the one having the maximum best associability compositions ratio ($mbacr$) is selected from the ordered $mbsc$.

3.4. Evaluating discrete uniform service rank distribution as composition fitness

This process depicts the uniform distribution of the ranks of cloud services assessed under DURD for each service related to the different QoS parameter considered. The process is similar to DURD. DURD considers the ranks assigned to the QoS parameters as the inputs to assess the service level fitness, whereas the discrete uniform service rank distribution (DUSRD) considers the ranks assigned to the cloud services to assess the composition fitness.

In order to assess the DUSRD of a cloud service S_i of task t, we measure the following three parameters:

- Mean value $\mu R(c_i)$ of the ranks of the composition.
- Correlations for each partition in a composition.
- Deviation of correlations in each composition.

3.4.1. Measuring mean value of ranks of the composition

Let the rank set of a composition $c = T_i s_j \exists j \in [1, 2, 3, \ldots, |T_i s|] \forall i \in [1, 2, 3, \ldots, |c|]$ (c being a composition of services related to all tasks in a sequence) be $rs(c_i) = [r(T_1 s_j \exists j \in [1, 2, 3, \ldots, |T_1 s|]), r(T_2 s_j \exists j \in [1, 2, 3, \ldots, |T_2 s|]), \ldots, r(T_m s_j \exists j \in [1, 2, 3, \ldots, |T_m s|])]$. Then, we find the mean of the ranks of all cloud services (μR) of composition c_i, defined as follows:

$$\mu R(c_i) = \left(\frac{\sum_{k=1}^{|rs(c_i)|} r(T_k s_j \exists j \in [1, 2, 3, \ldots, |T_k s|])}{|rs(s_i)|} \right) \tag{3.7}$$

Algorithm 3: Correlation for each partition of composition (c_i)

$\upsilon(Q_{c_i}) \leftarrow \phi$; {initializing a vector}
for *each k: 1, 2, …, q* **do**
  $\rho(Q_{(c_i)_k}) \leftarrow \phi$; {initializing a vector}
  for *each j: 1, 2, …, q* **do**
    $\rho(Q_{(c_i)_k}) \leftarrow cor(Qk_{rs(s_i)}, Qj_{rs(s_i)})$;
  end
  $\upsilon(Q_{c_i}) \leftarrow cov(\rho Q_k)$; {adding covariance of the values of vector $\rho(Q_k)$ to $\upsilon(Q_{s_i})$}
end

3.4.2. Measuring correlations for each partition in a composition

We partition the rank set $rs(c_i)$ into q partitions, which can be referred to as $Q1_{rs(c_i)}$, $Q2_{rs(c_i)}$, $Q3_{rs(c_i)}$, …, $Qq_{rs(c_i)}$. Then, we find the correlation of each part with all other parts and the covariance of these resultant correlations as given in Algorithm 3.

$\rho(Q_{(c_i)_k})$ is a vector of size q, and each entry of this vector is the correlation between partition $Qk_{rs(c_i)}$ and one of the partitions $Qj_{rs(c_i)}$ of $rs(c_i)$. $\upsilon(Q_{c_i})$ is a vector of size q, and each entry of this vector is the covariance of the vector $\rho(Q_{(c_i)k})$ entries (correlations observed between a part $Qk_{rs(c_i)}$ and all parts of $rs(c_i)$).

Further, we find the mean of all entries of the vector $\upsilon(Q_{c_i})$ for each composition c_i as follows:

$$\upsilon QV(c_i) = \frac{\sum_{j=1}^{|\upsilon(Q_{c_i})|} \upsilon(Q_{c_i})_j}{|\upsilon(Q_{c_i})|} \tag{3.8}$$

where $\upsilon(Q_{c_i})$ is a vector of size q and each entry of this vector is the covariance of $\rho(Q_{(c_i)_k})$. $\rho(Q_{(c_i)_k})$ is a vector of correlations observed between a part $Qk_{rs(c_i)}$ and all parts of the $rs(c_i)$ entries.

3.4.3. Measuring deviation of correlations in each composition

We measure the deviation $\sigma QV_{(c_i)}$ of composition c_i as follows:

$$\sigma QV_{(c_i)} = \sqrt{\frac{\sum_{k=1}^{|\upsilon(Q_{c_i})|}(\mu(\upsilon(Q_{c_i})) - \upsilon(Q_{c_i})_k)^2}{|\upsilon(Q_{c_i})|}} \tag{3.9}$$

The pseudocode representation of DUSRD is shown as Algorithm 4.

3.4.4. Compositions selection by DUSRD

We apply the processes explored in this section 3.4 for all compositions available as genotypes. We initially sort all compositions by their μR values in ascending order and discard the compositions with μR greater than the given threshold (generally the average of the μR values for all cloud services). Then, we sort the remaining compositions by their υQV values in ascending order and discard compositions with an υQV value greater than the given threshold (generally the average for all compositions). Afterwards, the remaining compositions are sorted by their σQV values in ascending order, and compositions with σQV values greater than the given threshold (generally the average of the σQV values for all compositions) will be discarded. The compositions surviving this filtering process for assessing DUSRD are the set of new genotypes for further GA evolution. An illustrated example of the DUSRD model is presented in Appendix B.

Algorithm 4: Discrete Uniform Service Rank Distribution (DUSRD)

Ranks of service: a set that contains ranks for each service of a composition.
Steps applied to all possible tasks to a composition:
Generate_composition(*set of tasks*); {Generate possible composition}
for *each task*(T_i) **do**
  selected_services ← *DURD*(s_i);
end
for *each composition* ($c_i, \forall i \in 0, 1, 2, \ldots, m$) **do**
  for *each task* ($T_j, \forall j \in 0, 1, 2, \ldots, n$) **do**
    $\mu R(T_i) \leftarrow \frac{\sum_{j=1}^{|rs(c_i)|}(r(T_j) \exists r(r_j) \in rs(c_i))}{|rs(c_i)|}$; {Mean of Ranks}
    $\mu R_{c_i} \leftarrow \mu R(T_i)$; {Move $\mu R(T_i)$ as an entry to vector μR_{c_i}}
    Partition the ranks set $rs(c_i)$ into q sets;
    for *each partition* ($Q_i \forall i \in 1, 2, 3, \ldots, q$) **do**
      for *each partition* ($Q_j \forall j \in 1, 2, 3, \ldots, q$) **do**
        $\upsilon(Q_{T_i}) \leftarrow$ *Correlation*(Q_j, Q_l) ; {Correlation of ranks}
        $\rho_{T_i}(Q_i) \leftarrow \upsilon(Q_{T_i})$;
      end
      $\upsilon(Q_{T_i}) \leftarrow$ *variance*($\rho_{T_i}(Q_l)$); {Find variance $\rho_{T_i}(Q_l)$}
    end
    $\upsilon VQ(c_i) \leftarrow$ *mean* ($\upsilon(Q_{T_i})$); {Find mean $\upsilon(Q_{T_k s_i})$}
    $\sigma Q(c_i) \leftarrow$ *deviation* ($\upsilon(Q_{T_i})$); {Find deviation $\upsilon(Q_{T_i})$}
  end
  $\upsilon VQ(c_i) \leftarrow$ *mean* ($\upsilon(Q_{T_i})$); {Find mean $\upsilon(Q_{T_k s_i})$}
  $\sigma Q(c_i) \leftarrow$ *deviation* ($\upsilon(Q_{T_i})$); {Find deviation $\upsilon(Q_{T_i})$}
end
$\mu R_C \leftarrow \mu R(c_i)$; $\upsilon VQ(C) \leftarrow \upsilon VQ(c_i)$;
$\sigma Q(C) \leftarrow \sigma Q(c_i)$;
Sort($\mu R(C)$); {Sort the services Ascending Order}
for *each composition* $c_i, \forall i \in 0, 1, 2, \ldots |C|$ **do**
  if *($\mu R(c_i) <$ Threshold)* **then**
    composition_variance.add(c_i);
  end
end
Sort (*composition_variance*); {Sort composition variance}
for *each composition* $c_i \forall i \in 0, 1, 2, \ldots |composition\_variance|$ **do**
  if *($\upsilon VQ(c_i) <$ Threshold)* **then**
    composition_deviation.add(c_i);
  end
end
Sort(*composition_deviation*); {Sort composition deviation}
for *each composition* $s_d \forall d \in 0, 1, 2, \ldots composition\_deviation$ **do**
  if *($\sigma Q(c_i) <$ Threshold)* **then**
    select_compositions.add(c_i);
  end
end
Best_rank (select_composition);

3.5. Optimal fitness-aware service composition using AGEGA

The inputs for AGEGA are as follows:

- A set of tasks $CT = \{T_1, T_2, T_3, \ldots, T_{|T|}\}$ involved in the given application
- A set of available candidate cloud services $CS = \{ST_1 = \{s_{1\,1}, s_{1\,2}, \ldots, s_{1\,i}\}, ST_2 = \{s_{2\,1}, s_{2\,2}, \ldots, s_{2\,j}\}, \ldots, ST_p = \{s_{p\,1}, s_{p\,2}, \ldots, s_{p\,m}\}\}$, where i, j, and m denote the number of candidate cloud services in ST_1, ST_2, and ST_p, respectively
- Set of all possible compositions: $C = \{c_1, c_2, c_3, \ldots, c_{|c|}\}$
- Associability count AC observed
- A set of task sets demanding associability $ST' = \{\{ST_i, ST_j, ST_k, \ldots\}, \{ST_x, ST_y, ST_z, \ldots\}, \ldots\}$
- Maximum evolution iteration threshold, *met*

Following are the preprocessing steps for AGEGA:

- Perform the service level fitness calculation (see Section 3.1.1).

- Perform the composition associability fitness calculation (see Section 3.2).
- Perform the composition fitness calculation (see Section 3.3).
- Sort the resultant compositions and select the best compositions by composition fitness value and composition associability fitness value (see Section 3.4.4).

The proposed OFASC using Adaptive Genotypes Evolution-Based GA is as follows (see Algorithms 5 and 6):

1. Find the common services from given any two compositions such that a common service must not be the crossover point if it currently has a desired associability in parent compositions and does not retain that associability in the resultant composition.
2. For all the common services found: (i) Divide the first and second compositions in the pair having a common service into two parts, denoted as lp_1 and rp_1 from the first composition and lp_2 and rp_2 from the second composition. Then, build the two new compositions by connecting lp_1 and rp_2, which forms the first one, and connecting lp_2 and rp_1, which forms the second. (ii) Consider any of the newly formed compositions to be best composition if its fitness is greater than the fitness of any of the parent compositions.
3. If any new compositions are formed in step 2, then add all of them to the composition set C.
4. Sort the composition set C and select the best compositions by composition and associability fitness values.
5. Continue the adaptive genotype evolution process on C up to the given maximum evolution iteration threshold *met*.
6. Verify the DUSRD (see Section 3.4) of each composition among the resultant compositions.
7. Sort the resultant compositions and select the best compositions through fitness score, associability score, and DUSRD (see Sections 3.3 and 3.4).

Algorithm 5: The pseudo code representation of the OFASC

```
pc ← true; {Best composition process}
met ← 0; {Maximum evolution threshold}
while pc do
    nC ← φ;
    for each composition {c_i ∀c_i ∈ C} find crossover points do
        for each composition {c_j ∃(c_j ∈ C ∧ j ≠ i)} do
            nC ← AGEGA(c_i, c_j);
        end
    end
    C̄ ← C; {Clone the C as C̄}
    C̄ ← nC; {Adding new compositions to C}
    DUSRD(C̄); {see section ??}
    if C ≥ C̄ then
        pc ← false; {Equaling by definition}
    end
end
```

4. Performance evaluation and results

To evaluate the scalability and effectiveness of the proposed approach, we conducted experiments on a personal computer with an Intel (R) core (TM) i5 2.60-GHz processor and 8 GB of memory, running Windows 8.1. There are some benchmark data sets for web services, [41] and [42], that are not suitable in the context of OFASC-AGEGA because these benchmark data sets describe only the values for response time and throughput. For our proposed approach, we used a synthetic data set comprising the QoS parameters such as availability, security, accessibility, cost, integrity, throughput, response time, and reliability. This synthetic data set

Algorithm 6: Adaptive Genotypes Evolutions based Genetic Algorithm (AGEGA)

```
AGEGA(c_i, c_j)
BEGIN
c_ij ← φ; {a set of compositions formed from crossover of c_i, c_j}
for each service s_k ∀s_k ∈ c_i do
    cop ← φ; {set of crossover points}
    for each service s_l ∀s_l ∈ c_j do
        if ((s_k ≅ s_l)&(a(e_(s_k−1→s_k)) = 0&a(e_(s_k→s_k+1)) =
        0)&(a(e_(s_l−1→s_l)) = 0&a(e_(s_l→s_l+1)) = 0)) then
            {(a(e_(s_k−1→s_k)) = 0&a(e_(s_k→s_k+1)) = 0) indicates that the
            s_k is not having associability with predecessor and
            successor services in composition}
            {(a(e_(s_l−1→s_l)) = 0&a(e_(s_l→s_l+1)) = 0) indicates that the s_l
            is not having associability with predecessor and
            successor services in composition}
            cop ← s_k;
        end
    end
    for each cp ∀cp ∈ cop do
        cross c_i at cross point cp and form ←c_i and →c_i ;
        cross c_j at cross point cp and form ←c_j and →c_j ;
        divide c_i in to two parts at cross point cp, and the left part
        label as ←c_i and right part as →c_i ;
        divide c_j in to two parts at cross point cp, and the left part
        label as ←c_j and right part as →c_j ;
        Form composition c_k as
        c_k ← φ; c_k ← ←c_i ; c_k ← cp; c_k ← →c_j ; c_ij ← c_k;
        Form composition r_k as
        c_l ← φ; c_l ← ←c_j ; c_l ← cp; c_l ← →c_i ; c_ij ← c_l;
    end
    for each new composition c ∀c ∈ c_ij do
        if (fs(c) < fs(c_i)&fs(c) < fs(c_j)) then
            delete c form c_ij;
        end
    end
    return c_ij;
end
END
```

comprises 2250 web services. We used the QoS data set described in [43] to assign the values to the services. The QoS data set has various QoS parameters and more than 2500 web services. From this data set, we randomly selected 2250 web services and their corresponding QoS parameter values. The missing QoS parameters and their corresponding values were randomly generated. Different combinations of tasks in the range of 70 (sparse) to 250 (dense) were used to conduct experiments in order to assess the performance. Each task comprises 7 to 16 services. We considered 450 different service composition scenarios. The number of tasks given for each service composition was between 7 and 25. We tested this data set using the Shapiro–Wilk test, and the results revealed that the given data set was normally distributed [44]. Several researches adopted synthetic data sets to validate their methods in service composition [45–49]. The empirical analysis of our proposed approach was implemented in Java. Statistical measures such as discrete uniform rank distribution and discrete uniform service rank distribution of the discovered compositions were performed using the R language. In our evaluations, several parameters were fixed for all approaches, controlling the execution time and specifying the population size. We set the population size to 100. Each experiment was executed continuously 30 times, and the mean of each run was duly noted due to the stochastic nature of algorithms. Table 1 presents the list of approaches (with name,

Table 1
Compared approaches with parameter settings.

Approach	Abbrevations	Parameter Setting
Genetic Algorithm [11]	GA	Crossover = 0.8, mutation = 0.001 and random selection approach is applied
Orthogonal Genetic Algorithm [39]	OGA	Crossover $P_c = 0.1$, mutation $P_m = 0.02$ and F = 4
Adaptive Genetic programming [40]	AGP	population size = 30, $g_{max} = 500$, $k_c = 2.5$, $k_m = 2.0$, Crossover $P_c = 0.9$, mutation $P_m = 0.01$
Genetic algorithm with simulated annealing / Genetic algorithm with Harmony Search [12]	GASA / GAHS	Crossover = 0.8, mutation = 0.01, SA application = 0.03
Transactional Genetic Algorithm [15]	TGA	population = 50, Crossover = 0.9, mutation = 0.1, η (incentive factor = 0.5) and ρ (penality factor = 0.3)

Table 2
Average fitness values observed for our proposed approach and other compared approaches.

		Mean of the fitness values						
Test Cases.	Number of Abstract Services	GA	TGA	GASA	GAHS	AGP	OGA	OFASC-AGEGA
65	10	0.327	0.3261	0.3333	0.323	0.3241	0.3875	**0.4138**
54		0.3166	0.3156	0.3159	0.3071	0.3229	0.3357	**0.4299**
49	30	0.2681	0.3083	0.3348	0.2369	0.2912	0.3132	**0.4108**
61		0.3297	0.2901	0.3451	0.2053	0.2874	0.3128	**0.4584**
65	50	0.2791	0.3049	0.3150	0.3083	0.3181	0.3351	**0.4330**
54		0.3138	0.3174	0.329	0.3138	0.3218	0.3315	**0.4603**
49	70	0.2873	0.2870	0.3343	0.3239	0.3189	0.3437	**0.4679**
61		0.3027	0.2907	0.3234	0.3358	0.3298	0.3473	**0.4897**
65	90	0.3071	0.3116	0.3435	0.3268	0.3014	0.4757	**0.4983**
54		0.327	0.3261	0.3333	0.323	0.3241	0.5075	**0.5138**
49	100	0.3235	0.4061	0.3781	0.3643	0.3591	0.5248	**0.5456**
65		0.327	0.4261	0.3833	0.3834	0.3739	0.5784	**0.6238**

abbreviations, and parameter settings) that we compared with our approach.

We evaluated the performance of our proposed algorithm in the following ways: (i) Evaluated the average fitness values by varying the number of abstract services. (ii) Evaluated the process completion time and computational complexity for a varying number of abstract services.

We compare our propose approach OFASC-AGEGA with other approaches such as GA [50], TGA [15], GASA [12], GAHS [12], AGP [40], and OGA [39]. Table 2 illustrates the average fitness values for different test cases with 10, 30, 50, 70, 90, and 100 abstract services.

The completion time of our proposed approach is presented in Fig. 1. Based on Fig. 1, we observe that our proposed approach takes less time to complete than other contemporary approaches. The completion times for TGA, GAHS, GASA, GA, AGP, and OGA to obtain the best solutions were 317.559, 221.898, 210.244, 159.920, 95.795, and 81.640 s respectively, whereas the completion time for the OFASC-AGEGA evolution iterations was 48.086 s. The computational time taken by the adaptive genotype evolution strategy is noticeably minimal (see Fig. 2) compared to other approaches [39, 50,40,12,15]. In GA and TGA, with each iteration, a new population is generated, and the individual fitnesses are evaluated (based on fixed and predetermined crossover and unguided mutation). These processes cause GA and TGA to converge slowly and to become easily stuck in local maxima. In GAHS and GASA, with each iteration, a updated population is generated, and the individual fitnesses are assessed (based on predetermined crossover and unguided mutation). It takes more time because of the higher number of parameter turnings and the single-point crossover. In the OGA algorithm, the orthogonal method is used to produce the new population, and the crossover strategy is employed on the two parents, thus causing the process to consume more time.

The evolution complexity obtained in our proposed model is linear (see Fig. 2). As observed from Figs. 1 and 2, our approach is more effective, scalable, and robust than other approaches.

The time complexity of OFASC-AGEGA is $O(n)$, whereas the observed complexity of the other models ranges as $O(n^2)$ or $O(n \log(n))$. The fitness distribution (DURD) of the services involved in the 10

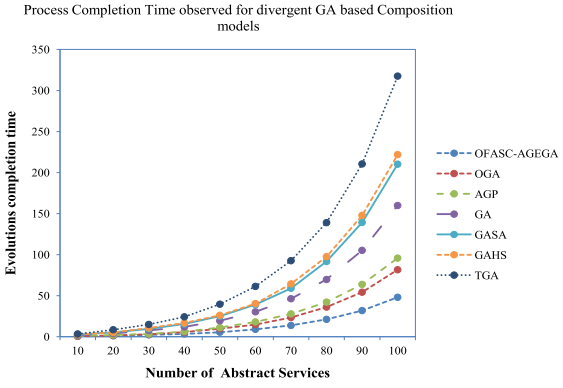

Fig. 1. Process completion time observed for OFASC-AGEGA with other approaches.

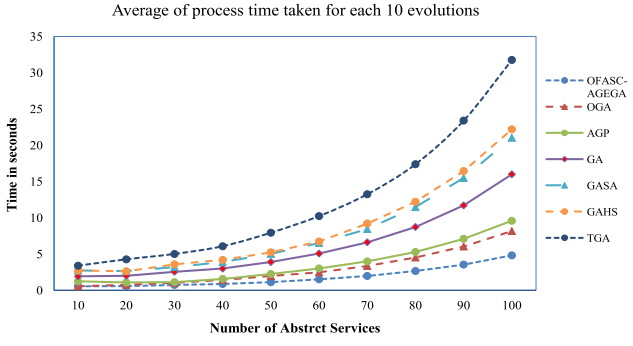

Fig. 2. Computational complexity observed for OFASC-AGEGA with other approaches.

best compositions recommended by OFASC-AGEGA is optimal, in contrast to other approaches (see Fig. 4).

The accuracy of the composition recommendations of OFASC-AGEGA is assessed using probability theory [51]. The success ratio of a composition is measured as the average success ratio of the

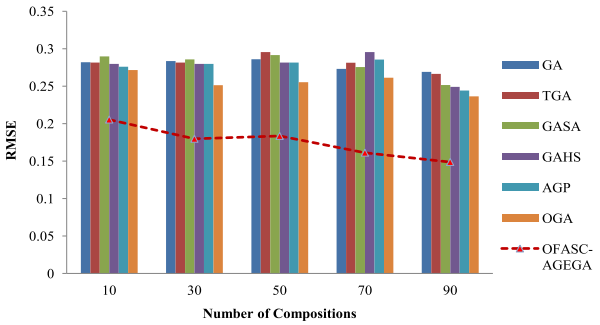

Fig. 3. Average RMSE of the recommended compositions.

services included in that composition. The assessment of the accuracy of a composition is defined as follows:

$$sup_S(S_j) = \frac{\sum_{i=1}^{|C_S|}(1 \exists S_j \in (c_i \forall c_i \in C_S))}{|C_S|} \tag{4.1}$$

where $sup_S(S_j)$ indicates the ratio (positive support) of service S_j in the set of compositions C_S that are labeled as S, c_i indicates the ith composition of set C_S, and $|C_S|$ indicates the number of compositions labeled as S.

$$sup_F(S_j) = \frac{\sum_{i=1}^{|C_F|}(1 \exists S_j \in (c_i \forall c_i \in C_F))}{|C_F|} \tag{4.2}$$

where $sup_F(S_j)$ indicates the ratio (negative support) of service S_j in the set of compositions C_F that are labeled as F, c_i indicates the ith composition of set C_F, and $|C_F|$ indicates the number of

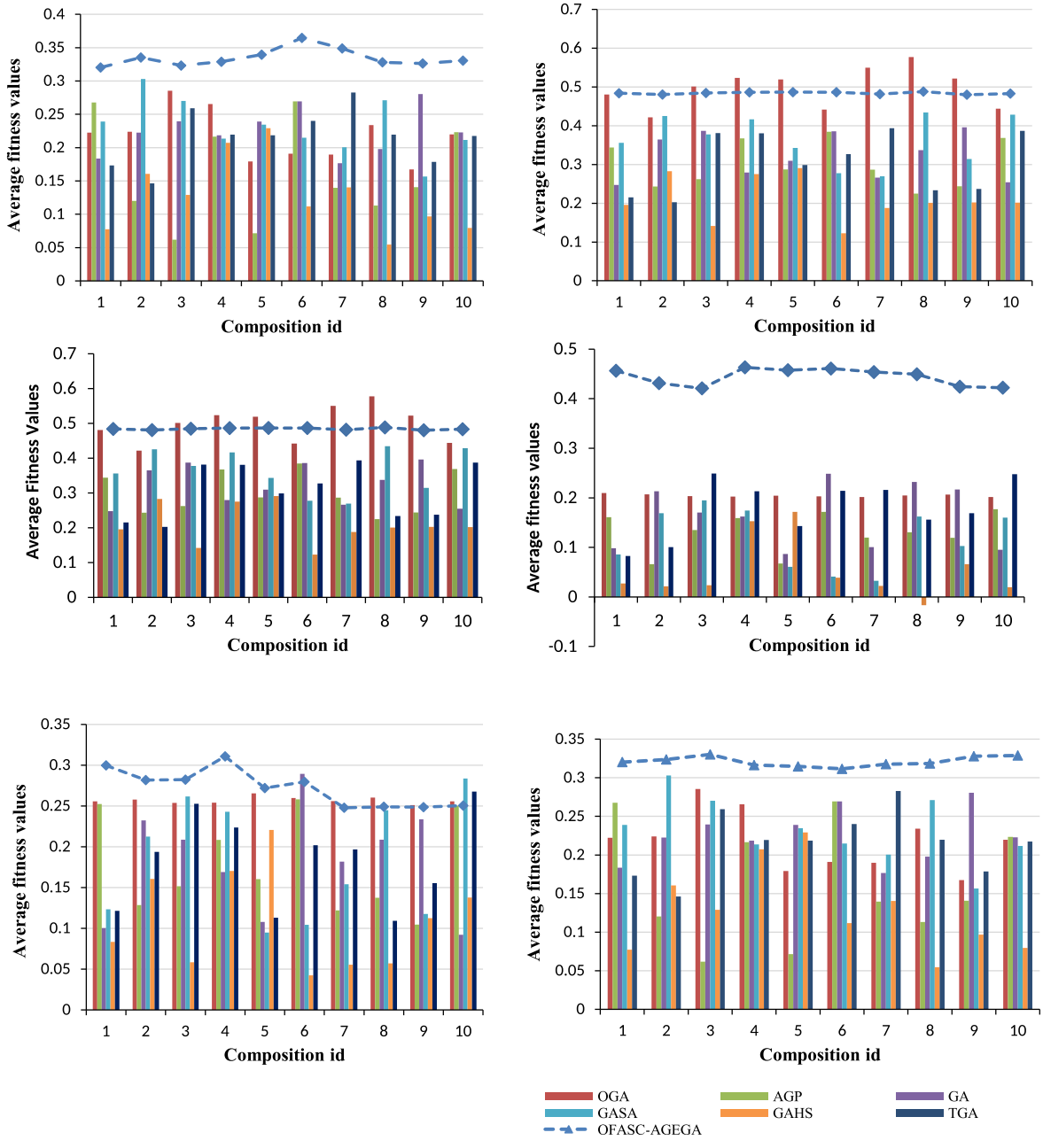

Fig. 4. Fitness distribution (DURD) of service involved in 10 resultant compositions.

Table 3
T-Test results for OFASC-AGEGA and other compared approaches.

Approaches		GA	TGA	GASA	GAHS	AGP	OGA	OFASC-AGEGA
GA		*	*	*	*	*	*	*
TGA		0.3213 (0.36103)	*	*	*	*	*	*
GASA	**T-Value/P-Value**	0.12301 (0.04559)	0.12411 (0.06998)	*	*	*	*	*
GAHS		1.6894 (0.05263)	1.7849 (0.00547)	1.6894 (0.052631)	*	*	*	*
AGP		0.1872 (0.40913)	0.1904 (0.41131)	0.1807 (0.42913)	0.20450 (0.4340)	*	*	*
OGA		1.50505 (0.07327)	1.69982 (0.05163)	1.53129 (0.06998)	2.77954 (0.00547)	1.62635 (0.059058)	*	*
OFASC-AGEGA		4.5245 (0.00008)	1.85282 (0.03869)	4.429244 (0.00015)	5.96342 (0.00001)	4.83606 (0.000039)	1.85282 (0.03869)	*

Table 4
Wilcoxon signed-rank test results for OFASC-AGEGA and other compared approaches.

Approaches		GA	TGA	GASA	GAHS	AGP	OGA	OFASC-AGEGA
GA		*	*	*	*	*	*	*
TGA		−1.1767 (0.119)	*	*	*	*	*	*
GASA	**Z-Value/ P-Value**	0.7452 (0.22663)	0.3922 (0.3482)	*	*	*	*	*
GAHS		3.0594 (0.00111)	3.0594 (0.00111)	3.0594 (0.00111)	*	*	*	*
AGP		0.6276 (0.26435)	3.0594 (0.00111)	0.2353 (0.40517)	3.0594 (0.00111)	*	*	*
OGA		2.8241 (0.0024)	3.0594 (0.00111)	2.7456 (0.00298)	3.0594 (0.00111)	2.8241 (0.00241)	*	*
OFASC-AGEGA		2.9025 (0.00187)	3.0594 (0.00111)	2.9025 (0.00187)	3.0594 (0.00111)	2.9025 (0.00187)	2.1181 (0.017)	*

compositions labeled as F.

$$SR(s_j) = sup_S(s_j) - sup_F(s_j) \tag{4.3}$$

where $SR(s_j)$ indicates the success ratio of a service s_j in composition formation. Further, the success ratio of a composition c_i can be measured as follows:

$$SR(c_i) = \frac{\sum_{j=1}^{|c_i|}(SR(s_j) \exists s_j \in c_i)}{|c_i|} \tag{4.4}$$

where $SR(c_i)$ indicates the success ratio of composition c_i, which is the average of the success ratios of services involved in composition c_i. The compositions recommended by the OFASC have a lower root mean square error (RMSE) [51,52] than the compositions recommended by [39,50,40,12,15] (see Fig. 3). The RMSE of a composition is measured as follows:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(osr - SR(c_i))^2}{n}} \tag{4.5}$$

where osr indicates the optimal success ratio (which is 1 in this context) and n represents the n best compositions recommended. Fig. 3, we observed that the RMSE values of the 90 compositions for GA, TGA, GASA, GAHS, AGP, and OGA are 0.2690, 0.2663, 0.2515, 0.2489, 0.2443, and 0.2363 respectively, whereas the RMSE value for OFASC-AGEGA is 0.1486.

We performed statistical tests (parametric and non-parametric) for our proposed OFASC-AGEGA and other contemporary algorithms to ascertain whether the results of the proposed algorithms are statistically significant. The T-test [53,54] and the Wilcoxon signed-rank test [55] were applied to evaluate whether the best mean values obtained for all algorithms have a significant difference with 58 degrees of freedom at a 1% level of significance. The results of the T-test and the Wilcoxon signed-rank test for all methods are presented in Tables 3 and 4, respectively, for 100 abstract services (with 100 candidate services). Based on Table 3, the values procured are statistically significant (all T-values are positive, and the P-values are near to zero). Based on Table 4, the values procured are statistically significant (all Z-values are obtained by positive ranks, and the P-values are near to zero).

5. Conclusions

In this article, we presented an Optimal Fitness Aware Cloud Service Composition using an Adaptive Genotypes Evolution based Genetic Algorithm. This approach assesses the fitness of candidate cloud services as well as the fitness of service compositions balancing the QoS parameters and satisfying the connectivity constraints. Our proposed method determines the service fitness and the service composition fitness by using DURD and DUSRD respectively, which assist in pruning services and compositions having non-optimal solutions and select only the best compositions, thereby gradually reducing the search space. The empirical study of our approach illustrates the better performance of our approach in comparison with the existing approaches. However, if there are very large number of candidate cloud services in the composition, our proposed method may result in an increase of search space within local optima. In future, we plan to work on more efficient dynamic services composition startegies considering multiple services connectivity constraints and multiple QoS parameters (similar to [5]). Further, we would like to develop new approaches for efficient cloud service composition on various parallel data processing platforms.

Appendix A. Discrete uniform rank distribution (DURD) based service selection

See Tables A.5–A.14.

Table A.5
Services, metrics and their ranks, mean and mean of four quarters.

	t_1s_1	t_1s_2	t_1s_3	t_1s_4	t_1s_5	t_1s_6	t_1s_7	t_1s_8	t_1s_9	t_1s_{10}
m1	6	9	3	3	9	9	2	10	10	7
m2	10	2	5	1	5	10	7	4	6	6
m3	3	3	9	2	10	4	1	10	2	2
m4	4	9	3	5	8	1	4	10	9	2
m5	1	5	6	8	4	5	6	5	6	1
m6	2	8	1	7	8	1	8	6	6	4
m7	10	2	8	3	3	9	3	7	10	3
m8	7	10	4	1	3	10	10	6	4	9
m9	4	3	5	10	4	7	8	3	2	2
m10	7	4	10	3	9	7	7	6	9	2
m11	6	4	4	6	1	1	5	1	10	6
m12	6	6	9	4	1	7	10	7	4	6
m13	1	7	7	1	3	4	9	6	9	10
m14	1	4	8	10	10	2	9	1	1	3
m15	2	9	7	4	2	9	9	8	3	3
m16	5	2	4	1	7	2	1	5	7	9
m17	9	10	3	2	2	5	6	1	10	1
m18	4	3	9	2	3	7	7	2	8	4
m19	2	1	2	1	2	7	4	3	8	8
m20	8	6	2	8	4	3	1	9	10	2

Table A.6
Q1 correlation with Q2, Q3 and Q4 and the variance of these correlations.

cor(Q1,Q1)	1	1	1	1	1	1	1	1	1	1
cor(Q1,Q2)	−0.28	−0.1	−0.24	−0.61	−0.63	0.32	0.32	−0.58	−0.26	0.73
cor(Q1,Q3)	0.34	−0.58	0.38	−0.05	0.18	0	0.79	0.05	0.4	0.32
cor(Q1,Q4)	−0.62	0.32	−0.39	−0.41	−0.58	−0.26	−0.34	−0.59	−0.22	0.16
VAR	0.51	0.45	0.41	0.51	0.59	0.3	0.35	0.56	0.35	0.15

Table A.7
Q2 correlation with Q1, Q3 and Q4 and the variance of these correlations.

cor(Q2,Q1)	0.28	0.04	0.24	0.22	−0.32	−0.31	−0.87	−0.57	−0.11	−0.06
cor(Q2,Q2)	1	1	1	1	1	1	1	1	1	1
cor(Q2,Q3)	0.03	0	0.69	0.98	−0.32	0.52	−0.37	0.65	−0.05	0.95
cor(Q2,Q4)	0.68	−0.51	−0.44	−0.42	0.74	0.75	−0.06	0	0.75	−0.02
VAR	0.18	0.4	0.39	0.47	0.48	0.32	0.62	0.48	0.31	0.35

Table A.8
Q3 correlation with Q1, Q2 and Q4 and the variance of these correlations.

cor(Q3,Q1)	0.8	−0.65	0.21	0.3	0.25	0.17	0.55	−0.77	−0.19	0.2
cor(Q3,Q2)	0.03	0	0.69	0.98	−0.32	0.52	−0.37	0.65	−0.05	0.95
cor(Q3,Q3)	1	1	1	1	1	1	1	1	1	1
cor(Q3,Q4)	0.55	0.48	−0.18	−0.31	−0.48	−0.14	0.62	0.16	−0.58	−0.11
VAR	0.17	0.49	0.27	0.39	0.45	0.24	0.34	0.59	0.46	0.3

Table A.9
Q4 correlation with Q1, Q2 and Q3 and the variance of these correlations.

cor(Q4,Q1)	0.35	−0.78	0.76	0.76	0.25	−0.63	−0.11	−0.18	−0.35	0.22
cor(Q4,Q2)	0.68	−0.51	−0.44	−0.42	0.74	0.75	−0.06	0	0.75	−0.02
cor(Q4,Q3)	0.55	0.48	−0.18	−0.31	−0.48	−0.14	0.62	0.16	−0.58	−0.11
cor(Q4,Q4)	1	1	1	1	1	1	1	1	1	1
VAR	0.08	0.7	0.49	0.53	0.43	0.58	0.29	0.27	0.62	0.25

Table A.10
Average and standard deviation of variance of all 4 quarters and average rank.

Avg-var	0.14	0.56	0.33	0.38	0.41	0.41	0.48	0.49	0.44
Stdev-var	0.05	0.14	0.15	0.17	0.08	0.16	0.19	0.16	0.13
Avg-rank	4.9	5.35	5.45	4.1	4.9	5.5	5.85	5.5	6.7

Table A.11
Services and the mean and standard deviation of each quarter variance with other quarters (Q1, Q2, Q3 and Q4) and average of service QoS metric ranks.

S_{id}	avg-var	stdev-var	avg-rank
t_1s_1	0.138339	0.050106	4.9
t_1s_2	0.562507	0.141016	5.35
t_1s_3	0.326343	0.147882	5.45
t_1s_4	0.380454	0.170691	4.1
t_1s_5	0.412157	0.08367	4.9
t_1s_6	0.411517	0.156025	5.5
t_1s_7	0.479163	0.192796	5.85
t_1s_8	0.492576	0.158607	5.5
t_1s_9	0.440441	0.13031	6.7
Average	0.392277	0.129055	5.275

Table A.12
Ordering and filtering the services by their average rank.

S_{id}	Avg-var	Stdev-var	Avg-rank
t_1s_4	0.380454	0.170691	4.1
t_1s_{10}	0.279272	0.059449	4.5
t_1s_1	0.138339	0.050106	4.9
t_1s_5	0.412157	0.08367	4.9
Above the Average Rank			
t_1s_2	0.562507	0.141016	5.35
t_1s_3	0.326343	0.147882	5.45
t_1s_6	0.411517	0.156025	5.5
t_1s_8	0.492576	0.158607	5.5
t_1s_7	0.479163	0.192796	5.85
t_1s_9	0.440441	0.13031	6.7
Average	**0.392277**	**0.129055**	**5.275**

Table A.13
Ordering and filtering the selected services by variance average.

S_{id}	Avg-var	Stdev-var	Avg-rank
t_1s_1	0.138339	0.050106	4.9
t_1s_{10}	0.279272	0.059449	4.5
t_1s_4	0.380454	0.170691	4.1
Above the average of variance			
t1s5	0.412157	0.08367	4.9

Table A.14
Services with Discrete Uniform distribution of QoS metric Ranks (t1s1, t1s10) and optimal service (t1s10)

S_{id}	Avg-var	Stdev-var	Avg-rank
t_1s_1	0.138339	0.050106	4.9
t_1s_{10}	0.279272	0.059449	4.5
Above the average of variance deviation			
t1s4	0.380454	0.170691	4.1

Table B.15
Compositions with desired fitness score and ranks of the involved services.

	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10
st_1	5	5.06	5.09	5.06	4	5.08	4.07	6.03	6.04	5.05
st_2	6.09	5.08	4.01	5.01	4.08	5.09	4.05	6.07	4.08	5.05
st_3	5.08	5.06	6.08	4.07	4.05	5.09	5.02	4.04	6.01	6.06
st_4	4.01	4.06	4.1	4.02	6	6.08	5.05	5.08	4.09	4.08
st_5	6.05	5.01	4.03	6.03	4.07	4.08	5.04	6.05	5.07	6.03
st_6	6.07	5.02	6.07	6.08	4.03	5.01	4.01	4.05	5.09	6.09
st_7	5.03	4.02	5.05	6.03	4.06	4	4.06	4.07	5.06	6
st_8	4.03	4.05	5.08	5.1	5.01	5.02	6.05	6.08	6.08	6.02
st_9	5.02	5.04	5.05	6.02	6.04	5.07	6.06	5.08	5.02	5.01
st_{10}	5.04	5.01	5.09	5.01	5.06	6.02	6.03	5.09	5.05	4.05
st_{11}	4.06	6.01	5.02	6.02	4.04	6.01	5.06	6.05	6.03	4.08
st_{12}	4.01	5.1	4.09	5.07	5.06	5.01	6.07	4.08	4.08	6.03
st_{13}	4.07	5.01	6.02	4.06	5.09	6.07	4	5.02	4.08	5.09
st_{14}	5.09	4.04	4.07	6.01	6.09	4.06	6.02	6.03	6.08	4.02

(*continued on next page*)

Table B.15 (*continued*).

	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10
st_{15}	5	5.01	6.1	6	6.08	5.02	6.06	4.06	5.07	6.01
st_{16}	5	4.03	6	5.1	4.04	4.1	5.01	5.03	5.08	4.09
st_{17}	6.02	4.03	6.01	5.06	6.04	6.09	4.1	6	5.05	4.08
st_{18}	6.08	4.07	4.1	4.07	5.04	4.06	5.06	6.07	5.06	5.07
st_{19}	5.03	4.02	6.08	6.07	6.09	5.08	5.07	4.01	6.04	4.02
st_{20}	4.03	4.03	4.07	4.01	5.07	6.02	4.07	4.07	6.03	4.02

Table B.16
Q1 correlation with Q2, Q3 and Q4 and the variance of these correlations.

cor(Q1,Q1)	1	1	1	1	1	1	1	1	1	1
cor(Q1,Q2)	−0.03	−0.46	0.28	−0.27	0.81	−0.48	1	−0.87	0.58	−0.12
cor(Q1,Q3)	−0.3	0.8	0.52	0.46	0.56	−0.4	−0.11	−0.27	0	0.61
cor(Q1,Q4)	−0.05	0.47	−0.39	−0.46	0.58	−0.34	0.19	−0.3	−0.44	0.55
VAR	**0.33**	**0.42**	**0.34**	**0.45**	**0.04**	**0.5**	**0.32**	**0.62**	**0.4**	**0.22**

Table B.17
Q2 correlation with Q1, Q3 and Q4 and the variance of these correlations.

cor(Q2,Q1)	−0.03	−0.46	0.28	−0.27	0.81	−0.48	1	−0.87	0.58	−0.12
cor(Q2,Q2)	1	1	1	1	1	1	1	1	1	1
cor(Q2,Q3)	−0.02	−0.04	0.02	0.38	0.8	−0.02	−0.12	−0.03	−0.56	−0.27
cor(Q2,Q4)	−0.45	−0.65	0.36	0.87	0.46	−0.03	0.18	0.02	−0.45	0.43
VAR	**0.38**	**0.54**	**0.17**	**0.33**	**0.05**	**0.39**	**0.33**	**0.58**	**0.59**	**0.33**

Table B.18
Q3 correlation with Q1, Q2 and Q4 and the variance of these correlations.

cor(Q3,Q1)	−0.3	0.8	0.52	0.46	0.56	−0.4	−0.11	−0.27	0	0.61
cor(Q3,Q2)	−0.02	−0.04	0.02	0.38	0.8	−0.02	−0.12	−0.03	−0.56	−0.27
cor(Q3,Q3)	1	1	1	1	1	1	1	1	1	1
cor(Q3,Q4)	−0.74	0.17	−0.93	0.46	0.66	−0.62	−0.62	−0.26	0.49	0.02
VAR	**0.54**	**0.25**	**0.68**	**0.08**	**0.04**	**0.52**	**0.47**	**0.36**	**0.45**	**0.33**

Table B.19
Q4 correlation with Q1, Q2 and Q3 and the variance of these correlations.

cor(Q4,Q1)	−0.05	0.47	−0.39	−0.46	0.58	−0.34	0.19	−0.3	−0.44	0.55
cor(Q4,Q2)	−0.45	−0.65	0.36	0.87	0.46	−0.03	0.18	0.02	−0.45	0.43
cor(Q4,Q3)	−0.74	0.17	−0.93	0.46	0.66	−0.62	−0.62	−0.26	0.49	0.02
cor(Q4,Q4)	1	1	1	1	1	1	1	1	1	1
VAR	**0.58**	**0.47**	**0.71**	**0.44**	**0.05**	**0.5**	**0.44**	**0.37**	**0.51**	**0.16**

Table B.20
Average and standard deviation of variance of all 4 quarters and average rank of the services involved in composition.

Compositions	**Avg-Var**	**Stdev-Var**	**Avg-Rank**
c_1	0.46	0.12	4.99
c_2	0.42	0.13	4.64
c_3	0.48	0.26	5.06
c_4	0.33	0.17	5.2
c_5	0.05	0.01	4.95
c_6	0.48	0.06	5.1
c_7	0.39	0.07	5
c_8	0.48	0.14	5.1
c_9	0.49	0.08	5.21
c_{10}	0.26	0.08	5
Average	**0.384**	**0.112**	**5.025**

Table B.21
Ordering and filtering the compositions by their average rank.

Compositions	Avg-Var	Stdev-Var	avg-rank
c_2	0.42	0.13	4.64
c_5	0.05	0.01	4.95
c_1	0.46	0.12	4.99
c_7	0.39	0.07	5
c_{10}	0.26	0.08	5
c_3	0.48	0.26	5.06
Compositions with rank above the mean			
c_2	0.4	0.1	5.2
c_8	0.48	0.14	5.1
c_4	0.33	0.17	5.2
c_9	0.49	0.08	5.21

Table B.22
Compositions with Discrete Uniform Service Rank distribution (c5, c7, c9, c8, c10, c4 and c1) and optimal compositions (c5, c7, c9, c8, c10) to reinitiate GA evolution.

	Avg-Var	Stdev-Var	Avg-Rank
c_5	0.05	0.01	4.95
c_{10}	0.26	0.08	5
c_7	0.39	0.07	5
Compositions with average variance above the mean			
c_2	0.42	0.13	4.64
c_1	0.46	0.12	4.99
c3	0.48	0.26	5.06

Appendix B. Discrete uniform service rank distribution (DUSRD) based composition selection

See Tables B.15–B.22.

References

[1] M. Cusumano, Cloud computing and SaaS as new computing platforms, Commun. ACM 53 (4) (2010) 27–29.

[2] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, I. Brandic, Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility, Future Gener. Comput. Syst. 25 (6) (2009) 599–616.

[3] D. Ardagna, B. Pernici, Adaptive service composition in flexible processes, IEEE Trans. Serv. Comput. 33 (6) (2007) 369–384.

[4] H. Zheng, W. Zhao, J. Yang, A. Bouguettaya, QoS analysis for web service compositions with complex structures, IEEE Trans. Serv. Comput. 6 (3) (2013) 373–386.

[5] C. Jatoth, G.R. Gangadharan, U. Fiore, Optimal fitness aware cloud service composition using modified invasive weed optimization, Swarm Evol. Comput. (2018) http://dx.doi.org/10.1016/j.swevo.2018.11.001.

[6] E. Al-Masri, Q.H. Mahmoud, Qos-based discovery and ranking of web services, in: Proceedings of the 16th International Conference on Computer Communications and Networks, IEEE, 2007, pp. 529–534.

[7] F. Seghir, A. Khababa, A hybrid approach using genetic and fruit fly optimization algorithms for QoS-aware cloud service composition, J. Intell. Manuf. (2016) 1–20, http://dx.doi.org/10.1007/s10845-016-1215-0.

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

[9] Z.-Z. Liu, D.-H. Chu, C. Song, X. Xue, B.-Y. Lu, Social learning optimization (SLO) algorithm paradigm and its application in QoS-aware cloud service composition, Inform. Sci. 326 (2016) 315–333.

[10] H. Jin, X. Yao, Y. Chen, Correlation-aware QoS modeling and manufacturing cloud service composition, J. Intell. Manuf. 28 (8) (2017) 1947–1960.

[11] G. Canfora, M. Di Penta, R. Esposito, M.L. Villani, An approach for QoS-aware service composition based on genetic algorithms, in: Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation, GECCO, 2005, pp. 1069–1075.

[12] A.E. Yilmaz, P. Karagoz, Improved genetic algorithm based approach for QoS aware web service composition, in: Proceedings of the IEEE International Conference on Web Services, ICWS, IEEE, 2014, pp. 463–470.

[13] M.B. Karimi, A. Isazadeh, A.M. Rahmani, QoS-aware service composition in cloud computing using data mining techniques and genetic algorithm, J. Supercomput. 73 (4) (2017) 1387–1415.

[14] S. Mistry, A. Bouguettaya, H. Dong, A.K. Qin, Metaheuristic optimization for long-term iaas service composition, IEEE Trans. Serv. Comput. 11 (1) (2018) 131–143.

[15] Z. Ding, J. Liu, Y. Sun, C. Jiang, M. Zhou, A transaction and QoS-aware service selection approach based on genetic algorithm, IEEE Trans. Syst. Man Cybern. 45 (7) (2015) 1035–1046.

[16] Z.-H. Zhan, X.-F. Liu, Y.-J. Gong, J. Zhang, H.S.-H. Chung, Y. Li, Cloud computing resource scheduling and a survey of its evolutionary approaches, ACM Comput. Surv. 47 (4) (2015) 63.

[17] J.O. de Carvalho, F. Trinta, D. Vieira, O.A.C. Cortes, Evolutionary solutions for resources management in multiple clouds: State-of-the-art and future directions, Future Gener. Comput. Syst. 88 (2018) 284–296.

[18] T. Thiruvenkadam, P. Kamalakkannan, Energy efficient multi dimensional host load aware algorithm for virtual machine placement and optimization in cloud environment, Indian J. Sci. Technol. 8 (17) (2015).

[19] G.F. Anastasi, E. Carlini, M. Coppola, P. Dazzi, QoS-aware genetic cloud brokering, Future Gener. Comput. Syst. 75 (2017) 1–13.

[20] R. Mennes, B. Spinnewyn, S. Latré, J.F. Botero, Greco: a distributed genetic algorithm for reliable application placement in hybrid clouds, in: Proceedings of the 5th IEEE International Conference on Cloud Networking, Cloudnet, IEEE, 2016, pp. 14–20.

[21] C. Guerrero, I. Lera, C. Juiz, Genetic algorithm for multi-objective optimization of container allocation in cloud architecture, J. Grid Comput. 16 (1) (2018) 113–135.

[22] J. Li, Y. Bai, N. Zaman, V.C. Leung, A decentralized trustworthy context and QoS-aware service discovery framework for the internet of things, IEEE Access 5 (2017) 19154–19166.

[23] M. Cuka, D. Elmazi, R. Obukata, K. Ozera, T. Oda, L. Barolli, An integrated intelligent system for IoT device selection and placement in opportunistic networks using fuzzy logic and genetic algorithm, in: Proceedings of the 31st International Conference on Advanced Information Networking and Applications Workshops, WAINA, IEEE, 2017, pp. 201–207.

[24] M. Aazam, E.-N. Huh, Dynamic resource provisioning through Fog micro datacenter, in: Proceedings of the 2015 IEEE International Conference on Pervasive Computing and Communication Workshops, IEEE, 2015, pp. 105–110.

[25] A. Papageorgiou, B. Cheng, E. Kovacs, Real-time data reduction at the network edge of Internet-of-Things systems, in: Proceedings of the 11th International Conference on Network and Service Management, CNSM, IEEE, 2015, pp. 284–291.

[26] L.M. Vaquero, L. Rodero-Merino, Finding your way in the fog: towards a comprehensive definition of fog computing, ACM SIGCOMM Comput. Commun. Rev. 44 (5) (2014) 27–32.

[27] Q. Li, R. Dou, F. Chen, G. Nan, A QoS-oriented Web service composition approach based on multi-population genetic algorithm for Internet of things, Intl. J. Comput. Intell. Syst. 7 (2) (2014) 26–34.

[28] O. Skarlat, M. Nardelli, S. Schulte, M. Borkowski, P. Leitner, Optimized IoT service placement in the fog, Serv. Oriented Comput. Appl. 11 (4) (2017) 427–443.

[29] S. Kalsi, H. Kaur, V. Chang, DNA cryptography and deep learning using genetic algorithm with NW algorithm for key generation, J. Med. Syst. 42 (1) (2018) 17.

[30] C. Wang, Y. Zhu, W. Shi, V. Chang, P. Vijayakumar, B. Liu, Y. Mao, J. Wang, Y. Fan, A dependable time series analytic framework for cyber-physical systems of iot-based smart grid, ACM Trans. Cyber-Phys. Syst. 3 (1) (2018) 7.

[31] B. Cao, J. Liu, Y. Wen, H. Li, Q. Xiao, J. Chen, QoS-aware service recommendation based on relational topic model and factorization machines for IoT Mashup applications, J. Parallel Distrib. Comput. (2018).

[32] C. Jatoth, G.R. Gangadharan, U. Fiore, R. Buyya, QoS-aware Big service composition using MapReduce based evolutionary algorithm with guided mutation, Future Gener. Comput. Syst. 86 (2018) 1008–1018.

[33] M. Faheem, V.C. Gungor, Energy efficient and qos-aware routing protocol for wireless sensor network-based smart grid applications in the context of industry 4.0, Appl. Soft Comput. 68 (2018) 910–922.

[34] H. Wu, K. Yue, C.-H. Hsu, Y. Zhao, B. Zhang, G. Zhang, Deviation-based neighborhood model for context-aware QoS prediction of cloud and IoT services, Future Gener. Comput. Syst. 76 (2017) 550–560.

[35] S. Tomovic, W. Cerroni, F. Callegati, R. Verdone, I. Radusinovic, M. Pejanovic, C. Buratti, An architecture for qos-aware service deployment in software-defined iot networks, in: Proceedings of the 20th International Symposium on Wireless Personal Multimedia Communications, WPMC, IEEE, 2017, pp. 561–567.

[36] D. Layzer, Genetic variation and progressive evolution, Am. Nat. (1980) 809–826.

[37] J. Chandrashekar, G.R. Gangadharan, Fitness metrics for QoS-aware web service composition using metaheuristics, in: Proceedings of the 7th International KES Conference on Intelligent Decision Technologies, Springer, 2015, pp. 267–277.

[38] N. Balakrishnan, V. Nevzorov, Discrete uniform distribution, Primer Stat. Distrib. (2005) 27–37.

[39] L. Bao, F. Zhao, M. Shen, Y. Qi, P. Chen, An orthogonal genetic algorithm for QoS-aware service composition, Comput. J. 59 (12) (2016) 1857–1871.

[40] Y. Yu, H. Ma, M. Zhang, An adaptive genetic programming approach to QoS-aware web services composition, in: Proceedings of the IEEE Congress on Evolutionary Computation, 2013, pp. 1740–1747.

[41] Z. Zheng, Y. Zhang, M. Lyu, Investigating QoS of real-world web services, IEEE Trans. Serv. Comput. 7 (1) (2014) 32–39.

[42] Y. Zhang, Z. Zheng, M. Lyu, WSPred: A time-aware personalized qos prediction framework for web services, in: Proceedings of the IEEE 22nd International Symposium on Software Reliability Engineering, 2011, pp. 210–219.

[43] E. Al-Masri, Q.H. Mahmoud, Investigating web services on the world wide web, in: Proceedings of the 17th International conference on World Wide Web, ACM, 2008, pp. 795–804.
[44] J.A. Villasenor Alva, E.G. Estrada, A generalization of Shapiro-Wilk's test for multivariate normality, Commun. Stat. Theory Methods 38 (11) (2009) 1870–1883.
[45] A. Mostafa, M. Zhang, Multi-objective service composition in uncertain environments, IEEE Trans. Serv. Comput. (2015) http://dx.doi.org/10.1109/TSC.2015.2443785.
[46] D. Wang, Y. Yang, Z. Mi, A genetic-based approach to web service composition in geo-distributed cloud environment, Comput. Electr. Eng. 43 (2015) 129–141.
[47] X. Zhao, B. Song, P. Huang, Z. Wen, J. Weng, Y. Fan, An improved discrete immune optimization algorithm based on PSO for QoS-driven web service composition, Appl. Soft Comput. 12 (8) (2012) 2208–2216.
[48] M. Liu, M. Wang, W. Shen, N. Luo, J. Yan, A quality of service (QoS)-aware execution plan selection approach for a service composition process, Future Gener. Comput. Syst. 28 (7) (2012) 1080–1089.
[49] P. Rodriguez-Mier, M. Mucientes, M. Lama, Hybrid optimization algorithm for large-scale QoS-aware service composition, IEEE Trans. Serv. Comput. 10 (4) (2017) 547–559.
[50] M.A. Amiri, H. Serajzadeh, QoS aware web service composition based on genetic algorithm, in: Proceedings of the 5th International Symposium on Telecommunications, IST, 2010, pp. 502–507.
[51] V. Vapnik, Statistical Learning Theory. 1998, Wiley, New York, 1998.
[52] V. Chang, The business intelligence as a service in the cloud, Future Gener. Comput. Syst. 37 (2014) 512–534.
[53] D.W. Zimmerman, Teachers corner: A note on interpretation of the paired-samples t test, J. Educ. Behav. Stat. 22 (3) (1997) 349–360.
[54] V. Chang, M. Abdel-Basset, M. Ramachandran, Towards a reuse strategic decision pattern framework – from theories to practices, Inform. Syst. Front. (2018) http://dx.doi.org/10.1007/s10796-018-9853-8.
[55] F. Wilcoxon, Individual comparisons by ranking methods, Biometrics Bull. 1 (6) (1945) 80–83.

Chandrashekar Jatoth received his B.E in Information Technology from Osmania University, M.Tech in Artificial Intelligence from University of Hyderabad, Ph.D form University of Hyderabad, Hyderabad, India in 2008, 2010, and 2018 respectively. He currently is working as an Associate Professor at the Sreenidhi Institute of Science and Technology, Hyderabad, India. His research interests focus on service composition, and computational intelligence techniques.

G.R. Gangadharan is an Associate professor at National Institute of Technology, Tiruchirappalli, India. His research interests focus on the interface between technological and business perspectives. Gangadharan received his PhD in information and communication technology from the University of Trento, Italy, and the European University Association. He is a senior member of IEEE and ACM. Contact him at geeyaar@gmail.com.

Rajkumar Buyya is a Fellow of IEEE, Redmond Barry Distinguished Professor of Computer Science and Software Engineering, Future Fellow of the Australian Research Council, 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 Pty Ltd., a spin-off company of the University, commercializing its innovations in Grid and Cloud Computing. Dr. Buyya has authored/co-authored over 625 publications. He is one of the highly cited authors in computer science and software engineering worldwide. For further information on Dr. Buyya, please visit: http://www.buyya.com.