J Supercomput (2017) 73:1387–1415
DOI 10.1007/s11227-016-1814-8

QoS-aware service composition in cloud computing using data mining techniques and genetic algorithm

Mohammad Bagher Karimi[1] · **Ayaz Isazadeh**[2] · **Amir Masoud Rahmani**[1]

Published online: 16 August 2016

Abstract One of the requirements of QoS-aware service composition in cloud computing environment is that it should be executed on-the-fly. It requires a trade-off between optimality and the execution speed of service composition. In line with this purpose, many researchers used combinatorial methods in previous works to achieve optimality within the shortest possible time. However, due to the ever-increasing number of services which leads to the enlargement of the search space of the problem, previous methods do not have adequate efficiency in composing the required services within reasonable time. In this paper, genetic algorithm was used to achieve global optimization with regard to service level agreement. Moreover, service clustering was used for reducing the search space of the problem, and association rules were used for a composite service based on their histories to enhance service composition efficiency. The conducted experiments acknowledged the higher efficiency of the proposed method in comparison with similar related works.

Keywords Cloud computing · Service composition · Clustering · Association rules · Genetic algorithm

✉ Ayaz Isazadeh
isazadeh@tabrizu.ac.ir

Mohammad Bagher Karimi
m_karimi@iaut.ac.ir

Amir Masoud Rahmani
rahmani@srbiau.ac.ir

1 Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran

2 Department of Computer Science, University of Tabriz, Tabriz, Iran

1 Introduction

Cloud computing has recently emerged as the new generation of distributed computing where its objective is to realize the idea of providing computational services as the fifth utility service after water, electricity, telephone and gas services [1]. In line with the needs of different areas of IT, the following three different types of services can be provided by cloud computing: IaaS (infrastructure as a service), PaaS (platform as a service) and SaaS (software as a service) [2,3]. By inheriting several achievements such as pervasive computing, grid computing, service-oriented architecture and Web 2.0 from information technology, cloud computing has led to the development of new software engineering generation in which software application is provided as a SaaS on the Internet rather than as a package [4].

In a cloud environment, users can submit their requests for a software application service on the internet. In many cases, users might request complex services in a cloud environment which cannot be provided by a single service. Hence, under such circumstances, a user's request has to be accomplished as a composite service; this composite service includes individual services which are constructed in advance [5,6]. Due to the high inclination of numerous IT companies to provide cloud services, there are many candidates for each abstract service of each request workflow with the same functionality but differing in quality of service (QoS) attributes such as cost, response time and availability [7]. As a result, the service composition system should select those services among the concretes of the abstract services which have the highest quality. However, the global quality of selected services should not exceed the quality constraints specified by the users, documented in service level agreement (SLA) contracts [8–10]. Hence, QoS-aware service composition is a multi-constraint optimization problem (MCOP) which can be mapped into multi-dimensional multiple-choice knapsack problem (MMKP) [11–15].

In the cloud environment, composite services are produced on-the-fly based on users' on-demand requests. Under these conditions, composing service within a very short time is the critical significance, i.e., there should be a trade-off between optimality and execution time. The QoS-aware service composition problem is known as an optimization problem with NP-hard characteristics. Consequently, some of the researchers have used genetic algorithm and other heuristic methods to find an optimal global solution [16–19]. However, regarding the inherent features of these algorithms, especially genetic algorithm, there are some problems. First, these algorithms select initial population from the search space randomly. In dynamic cloud environment, sometimes, the number of services increases remarkably [20–23], so the convergence speed of these algorithms is reduced, and service composition time increases [18,24–27]. Second, since these heuristic algorithms can continue endlessly but by specifying a termination criterion usually to be restricted, so such execution led to the reduction of qualitative optimization of the selected composite service [18,21,26,28].

For handling this problem, researchers have used different hybrid heuristic algorithms [23–25,29–31]. Some researchers have used service clustering techniques to reduce the search space of the problem [30,32–34]. However, in the related works, there are a few disadvantages. For example, in these works, clustering is repeated online for each new request. In regard to high growing number of cloud services, this massive

time overhead could make the corresponding methods inefficient for real multi-cloud and public cloud environments. In the previous works, simple utility function and a hierarchical cluster selection method have been used [32,33]. The QoS attributes are globally specified by the users per composite service, not individually per its concrete services. Hence, selection of appropriate clusters could not be guaranteed. Thus, because of inappropriate selection of the clusters, inefficient initial population will be generated; consequently, the inefficient initial population will decrease the optimality of the solutions and convergence of these methods.
In this paper, using a combination of genetic algorithm and data mining techniques, an efficient method is presented for dealing with the above-mentioned challenges. Our proposed method targeted all types of cloud, especially public clouds. The major contributions and the novelty of the present study are as follows:

- Service clustering technique is used to reduce the search space of the problem which resulted in the reduction of the search time. Unlike previous works [30,32, 34], in the proposed method, clustering is conducted at the beginning of system initialization, and it is repeated in the determined intervals based on a specific rate of service increase. Consequently, it leads to the reduction of computational overhead.
- Other contribution of our work is that the association rule mining is used for selecting appropriate clusters based on semantic relations between clusters in composite services history. Such appropriate clusters will be the input of the genetic algorithm as the initial population. Simulation results show that our algorithm performs well in terms of solution feasibility, optimality and time complexity compared with other works.

The rest of the paper is organized as follows: In Sect. 2, we describe service composition problem formally and briefly review the related studies. In Sect. 3, a detailed account of the proposed genetic-based algorithm for service composition is given. In Sect. 4, the simulation of the proposed method is described, and the related experiments for verifying and analyzing it are elaborated. Finally, in Sect. 5, the conclusion of the study and directions for further research are mentioned.

2 Background and related works

The issue regarding service composition process is selecting an appropriate service among many similar services for executing a task and ultimately producing a composite service (Fig. 1). Inasmuch as service composition component is the core of supplying cloud applications, hence, due to its vital significance, there is an essential need for embedding a section in the middleware of cloud computing systems as the service composition unit. There are two general strategies for QoS-aware service composition: local service selection and global optimization of composite services [35–37]. Each of these strategies is briefly described below.

- Local selection of a service: It is appropriate for distributed systems where centralized management is not practical. In this strategy, a service is selected from among candidate services for an abstract service regardless of other abstract ser-

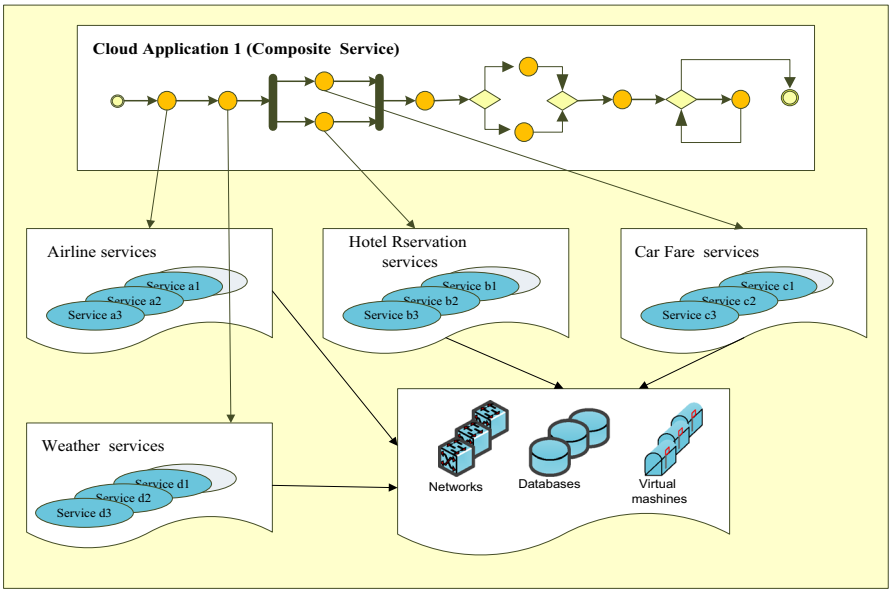

Fig. 1 An example of a composite service

vices within a workflow. This strategy does not consider the global constraints specified by the user.

- Global optimization of composite services: In this strategy, an appropriate service is not independently selected for an abstract service in the workflow. Rather, the selection is made with regard to concrete services chosen for all other abstract services of the workflow. The purpose of this method is to select the best concrete service for each of the abstract services so that a composite service with the highest quality within user-determined constraint framework can be obtained.

2.1 Service quality model

As shown in Eq. (1), the global quality of a composite service should be measured based upon the workflow structure of that composite service.

In many previous studies [16,32,37], the following four pattern types are considered in the internal structure of composite services which are supported by business process execution language (BPEL) [17]:

- Sequence: sequential execution of services.
- AND: parallel execution of services.
- Branch: branching or conditional execution of services.
- Loop: repetitive execution of services.

Table 1 indicates the manner of measuring global values for QoS parameters in a composite service [32,37].

Table 1 Method of calculating qualitative values

Qos parameters	Composition patterns			
	Sequence	AND	Branch	Loop
Response time	$\sum_{i=1}^{n} q_{s_i}^1$	$\text{Max}_{i=1}^{n} q_{s_i}^1$	$\text{Max}_{i=1}^{n} q_{s_i}^1$	$\sum_{i=1}^{k} q_s^1$
Price	$\sum_{i=1}^{n} q_{s_i}^2$	$\sum_{i=1}^{n} q_{s_i}^2$	$\text{Max}_{i=1}^{n} q_{s_i}^2$	$\sum_{i=1}^{k} q_s^2$
Availability	$\prod_{i=1}^{n} q_{s_i}^3$	$\prod_{i=1}^{n} q_{s_i}^3$	$\text{Max}_{i=1}^{n} q_{s_i}^3$	$\prod_{i=1}^{n} q_s^3$
Successability	$\text{Avg}_{i=1}^{n} q_{s_i}^4$	$\text{Avg}_{i=1}^{n} q_{s_i}^4$	$\text{Avg}_{i=1}^{n} q_{s_i}^4$	q_s^4

$q_{s_i}^l$, $1 \le l \le 4$ shows response time, price, availability and successability of concrete service S_i

2.2 Statement of the problem

In composing a service, the issue which should be considered is the selection of individual and atomic services with high-value qualitative parameters. Indeed, this issue should be taken into consideration so that the total parameters should not exceed the constraints specified by the users. Service composition issue can be formalized by Eqs. 1 and 2.

$$\begin{cases} \text{Max} \sum_{l=1}^{p} Q_l\,(\text{cs})\,.W_l \\ \quad \text{Subject To } Q_j\,(\text{CS}) \le \text{SLAL}_j\,(\text{CS}) \quad \text{for negative attributes} \\ \quad \text{Subject To } Q_j\,(\text{CS}) \ge \text{SLAL}_j\,(\text{CS}) \quad \text{for positive attributes} \end{cases} \tag{1}$$

$$\sum\nolimits_{l=1}^{p} W_l = 1 \tag{2}$$

The workflow of a composite service is defined by the vector $\text{WF} = (T_1, T_2, \ldots, T_n)$, and a candidate composite service is denoted by $\text{CS} = (S_1, S_2, \ldots, S_n)$, where T_i refers to a task with an abstract service in the workflow and S_i denotes a selected concrete service for the T_i service in the candidate composite service. The qualitative constraints specified by the customer are represented by the vector $\text{SLA} = (\text{SLA}_1, \text{SLA}_2, \ldots, \text{SLA}_p)$. In Eq. (1), $\text{SLA}_j(\text{CS})$ function determines the value of the jth global qualitative parameter specified by user for the CS composite service. Also, Q_j (CS) function determines the jth qualitative parameter for the CS composite service, and W_L indicates the weight of the Lth qualitative parameter. By taking Eqs. (1) and (2) into account, optimal composite service should be achieved.

2.3 Related works

Before the emergence of cloud computing, service composition was accomplished in service-oriented architecture (SOA) and grid environment. The SOA service composition is involved with the realization of a workflow using concrete services. Also, service composition is another term for resource scheduling in grid environment [16]. However, it should be pointed out that the first impression about service composition

in cloud environment appears to be a combination of service composition in SOA and grid computing. It is a comprehensive process which includes the discovery and selection of software services which are deployed on infrastructure services. Such services are selected based on customers' non-functional requirements and constraints specified through SLAs.

QoS-aware service composition is an optimization problem which has several potential solutions, and only some solutions are optimal. Furthermore, each of the studies reviewed in this section focused on one or more of the following criteria: SLA contract support, centralization or decentralization of the composition algorithm, automaticity of the composition process, consistency and compatibility and concentration on inherent features of cloud environment including economic aspects, longevity of composition, scalability, etc. Some of the related works, which are briefly overviewed below, tried to solve service composition optimization problem using classic algorithms [14]. In [37], a two-stage hybrid method was proposed in which global optimization was combined with local selection to benefit from the advantages of both methods. In this study, mixed integer programming (MIP) was used for decomposing global QoS constraints into local constraints, so that services can be selected both locally and in a parallel form. In [38], a two-stage method was used for optimal service composition so that services can be composed dynamically and reliably. In the first stage, culture genetic algorithm (CGA) was used for selecting the best service composition schemes. Then, global QoS constraints were decomposed into local constraints. In the second stage, proper concrete services for each of the workflow tasks were selected based on improved case-based reasoning. This group of algorithms just has aimed to solving service composition as optimization problem regardless of time complexity of algorithms.

Achieving an optimal solution within a short time has motivated researchers to use combinatorial algorithms [15,39]. The proposed method in [8] produces composite services using QoS requirements in the cloud environment for service composition; the corresponding authors tried to produce composite services with appropriate qualitative attributes within acceptable time. In [40], genetic algorithm was used to compose services by taking customers' qualitative requests and SLA constraints into consideration. Skyline notion was used in this algorithm to enhance composite service quality and convergence speed. Moreover, to overcome the time-consuming issue of service composition problem, the paper [18] proposed the use of genetic programming for web service composition, investigating three variations to ensure the creation of functionally correct solutions. It also optimized the composition according to their quality of service.

Particle swarm optimization (PSO) with integer array coding was used to solve the service composition problem in rational time [29]. To achieve this goal, the skyline operator with binary decision variables was used to eliminate improper services from the search space. The time complexity of these combinatorial algorithms is much better than the classic algorithms. However, their execution time is not acceptable for an increasing number of services yet [24–26]. Because of on-demand and on-the-fly form of service composition, it should be made a trade-off between optimal solution and composition time. But result evaluations show that with increasing the number of concrete services led to much more increasing of time complexity. To resolve this

critical problem, some researchers applied local selection and global optimization together.

In [24], an improved merging genetic algorithm and ant colony algorithm was used for solving the time complexity of service composition. In first stage global optimal paths is found by genetic algorithm and in second stage globally optimal solution by running ant colony algorithm. In [41], researchers combined the optimized gravitational attraction search algorithm with the imperialist competitive algorithm to develop a new memetic algorithm. This algorithm was used for simultaneously obtaining response time and optimal or near-optimal execution cost to compose services in the cloud environment. In [42], Niche particle swarm optimization (NPSO) was used to achieve higher cognitive speed and find optimal composite service. Moreover, NPSO and intelligent multi-objective PSO were used to produce a set of Pareto optimal composite services by optimizing various objective functions simultaneously. A new proposed method in [31] combines two composition strategies to solve time and local optima issues of service composition problem. The global search process is performed by the genetic algorithm, which applies a novel roulette wheel selection to avoid premature convergence. To reduce the time complexity, the local search is done by the fruit fly optimization phase.

In [43], contextual data, such as geographical information, were used to identify similar neighbors for each requester. Then, QoS records of the similar neighbors were used to obtain more accurate prediction values. The basic rationale behind this study is that the users or services with similar service invocation context are more likely to receive similar QoS values. Due to publishing services on multiple clouds, an algorithm was proposed in [44] for cloud service composition which can compose services from different clouds. The algorithm selects the cloud with the maximum number of services which increases the feasibility of composition with minimal time overhead. The method proposed in [45] was intended to efficiently find a composition across multiple clouds. It included a greedy algorithm and ant colony optimization-based algorithm which was aimed at selecting feasible composite services using the minimum number of clouds that led to less computational time.In [46], a heuristic algorithm was implemented along with service clustering technique for grouping of similar services. This algorithm was capable to reduce the execution time and nearly promised the optimal result as well. Because of dynamic nature of the cloud environment, sometimes, the number of services increases remarkably [20–23]; consequently, these algorithms are not sufficiently effective in terms of convergence speed and optimality [18,24–27].

Some other works have also used clustering techniques with heuristic algorithms to improve composition speed by reducing problem space. In [32], researchers first clustered all service classes. Then, they used quality utility function and a heuristic method to find a service close to the optimal service. In case a problem is observed and detected by continuous monitoring, alternative compositions can be used. Using the pre-built service clusters in [34], a method has been presented for selecting candidate service set with the effectively reduced searching space and semantic comparison complexity. It could obtain more optimal compositions by filtering services with the dynamically determined threshold based on the best composition QoS in the process of composition. In [30], authors have used clustering for categorizing the web services

to find similar services. Then, an ant colony algorithm is applied for finding the best set of services.

As mentioned above, clustering technique is used for local selection in the previous works. However, none of these works has an efficient pattern to select service clusters for global optimization. There are two main problems in these works:

- First, clustering is repeated online for each new request. In regard to high growing number of cloud services, this massive time overhead could make the corresponding methods inefficient for real multi-cloud and public cloud environments.
- Second, in the previous works, simple utility function and a hierarchical cluster selection method. QoS attributes are globally specified by the users per composite service, not individually per its concrete services. Hence, selection of appropriate clusters could not be guaranteed. Because of inappropriate selection of the clusters, inefficient initial population will be generated; consequently, the inefficient initial population will decrease the optimality of the solutions and convergence of these methods.

In this paper, to solve the above-mentioned problems, we propose a new approach using genetic algorithm and data mining techniques toward service composition in cloud environment. Each abstract service is initially clustered. When a new request is received, appropriate service cluster is selected based on semantic relations between service clusters; semantic relation is extracted by association rule mining. After selecting the appropriate service clusters, as initial population, the genetic algorithm is used to find out optimal or near-optimal solutions (composite service) regarding the QoS constraints determined by users.

3 The proposed method

As discussed above, due to the rapid increase in the number of cloud services and cloud users, the previous combinatorial methods are not adequately efficient in the dynamic cloud environment. With regard to this research problem, unsupervised data mining techniques were used to propose a novel method. Clustering and association rules were used for local selection of services, and genetic algorithm was used to achieve a global optimal solution within the criteria announced by the user. Different phases of the proposed service composition method include pre-processing phase, contract-making phase, service composition and customer request execution phase and maintenance phase, each of which is described below.

3.1 Pre-processing phase

This phase includes the following four stages: normalization of the qualitative attributes of services, service clustering, clustering of users' SLA and discovering association rules between users' SLA clusters and service clusters. Algorithm 1 shows the pre-processing phase.

```
1 Void PreProcessing( )
2 {
3     Cluster User's SLAs;
4     Cluster Services;
5     Extract Association Rules of Service Clusters per each SLA Cluster;
6 }
```

Algorithm 1. Pre-processing algorithm

3.1.1 Normalization of qualitative attributes of services

The scales of qualitative attributes of services are different. As a case in point, in the experimental environment used in this study, while reliability index was between 0 and 1, response time was a figure which could vary from 100 to 3000 ms for each service. Consequently, measuring high-value attributes led to a bias against low-value attributes.
Thus, Eq. (3) is used to normalize all the attributes into a range from 0 to 1 [16,26]. In this study, the high and low values for QoS attributes are obtained by searching for the minimum and maximum values for the intended QoS attributes among samples for each respective service [12,25,26]. Furthermore, when a new service is added, the related information should be updated.

$$q^{l}_{S_{ij}} = \begin{cases} \left(q^{l}_{S_{ij}} - \mathrm{Min}\left(q^{l}_{S_i}\right)\right) \Big/ \left(\mathrm{Max}\left(q^{l}_{S_i}\right) - \mathrm{Min}\left(q^{l}_{S_i}\right)\right) & \text{for positive attributes} \\ \left(\mathrm{Max}\left(q^{l}_{S_i}\right) - q^{l}_{S_{ij}}\right) \Big/ \left(\mathrm{Max}\left(q^{l}_{S_i}\right) - \mathrm{Min}\left(q^{l}_{S_i}\right)\right) & \text{for negative attributes} \end{cases} \tag{3}$$

In Eq. (3), $q^{l}_{S_{ij}}$ refers to the lth qualitative attribute of the jth service from the ith abstract service. Also, $\mathrm{Min}(q^{l}_{S_i})$ and $\mathrm{Max}(q^{l}_{S_i})$ were the lowest and highest values of the lth qualitative attribute for the ith abstract service.

3.1.2 Service clustering

Regarding the hypothesis that users' requests with identical requests and identical SLA contracts require similar service qualities, hence, we should classify concrete services of the abstract services in the cloud system. In public cloud systems and the systems which supply web services, announcements, interfaces and the qualitative attributes of the services are published by their providers on UDDI (universal description, discovery, and integration) registries [47]. In the system proposed in this study, the method for producing information functions in this way: service information is collected through UDDI registries, and clustering is applied on them. When a new service is discovered, it is added to the related similar cluster based on mapping. Whenever the development rate, i.e. the number of services, reaches the threshold which was specified by the managers, clustering is updated again.

For classifying concrete services in the proposed method, K-means algorithm was used which is simple, fast and efficient [48–50]. In this algorithm, after the number of clusters for each abstract service is determined and cluster centers are selected, services are clustered according to the minimal Euclidean distance between cluster centers and services. Distance is measured using the following equation:

$$\sqrt{\sum_{l=1}^{k}\left(q_{C_{ij}}^{l}-q_{S_{ij}}^{l}\right)^{2}} \tag{4}$$

In this equation, $q_{C_{ij}}^{l}$ denotes the value of lth qualitative attribute for the jth cluster center, and ith stands for abstract service. Thus, $q_{S_{ij}}^{l}$ indicates the value of the lth qualitative attribute for the jth concrete service of the ith abstract service. In service clustering, weight was not considered for the qualitative attributes. Indeed, weight should be given according to users' views for each request. However, if so, clustering has to be repeated per request. Consequently, it will impose excessive computing overhead and long execution time.

3.1.3 SLA clustering

In this system, content of users' requests for composite services including operational and non-operational needs are processed. For meeting operational needs, a workflow is defined for executing the intended request. Nevertheless, regarding the non-operational need which includes the wanted qualitative features, an SLA contract is established with the user. Indeed, each request is translated into a workflow and an SLA contract and is stored in the database. At the outset of the establishment of the system, clustering is carried out based on the available records in its history. With the arrival of any new request, the record related to the respective cluster is mapped and is registered on it; whenever the development of the database grows at a particular rate which has been determined by the system managers, clusters are updated via the repetition of clustering. Clustering users' requests is also carried out based on the similarity of the content of SLA contracts. K-means algorithm is also used here. SLA contracts are clustered through Eq. (5) according to the minimal Euclidean distance.

$$\sqrt{\sum_{l=1}^{k}\left(\mathrm{SLA}_{ci}^{l}-\mathrm{SLA}_{uj}^{l}\right)^{2}} \tag{5}$$

In this equation, SLA_{ci}^{l} refers to the lth qualitative attribute included within c_i cluster center. Hence, SLA_{uj}^{l} stands for the lth qualitative attribute of the jth request of the uth user.As discussed above, an identical weight was given to all the attributes.

3.1.4 Discovering association rules between contracts and cluster services

As mentioned before, the underlying rationale for the proposed method is that for accomplishing similar requests with similar qualitative needs, the services with similar qualitative attributes should be selected. In general, such services are included within the same cluster. For enhancing the efficiency of service composition, at first, we should determine the clusters from which services were selected for realizing previous

Table 2 Service composition history

Request	SLA cluster	History
1	0	S3_C2, S2_C0, S1_C3, S7_C3, S9_C5, S6_C4
2	0	S3_C2, S2_C9, S1_C3, S7_C3, S9_C5, S6_C9
3	0	S3_C2, S2_C9, S1_C0, S7_C3, S9_C5, S6_C9
4	0	S3_C2, S2_C1, S1_C3, S7_C3, S9_C5, S6_C9
5	0	S3_C2, S2_C0, S1_C0, S7_C5, S9_C5, S6_C9
6	0	S3_C2, S2_C0, S1_C3, S7_C5, S9_C5, S6_C9
7	0	S3_C9, S2_C0, S1_C3, S7_C5, S9_C5, S6_C4
8	0	S3_C2, S2_C1, S1_C3, S7_C3, S9_C5, S6_C2
9	0	S3_C9, S2_C9, S1_C0, S7_C3, S9_C5,S6_C9
10	0	S3_C2, S2_C9, S1_C3, S7_C9, S9_C3, S6_C4
		…

composite services. Apriori association rule extraction algorithm was used on the history data of the composite services requested by users [48–50]. As a result, we could discover the related clusters of different abstract services. In this study, the term related service clusters refers to the clusters from which services have already been used for realizing the workflows of similar requests. In the next stage, for selecting a concrete service of a request, we only refer to the pre-specified clusters of the related requested abstract services in workflow. This operation has two significant results: first, service composition time decreases due to the reduced search space. Second, the compatibility between qualitative attributes and user's requests increases because previously experienced service clusters are used. The example illustrated in Tables 2 and 3 shows the details about the extraction of association rules using the Apriori algorithm in our method. Table 2 contains part of a sample composite services history. In this table, Si_Cj is an ordered pair that Si shows an abstract service, and Cj shows the corresponding selected cluster. Also, Table 3 shows part of association rules extracted by Apriori algorithm through composition histories illustrated in Table 2.

3.2 Contract-making phase

In this phase, user request for a needed composite service, qualitative features and quality constraints specified by the customer are negotiated and determined. Then, they are registered in the form of an SLA contract in the system. The SLA obtained in this phase creates the foundation for the development of composite service. That is, composite service is produced based on supporting qualitative attributes which are registered in the SLA contract.

In this system, there is a subsystem which monitors the implementation and performance of the contract. At any moment, it monitors the qualitative attributes of the service and records the changes. In case the service provider breaches the guaranteed service attributes, the customer should be compensated for the damages according to the contract. The operation of the SLA monitoring and maintenance system is pre-

Table 3 Association rules

	Association rule	Confidence (%)	Support
1	S1_C3−−>S9_C5,S3_C2,S7_C3	57	24.5
2	S3_C2,S1_C3−−> S7_ C3,S1_C3	67	27.2
3	S3_C2,S1_C3−−> S9_ C5,S1_C3	67	23.2
4	S3_C2,S1_C3−−> S9_ C5,S7_C3	67	22.2
5	S7_C3,S1_C3−−> S9_ C5,S3_C2	100	23.2
6	S9_C5,S1_C3−−> S3_ C2,S1_C3	67	27.78
7	S9_C5,S1_C3−−> S7_ C3,S3_C2	67	26.55
8	S9_C5,S1_C3−−> S9_ C5,S7_C3	67	28.7
	…		

sented in [51], and the adaptability model and the modification and revision of the qualitative loss of system services are demonstrated in [52,53]. Also, in our system, after the configuration of the composite service and the beginning of its execution, SLA monitoring unit continuously examines the qualitative attributes of each of the services and compares them with their reference values which are stored in Service QoS database. If the service is impaired with regard to QoS, hence, selecting and replacing the appropriate alternative service lead to the desirable continuation of service based on SLA content which is stored in SLA database, and the shortcoming of the service provision is stored in the SLA violation database for compensation (Fig. 2).

3.3 Service composition

In this phase, service composition system maps the customer to the related cluster according to the values and contents of the SLA contract. Then, based on the service clusters related to that SLA cluster, the system develops the composite service for accomplishing customer's request.

The first phase of the proposed method which includes the clustering of services and SLAs is passive. It is done at the beginning of system initialization. The clusters should be updated for increasing a certain number of services or SLA contracts. However, the other three phases are active which are once executed for each composite service request. Service composition phase includes two stages: finding appropriate service clusters and using genetic algorithm for global optimization. In first stage, local services are first selected. Local service selection includes mapping a user's SLA contract into a similar SLA cluster. Then, service clusters used in the history of that cluster are extracted for instantiating each abstract service. In the second stage, genetic algorithm is implemented to find the globally optimal service composition. That is, services leading to the highest quality composition are selected from the determined clusters of abstract services. However, the final composite services should not exceed the quality constraints determined by the user. The steps of the proposed method are described as follows:

1. Finding appropriate service clusters: In this stage, user SLA is first mapped to an appropriate SLA cluster via mapping function. This procedure is accomplished

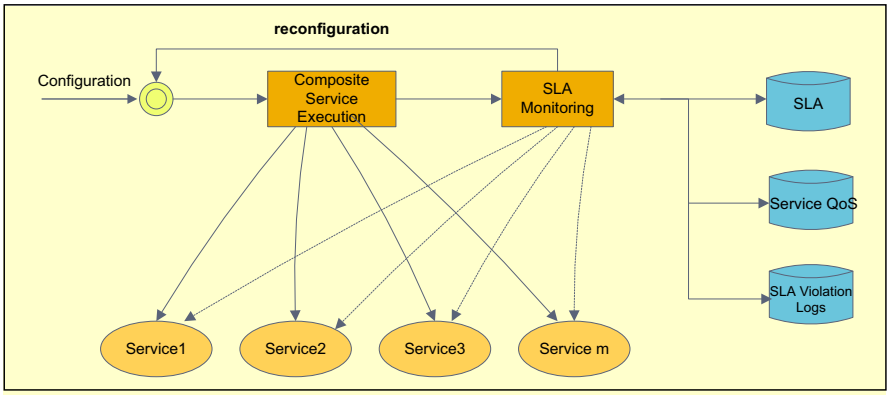

Fig. 2 SLA monitoring system

based on the closest Euclidean distance between user SLA contract and the centers of the SLA clusters. After the specification of the related cluster, relevant association rules to that cluster are fetched. Then, using algorithm (2), the most useful service clusters are selected for realizing the workflow of the requested service. The process of finding appropriate clusters is shown in Fig. 3. Table 4 shows an example of which clusters is selected for a sample workflow by algorithm (2). Table 4 shows the results of execution of algorithm (2) to find appropriate clusters for a workflow shown in Fig. 4; this algorithm uses the association rules shown in Table 3. SLA of this request has been assigned to SLA cluster 0. Default cluster has been assigned for each abstract service that could not find any cluster for it using the algorithm.

2. Global optimization by the genetic algorithm: The primary focus of the method proposed in this paper is to reduce the execution time and enhance the coverage of users' qualitative requirements. Time reduction is a result of search space reduction through service clustering. Coverage enhancement of users' requests is related to fetching services from appropriate clusters. That is, services should be selected from clusters

which have already been experienced. Clustering leads to the substantial reduction of the search space problem; however, the remarkable increase in the number of cloud environment services leads to an increase in the population of each service cluster. Hence, for performing the search within an acceptable time limit, the genetic algorithm was used for searching a solution with global optimization [54,55]. Indeed, several heuristic methods, such as PSO, imperialist competitive algorithm, etc., have been proposed for finding an optimal solution. Each method has specific capabilities and features which are considered to be appropriate for a solving particular issues and problems. In different studies [27,56,57], researchers have demonstrated the efficiency and efficacy of the genetic algorithm in solving different optimization problems in comparison to other global optimization methods. The investigations indicated that the genetic algorithm had significantly better performance in terms of response optimality and convergence speed. As discussed above, there are two reasons and justifications for using the genetic algorithm. The first reason is that this algorithm has special features

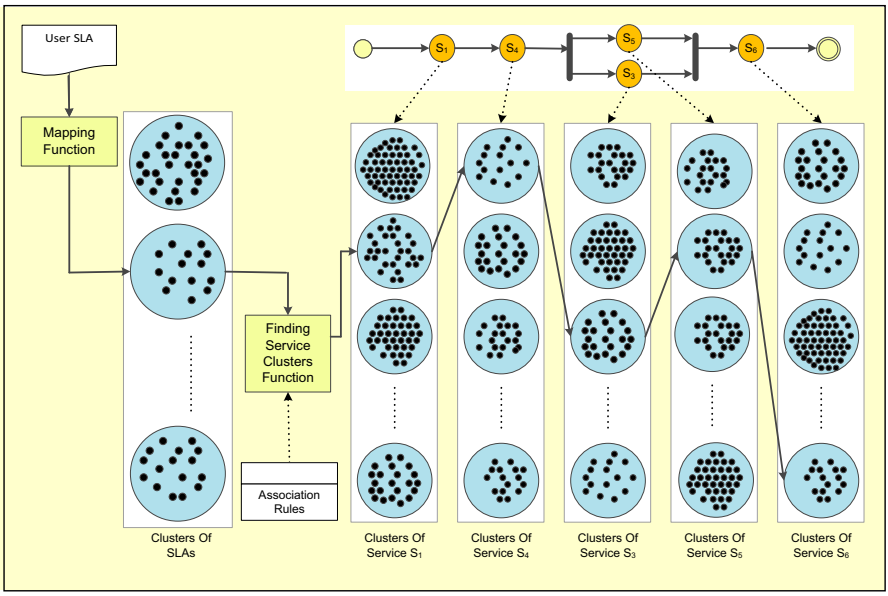

Fig. 3 Finding appropriate service clusters

Table 4 Selected clusters of workflow

Abstract services	S3	S2	S1	S7	S9	S6
Cluster	C2	Default	C3	C3	C5	Default

which make it suitable for solving the intended problem discussed in this study. It evolves solutions based on the population rather than the individual. For selecting an optimal solution, it operates only based on the fitness function with simple arithmetic expressions. Furthermore, it does not require complex calculations for producing the initial population which is simply produced based on probability. The second major justification for using this algorithm is that it has been extensively used by researchers in the past for solving the service composition problem. Previous studies have proved several advantages and the desired performance of the genetic algorithm in comparison with other methods [16,40]. In this paper, the researchers demonstrated the merits of the novel combinatorial method in comparison with other methods.

Different stages of the proposed method are illustrated in algorithm (3). After getting a user's request and making an SLA contract, the suitable clusters for selecting the services required for the composite service are found through algorithm (2). Then, the first generation of the solutions is produced by randomly selecting specified number of individuals of the population from the clusters related to each abstract service. Next, the population of solutions evolves and develops by the genetic algorithm. This operation is done through a specific number of repetitions and through achieving optimal conditions. The required stages and components of the genetic algorithm for solving the intended problem are given below.

```
1 Void FindClusters(Abstract Service WorkFlow[], Number Clusters[], Rule RuleSet[],
  float α, float β)
    // Workflow array contains abstract services of the requested composite service
    // Clusters array returns the selected clusters for concrete services of each of abstract services in
    // RuleSet is set of rules. Each rule is contained items in form of abstract services and those selected
       cluster number.
    // α and β parameters are defined relative to problem space
2 {
3   For(each abstract service Si in WorkFlow which no assigned any cluster in
    cluster vector)
4      {
5       Find a Rule R in RuleSet such that:
6           R covers Si with support>α and confidence> β;
7       IF(Found)
8          {
9           add R to FoundRuleSet; // Rule FoundRuleSet[]
10          insert cluster number of Si to Clusters vector;
11          For(each reminded abstract service )
12              {
13               Find any Sj in WorkFlow such that:
14                   R covers cluster of Sj;
15               IF(Found)
16                  insert cluster number of Sj to Clusters vector;
17               Else
18                   {
19                    Find any Sj in WorkFlow such that:
20                        Sj is contained in consequence of R;
21                        IF(Found)
22                          insert cluster number of Sj to Clusters vector;
23                    }
24                }
25            }
26     }
27   For(each reminded abstract service Si in WorkFlow which no assigned any cluster
     in cluster vector)
28      {
         Find a Rule R in RuleSet such that:
29          R covers cluster of Si with support>0 and confidence>0;
30       IF(Found)
31         insert cluster number of Sj to Clusters vector;
33       }
34 }
```

Algorithm 2. Finding appropriate service clusters algorithm

```
1 CompositService  MakeCompositService( )
2 {
3     Read User Request;
4     Read Workflow;
5     Read QoS Parameters Defined in SLA;
6     Call FindClusters;
7     Generate initial Population;
8     Repeat
9         Calculate Fitness Values for all Individuals;
8         Sort Individuals based on Fitness Value;
9         If (Feasible Solution is Found)
10           Return Solution;
11       Else
12          Generate Next Population:
13               Select Operation;
14               CrossOver Operation;
15               Mutation Operation;
16    Until Termination Criteria Satisfied;
17 }
```

Algorithm 3. The proposed service composition algorithm

Gene and chromosome: In this problem, each gene stands for a selected concrete service for instantiating an abstract service. Also, infrastructure services are used for executing that service. A service can be indicated as $\text{Genome}_\text{i} = (S_{ij}, Vm_l, Net_k, Db_m)$ in which Vm, Net and Db as infrastructure services refer to virtual machine, network bandwidth and data storage, respectively, which are used for executing S_ij service. Each chromosome refers to an individual from the generation population which indicates a solution. Chromosome is a set of genes which is denoted by $chrom_n = (genome_1, genome_2, \ldots genome_m)$ in which m refers to the number of the abstract services of a workflow.

Fitness function: For solving the problem through the genetic algorithm, fitness degree should be used as a criterion for selecting an individual for the next generation of the population. In this method, fitness function specifies an individual's fitness according to the highness of positive qualities and the lowness of negative qualities. Furthermore, this function satisfies the constraints specified by the user in measuring the value of fitness as formulated in Eq. (6).

$$\text{fitness}\,(C_\text{n}) = \sum_{l=1}^{p} W_l.\text{NQ}_l\,(\text{CS}) \tag{6}$$

$$\mathrm{NQ}_l\,(\mathrm{CS}) = \begin{cases} \frac{Q_l(\mathrm{CS}) - Q_l(\mathrm{SLA})}{Q_l(\mathrm{SLA})} & \text{for negative attributes} \\ \frac{Q_l(\mathrm{SLA}) - Q_l(\mathrm{CS})}{Q_l(\mathrm{SLA})} & \text{for positive attributes} \end{cases} \tag{7}$$

In Eq. (6), C_n, W_l and $\mathrm{NQ}_1(\mathrm{CS})$ refer to chromosome, vector of the weights of qualitative attributes and the normalized value of the lth qualitative attribute of the composite service, respectively. Moreover, in Eq. (7), $Q_1(\mathrm{CS})$ and $Q_1(\mathrm{SLA})$ indicate the lth qualitative parameter of the composite service and the agreed value for the lth qualitative parameter of the composite service in the SLA contract, respectively.

Selection operator: Genetic evolutionary algorithm can lead us to producing the best generation if the best individuals from the current generations have more chance for going to the next generation. For doing so, roulette wheel algorithm was used for selecting the individuals. In this method, a cumulative probability is assigned to each individual with respect to its fitness value. Using this value, the selection chance of each individual is determined.

Crossover operator: For executing the crossover operator on a generation, at first, the current generation is randomly divided into two groups, i.e., father and mother. Then, using probability, a pair from the two sets is selected, and a random cut-off point is selected (Fig. 4). For producing a new pair as the children who inherit features from their parents, the genes above the cut-off point are repeated without any changes and the genes under it are exchanged with other genes. Then, the children's chromosomes are added to the population list. Figure 5 depicts an example of the crossover operator in the proposed method. For the sake of simplification, only the concrete service of each gene was illustrated, but the infrastructure services were not included in the figure.

Mutation operator: This operator was used for providing participation opportunity for the potential genes which could not participate in the initial generation of the population. This operator leads to the production of new generations without resulting in early convergence.
Consequently, evolution proceeds until an optimal or a near-optimal solution can be achieved. It selects a chromosome based on probability which is a concrete service of an abstract service and its infrastructure services. Then, the gene is randomly replaced with another gene (Fig. 6).

Fig. 4 Workflow of composite service request

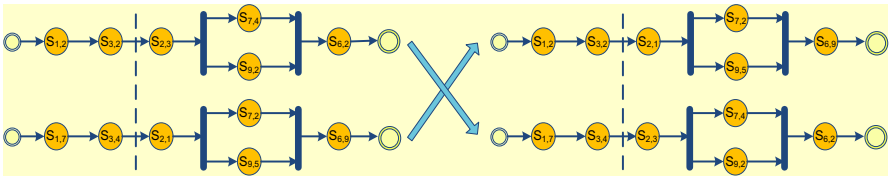

Fig. 5 An example of crossover operator

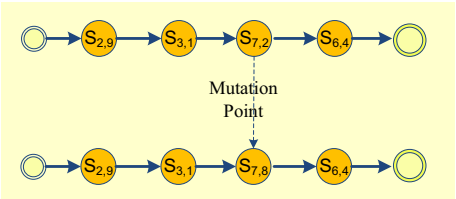

Fig. 6 An example of mutation operator

Execution and termination policy of the algorithm: The execution of this algorithm is repeated until the algorithm can satisfy the users' constraints most optimally. While executing this algorithm, if an optimal response is not produced, the algorithm will be repeated until the convergence is obtained and a near-optimal response is produced. However, if the convergence is not achieved after the maximum number of repetitions, it can be argued that it is not feasible to construct an acceptable composite service within this system for the requested service.

3.4 Maintenance phase

In this phase, the selected composite service is executed and monitored. In case a problem occurs, the composite service will be repaired, recovered, recomposed and executed. We have considered an online explore unit in this system which searches service UDDIs over the public cloud systems. If a new service is found, based on distance from the cluster centers, it will be added to one of the related clusters. If a specific threshold is met, updating and reorganizing the database including service clusters, SLA clusters and association rules are done in an offline way. As a case in point, the experimental results reveled that when concrete services of an abstract service and SLAs increase for more than one percent, services and SLAs should be re-clustered. Also, in our experiments, when composite services grow for one percent, association rules should be re-extracted.

4 Simulations

4.1 Simulation environment

In this section, the characteristics of the simulation environment and the data used in the present study are described. The proposed method was simulated by *VC#.net 2013* language. The hardware used for simulating the method was an *Intel Core 2 Duo, 2.5 GHZ* processor with 3 GB RAM memory, and the utilized operating system was Windows XP.

Regarding the production of datasets, since there were not any benchmarked datasets, we used the QWS dataset in the simulation which was collected by Al-Masri and Quasay [58]. The QWS dataset has been frequently used by several researchers in this field such as [17,18,21]. It includes 10 qualitative attributes for 2506 web services. In fact, this dataset for testing the hypothesis of the study, i.e., the investigation of the impact of local selection via clustering on the improvement of service composition

Table 5 Distribution of services on clusters

Clusters	C0	C1	C1	C2	C3	C4	C5	C6	C7	C8	C9	Total
Service types												
S1	304	15	15	75	193	87	54	19	38	26	159	970
S2	60	101	101	116	17	298	25	33	195	43	219	1107
S3	91	62	62	207	21	215	133	21	23	31	289	1093
S4	14	106	106	38	126	44	296	81	207	10	178	1100
S5	173	238	238	12	43	89	193	97	20	16	28	909
S6	129	63	63	19	28	303	206	16	18	101	220	1103
S7	14	106	106	38	126	44	296	81	207	10	178	1100
S8	16	301	301	200	103	97	244	33	179	14	14	930
S9	16	301	301	200	103	97	244	33	179	14	14	930
S10	26	374	374	199	12	42	19	8	98	258	26	1062

time complexity, we needed a dataset including much more services, we expanded the number of services to 10,000 services for which we used a controlled random generation methodology based on normal distribution. We selected three quality attributes of availability, response time and successability from QWS services and calculated price for them within the intervals of 5 to 100 dollar depending on three other qualities. Algorithm (4) has used for creating a service for the expanding of QWS. In this dataset, 10 types of abstract services were considered, each of which was clustered within 10 clusters by means of the K-means algorithm. The clusters are shown in Table 5. The set of used data included only web services (software). For infrastructure services (virtual machine, network bandwidth and database), three sets of services including 100 services and 4 above-mentioned attributes were randomly produced.

```
1 Service Random Service Generator ( )
2 {
3    Generate Rt as Response Time between minimum and maximum value of
     Response time attribute in QWS.
4    Find range (Amin and Amax) of Availability attribute in QWS for
     services with Response time near to RT
5    Generate Av as Availability  between Amin and Amax
6    Find range (Smin and Smax) of Successability  attribute in QWS for
     services with Response time near to RT  and  Availability  near to AV
7    Generate Sc as Successability  between Smin and Smax
8    Produce Price based on Availibility, Response Time and Successability
     values
9 }
```

Algorithm 4 . Controlled random service generator method

Table 6 Distribution of SLAs on clusters

SLA clusters	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	Total
SLA	206	216	207	189	208	198	192	207	182	195	2000

For the workflows, we designed 50 samples, ranging from simple (linear) samples to complex (including loop and parallel service execution) samples. However, since we could not find a real set for the SLA contracts and inasmuch as we did not get any results from communicating and corresponding with business companies which provide cloud services, an SLA dataset consisting of 2000 sample SLAs was randomly produced. It was clustered through K-means algorithm (Table 6).
The simulations were planned and carried out for testing and substantiating the following three hypotheses:

- Hypothesis one: for realizing composite service requests with similar qualitative requirements included within identical SLA clusters, most services are selected from the same service clusters.
- Hypothesis two: due to the search space reduction of the problem, the time complexity of the algorithm will be better than other algorithms.
- Hypothesis three: since all similar and appropriate services of a request are clustered in a service cluster, the obtained fitness value is higher at a certain time or for a certain number of repetitions. Consequently, it results in a higher user satisfaction.

4.2 Experiments and evaluation of the results

For evaluating the advantages of the proposed method (CGA), it was analyzed and compared with service composition method with normal genetic algorithm NGA [16] and SCA [40] with skyline-based genetic algorithm. The efficiency of the proposed method was compared with the above-mentioned methods in terms of different parameters. The population size for all genetic algorithms was 500. The convergent criterion for all genetic algorithms was that the best fitness value does not improve over the last 100 iterations.

4.2.1 Distribution of service selection on clusters

This experiment was related to testing the first hypothesis. Indeed, if requests for identical composite services have similar SLA contracts, i.e. they are located within the same cluster, then the concrete services selected for producing those composite services will be placed in the same cluster. As a result, two workflows were produced, and the real service clusters of each abstract service were once clustered within five clusters and for the second time, they were clustered within ten clusters. The experiment was repeated for 100 times by randomly selecting SLA contracts from two different clusters. In this experiment, normal genetic algorithm was used without clustering. After

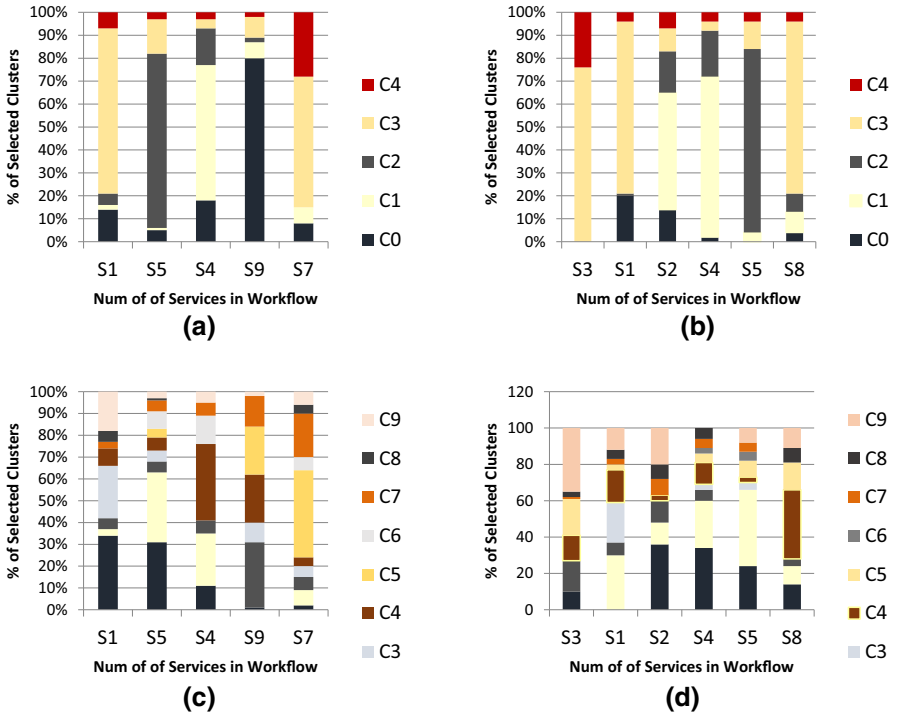

Fig. 7 Distribution of selected services on clusters

selecting services, we specified which services belong to which clusters. As shown in Fig. 7a, b, this experiment indicated that in case the number of service clusters is assumed to be five, the absolute majority of the selected services will belong to one cluster. Nevertheless, as depicted in Fig. 7c, d, if there are ten clusters, the relative majority of selected services will be from one or two clusters. Our analysis shows that in a system with five clusters on average, 69 % of services are selected from a cluster and in a system with ten clusters on average, 60 % of services are selected from two clusters. These results confirm hypothesis one.

4.2.2 Time and speed of service composition

For testing the second hypothesis of the study, the researchers designed and implemented experiments so as to investigate and compare service composition time in the proposed method with those of the above-mentioned methods. Hence, the experiment was carried out on six different workflows and was repeated for ten times on each different workflow, and the average efficiency was computed for each of them. The same experiment was also implemented for SGA and NGA methods. As illustrated in Fig. 8, it can be observed that the average service composition time for the proposed method was less than those for the other methods. Although service composition time increases as the number of requested services in the workflows increases, the slope for

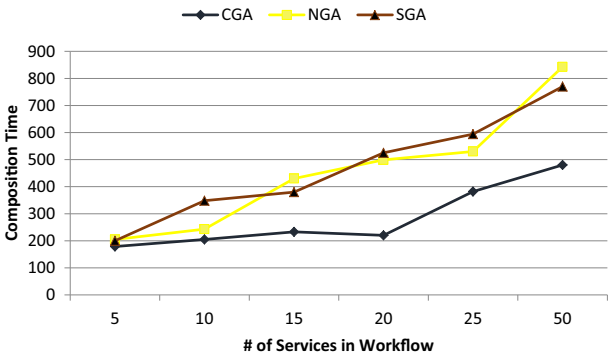

Fig. 8 Service composition time

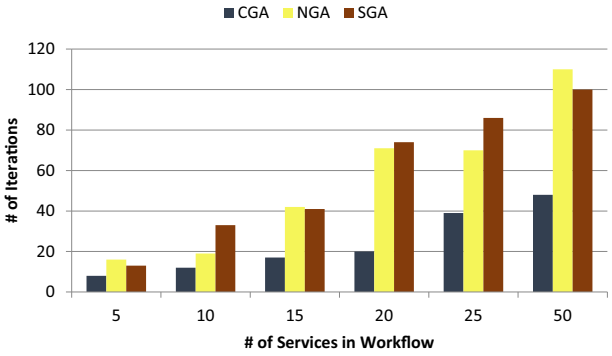

Fig. 9 Number of the repetitions of the genetic algorithm

the service composition time in the proposed method was less than those of the other methods. Moreover, it was found that service composition time of the proposed method did not have a regular linear growth. That is, when the number of abstract services increases significantly, slope of the figure does not increase accordingly. Indeed, this is attributed to the random nature of the genetic algorithm. Furthermore, the results indicate that the proposed method is a scalable method, and its efficiency does not decrease as the number of concrete services increases. Figure 9 indicates that the convergence speed of the proposed method is better than those of other two methods. Despite the fact that the number of repetitions for reaching a reply increases along with an increase in the number of abstract services in the workflow, the performance of the proposed method is better than that of the other two methods.

4.2.3 Optimality evaluation

For testing and substantiating the third hypothesis of the study, we designed experiments through which the fitness value of the selected solution of the proposed method can be analyzed and compared with those of the two other algorithms. This experiment was conducted within the same conditions of the preceding experiment. Figure 10

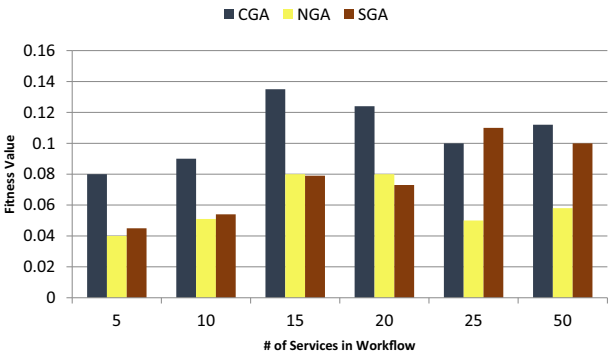

Fig. 10 Fitness value

shows the results of this experiment. In addition to the high significance of speed in achieving a response, it should be noted that attaining a response with high optimality can enhance customer satisfaction. In other words, the result indicates that as the number of abstract services in the workflow increases, the efficiency of the proposed method does not decrease in terms of the optimality degree of the solutions.

It should be noted that some of the results of the NGA method implemented in this study which are reported in Figs. 8 and 9 are different from the results published in [16]. This diversity is attributed to the difference between range of data in our dataset and their dataset. Also, our fitness function is different from their function which may affect the results shown in Fig. 10.

We compared the proposed method with other methods in terms of composition time, convergence speed and the degree of fitness of the produced composite service. Regarding the obtained average times for different methods in relation to the number of services, it was found that the average composition execution time for the proposed method was 343.62; however, the obtained values for the other methods were 656.25 and 637.87. Such a difference in average time between the proposed method and the other methods indicates the significant efficacy of the proposed method on optimizing composition time. Convergence speed was experimented and investigated with the number of repetitions. The average number of repetitions in the proposed method was 24, and in the other methods, it was 54 and 58, respectively. Indeed, the reason for clustering and selecting service clusters is the fitness with user request which is obtained from the association rules. These results indicate the superiority of the proposed method to other methods with regard to convergence speed. The average fitness value which was obtained for the composite services by the proposed method and the other two methods was 0.107, 0.06 and 0.077, respectively. These results indicate the superiority of the degree of optimality of the proposed method to the other methods. The reason for which is the accumulation of the population of similar services in clusters and the selection of appropriate clusters as the input of the genetic algorithm. Since the selection of the genetic algorithm population is carried out in a guided way rather than randomly, hence, the produced composite services have a high degree of optimality.

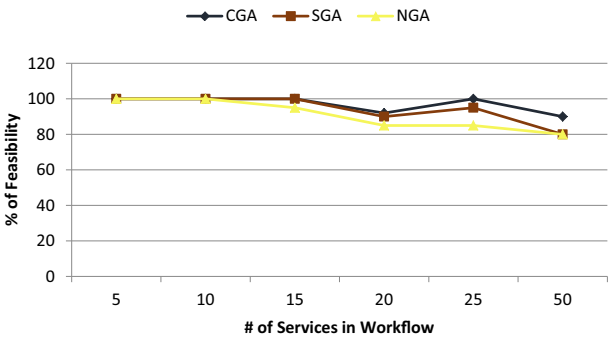

Fig. 11 Feasibility

4.2.4 Feasibility rate

In addition to optimality and convergence speed of the service composition algorithm, it can be argued that having a great chance of reaching a reasonable response in a method is also important. For examining and testing the proposed method with respect to the probability of success and feasibility, six workflows were defined. Then, the proposed method and the two other methods were tested for each workflow for 50 times under identical qualitative criteria. Figure 11 indicates that the feasibility of the proposed method is higher than that of SGA and NGA. The reason for this result is that the proposed method selects services from experienced clusters. Thus, success probability within a specific time or specific number of repetitions is significantly enhanced.

4.2.5 Impact of clustering population

The preceding experiments revealed that, due to search space reduction of the genetic algorithm, clustering can lead to the improvement of this algorithm in terms of service composition time, optimality, convergence speed and the degree of feasibility. The question arising at this point is that how many more clusters can we increase and how much can we reduce the space? Is there an intermediate point for the number of clusters? For finding answers to these questions, we designed an experiment in which concrete services of each abstract service were clustered within 0, 5, 10, 20 and 30 clusters in distinct datasets. Then, using six different workflows, the experiments were conducted on time, optimality and convergence speed of the dataset. As shown in Figs. 12 and 13, it can be observed that even in case the clustering number is low, the performance of the proposed method is better than the normal genetic algorithm. It should be pointed out that the efficiency is enhanced as the number of clusters increases. As a case in point, when there were 10 and 20 clusters for concrete services, service composition time and convergence speed of the proposed algorithm were better than the mode with 5 clusters. However, the performance of the algorithm decreased when there were 30 clusters. Thus, it should be highlighted that selecting an appropriate number of clusters has a remarkable impact on the efficiency of the algorithm. In

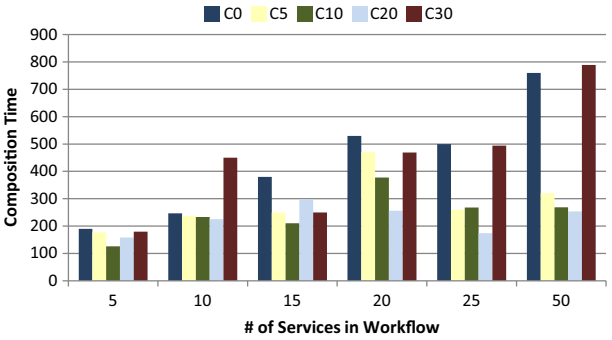

Fig. 12 Composition time for each clustering type each clustering

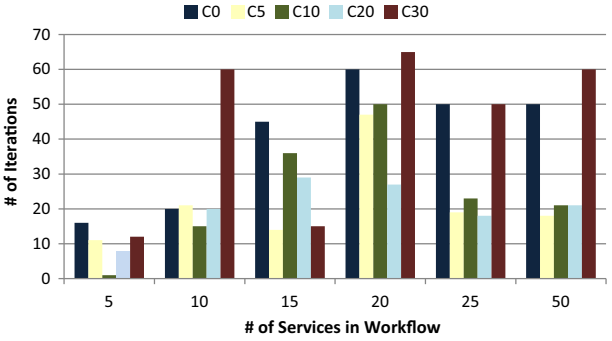

Fig. 13 Number of iterations for each clustering type

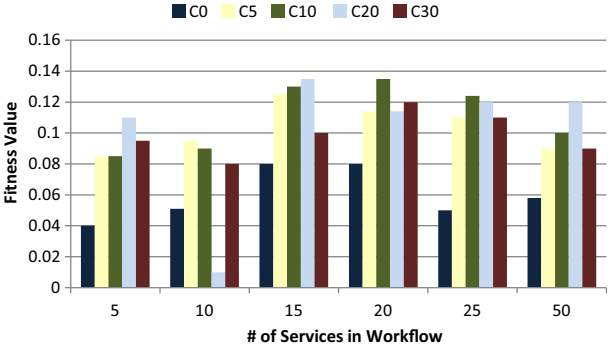

Fig. 14 Fitness values for each clustering type

contrast, as depicted in Fig. 14, the number of clusters has no significant impact on the fitness value of the solutions obtained by the proposed method.

To evaluate the significance of clustering impact on the fitness values of solutions, we used the ANOVA (analysis of variance) as a statistical test. In this test, each group refers to a specific fitness values per different cluster numbers. Table 7 shows the results of ANOVA test. As Table x indicates, the obtained P value (0.04) and the

Table 7 ANOVA test for time complexity

ANOVA					
Source of variation	SS	*df*	MS	*F*	*P* value
Between groups	251,803.6	4	62,950.9	2.87	0.04
Within groups	546,826.6	25	21,873.06		
Total	798,630.2	29			

Table 8 ANOVA test for evaluating impact of clustering on fitness value

ANOVA					
Source of variation	SS	*df*	MS	*F*	*P* value
Between groups	0.00965	4	0.002413	3.68	0.01
Within groups	0.016367	25	0.000655		
Total	0.026017	29			

Table 9 ANOVA test for evaluating impact of cluster numbers on fitness value

ANOVA					
Source of variation	SS	*df*	MS	*F*	*P* value
Between groups	0.000445	3	0.000148	0.198064	0.89
Within groups	0.014983	20	0.000749		
Total	0.015428	23			

obtained *F* ratio (2.87) confirm a significant difference between the composition time using different cluster numbers. Therefore, the null hypothesis is rejected, and hence, the clustering has significant impact on the composition time. The results shown in Table 7 confirms hypothesis x.

To show that use of clustering has significant impact on the fitness values of solutions, we used the ANOVA test. In this test, each group refers to a specific fitness values per different cluster numbers. Table 8 illustrates the results of ANOVA for groups of experimental data. As Table x indicates, the obtained *P* value (0.01) and the obtained *F* ration (3.68) confirm a significant difference between the fitness values of obtained solution with and without clustering. Therefore, the null hypothesis is rejected, and hence, the clustering has significant impact on the fitness values. The results shown in Table 8 confirm hypothesis x.

To evaluate the effects of number of clustering on the fitness values of the solutions, we used the ANOVA test; each group refers to a specific fitness value per different cluster numbers. As Table 9 shows, the obtained *P* value (0.89) and the obtained *F* ration (0.19) confirm a non-significant difference between the fitness values of different cluster numbers. Therefore, the null hypothesis is accepted, and hence, the number of clustering has not significant impact on the fitness values.

5 Conclusion and directions for further research

In this paper, an attempt was made to identify the requirements of service composition based on SLA contract in cloud computing environments. Inasmuch as services have been remarkably enhanced in cloud environment and customers are concerned with the speed of requested service delivery as well as service quality, researchers in this study used data mining techniques such as clustering, association rules in service composition and genetic algorithm for reducing the search space of a problem. Based on the findings of the study, it can be concluded that these techniques can result in the reduction of service composition time and the enhancement of the optimality of compound services. Furthermore, the results of the study indicated that the proposed method is scalable; hence, it is highly appropriate for dynamic cloud environments. Nevertheless, in this study, sufficient attention was not given to fitness function which should be addressed in future studies. Although the total fitness value of a composite service is good, some qualitative features might sometimes bias other features which can reduce customer satisfaction. As a direction for further research, fuzzy logic might be used in the fitness function to handle this problem. Also, optimizing the clustering method might enhance the efficiency of the proposed method which should be addressed in further research.

References

1. Buyya R, Yeo CS, Venugopal S, Broberg J, Brandic I (2009) Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. Future Gener Comput Syst 25(6):599–616
2. Liu F, Tong J, Mao J, Bohn R, Messina J, Badger L, Leaf D (2011) NIST cloud computing reference architecture. NIST Spec Publ 500:292
3. Garg SK, Versteeg S, Buyya R (2013) A framework for ranking of cloud computing services. Future Gener Comput Syst 29(4):1012–1023
4. Pressman RS (2005) Software engineering: a practitioner's approach. Palgrave Macmillan
5. Chiu D, Deshpande S, Agrawal G, Li R (2009) A dynamic approach toward QoSAware service workflow composition. In: IEEE International Conference on Web Services, pp 655–662
6. Joshi KP, Yesha Y, Finin T (2014) Automating cloud services life cycle through semantic technologies. IEEE Trans Serv Comput 7(1):109–122. doi:10.1109/TSC.2012.41
7. Rosenberg F, Celikovic P, Michlmayr A, Leitner P, Dustdar S (2009) An end-to-end approach for QoS-aware service composition. In: Enterprise Distributed Object Computing Conference, 2009. EDOC'09. IEEE International. IEEE, pp 151–160
8. Teixeira M, Ribeiro R, Oliveira C, Massa R (2015) A quality-driven approach for resources planning in service-oriented architectures. Expert Syst Appl 42(12):5366–5379
9. Karim R, Ding Chen, Miri A (2013) An end-to-end QoS Mapping approach for cloud service selection. In: 9th World Congress on Services, pp 341–348
10. Baset Salman A (2012) Cloud SLAs: present and future. ACM SIGOPS Oper Syst Rev 46(2):57–66
11. Kofler K, Haq Iu, Schikuta E (2010) User-Centric, heuristic optimization of service composition in clouds. In: International Conference on Euro Parallel Processing. Springer, Berlin, Heidelberg, pp 405–417
12. Kofler K, Schikuta E (2009) A parallel branch and bound algorithm for workflow QoS optimization. In: IEEE International Conference on Parallel Processing, pp 478–485
13. Huang J, Liu Y, Duan Q (2012) Service provisioning in virtualization-based cloud computing: modeling and optimization. In: GLOBECOM, pp 1710–1715

14. Yong Z, Wei L, Junzhou L, Xiao Z (2012) A novel two-phase approach for QoS-aware service composition based on history records. In: 5th IEEE International Conference on Service-Oriented Computingand Applications, pp 1–8
15. Jula A, Sundararajan E, Othman Z (2014) Cloud computing service composition: a systematic literature review. Expert Syst Appl 41(8):3809–3824
16. Ye Z, Zhou X, Bouguettaya A (2011) Genetic algorithm based QoS-aware service compositions in cloud computing. In: International Conference Database Systems for Advanced Applications. Springer, Berlin, Heidelberg, pp 321–334
17. Moscato F, Mazzocca N, Vittorini V, Lorenzo GD, Mosca P, Magaldi M (2005) Workflow pattern analysis in web services orchestration: the BPEL4WS example. In: International Conference on High-Performance Computing and Communications, pp 395–400
18. da Silva AS, Ma H, Zhang M (2016) Genetic programming for QoS-aware web service composition and selection. Soft Comput 1–17. doi:10.1007/s00500-016-2096-z
19. Yu Y, Ma H, Zhang M (2013) An adaptive genetic programming approach to qos-aware web services composition. In: Evolutionary computation (CEC), 2013 IEEE Congress. IEEE, pp 1740–1747
20. AlSedrani A, Touir A (2016) Web service composition in dynamic environment: a comparative study. Comput Sci Inf Technol 75–84. doi:10.5121/csit.2016.60508
21. Yu Y, Ma H, Zhang M (2014) A genetic programming approach to distributed qos-aware web service composition. In: 2014 IEEE Congress on Evolutionary Computation (CEC). IEEE, pp 1840–1846
22. AllamehAmiri Mohammad, Derhami Vali, Ghasemzadeh Mohammad (2013) QoS-Based web service composition based on genetic algorithm. J AI Data Min 1(2):63–73
23. Liu B, Meng P (2008) Hybrid algorithm combining ant colony algorithm with genetic algorithm for continuous domain. In: Young computer scientists, 2008. ICYCS 2008. The 9th International Conference. IEEE, pp 1819–1824
24. Zhao Z, Hong X, Wang S (2015) A web service composition method based on merging genetic algorithm and ant colony algorithm. In: Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), 2015 IEEE International Conference. IEEE, pp 1007–1011
25. ZHAO C, WANG J, Qin Jie, Zhang W-Q (2014) A hybrid algorithm combining ant colony algorithm and genetic algo-rithm for dynamic web service composition. Open Cybern Syst J 8:146–154
26. Elbeltagi E, Hegazy T, Grierson D (2005) Comparison among five evolutionary-based optimization algorithms. Adv Eng Inf 19(1):43–53
27. Zhao C-Y, Wang J-L, Qin J, Zhang W-Q (2014) A hybrid algorithm combining ant colony algorithm and genetic algorithm for dynamic web service composition. Open Cybern Syst J 8:146–154
28. Gao F, Curry E, Ali MI, Bhiri S, Mileo A (2014) Qos-aware complex event service composition and optimization using genetic algorithms. In: International Conference on Service-Oriented Computing. Springer, Berlin, Heidelberg, pp 386–393
29. 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
30. Rostami NH, Kheirkhah E, Jalali M (2014) An optimized semantic web service composition method based on clustering and ant colony algorithm. arXiv:1402.2271
31. 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. doi:10.1007/s10845-016-1215-0
32. Mabrouk NB, Beauche S, Kuznetsova E, Georgantas N, Issarny V (2009) QoS-aware service composition in dynamic service oriented environments. In: International Conference on Middleware. Springer, Berlin, Heidelberg, pp 123–142
33. Xia Y, Chen P, Bao L, Wang M, Yang J (2011) A QoS-aware web service selection algorithm based on clustering. In: IEEE International Conference on Web Services (ICWS), pp 428–435
34. Deng SY, Du YY (2013) Web service composition approach based on service cluster and Qos. J Comput Appl 33(8):2167–2166
35. Ardagna D, Pernici B (2006) Global and local QOS quarantee in web service selection. In: Business Process Management Workshops, pp 32–46
36. Sun SX, Zhao J (2012) A decomposition-based approach for service composition with global QoS quarantees. J Inf Sci 199(1):138–153
37. Alrifai M, Risse T, Nejdl W (2012) A hybrid approach for efficient web service composition with end-to-end QoS constraints. ACM Trans Web 6(2):7

38. 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. doi:10.1016/j.knosys.2016.01.010
39. Tao F, LaiLi Y, Xu L, Zhang L (2013) FC-PACO-RM: a parallel method for service composition optimal-selection in cloud manufacturing system. IEEE Trans Ind Inf 9(4):2023–2033
40. Wang D, Yang Y, Mi Z (2014) A genetic-based approach to web service composition in geo- distributed cloud environment. Comput Electr Eng 43:129–141. doi:10.1016/j.compeleceng.2014.10.008
41. Jula A, Othman Z, Sundararajan E (2013) A hybrid imperialist competitive-gravitational attraction search algorithm to optimize cloud service composition. In: IEEE Workshop on Memetic Computing, pp 37–43
42. Fan X-Q (2013) A decision-making method for personalized composite service. Expert Syst Appl 40(15):5804–5810
43. Xu Y, Yin J, Deng S, Xiong NN, Huang J (2016) Context-aware QoS prediction for web service recommendation and selection. Expert Syst Appl 53:75–56. doi:10.1016/j.eswa.2016.01.010
44. Kurdi H, Al-Anazi A, Campbell C, Al Faries A (2015) A combinatorial optimization algorithm for multiple cloud service composition. Comput Electr Eng 42:107–113
45. Yu Q, Chen L, Li B (2015) Ant colony optimization applied to web service compositions in cloud computing. Comput Electr Eng 41:18–27
46. Xia Y, Chen P, Bao L, Wang M, Yang J (2011) A QoS-aware web service selection algorithm based on clustering. In: Web services (ICWS), 2011 IEEE International Conference. IEEE, pp 428–435
47. Newcomer E, Lomow G (2005) Understanding SOA with web services. Addison-Wesley, Boston, Mass
48. Ghazanfari M, Alizadeh S, Teymurpour B (2013) Data mining and knowledge discovery. Science and Technology Publication Co, 1st edn. ISBN: 9789644541780
49. Agrawal R, Srikant R (1994) Fast algorithms for mining association rules. In: Proceedings of 20th International Conference on Very Large Data Bases, VLDB, vol 1215, pp 487–499
50. Han Jiawei, Kamber Micheline, Pei Jian (2011) Data mining: concepts and techniques. Elsevier, Amsterdam
51. Bianco P, Lewis GA, Merson P (2008) Service Level agreements in service-oriented architecture environments. Technical note CMU/SEI-2008-TN-021. http://www.sei.cmu.edu
52. van Steen M, Tanenbaum AS (2007) Distributed systems: principles and paradigms
53. Nai-zhong WU (2013) Dynamic composition of web service based on cloud computing. Int J Hybrid Inf Technol 6(6):389–398
54. Mitchell Melanie (1998) An introduction to genetic algorithms. MIT Press, Cambridge
55. Wright AH (1991) Genetic algorithms for real parameter optimization. Found Genet Algorithms 1:205–218
56. Lim SP, Haron H (2013) Performance comparison of genetic algorithm, differential evolution and particle swarm optimization towards benchmark functions. In: Open Systems (ICOS), 2013 IEEE Conference. IEEE, pp 41–46
57. Hassan R, Cohanim B, De Weck O, Venter G (2005) A comparison of particle swarm optimization and the genetic algorithm. In: Proceedings of the 1st AIAA Multidisciplinary Design Optimization Specialist Conference, pp 18–21
58. http://www.uoguelph.ca/~qmahmoud/qws/