Soft Comput (2018) 22:8353–8378
https://doi.org/10.1007/s00500-017-2783-4

METHODOLOGIES AND APPLICATION

CSA-WSC: cuckoo search algorithm for web service composition in cloud environments

Mostafa Ghobaei-Arani[1] · **Ali Asghar Rahmanian**[2] · **Mohammad Sadegh Aslanpour**[3] · **Seyed Ebrahim Dashti**[3]

Published online: 21 August 2017

Abstract In recent years, service-based applications are deemed to be one of the new solutions to build an enterprise application system. In order to answer the most demanding needs or adaptations to the needs of changed services quickly, service composition is currently used to exploit the multi-service capabilities in the Information Technology organizations. While web services, which have been independently developed, may not always be compatible with each other, the selection of optimal services and composition of these services are seen as a challenging issue. In this paper, we present cuckoo search algorithm for web service composition problem which is called 'CSA-WSC' that provides web service composition to improve the quality of service (QoS) in the distributed cloud environment. The experimental results indicate that the CSA-WSC compared to genetic search skyline network (GS-S-Net) and genetic particle swarm optimization algorithm (GAPSO-WSC) reduces the costs by 7% and responding time by 6%, as two major reasons for the reduction of improvement of the quality of service. It also increases provider availability up to 7.25% and the reliability to 5.5%, as the two important QoS criteria for improving the quality of service.

Keywords Cloud computing · Web service composition · Quality of service · Cuckoo search algorithm

Communicated by V. Loia.

✉ Mostafa Ghobaei-Arani
mostafaghobaye@yahoo.com; m.ghobaei@qom-iau.ac.ir

✉ Ali Asghar Rahmanian
ali.a.rahmanian@ieee.org

Mohammad Sadegh Aslanpour
aslanpour.sadegh@gmail.com

Seyed Ebrahim Dashti
sayed.dashty@gmail.com

[1] Department of Computer Engineering, Qom Branch, Islamic Azad University, Qom, Iran

[2] Department of Computer Science and Engineering and IT, College of Electrical and Computer Engineering, Shiraz University, Shiraz, Iran

[3] Department of Computer Engineering, Jahrom Branch, Islamic Azad University, Jahrom, Iran

1 Introduction

In the cloud computing environment, computation is not performed on the local computers, but it is done by the cloud servers located in data centers, which provide infrastructure, software and platform as Internet-based service. In fact, the goal of cloud computing is to integrate hardware and software as a service accessible to users through the Internet (Aslanpour et al. 2017; Buyya et al. 2010; Ghobaei-Arani et al. 2016, 2017a). The rapid development of cloud computing leads to publishing many different web services throughout the world (Ghobaei-Arani and Shamsi 2015). Nowadays, one of the new solutions to build an enterprise application system is the service-based applications. Also, service-oriented architecture is seen as a leading method in the architecture of enterprise solutions. It provides a rapid response to changes, requirement and service adaptation, and demands several services available in the organization. For this reason, it is necessary to identify the key elements of service-oriented architecture (SOA) and restrictions of service composition support, composition verification and automated composition (Simon et al. 2013; Piprani et al. 2013). The quality of service (QoS) of web services refers to different non-functional properties such as response time, availability and

reliability. For a combination request, several candidate services can be achieved so that they provide similar functions with different QoS. Regarding the non-functional properties of QoS-based service, web services composition engine establishes the best candidate services from the collection services (Portchelvi et al. 2012). By increasing demands of cloud service customers, cloud services have been expanded and therefore they have been being grown rapidly (Rahmanian et al. 2017). Since the capacity of a data center is limited, to achieve more stable services, it seems that the load distribution over global data centers can be a suitable approach (Ghobaei-Arani et al. 2017b). Most web services are established on cloud data centers distributed geographically around the world. The cloud data centers interdepend on the network to communicate with each other and cloud users. The network communication may have some effects on the performance of service provided by the involved data centers. The network QoS is a significant metric for web service composition problem, and also it is important to avoid violating the service-level agreement (SLA) during the service composition process (Gholami and Ghobaei-Arani 2015; Jula et al. 2014). In this paper, we propose a new algorithm for web service composition problem in the geographically distributed cloud environment. In proposed algorithm, we used the cuckoo search technique for web service composition problem; we then evaluated the response time, cost, availability and reliability of the composition process as four major QoS criteria. The service composition problem is known as an NP-hard optimization problem in that a larger service is provided by services integrating processes. In many cases, the web services are deployed by multiple service providers, and no single service can satisfy a user's needs. Therefore, we need a service composition algorithm to combine several web services from different service providers and fulfill the user's requirements. In this paper, we apply a new evolutionary optimization algorithm called cuckoo search algorithm (CSA) (Rajabioun 2011) for web service composition problem in the geographically distributed cloud environment called CSA-WSC. The CSA is a novel nature inspired by lifestyle of a bird family called cuckoo that is appropriate for where there is incomplete information regarding the environment and in dynamic, complex, or random environments with a large number of uncertainties such as cloud environments, and it outperforms compared to other well-known algorithms like particle swarm optimization (PSO), ant colony optimization (ACO), genetic algorithm (GA).

The main contributions of this research can be summarized as follows:

- We present a meta-heuristic algorithm called CSA-WSC to reduce the complexity of the approach compared with other algorithms.
- Our algorithm considers the distributed network environment. Also, our algorithm considers not only the QoS of the web services but also the network QoS.
- The cuckoo algorithm applied has more advantages such as faster higher convergence speed, higher precision, ability to search for local as well as global search, much less likely trapped in local optimum points.

The rest of this paper is organized as follows: In Sect. 2, we focus on a survey of related works. Section 3 provides the necessary backgrounds for the proposal of this paper. In Sect. 4, we describe the proposed method. Section 5 presents an experimental evaluation of the proposal, and we finally present the conclusions and future works in Sect. 6.

2 Related works

This section will refer to some research about service composition in the cloud environment.

Zhou and Yao (2016) proposed a hybrid artificial bee colony (HABC) algorithm for optimal selection of QoS-based cloud manufacturing service composition. The HABC algorithm employs both the probabilistic model of Archimedean copula estimation of distribution algorithm (ACEDA) and the chaos operators of a global best-guided artificial bee colony to generate the offspring individuals with consideration of quality of service and cloud manufacturing environment. Experimental results have shown that the HABC algorithm could find better solutions compared with such algorithms as a genetic algorithm, particle swarm optimization and basic artificial bee colony algorithm.

Yu et al. (2015) presented an ant colony optimization-based algorithm for web service composition called ACO-WSC, which attempts to select cloud combinations that are feasible and use the minimum number of clouds. In the ACO-WSC algorithm, artificial ants travel on a logical digraph to construct cloud combinations. The ACO-WSC algorithm achieves a superior tradeoff between time and quality, and it is a practical solution for deploying in multi-cloud web service provision environments.

Wang et al. (2015) developed a genetic-based approach to web service composition problem in a geo-distributed cloud environment so that simultaneously minimizing SLA violations. To deal with cases where cloud datacenters are located geographically, a QoS-based composition model for cloud network environment is considered.

Seghir and Khababa (2016) a hybrid genetic algorithm (HGA) to solve the QoS-aware cloud service composition is proposed. The HGA combines two phases to perform the evolutionary process search, including genetic algorithm phase and fruit fly optimization phase. In genetic algorithm phase, the global search process is performed, while the local

search process was carried out by the fruit fly optimization phase.

Chen et al. (2016) designed a flexible QoS-aware web service composition (QWSC) method by multi-objective optimization in cloud manufacturing. The QWSC method considers both the QoS performance and the QoS risk constraints as the optimization objectives. Moreover, a ε-dominance multi-objective evolutionary algorithm (EDMOEA) is developed to solve the large-scale QWSC.

Karimi et al. (2016) proposed a QoS-aware service composition approach in cloud computing using data mining techniques and genetic algorithm. In their proposed approach, a genetic algorithm was used to achieve global optimization about 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.

Kurdi et al. (2015) proposed a novel combinatorial optimization algorithm for cloud service composition (COM2) that can efficiently utilize multiple clouds. The COM2 algorithm ensures that the cloud with the maximum number of services will always be selected before other clouds, which increases the possibility of fulfilling service requests with minimal overhead.

Liu and Zhang (2016) an approach of synergistic elementary service group-based service composition (SESG-SC) for QoS-aware service composition problem in cloud manufacturing is presented. Their approach releases the assumption of a one-to-one mapping between elementary services and subtasks, allowing a free combination of multiple functionally equivalent elementary services into a synergistic elementary service group (SESG) to perform each subtask collectively, thereby bettering the overall QoS and achieving more acceptable success rate.

Qi et al. (2017) applied a knowledge-based differential evolution (KDE) algorithm to solve web service composition optimizing problem in cloud environments. Their algorithm improves the accelerate convergence velocity by importing structure knowledge. The simulation results indicate the effectiveness of KDE in solving a complex optimization problem with large-scale solution space compared with the original differential evolution, particle swarm optimization algorithms.

Wang et al. (2013) presented a particle swarm optimization approach with skyline operator for fast cloud-based web service (CWS) composition in cloud environments. Their approach adopts skyline operator to prune redundant CWS candidates and then employs particle swarm optimization (PSO) to select CWS from amount of candidates for composing single service into a more powerful composite service.

Lartigau et al. (2015) described a method based on QoS evaluation along with the geo-perspective using an improved artificial bee colony (ABC) optimization algorithm in cloud manufacturing. The modification of the original ABC initialization gives certain advantages at the start, targeting the neighborhood directly around the shortest transportation path. Also from a computational efficiency perspective, their proposed algorithm is surely faster than PSO and GA optimization.

Huo et al. (2015) proposed the discrete Gbest-guided artificial bee colony (DGABC) algorithm for cloud service composition which simulates the search for the optimal service composition solution through the exploration of bees for food. The DGABC algorithm has advantages in terms of the quality of solution and efficiency compared with other algorithms, especially for large-scale data.

Klein et al. (2014) developed the self-adaptive network-aware approach to service composition in cloud environments. Their approach employs a self-adaptive genetic algorithm which balances the optimization of latency and other QoS as needed, and it handles the QoS of services and the QoS of the network independently, improves the convergence speed and finds compositions with low latency for a given execution policy.

Zhao et al. (2015) presented a systematic approach based on a fuzzy preference model and evolutionary algorithms for SLA-constrained service composition problem. Authors model this multi-objective optimization problem using the weighted Tchebycheff distance rather than a linear utility function and present a fuzzy preference model for preference representation and weight assignment. Also, they present two evolutionary algorithms (EA), single_EA and hybrid_EA, that attempt to determine different types of optimization solutions for service composition. Faruk et al. (2016) presented a unique heuristic method to resolve the QoS-aware cloud service selection using an enhanced genetic particle swarm optimization (GPSO) algorithm which is broadly utilized to crack hefty scale optimization issues. Also, the adaptive non-uniform mutation (ANUM) approach is proposed to attain the best particle globally to boost the population assortment on the motivation of conquering the prematurity level of GPSO algorithm.

Table 1 sums up some of the most relevant works related to web service composition techniques in cloud environments based on four characteristics: (1) optimization techniques used, (2) QoS criteria, (3) advantages and (4) disadvantages.

3 Preliminary

In this section, we first introduce the QoS-based web services, location-based web services and composition services. We then provide a brief overview of the cuckoo search algorithm.

Table 1 Survey of works related to web service composition techniques in cloud environment

Refs	Optimization technique	QoS criteria	Advantages	Disadvantages
Zhou and Yao (2016)	Bee colony	Time, cost, availability, reliability	High efficiency, high stability	High time complexity
Yu et al. (2015)	Bee colony	Execution time, cost	High efficiency	Low scalability
Wang et al. (2015)	Skyline-based genetic algorithm	Response time, price, availability, reliability	Low computation time, high scalability	Incompatibility
Seghir and Khababa (2016)	Genetic algorithm+fruit fly	Time, price, availability, reliability	High-speed convergence, low computation time	High cost
Chen et al. (2016)	Genetic algorithm+evolutionary algorithm	Cost, execution time, latency, availability, reliability	High-speed convergence	Low scalability
Karimi et al. (2016)	Genetic algorithm+data mining techniques	Response time, price, availability, successability	High scalability, high efficiency, low time	High overhead
Kurdi et al. (2015)	Combinational optimization	Execution time, cost	Low execution time, Low overhead	High cost
Liu and Zhang (2016)	Genetic algorithm	Time, cost, reliability	High success rate, low time convergence	High time complexity
Qi et al. (2017)	Differential evolution	Response time, availability, reliability	High-speed convergence, low computation time	Low scalability
Wang et al. (2013)	Particle swarm optimization+skyline operator	Response time, price, reputation	Low cost, high efficiency	Low stability
Lartigau et al. (2015)	Bee colony	Price, time, availability, reliability, maintainability	High scalability, high efficiency	High time
Huo et al. (2015)	Bee colony	Response time, price, reputation, availability, throughput	High scalability	High cost
Klein et al. (2014)	Genetic algorithm	Latency, price	Low time	Low scalability
Zhao et al. (2015)	Fuzzy preference+evolutionary algorithm	Response time, price, availability, throughput	Low time	Low scalability
Faruk et al. (2016)	Particle swarm optimization+genetic algorithm	Response time, price, availability, reputation	High-speed convergence	High time complexity

3.1 The QoS-based web services

The QoS of web services describes the non-functional properties such as availability, reliability, response time. The QoS of individual services is supplied by service providers, and features of every QoS criterion are collected by user's feedback. In this paper, we focused on four QoS criteria that include: response time, cost, availability and reliability as shown in Table 2. Also, SLA is defined by these four QoS criteria.

3.2 Location-oriented web services

Since there are several individual services in different data centers, the degree of distribution of these services effects on the composited services QoS. For example, the individual services located in the same data center compared with two individual services on data centers that have been deployed in Asia and Europe are different, and network delay between them in communicating with each other is a significant parameter. The performance of the network is critical to the composited services. Therefore, there are two types of network latency: the network latency between services and the network latency between services and the users. The network latency between services, which is defined by variable *dt1* in Table 3, is mainly determined by the geographical location of data centers in which those services are deployed. The latency between data centers is measurable and predictable since the number of data centers for cloud providers is limited and stable. The cloud providers can save the network latency between data centers to use easily in their cache. The network latency between the service provider and the user, which is defined by variable *dt2* in Table 3, is mainly determined by the network environment among the services, and it can also be obtained from the network feedback (Wang et al. 2015).

3.3 Composited services

SLA refers to a contract between users and providers for the guaranty of QoS criteria. In order to find whether a service composition can meet the SLA, it is necessary to check the quality of service by collecting of individual services QoS criteria and network QoS. The composition service QoS is related to composition path structure. As shown in Fig. 1, there are three composition structures: sequential, parallel and conditional (Wang et al. 2015). Computing of sequential structure QoS provides a basis for computing of other structures QoS. Granulation functions for computing the QoS criteria of composition services in Table 3 are shown where t_i represents the response time, p_i represents the price, a_i represents availability, and r_i represents collected reliability of the ith consecutive branches.

3.4 Cuckoo search algorithm

Cuckoo search algorithms (Rajabioun 2011; Wang et al. 2016a, b; Zhou et al. 2016; Fouladgar and Lotfi 2016) find the proper answer at a certain distance from the optimal solution which is called suboptimal solutions. The cuckoo search algorithm is one of the strongest evolutionary algorithms, which has a greater ability to find the global optimum compared with other evolutionary algorithms. Algorithm 1 provides the pseudo-code of the cuckoo search algorithm. This algorithm starts by an initial population. The population of cuckoo has some eggs, which will be put in the nests of some host eggs.

Algorithm 1: Cuckoo search via Levy flight algorithm

1: ***Begin***
2: Objective function $f(x), x=(x_1,x_2,x_3,...,x_d)^T$
3: Generate initial population of n host nests $x_i\ (i=1,2,3,...,n)$
4: ***While*** (t<Max Generation) or (stop criterion) ***do***
5: ***begin***
6: Get a cuckoo randomly by Levy flight
7: Evaluate its quality/fitness F_i
8: Choose a nest among n (say, j) randomly
9: ***If***$(F_i > F_j)$ Replace j by the new solution;
10: A fraction (p_n) of worse nests are abandoned and new ones are built;
11: Keep the best solutions (or nests with quality solutions);
12: Rank the solutions and find the current best
13: ***end while***
14: Post process results and visualization
15: ***End***

Those eggs that are similar to the eggs of host bird have more chance to grow and become mature cuckoo. The other eggs are recognized by the host bird, and they vanish. The more eggs indicate the nests suitability of that area. If more eggs can live and can also be rescued from that region, we should pay more attention to that area. Therefore, the situ-

Table 2 Examples of QoS criteria of single and independent services

QoS criteria	Unit	Description
Response time	Millisecond	The time between the moment a request comes in and the moment the result is obtained
Availability	Percent	The probability that a service is accessible
Cost	Dollar	The monetary cost that the user should pay to the service provider to use the service
Reliability	Percent	The probability that a service is trustworthy

Table 3 Aggregation functions for quality of service computation (Wang et al. 2015)

	Availability	Reliability	Cost	Response time
Sequential	$t = \sum_{i=1}^{N} st^i + \sum_{i=1}^{N-1} dt_1^i + \sum_{i=1}^{2} dt_2^i$	$a = \prod_{i=1}^{N} sa^i$	$p = \sum_{i=1}^{N} sp^i$	$r = \text{Avg}_{i=1,2,\ldots,N} sr^i$
Parallel	$t = \max_{i=1,2,\ldots,N} T^i$	$a = \prod_{i=1}^{N} sa^i$	$p = \sum_{i=1}^{N} sp^i$	$r = \text{Avg}_{i=1,2,\ldots,N} sr^i$
Conditional	$t = \text{Avg}_{i=1,2,\ldots,N} T^i$	$a = \text{Avg}_{i=1,2,\ldots,N} sa^i$	$p = \text{Avg}_{i=1,2,\ldots,N} sp^i$	$r = \text{Avg}_{i=1,2,\ldots,N} sr^i$

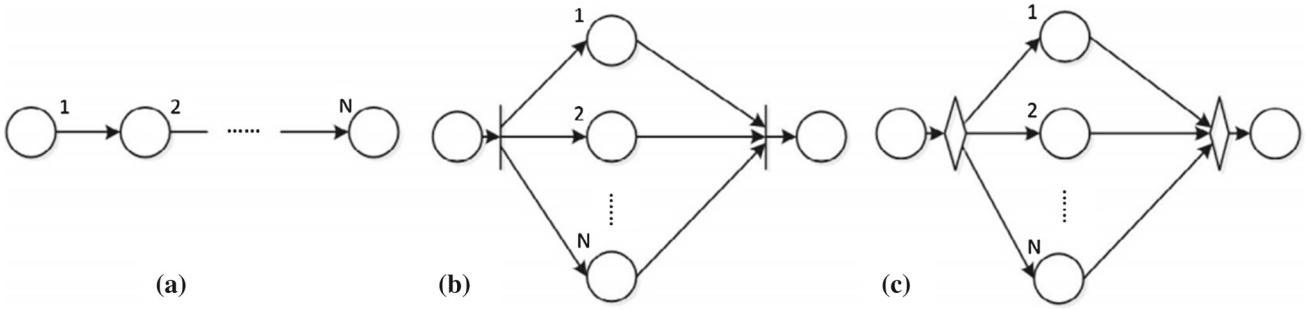


Fig. 1 The structure of the composition services. **a** Sequential. **b** Parallel. **c** Conditional

ation in which more eggs are rescued will be a parameter for the cuckoo search algorithm (CSA) to optimize it. The cuckoos search the best place for rescuing more eggs. After hatching and becoming an adult cuckoo, they come together to make homogenous groups. Each group selects a place to live. The best place of all groups is the destination for the next group. Everyone of the groups emigrates to that optimal place, and each group settles near the best place. By considering the number of eggs for each cuckoo and also the distance of cuckoo from the optimal place, they take into accent the radius of laying. After that, cuckoo starts to lay into the next of radius of laying randomly. This process continues to reach the optimal play for laying. That optimal place is the place in which many cuckoos are gathered.

4 The proposed algorithm

In this section, we first model the cloud web service composition problem. After offering a cloud web service composition model, a workflow example in cloud computing is provided to show the way in which web service composition is applied to serve tasks of a cloud application. The objective function of the given problem is also stated. Finally, the proposed algorithm is described as the CSA-WSC.

4.1 Web service composition model

The abstract model of the web service composer is depicted in Fig. 2. A workflow of user's requests is specified by the service requester. Service composition request section in the figure is used to serve the given tasks. Cloud combiner determines the candidate cloud web services for the requested tasks. Composition planner selects a subset of candidate cloud web services to execute the tasks. Finally, a service composition sequence for the given workflow is the output of the cloud web service composition model.

Let $\text{sc} = \{s_1, s_2, \ldots, s_n\}$ denote a collection of web services where $s_i (1 \leq i \leq n)$ denotes the ith web service. Each service provider published a set of web services as shown in Fig. 2. A cloud set (CS) is a set of m clouds $\text{CS} = \{C_1, C_2, \ldots, C_m\}$ where $C_i (1 \leq i \leq m)$ denotes the ith cloud.

Generally, a cloud web service composition problem can be defined by a triple $< I, G, W >$ where I shows the start point that is provided by the customer's request (initial interface); G indicates the goal interface that is provided

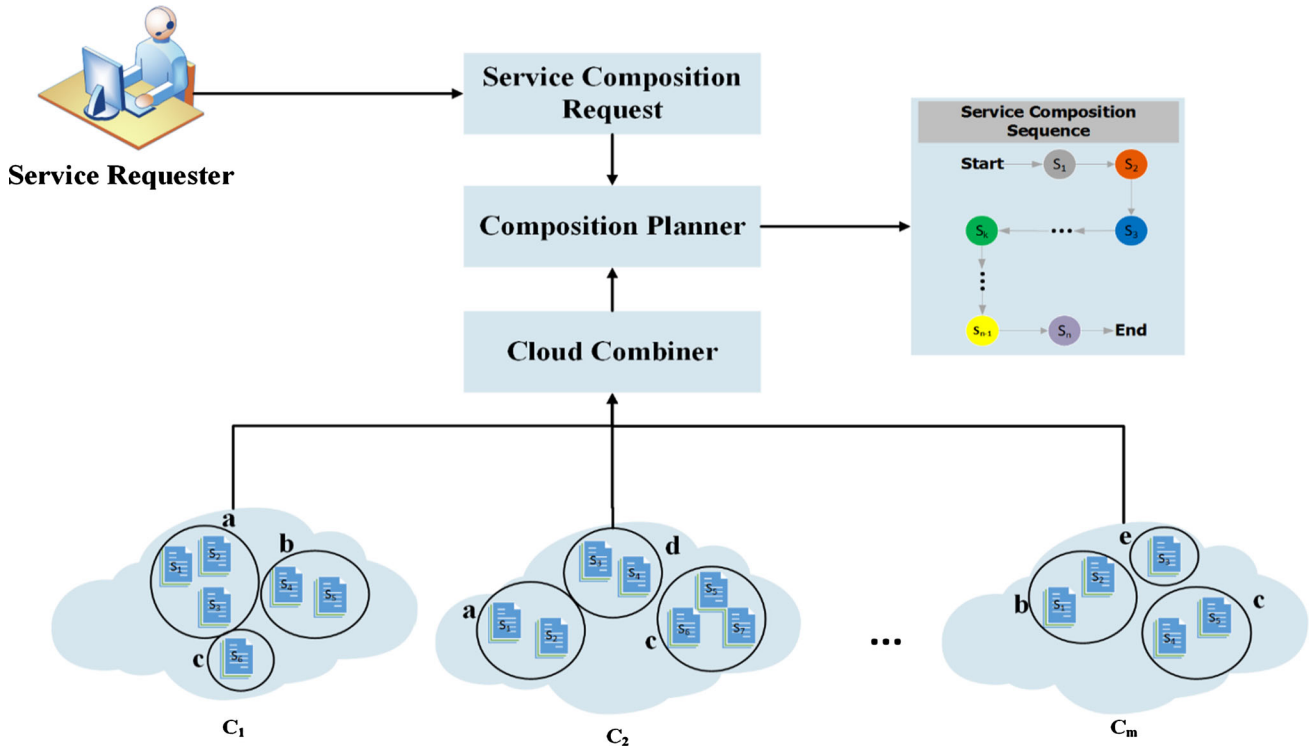


Fig. 2 Service composer model for cloud computing environments

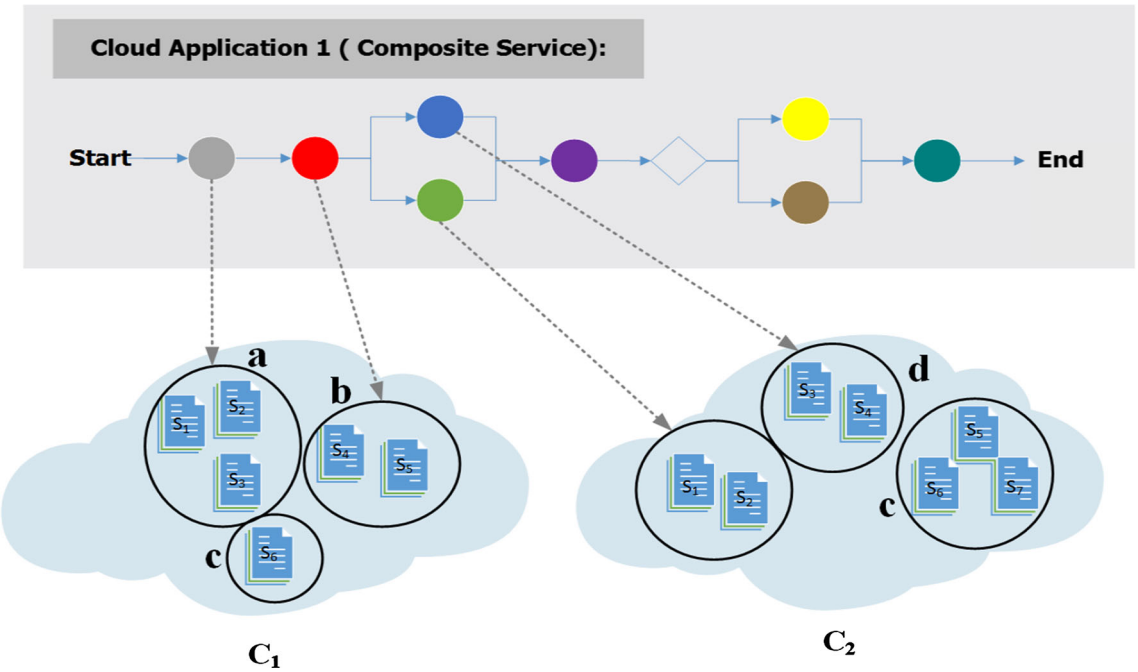


Fig. 3 An example of web service composition

by customer's requests (goal interface); and W denotes a set of available web services in clouds that can be used (web services). A sequence of web services ordered by their usage is the solution to the given problem such that the solution sequence $\pi \subseteq w$.

Hence, a composite service is the process of selecting a subset of provided web services for tasks. As an example, Fig. 3 shows that several similar services are available for each task.

4.2 Objective function of the WSC problem

According to the described problem and the mentioned aggregation functions for the quality of service, the objective function of web service composition problem can be formulized as follows:

$$\text{Max} \sum_{i=1}^{x} \sum_{j=1}^{k} \text{Quality}_j(\pi_i).w_j \tag{1}$$

Subject to :

$$\text{Quality}_j(\pi_i) \leq \text{SLA}_{i,j} \tag{2}$$

$$0 \leq w_j \leq 1 \tag{3}$$

$$\sum_{j=1}^{k} w_j = 1 \tag{4}$$

where π_i is a subset of candidate web services that are chosen in a sequence to execute tasks; $\text{Quality}_j(\pi_i)$ function calculates the jth qualitative parameter that is specified for the ith workflow by the user. $\text{SLA}_{i,j}$ $(1 \leq i \leq x$ and $1 \leq j \leq k)$ denotes the jth qualitative constraint for the ith workflow that is specified by the user. The weight of the jth qualitative parameter is denoted by w_j $(1 \leq j \leq k)$.

4.3 The CSA-WSC

In this section, the proposed algorithm, i.e. the cuckoo search algorithm for web service composition (CSA-WSC), is provided step by step based on the structure of cuckoo search algorithm. The steps of the proposed algorithm for selecting web services are shown in Fig. 4.

4.3.1 The first step: the production of the initial habitat of cuckoo based on the needed services

In order to solve the given problem, the values of variables of the problem should be in the form of an array. In genetic algorithm (GA) and particle swarm optimization (PSO), these arrays are distinguished as chromosome and particle position, while at the cuckoo search algorithm this array is called *habitat*. In proposed algorithm, for each cuckoo two criteria to be considered, and the sample services table with their index are stored. The initial population of cuckoos is shaped, due to the needed services for each request. For better understanding about how to the production of initial population and proposed algorithm, an example is explained. If the number of sample service is considered to 30, each request may need between 1 and 30 services and the sum of the atomic services number for each sample is 200. For example, if a request needs six types of services, it is clear that the algorithm parameter of cuckoo will determine like Table 4. Based on the parameters for the given an example in Table 4, the length of each cuckoo from the population is 6, as shown in Table 5.

If the solution of the problem needs finding N_{var} responses at the next optimal problem, next N_{var} that will be a habitat and array which has the current position of the life of cuckoo. It is defined in the format of Habitat $= [X_1, X_2, X_3, ..., X_{N_{\text{var}}}]$. Here each X is corresponding to the position of a cuckoo. Each cuckoo has two parts; the habitat and its QoS. In Table 6, the position of sample cuckoo is provided. Each X index of a service is the sum of atomic services in request sample.

In order to improve the quality of solution and the speed of convergence, one part of initial population is produced based on the concept of horizon line according to Eq. (5) and the rest are produced randomly, so that, P is the initial population, N is the length of habitat array, and SL is the number of selected services obtained in the skyline series.

$$P = \text{SL}/N + (N - \text{SL})/N \tag{5}$$

For example, if the number of cuckoo habitats (variable X) is six, therefore, the half of the services (three services) are specified by the skyline series and the other three services are selected randomly. In fact, three-sixth of the initial population is generated based on the concept of the skyline, and the other three-sixth population is randomly generated.

Definition 1 (*Dominance*) for two individual services S_1 and S_2 in service set, called S_1, dominates on S_2 when the following conditions are met:

$$\left\{\forall \left(q_{i.1}^{-}.q_{i,2}^{-}\right) | q_{i.1}^{-} \leq q_{i.2}^{-}\right\} \wedge \left\{\forall \left(q_{i.1}^{+}.q_{i.2}^{+}\right) | q_{i.1}^{+} \geq q_{i.2}^{+}\right\} \tag{6}$$

$$\left\{\exists \left(q_{i.1}^{-}.q_{i.2}^{-}\right) | q_{i.1}^{-} < q_{i.2}^{-}\right\} \vee \left\{\exists \left(q_{i.1}^{+}.q_{i.2}^{+}\right) | q_{i.1}^{+}) q_{i.2}^{+}\right\} \tag{7}$$

where $q_{i,j}^{-}$ is the *ith* negative QoS criteria of S_j atomic service and $q_{i.j}^{+}$ also is the *ith* positive QoS criteria of S_j atomic service. In the following, we can see an example of the definition of dominance in Figs. 5, 6 and 7 and corresponding Tables 7, 8 and 9. Four criteria are used to cover the quality of service, which contain two positive QoS criteria: reliability and availability, and two negative QoS criteria: cost and response time.

According to Fig. 5, individual service S_5 dominates an individual service S_3, or individual service S_2 is not determined by other individual services. That is, the individual service S_2 has no dominance.

According to Fig. 6, the individual service S_1 is dominated by individual service S_3, or the individual service S_5 is not determined by other individual services. In other words, the individual service S_5 has no dominance. Right now, we can get all four QoS criteria and obtain the average of positive and negative QoS criteria in Fig. 7.

Definition 2 (*horizon line set*) It is a subset of services, and it has produced alone available service in service set, which

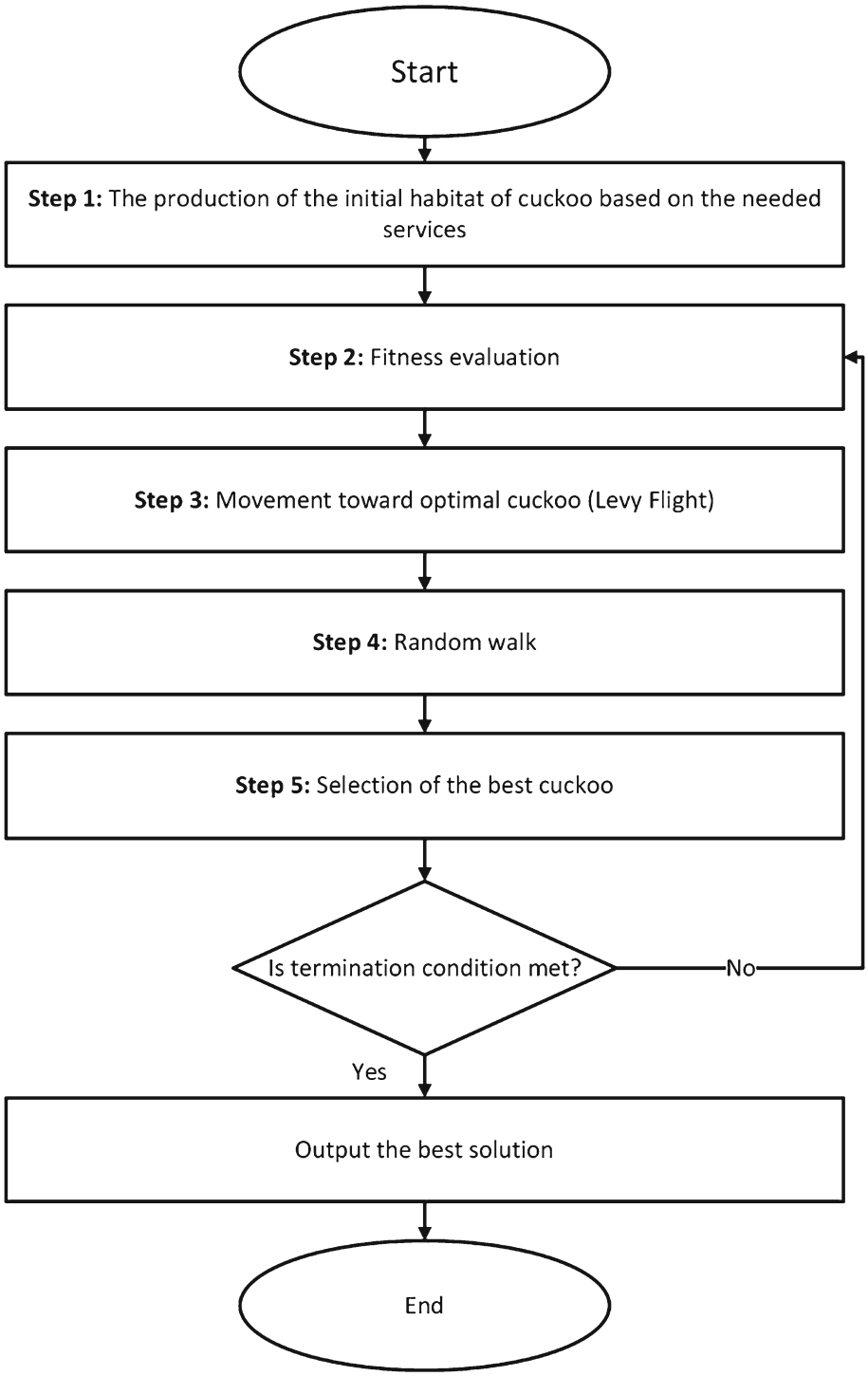


Fig. 4 The flowchart of the CSA-WSC

Table 4 The parameters initialization of cuckoo search algorithm

Parameter	Amount in example	Description
N_{var}	6	The number of sample services in request
LB	1	The lower bound of atomic service set
UB	200	The upper bound of atomic service set
N_{pop}	50	The number of the first cuckoos
Max iteration	200	Maximum number of iterations for the algorithm

Table 5 The sample of request structure

# Request 1	# Request 2	# Request 3	# Request 4	#Request 5	# Request 6
7	15	4	9	20	29

Table 6 The sample of the cuckoo structure to sample of request structure Table 5

X_1	X_2	X_3	X_4	X_5	X_6
80	190	50	3	120	40

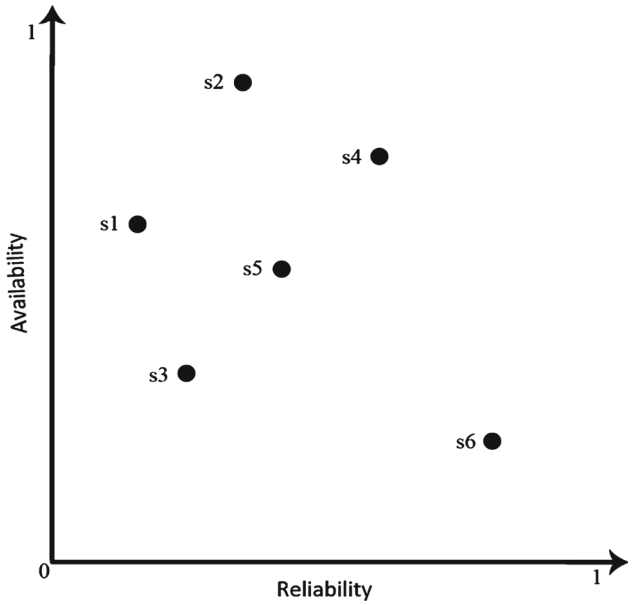


Fig. 5 Weight of the positive QoS criteria

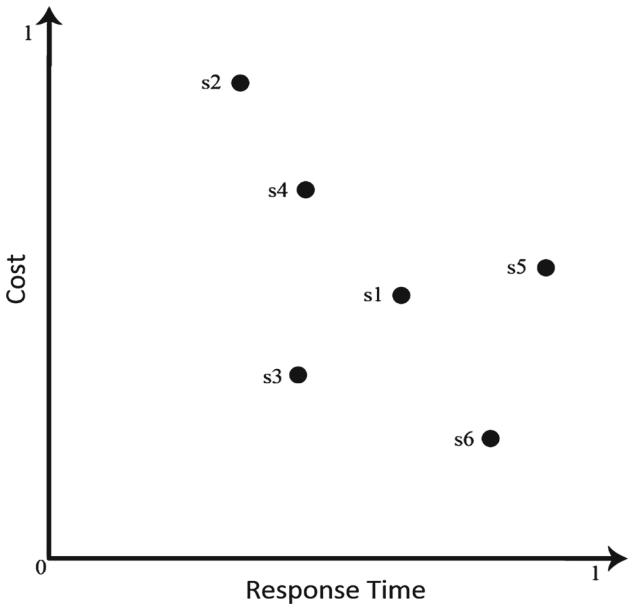


Fig. 6 Reverse weight of negative QoS criteria

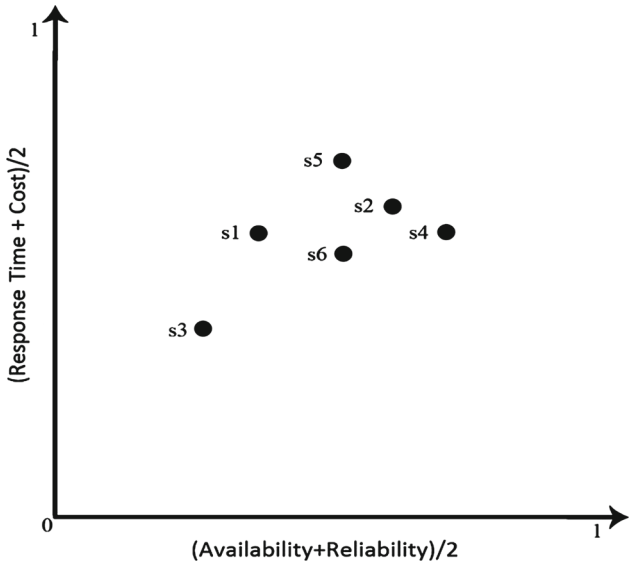


Fig. 7 The average of positive and negative QoS criteria

Table 7 The services weight for each criterion

	A	*R*	*C*	*T*
S1	0.6	0.1	0.5	0.4
S2	0.9	0.3	0.1	0.7
S3	0.3	0.2	0.7	0.6
S4	0.8	0.6	0.3	0.6
S5	0.5	0.5	0.5	0.1
S6	0.2	0.8	0.8	0.2

Table 8 Reverse the negative weight

	C	*T*
S1	0.5	0.6
S2	0.9	0.3
S3	0.3	0.4
S4	0.7	0.4
S5	0.5	0.9
S6	0.2	0.8

has no domination. For example, according to Fig. 7, horizon line set of positive values is (S_2, S_4, S_6) and the horizon line of negative values is (S_2, S_4, S_5) and the horizon line of each four values is (S_5, S_2, S_4).

Therefore, the computation cost for horizon line set become more expensive while the number of individual services increases.

Table 9 Average of positive and negative QoS criteria

	Inverse of $(T+C)/2$	$(T+C)/2$	$(A+R)/2$
S1	0.55	0.45	0.35
S2	0.6	0.4	0.6
S3	0.35	0.65	0.25
S4	0.55	0.45	0.7
S5	0.7	0.3	0.5
S6	0.5	0.5	0.5

4.3.2 *The second step: the evaluation of initial population using fitness function*

The fitness function is used to evaluate the suitability of the current habitat. Since we give each position of a cuckoo one adaptable value based on SLA, we consider two QoS criteria: the positive and negative QoS criteria. The increase in positive QoS criteria like availability and reliability is useful. The decrease in negative QoS criteria like time and cost is also important for users. The fitness function should reinforce the increase in positive QoS criteria and decrease in negative QoS criteria. Also, fitness function also needs to show the priority of users. The users prefer different QoS criteria. For example, some users prefer services with a higher availability and lower response time in SLA. For calculating fitness value, the value of QoS criteria needs to be normalized according to Eqs. (8) and (9) for positive and negative QoS criteria, respectively.

$$\zeta_i^- (\mathrm{CS}) = \frac{Sq_i^- - q_i^- (\mathrm{CS})}{Sq_i^+} \tag{8}$$

$$\zeta_i^+ (\mathrm{CS}) = \frac{q_i^+ (\mathrm{CS}) - Sq_i^+}{Sq_i^+} \tag{9}$$

where ζ_i^- and ζ_i^+ are normalized values of the ith QoS criterion of composited services, q_i is the ith of QoS criterion, and Sq_i^- is the ith of QoS constrain specified in SLA. To avoid SLA violations, positive QoS criteria should be increased at the same time negative QoS criteria is decreased. We also need to consider the user's preference in fitness value calculation. We use $\propto_i$ as a parameter to reflect the user's preference. The higher value of $\propto_i$ indicates the high level of QoS criterion. As mentioned above, fitness function is expressed by Eq. (10):

$$f (\mathrm{cs}) = \sum_{i=1}^{o} \propto_i \times \zeta_i (\mathrm{cs}) \tag{10}$$

$$\sum_{i=1}^{o} \propto_i = 1 \tag{11}$$

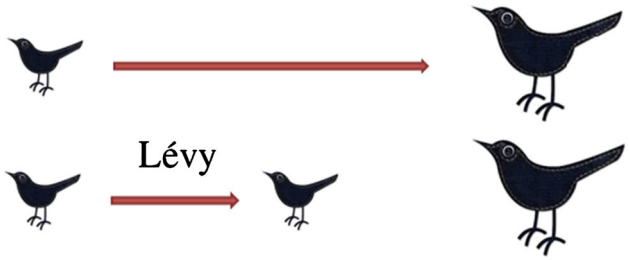


Fig. 8 Levy flight to move toward to the optimal cuckoo

where o refers to the number of QoS criteria. The aim of proposed algorithm is to find composited services with high fitness value.

4.3.3 *The third step: movement toward optimal cuckoo with applying Levy flight*

According to the fitness function, the value of each cuckoo or in other words the rate of quality function of the provided service is measured. In this step, firstly, the most valuable cuckoo is chosen. Afterward, cuckoos move toward it with limited step using Levy flight. This means that the selected services have the best value based on the cost and the response time as two negative QoS criteria and availability and reliability as two positive QoS criteria. Figure 8 shows the sample of Levy flight.

It is important to note that if less valuable cuckoos are moved to the exact position of the optimal cuckoo, we will have no new responses and thus the movement and exchange will be useless. The flight function each Levy is according to Eq. (12):

$$L (s.\gamma.\mu) = \sqrt{\frac{\gamma}{2\pi}} \frac{1}{(s-\mu)^{\frac{3}{2}}} \exp\left(-\frac{\gamma}{2(s-\mu)}\right) 0 < \mu < s < \infty \tag{12}$$

where μ is minimum steps and γ is the size parameter. If $s \to \infty$:

$$L (s.\gamma.\mu) = \sqrt{\frac{\gamma}{2\pi}} \frac{1}{(s-\mu)^{\frac{3}{2}}} \quad s \to \infty \tag{13}$$

We use Eq. (14) to generate the random step s.

$$s = \frac{u}{|v|^{1/\beta}} \tag{14}$$

where v and u are usually random variables.

$$u \sim N\left(0.\sigma_u^2\right), \quad v \sim N\left(0.\sigma_u^2\right) \tag{15}$$

Subject to:

$$\sigma = \left\{ \frac{\Gamma(1+\beta)\sin(\pi\beta/2)}{\Gamma|(1+\beta)/2|\,\beta 2^{(\beta-1)/2}} \right\}^{1/\beta} \tag{16}$$

where Γ is the gamma function. A distribution obtained from Eq. (14) for s will be a Levy distribution to $|S| \geq |S_0|$ that S_0 is the smallest step. After Levy flight, if the next position has much superior fit, the next position should be replaced as a better position. Otherwise, the previous position is preserved.

4.3.4 The fourth step: creation random movement pattern for all cuckoo's population

To make a better distribution between lower bound (LB) and upper bound (UB) to achieve the more probable answers, a random movement is used for the entire population with the Levy flight. In this step, the value of S is calculated by Eq. (17). Thus, those two cuckoo positions will be selected randomly. The difference between those two positions is calculated, and the result is multiplied by the random value.

$$\begin{aligned} S &= \text{rand.}\,(\text{nest}\,(\text{randperml}\,(\text{n})\,.\,:) \\ &\quad -\text{nest}\,(\text{randperm2}\,(n)\,.\,:)) \\ \text{nest}^{t+1} &= \text{nest}^{t} + S * P \end{aligned} \tag{17}$$

where *nest* is the position matrix of all cuckoo's, nest^t is the current position cuckoo's and nest^{t+1} will be the next position cuckoo's. To calculate the next position, we should calculate the value of P according to Eq. (18):

$$P = \begin{cases} 1 & \text{if rand} < P.a \\ 0 & \text{if rand} \geq P.a \end{cases} \tag{18}$$

The value of P is determined by the initial parameter Pa. Thus, a random number is generated (*rand*); if the random number is larger than Pa, the previous position is maintained, whereas if the random number is smaller than Pa, the next position based on the S is achieved.

4.3.5 The fifth step: selection of the best cuckoo

The best cuckoo will be selected as a generation response. After a few iterations of all cuckoo's population, they will have the greatest overall benefit to a point which is an optimum level of superior fit and in which the lowest number of eggs will be disappear. Best position specifically is those services that are selected to offer the request.

5 Performance evaluation

In this section, we evaluate the effectiveness of applying the proposed CSA-WSC for combining web services in distributed cloud environments. Some web services are selected and then composited and given to the applicant as a service set. The selection of the best web services for a workflow based on QoS criteria is the major role of the proposed algorithm. First, we describe the performance criteria and the simulation settings, and then we present and discuss experimental results.

5.1 Performance criteria

We applied the following the performance QoS criteria for a comparison of proposed algorithm with other algorithms:

Response time is the amount of time between the start of the request and the receipt of the response by the user from the cloud.

Cost is the amount of money the user pays for the request for any virtual machine, based on the amount of memory, processes, and bandwidth consumed. It is calculated by Eq. (19):

$$\text{Cost} = \sum_{i=1}^{K} C_i \times T_i \tag{19}$$

where $\boldsymbol{K}$ is the number of virtual machines allocated to user requests, C_i is the cost of a virtual machine, and T_i is the time for which the user can use the virtual machine.

Reliability is an indicator of the successful running of a task by the virtual machine in a datacenter and is expressed by Eq. (20) as follows (Koren and Krishna 2010):

$$\text{Reliability(RE)}\,V_k = \frac{C_k}{A_k} \tag{20}$$

where A_k is the number of tasks accepted by a virtual machine V_k, and C_k is the number of tasks completed by the virtual machine of V_k in a time limit T.

Availability is the possibility of accessing the virtual machines requested by a user from a datacenter. Suppose $V_1, V_2, V_3, ..., V_n$ are virtual machines in a datacenter; for any $k = 1, 2, 3, ..., n$ the percentage availability for a virtual machine of a datacenter is calculated using Eq. (21) as follows (Bauer and Adams 2012):

$$\text{Availability(AV)}\,V_k = \frac{\text{MTTF}_k}{\text{MTBF}_k} = \frac{\text{MTTF}_k}{\text{MTTF}_k + \text{MTTR}_k} \tag{21}$$

where ***MTTF*** is the mean time for which the resource works correctly without failure; ***MTTR*** is mean time to repair the

Table 10 The datacenters specification

Architecture	x86
Operation system	Cloud linux
Virtual machine manager	XEN

Table 11 The host specification in a datacenter

Name	Processor type	Number of core	Frequency (MIPS)	Main memory (GB)	Bandwidth
Host1	Intel Xeon 2540	4	512	4	1 Gbit/s

resource after failure, and ***MTBF*** is the mean time between two failures in a resource.

Note that for any datacenter, the average availability, the average reliability, the average cost and the average response time can be computed.

5.2 Experimental setup

The experiments presented in this section were developed using cloudsim toolkit (Calheiros et al. 2011). The structure of all the datacenters used in the simulation is shown in Table 10. In each data center, there is a host. In Table 11, each host specification is detailed. When the hardware hosted is more powerful and of a higher level, the cost of access to the resources on the virtual machine on this host increases.

Since one of the effective criteria is response time, the amount of latency or user interval from the datacenter is set to a random value with a normal distribution based on the distance between users and the datacenter, user and the service set, and the datacenters shown in Table 12. Furthermore, parameter settings for the algorithms are provided in Table 13. Also, the values of four QoS criteria of datacenters, response time, cost, availability and reliability were produced at the start of the simulation process shown in Table 14.

5.3 Experimental analysis

To evaluate the proposed algorithm, the experimental evaluation is formed in three different scenarios according to Table 15. We compared the proposed algorithm with three most effective WSC algorithms as follows: A genetic-based generic algorithm which considers the network delay (GS-S-Net) (Wang et al. 2015); a genetic particle swarm optimization algorithm (GAPSO-WSC) (Faruk et al. 2016) is used for discovering optimum regions from complex search spaces via the collaboration of individuals in a crowd of particles; a greedy-based algorithm for web service composition (Greedy). In the Greedy algorithm, the user's region

Table 12 Values of communication delay time between user, datacenter and service set

Communication	Produce the initial values
Between user and datacenter	The random normal distributed between 20 and 500
Between user and service set	The random normal distributed between 50 and 400
Between datacenters	The random normal distributed between 5 and 50

is first determined. Then, the largest datacenter in terms of free resources is chosen from the available datacenters and allocated to the user. Also, the simulation results of QoS criteria under CSA-WSC, GAPSO-WSC, GS-S-Net and Greedy algorithms in each scenario are reported.

5.3.1 The first scenario

In this scenario, the number of sample services is set 10, and the number of services is set 200, 400 and 500, respectively. First, the effect of the number of atomic services per set on the function values of GS-S-Net and CSA-WSC algorithms is evaluated (see Fig. 9). The fitness function values of the CSA-WSC outperform GS-S-Net and GAPSO-WSC algorithms, while it has higher integration in finding solutions. It is also can be seen that the CSA-WSC has less deviation in fitness function because it has faster convergence than the GS-S-Net algorithm and thus there is less possibility not to give the expected fitness value. The GAPSO-WSC algorithm has better performance compared with GS-S-Net because it uses PSO algorithm besides genetic algorithm, which provides better exploitation and exploration in the search space. While the increase in fitness results in the decrease in cost and Fig. 9 shows that the CSA-WSC provides better fitness compared with other algorithms, the cost incurred by applying CSA-WSC is reduced (see Fig. 10).

Based on the QoS criteria discussed in Sect. 5.1, Fig.10 shows the cost of three CSA-WSC, GS-S-Net and Greedy algorithms with the number of requests from 1000 to 10,000.

According to Fig. 10, it is clear that the cost of the Greedy algorithm is more than that of other algorithms, while it has a very simple idea of finding a suboptimal solution without any complexity. In contrast, the proposed CSA-WSC decreased the WSC cost to an acceptable level. Actually, cuckoos are able to make better solutions compared to the other algorithms and thus the incurred cost of dedicated and service composition is reduced. According to Fig. 10,

Table 13 Parameter settings of the algorithms

Variant	CSA-WSC	GAPSO-WSC	GS-S-Net
Termination condition	Number of generation	Number of generation	Number of generation
Number of initial population	50	50	50
Number of generation	200	200	200
Cuckoo radius move	3.2	×	×
Step size	0.01	×	×
Selection operator	×	Rolette wheel	Rolette wheel
Mutation (probability)	×	Boundary (0.1)	Boundary (0.1)
Crossover (probability)	×	Single-point(0.9)	Single-point(0.9)
Number of particles	×	25	×

Table 14 Values of four QoS criteria of datacenters

Factor	Produce the initial values
Response time	The random normal distributed between 20 and 1500
Cost	The random normal distributed between 2 and 15
Availability	The random normal distributed between 0.95 and 1
Reliability	The random normal distributed between 0.4 and 1

Table 15 Proposed scenarios for evaluating algorithms

Scenario	Various services	Number of services in each set	Objective
First scenario	10	200, 400, 500	Studying the simulation results CSA-WSC, GAPSO-WSC, GS-S-Net and Greedy algorithms in terms of QoS criteria including response time, cost, reliability, availability under scenario different
Second scenario	30	200, 400, 500	
Third scenario	60	200, 400, 500	

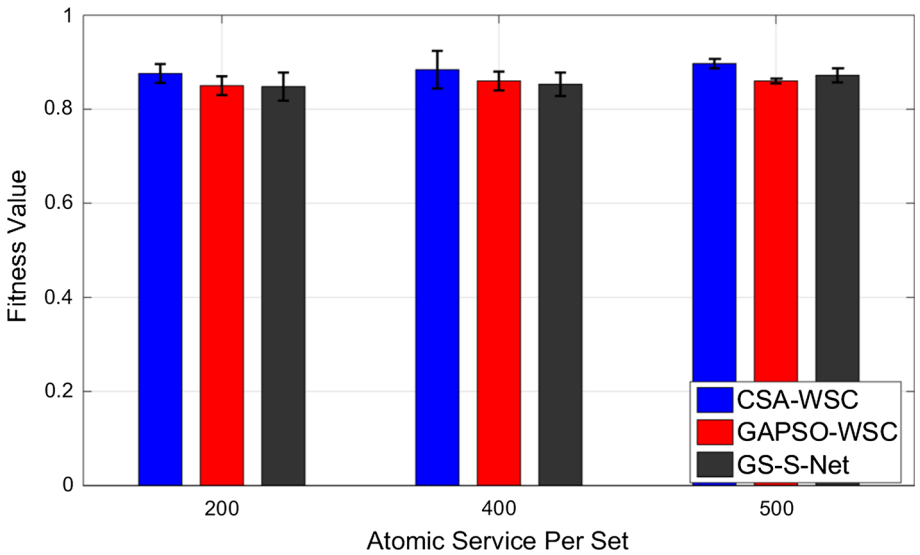


Fig. 9 The comparison of fitness values in the CSA-WSC, GAPSO-WSC and GS-S-Net algorithms in the first scenario

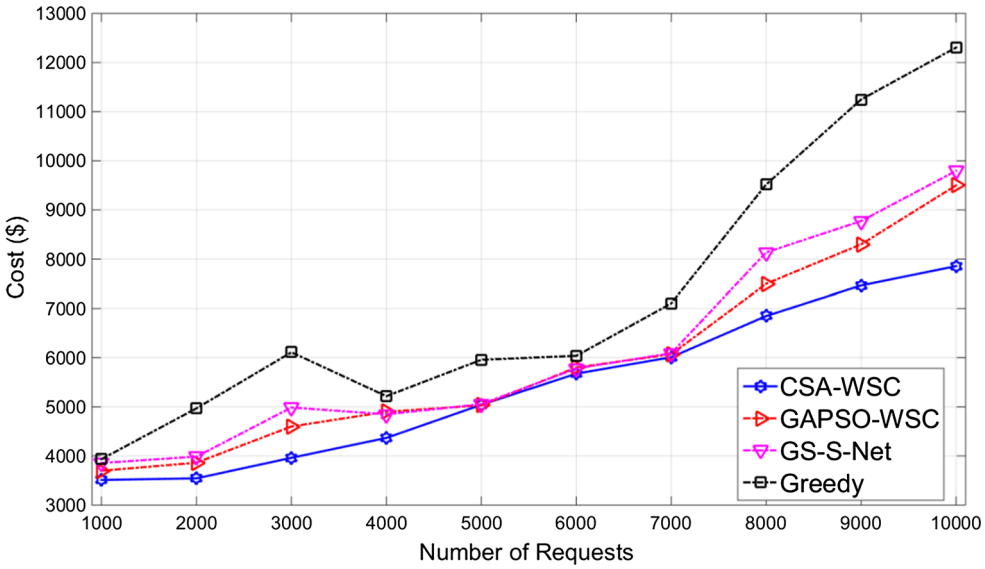


Fig. 10 The cost of CSA-WSC, GAPSO-WSC, GS-S-Net and Greedy algorithms in the first scenario

when the number of requests increases, the value of reduction cost increases significantly. The second most important criterion is the availability of the service. Figure 11 shows the provider's availability over a different number of requests for the first scenario.

According to Fig. 11, provider's availability in the CSA-WSC is higher than other algorithms, which specifies that the proposed CSA-WSC provides a more appropriate distribution of requests over web services. This means that requests are better distributed among service sets that lead to increasing in availability. The reliability or the trust of the service provider is another important criterion. Figure 12 shows the service providers reliability level of different CSA-WSC for the first scenario.

Figure 12 indicates that the CSA-WSC has a higher reliability. When a service is selected appropriately, the number of failed or uncompleted tasks is reduced in a provider. When a number of failed tasks is low in a service provider, this means that the reliability of the provider is in high level. Another important criterion is the response time of requests. We executed each simulation five times, and the average of results is provided in figures. The results of Fig. 13 show that the Greedy algorithm has less response time and has high speed in responding. However, when the request increases, the execution time of the Greedy algorithm increases more. Hence, GS-S-Net and CSA-WSC algorithms have higher execution time in executions with a lower number of requests. In contrast, by increasing the number of requests, the incurred cost of the Greedy algorithm compared to other algorithms becomes more significant, while GS-S-Net and CSA-WSC algorithms still able to choose the best composition of web services regardless of increasing the number of possible solutions. Thus, we can conclude that the better service composition compared to the execution time is more important in case inaccurate service composition results in cost, less reliability and/or less availability. Furthermore, Fig. 14 illustrates the box plot of different algorithms for different QoS criteria. As can be seen, the median line of the positive QoS criteria (availability and reliability) for the proposed CSA-WSC is higher than the comparing algorithms, which shows improvement in this terms. Also, the median line of negative QoS criteria (cost and time) for the proposed algorithm is lower than the comparing algorithms. Thus, the ranges of all quartiles in all QoS criteria confirm that the proposed algorithm outperforms other algorithms.

5.3.2 The second scenario

In this scenario, the number of services is 30, and the service set is 200, 400 and 500. Figure 15 shows the fitness function values for CSA-WSC, GS-S-Net and GAPSO-WSC algorithms over a different number of atomic service per set. Due to the great convergence of CSA-WSC, its fitness compared to the GAPSO-WSC and GS-S-Net algorithm is better. While the increase in fitness results in the decrease in cost, the proposed CSA-WSC provides a better fitness compared to other algorithms and thus the cost incurred by applying CSA-WSC is reduced (see Fig. 15).

Figure 16 shows the amount of cost in CSA-WSC, GAPSO-WSC, GS-S-Net and the Greedy algorithms with a different number of requests categorized between 1000 to10000. As seen in Fig. 16, the incurred cost by the Greedy algorithm is more than other algorithms. In comparison with GAPSO-WSC and GS-S-Net algorithms, CSA-WSC

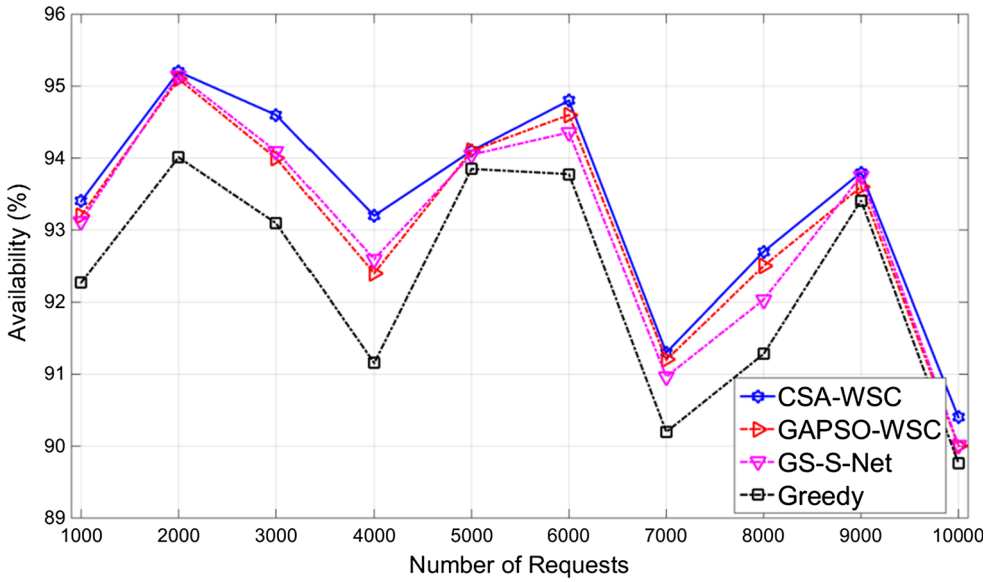


Fig. 11 The service availability of the CSA-WSC, GAPSO-WSC, GS-S-Net and Greedy algorithms in the first scenario

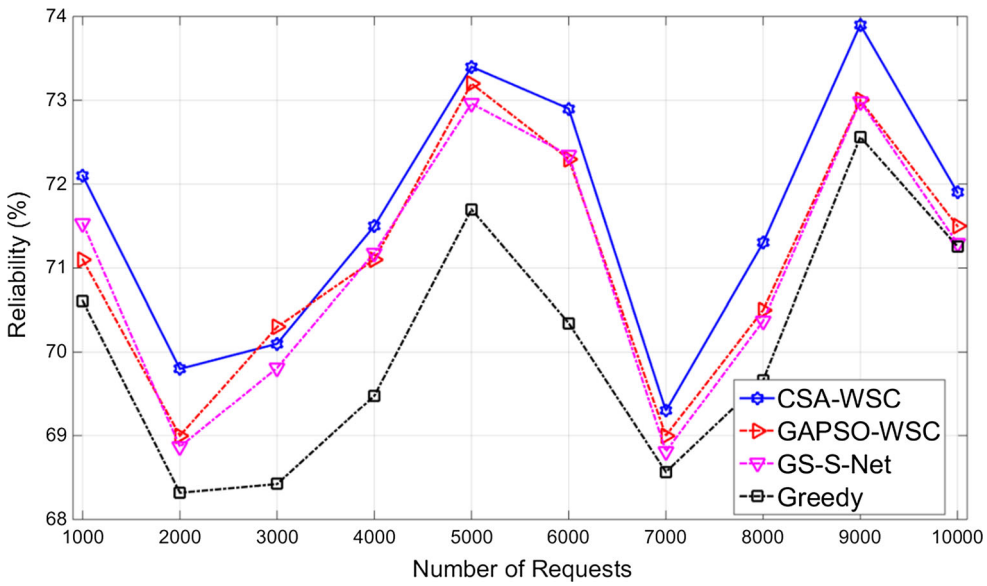


Fig. 12 The service reliability of the CSA-WSC, GAPSO-WSC, GS-S-Net and Greedy algorithms in the first scenario

decreases the amount of dedicated cost and the composition service. Accurate composition of web services by cuckoos reduced the incurred costs. Figure 16 illustrates that the increase in the number of requests brings more and more considerable decrease in the incurred cost by the proposed CSA-WSC compared to other algorithms. It is also done because of the availability of service set. Figure 17 shows the amount of availability of providers in the whole process of the second scenario.

According to Fig. 17, provider's availability in the CSA-WSC is higher than that of the other algorithms, which indicates that proposed algorithm provides more appropriate distribution requests. This means that requests are better distributed among service sets. Decreasing the busy time of service sets and increasing their availability depend on the more suitable distribution. The third important criterion is service provider reliability. Figure 18 shows the provider's service reliability in the whole second scenario process.

Figure 18 indicates that reliability rate in the CSA-WSC algorithm is higher. When a service is selected appropriately, failed or uncompleted task occurs less in a provider. When the number of failed-task is reduced in a service provider, this means that the reliability of the provider is increased. Another important criterion is request response time. Figure 19 shows

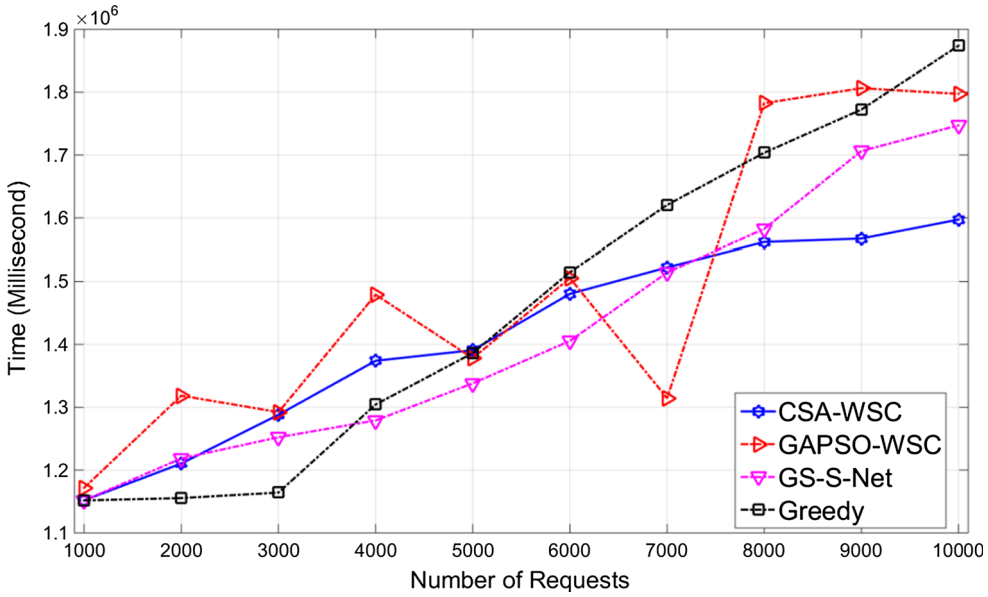


Fig. 13 Average of requests response time in the CSA-WSC, GAPSO-WSC, GS-S-Net and Greedy algorithms in the first scenario

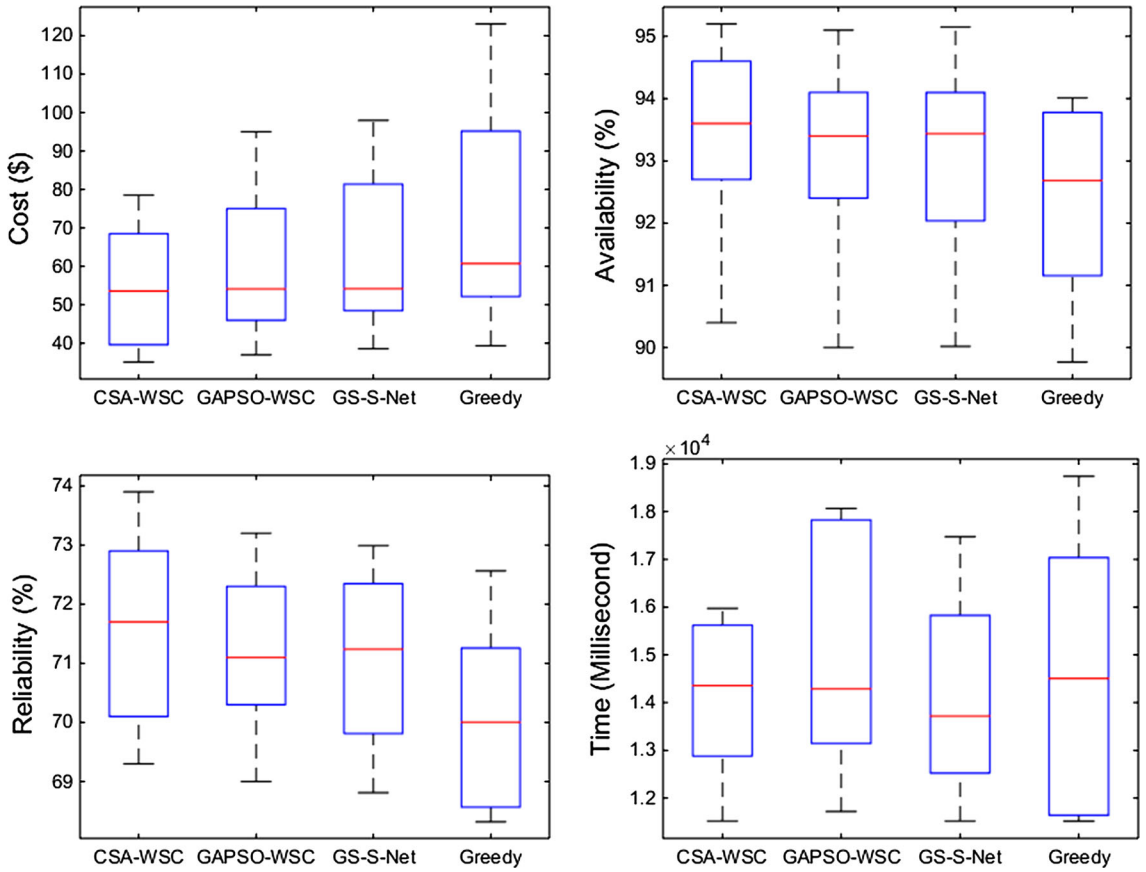


Fig. 14 Boxplot of different algorithms for different QoS criteria in the first scenario

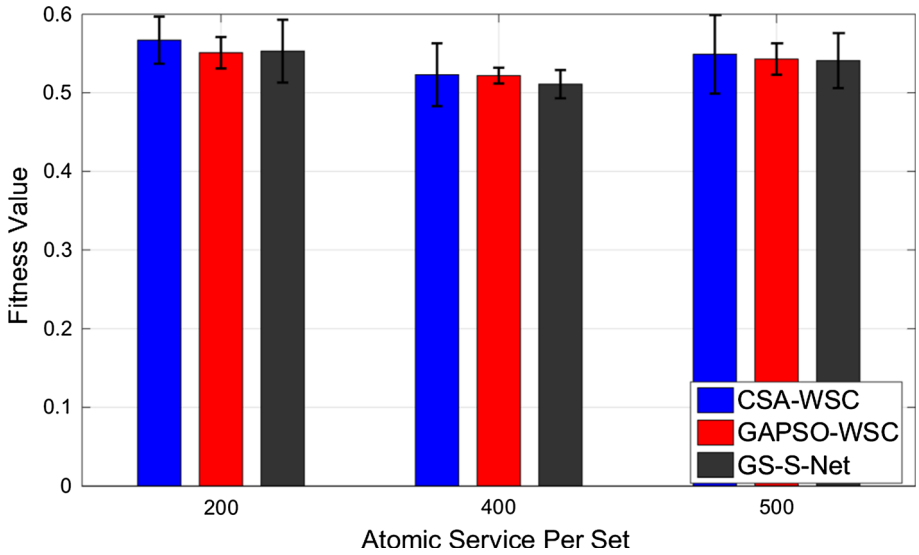


Fig. 15 The comparison of the fitness values in the CSA-WSC, GAPSO-WSC and GS-S-Net algorithms in the second scenario

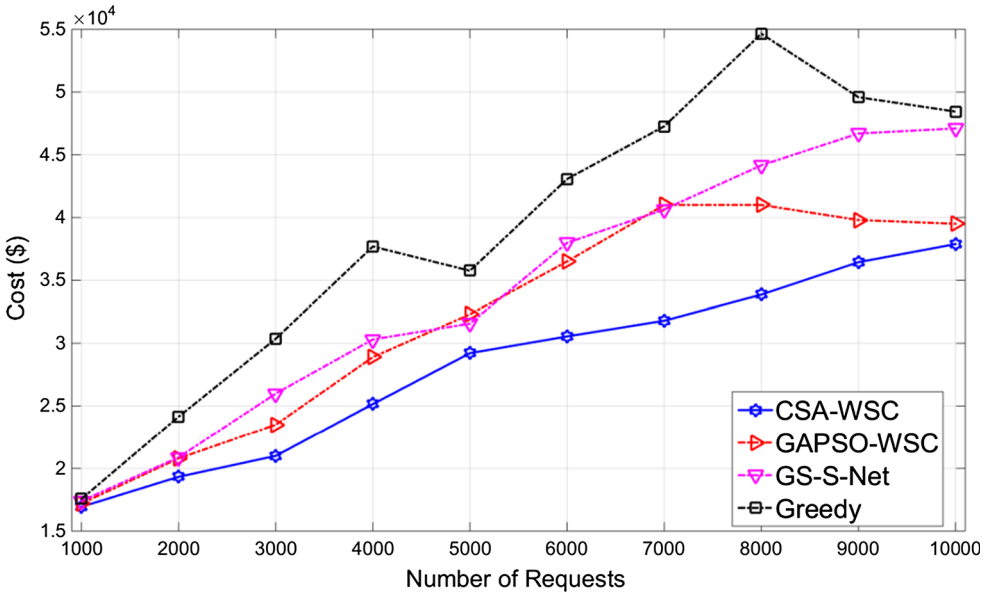


Fig. 16 The cost of CSA-WSC, GAPSO-WSC, GS-S-Net and Greedy algorithms in the second scenario

the average requests response time for different algorithms over a different number of requests.

Figure 19 shows that the Greedy algorithm has a lower response time and has high speed in responding fore case with a number of requests lower than three thousand. However, an increase in the number of requests results in higher response time for the Greedy algorithm compared to the other algorithms. The reason that the algorithms are low in the light number of requests is low-selected action in the cuckoo search algorithm and genetic algorithm. Therefore, the increase in the number of requests leads to better selection of web service composition sets. Thus, we can conclude that service composition quality compared to the execution time is more important in case inaccurate web service composition for cloud applications results in cost, less reliability and/or less availability. Figure 20 depicts the box plot of different algorithms for different QoS criteria. As shown, the median line of the positive QoS criteria for the proposed CSA-WSC is higher, while the median line of the negative QoS criteria of the CSA-WSC is lower compared to GAPSO-WSC, GS-S-Net and Greedy algorithms. Therefore, the ranges of the upper and lower quartiles in QoS criteria show that the CSA-WSC outperforms other algorithms.

5.3.3 The third scenario

Similarly to other scenarios, the sample size is considered 60 and the number of services set 200, 400 and 500, respectively,

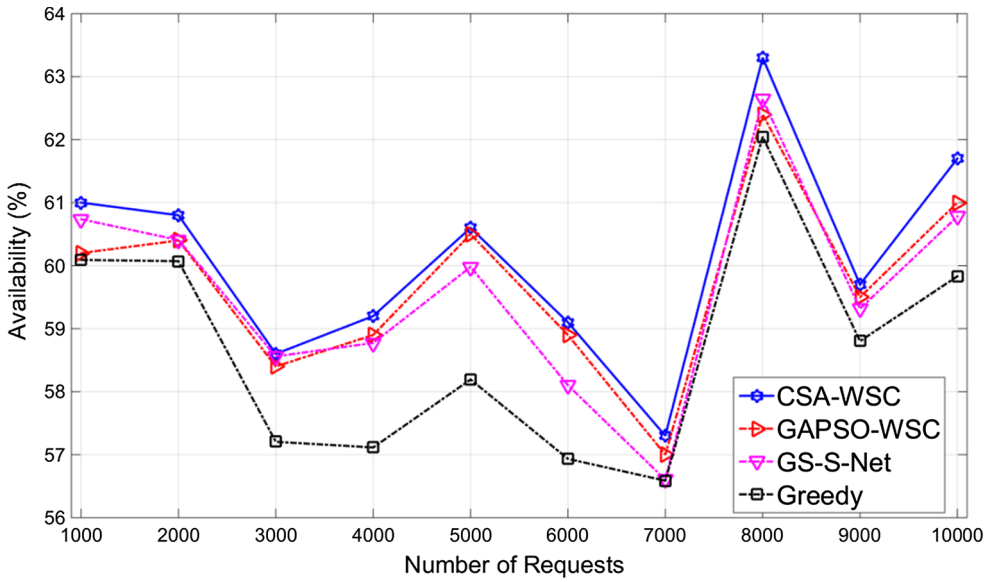


Fig. 17 The service availability of CSA-WSC, GAPSO-WSC, GS-S-Net and Greedy algorithms in the second scenario

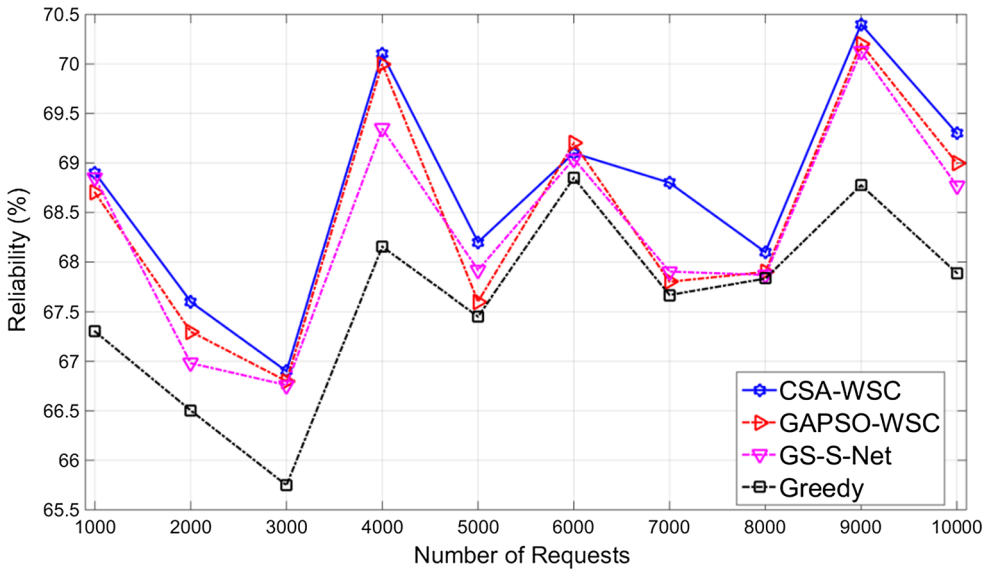


Fig. 18 The service reliability of CSA-WSC, GAPSO-WSC, GS-S-Net and Greedy algorithms in the second scenario

in this scenario. To compare the performance of the GAPSO-WSC, GS-S-Net and CSA-WSC algorithms, fitness function values are evaluated over a different number of atomic services per set in Fig. 21. As can be seen, due to higher integration of CSA-WSC, the fitness function value generated by this algorithm is better than by GAPSO-WSC and GS-S-Net algorithms. Given that the task of fitness function is to minimize the cost, Fig. 22 indicates that the CSA-WSC reduces the incurred cost more than GAPSO-WSC and GS-S-Net algorithms.

Given the important criteria discussed at the beginning of the evaluation, Fig. 22 shows the cost of the three CSA-WSC, GAPSO-WSC, GS-S-Net and Greedy algorithms with the number of requests between 1000 and 10,000. According to Fig. 22, it is clear that the cost of Greedy algorithms is more than other algorithms. It specifically becomes more significant as the number of requests increases while in this case the algorithm faces with a wider pool of solutions for the problem. In comparison with GAPSO-WSC and GS-S-Net algorithms, the proposed CSA-WSC reduced the total incurred costs. The reason of reducing the total costs has faster convergence that causes better fitness function values, and thus it brings more optimal solutions.

According to Fig. 22 when the requests number increases, the cost also significantly increases. However, the proposed algorithm is more stable in finding near-optimal solutions

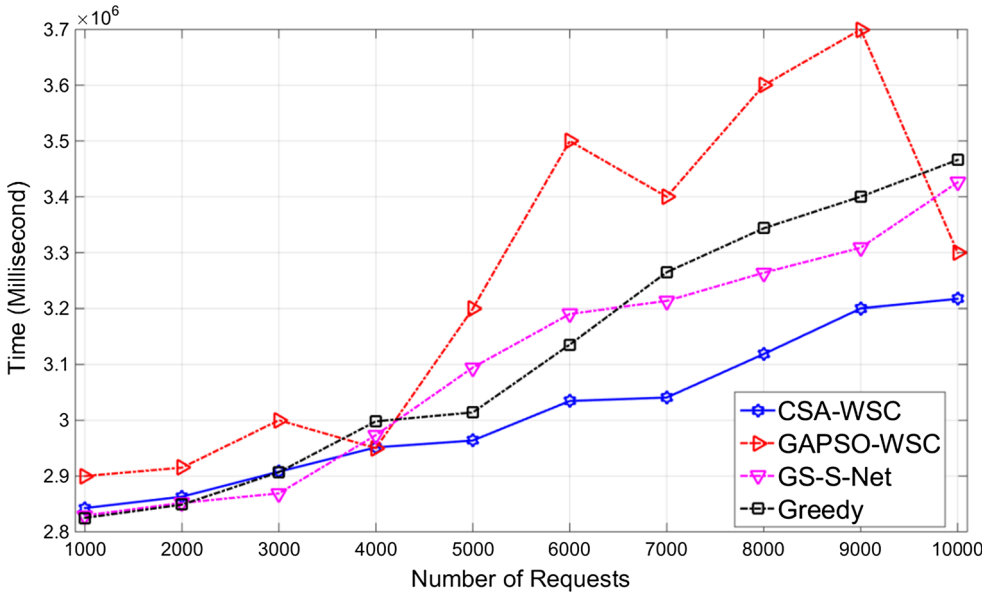


Fig. 19 Average requests response time on three CSA-WSC, GAPSO-WSC, GS-S-Net and Greedy algorithms in the second scenario

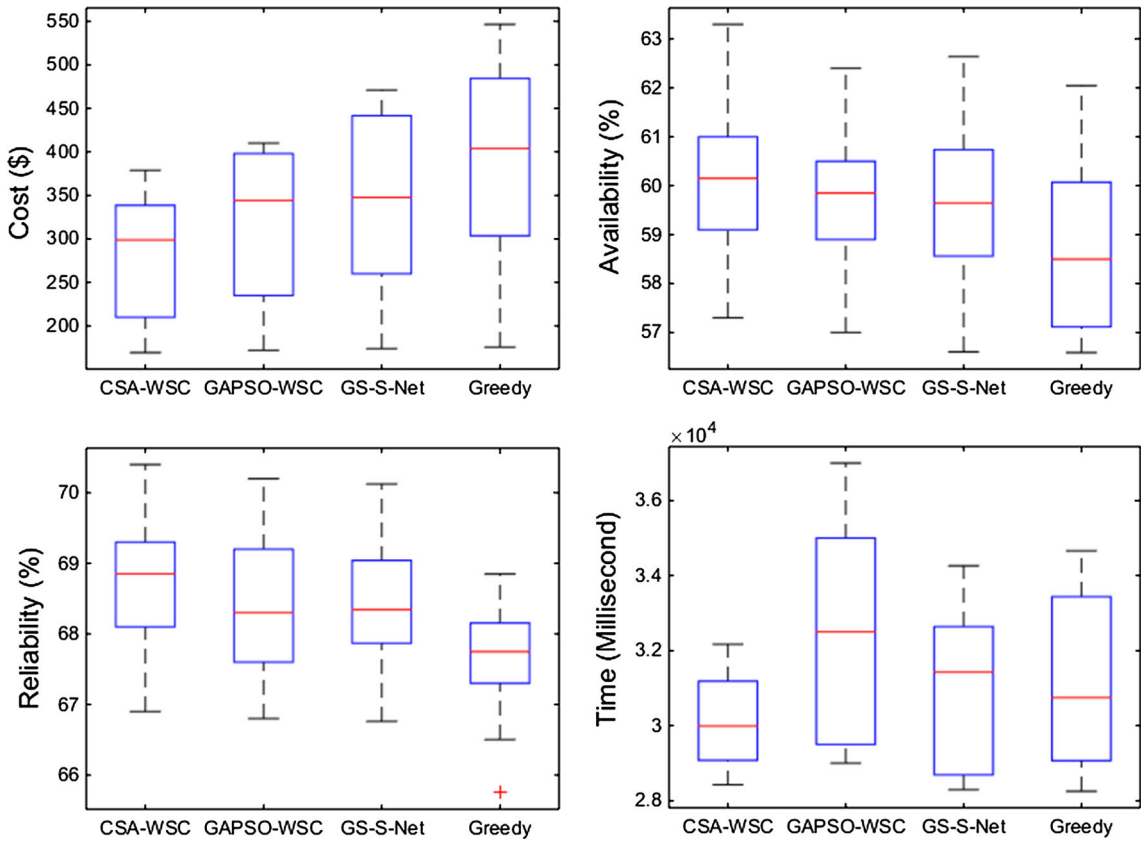


Fig. 20 Boxplot of different algorithms for different QoS criteria in the second scenario

while the number of requests increases. Consequently, the incurred costs of the proposed CSA-WSC are much less than other algorithms in case the number of requests is high. Service availability, as another important criterion, for the proposed CSA-WSC has a better situation compared to other algorithms (see Fig. 23). It shows that requests are better distributed among service sets. Decreasing the busy time of

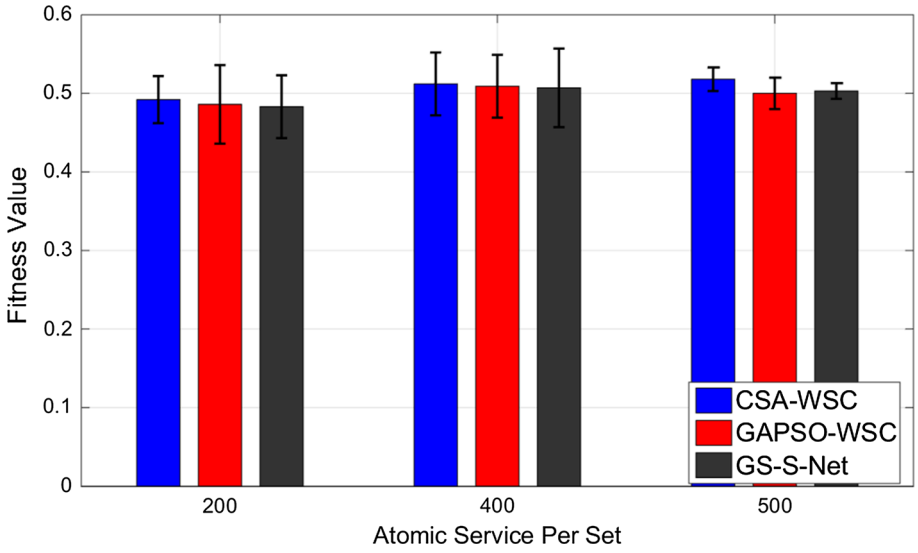


Fig. 21 The comparison of fitness values in the CSA-WSC, GAPSO-WSC and GS-S-Net algorithms in the third scenario

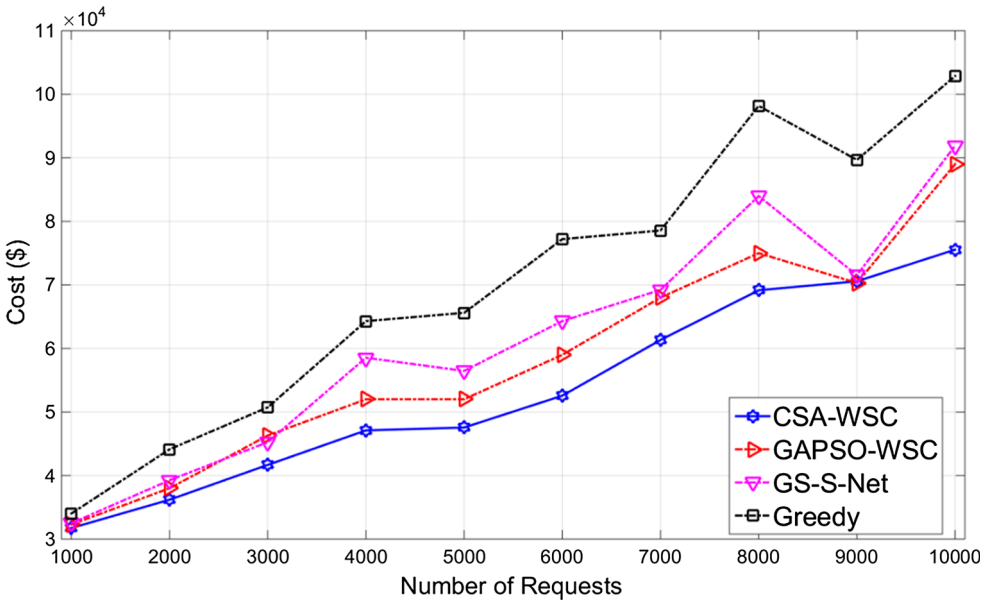


Fig. 22 The cost of CSA-WSC, GAPSO-WSC, GS-S-Net and Greedy algorithms in the third scenario

services sets and increasing their availability depend on the more suitable distribution.

Figure 24 shows providers service reliability in the third scenario. As can be seen, it indicates that the reliability of the CSA-WSC is higher. When a service set is selected appropriately, the number of failed or incomplete tasks occurs less. When the number of failed tasks is reduced in a service provider, this means that the reliability of the provider is increased. Figure 25 shows request response time for different algorithms over a different number of requests.

Results Fig. 25 specifies that the increase in the number of services results in the higher incurred costs for GAPSO-WSC, GS-S-Net and Greedy algorithms, while the proposed CSA-WSC algorithm was not much affected by the number of requests. Thus, it works better because of finding better solutions. The CSA-WSC in a pool of very large solutions can be suitable as well as in the smaller solutions, while its fitness convergence is quite faster than other algorithms. Figure 26 illustrates the box plot of different algorithms for different QoS criteria. As can be seen, the median line of the positive QoS criteria for the proposed CSA-WSC is higher, while the median line of the negative QoS criteria of the CSA-WSC is lower compared to GAPSO-WSC, GS-S-Net and Greedy algorithms. Thus, the ranges of the upper and lower quartiles in the QoS criteria show that the CSA-WSC outperforms other algorithms.

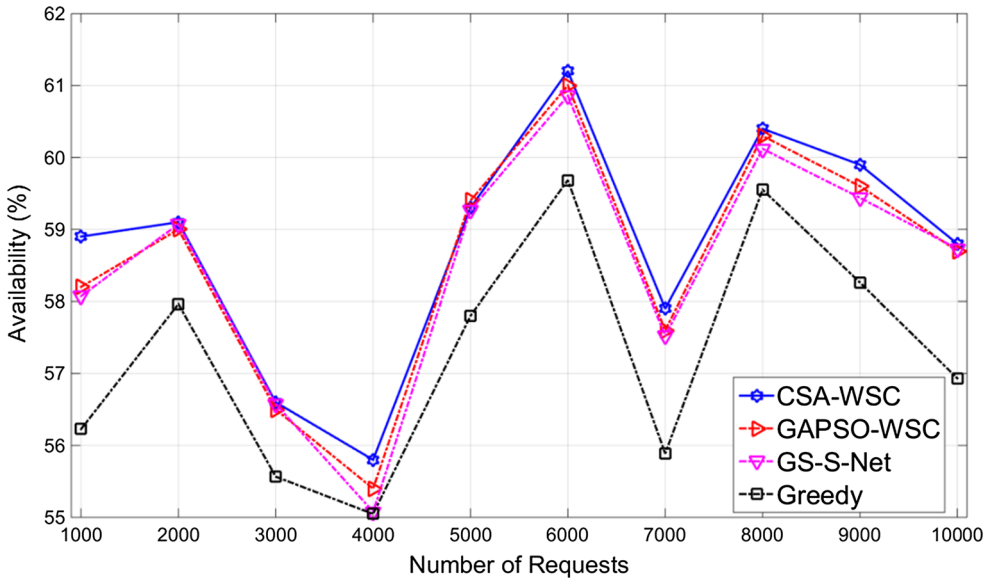


Fig. 23 The service availability of CSA-WSC, GAPSO-WSC, GS-S-Net and Greedy algorithms in the third scenario

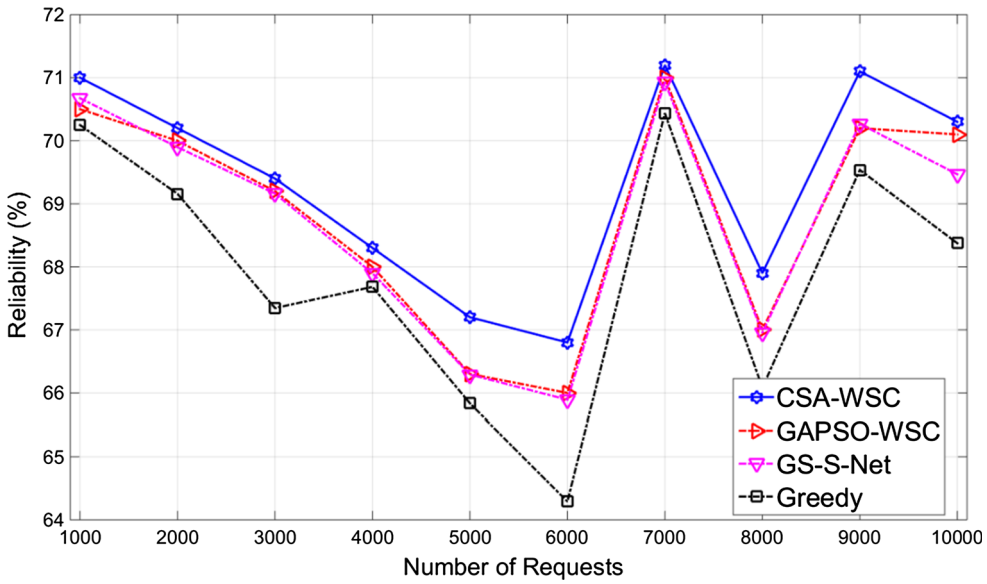


Fig. 24 The service reliability of CSA-WSC, GAPSO-WSC, GS-S-Net and Greedy algorithms in the third scenario

5.3.4 Total comparison

In this section, we sum up all the evaluations of the three prior scenarios. To do so, the superiority of the algorithms regarding the quality of service is evaluated.

As mentioned in the previous subsections, the convergence speed of the cuckoos is very effective on finding the near-optimal solutions. Here, we first evaluate the convergence speed of the proposed CSA-WSC in three scenarios. Figure 27 illustrates the fitness function values of the three scenarios over time generation. As is seen, the fitness values for the first scenario grow faster than other scenarios, while it seems that the problems in this scenario have been easier to be solved. Furthermore, the fitness function values in the second scenario are also increased faster than the third scenario. It shows that the most challenging scenario is the last one as the fitness function over time generation indicates this fact.

According to Fig. 27, proposed CSA-WSC converges at a faster rate compared with other baseline algorithms. Besides from converge speed, the proposed algorithm provides better fitness in all scenarios. Thus, Fig. 27 proves the fact that cuckoos provide a fast increase in the fitness value over time

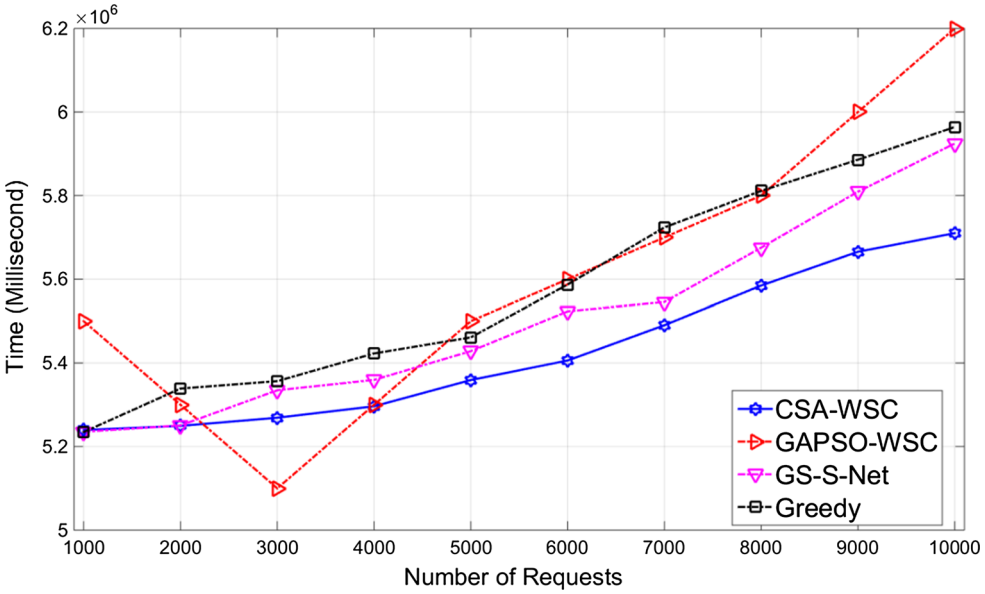


Fig. 25 Average requests of response time of CSA-WSC, GAPSO-WSC, GS-S-Net and Greedy algorithms in the third scenario

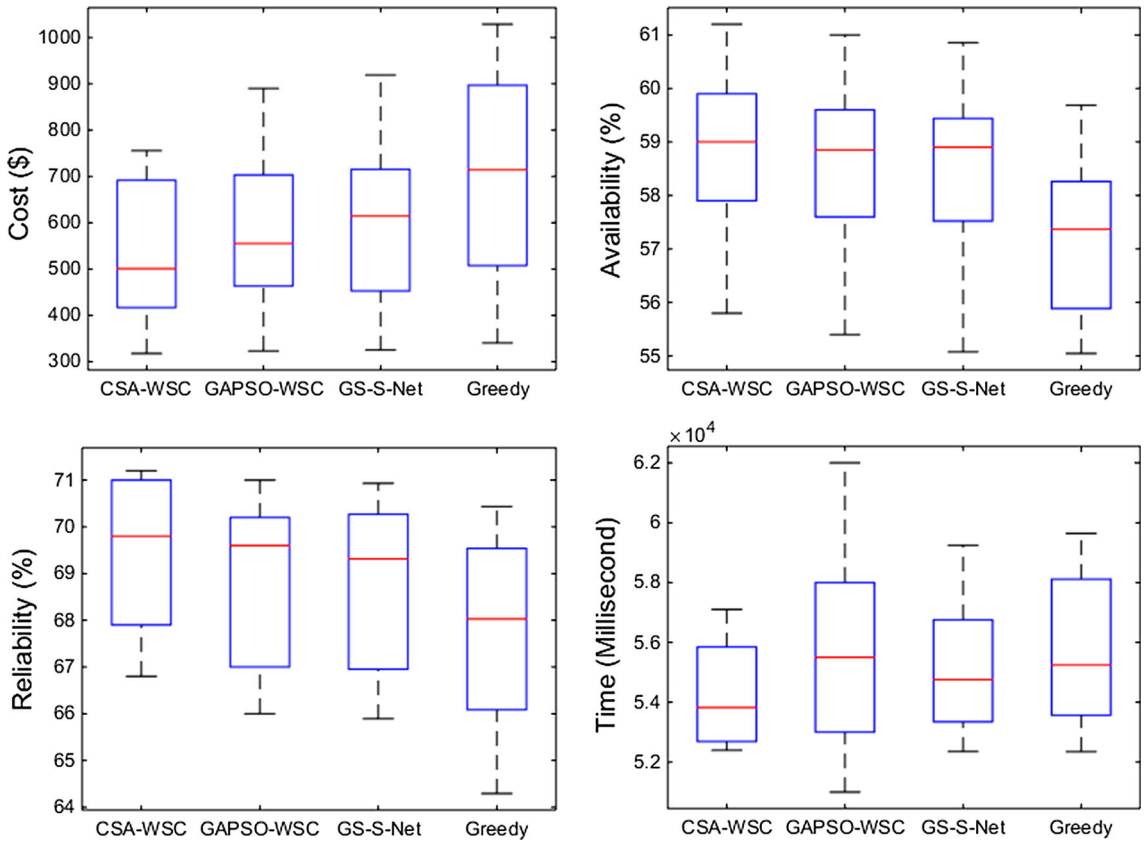


Fig. 26 Boxplot of different algorithms for different QoS criteria in the third scenario

generation and thus convergence of the algorithm to the target solution is quite fast.

According to Fig. 28, it is clear that the average requests of response time on the Greedy algorithm are higher than those on other algorithms. By comparing the values of GAPSO-WSC, GS-S-Net and CSA-WSC algorithms in Fig. 28, it becomes clear that the low diversity of service and proportion of the number of low service set, there is a slight

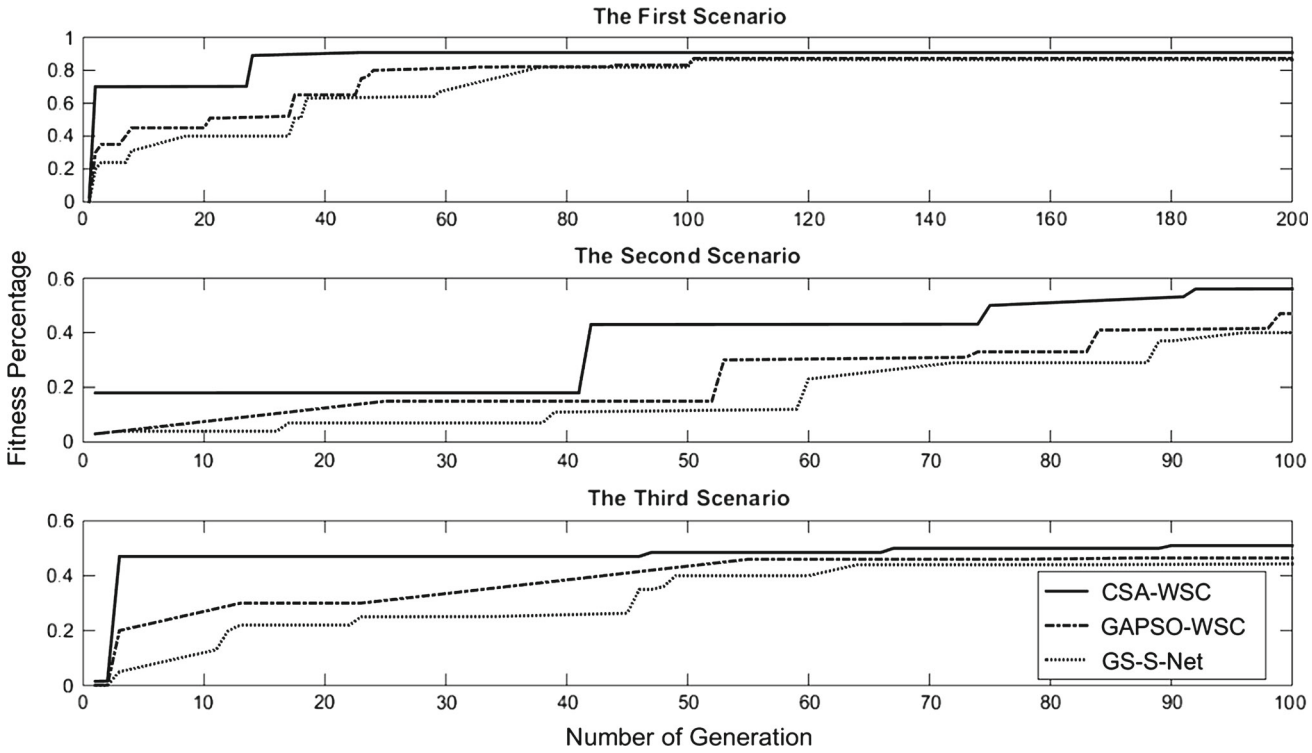
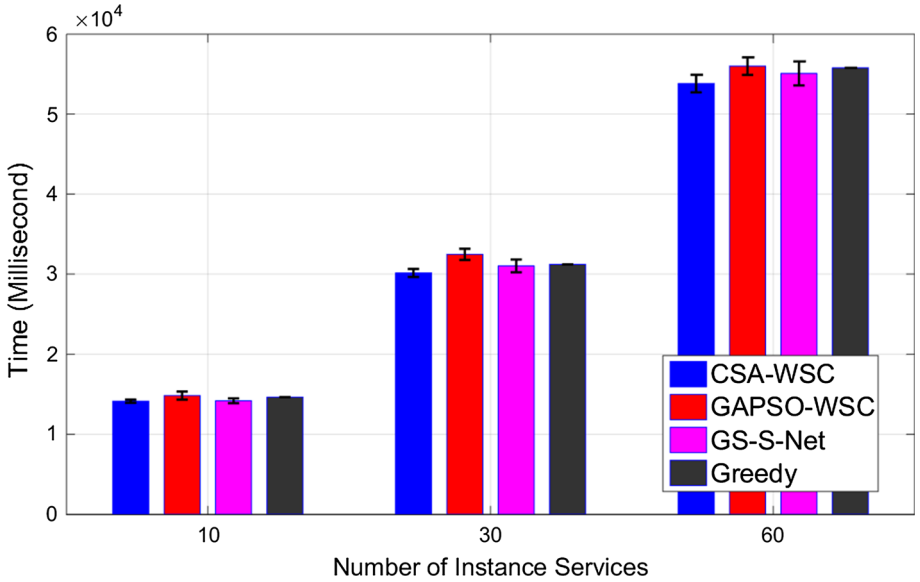


Fig. 27 Performance of CSA-WSC, GAPSO-WSC, GS-S-Net algorithm over generation

Fig. 28 Average requests response time in three positions 10, 30 and 60 samples of service



time difference between algorithms. By increasing any service diversity, CSA-WSC response time reduces compared to GAPSO-WSC and GS-S-Net algorithms, which represents the optimal performance of this algorithm in the high services diversity.

Figure 29 shows the resource allocation costs on three algorithms. According to Fig. 29, it is clear that when the numbers of providers and thus the number of sample services increase, the CSA-WSC decreases the availability of the composition cost and dedicates services. Regarding two main criteria such as response time and resource costs, it is concluded that the proposed CSA-WSC. Finally, the proposed CSA-WSC is a more suitable algorithm for web service composition problem in the geographically distributed cloud environment. While experimental results in some cases and criteria seem to be hard for evaluation, we here conduct paired *t* test in order to show whether there are significance differences between the performances of the algorithms. To do so, we executed the algorithms under three different number of instance services and ten different number of requests.

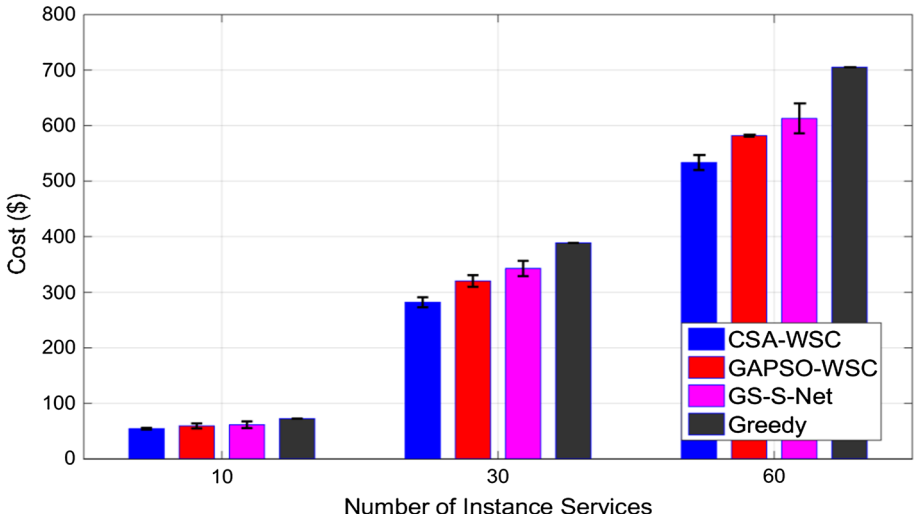


Fig. 29 Average cost of service composition in three positions 10, 30 and 60 samples of service

Table 16 Statistical comparison of proposed CSA-WSC with other baseline algorithms

Criteria	Algorithm	*N*	*Mean*	*SD*	*df*	*t* value	*p* value
Fitness	GAPSO-WSC	30	0.013678	0.014263	29	2.876899	0.010305
	GS-S-Net	30	0.012222	0.011234	29	3.264009	0.005729
Cost	GAPSO-WSC	30	23.05756	38.46142	29	3.283588	0.001339
	GS-S-Net	30	38.14815	53.1205	29	3.933435	0.00024
	Greedy	30	98.77608	86.63239	29	6.244995	0.040×10^{-09}
Availability	GAPSO-WSC	30	23.05756	38.46142	29	3.283588	0.001339
	GS-S-Net	30	38.14815	53.1205	29	3.933435	0.00024
	Greedy	30	98.77608	86.63239	29	6.244995	0.048×10^{-09}
Reliability	GAPSO-WSC	30	0.003033	0.006815	29	2.437818	0.01057
	GS-S-Net	30	0.003355	0.007316	29	2.511508	0.008922
	Greedy	30	0.013639	0.005954	29	12.54715	0.015×10^{-15}

Hence, we had 30 different experiments. Each experiment with a specific number of instance service and the number of requests is executed ten times, and the median results for the experiment are considered as the experiment result.

Table 16 shows the statistical results of paired t-test that determine the significance level of proposed algorithm compared to other algorithms regarding fitness, cost, availability and reliability. The table provides criteria, an algorithm to be compared with the proposed algorithm, the number of tests, mean and standard deviation differences between the proposed and baseline algorithms, degrees of freedom, t value and p value. To do a statistical evaluation, a paired t test with a significance level of $p < 0.05$ is done to evaluate if the differences were statistically significant. As is seen in Table 16, there is a meaningful difference between proposed algorithm and other algorithms, while p value in all cases is lower than 0.05. Given certainty, more than 0.95 proves that proposed algorithm has significant improvement in terms of all criteria. Thus, the null hypothesis is rejected, and it is shown that the differences, compared with the baseline algorithms, are significant.

6 Conclusion and future work

Today, users are increasingly accustomed to using the Internet to gain software resources in the form of web services. Through service composition technologies, loosely coupled services that are independent of each other can be integrated into value-added composited services. Most web services are deployed on cloud data centers distributed geographically around the world. In this paper, we have addressed the problem of web service composition in geo-distributed cloud environments. We have proposed a cuckoo search algorithm to solve the web service composition problem. Our algorithm considers not only the QoS of the web services but also the network QoS. The results of the simulation indicate that our algorithm can achieve a close to optimal result in terms of QoS criteria. In our future work, we aim at developing a

linear programming strategy. This will make our algorithm more practical and effective.

Compliance with ethical standards

Conflict of interest We have no conflict of interest to declare.

Ethical approval All procedures performed in studies involving human participants were in accordance with the ethical standards of the institutional and/or national research committee and with the 1964 Declaration of Helsinki and its later amendments or comparable ethical standards.

Human and animal participants This article does not contain any studies with human participants or animals performed by any of the authors.

Informed consent Informed consent was obtained from all individual participants included in the study.

References

Aslanpour MS, Ghobaei-Arani M, Toosi AN (2017) Auto-scaling web applications in clouds: a cost-aware approach. J Netw Comput Appl 95:26–41. doi:10.1016/j.jnca.2017.07.012

Bauer E, Adams R (2012) Reliability and availability of cloud computing. Wiley, Hoboken

Buyya R, Broberg J, Goscinski AM (2010) Cloud computing: principles and paradigms, vol 87. Wiley, Hoboken

Calheiros RN, Ranjan R, Beloglazov A, De Rose CA, Buyya R (2011) CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw Pract Exp 41(1):23–50

Chen F, Dou R, Li M, Wu H (2016) A flexible QoS-aware Web service composition method by multi-objective optimization in cloud manufacturing. Comput Ind Eng 99:423–431

Faruk MN, Prasad GLV, Divya G (2016) A genetic PSO algorithm with QoS-aware cluster cloud service composition. In: Thampi MS, Bandyopadhyay S, Krishnan S, Li K-C, Mosin S, Ma M (eds) Advances in signal processing and intelligent recognition systems. Springer, Cham, pp 395–405

Fouladgar N, Lotfi S (2016) A novel approach for optimization in dynamic environments based on modified cuckoo search algorithm. Soft Comput 20(7):2889–2903

Ghobaei-Arani M, Shamsi M (2015) An extended approach for efficient data storage in cloud computing environment. Int J Comput Netw Inf Secur 7(8):30

Ghobaei-Arani M, Jabbehdari S, Pourmina MA (2016) An autonomic approach for resource provisioning of cloud services. Cluster Comput 19(3):1017–1036

Ghobaei-Arani M, Jabbehdari S, Pourmina MA (2017a) An autonomic resource provisioning approach for service-based cloud applications: a hybrid approach. Future Gener Comput Syst. doi:10.1016/j.future.2017.02.022

Ghobaei-Arani M, Shamsi M, Rahmanian AA (2017b) An efficient approach for improving virtual machine placement in cloud computing environment. J Exp Theor Artif Intell. doi:10.1080/0952813X.2017.1310308

Gholami A, Ghobaei-Arani M (2015) A trust model based on quality of service in cloud computing environment. Int J Database Theor Appl 8(5):161–170

Huo Y, Zhuang Y, Gu J, Ni S, Xue Y (2015) Discrete gbest-guided artificial bee colony algorithm for cloud service composition. Appl Intell 42(4):661–678

Jula A, Sundararajan E, Othman Z (2014) Cloud computing service composition: a systematic literature review. Expert Syst Appl 41(8):3809–3824

Karimi MB, Isazadeh A, Rahmani AM (2016) QoS-aware service composition in cloud computing using data mining techniques and genetic algorithm. J Supercomput 73(4):1387–1415

Klein A, Ishikawa F, Honiden S (2014) SanGA: a self-adaptive network-aware approach to service composition. IEEE Trans Serv Comput 7(3):452–464

Koren I, Krishna CM (2010) Fault-tolerant systems. Morgan Kaufmann, Burlington

Kurdi H, Al-Anazi A, Campbell C, Al Faries A (2015) A combinatorial optimization algorithm for multiple cloud service composition. Comput Electric Eng 42:107–113

Lartigau J, Xu X, Nie L, Zhan D (2015) Cloud manufacturing service composition based on QoS with geo-perspective transportation using an improved Artificial Bee Colony optimization algorithm. Int J Prod Res 53(14):4380–4404

Liu B, Zhang Z (2016) QoS-aware service composition for cloud manufacturing based on the optimal construction of synergistic elementary service groups. Int J Adv Manuf Technol 88(9–12):2757–2771

Piprani B, Sheppard D, Barbir A (2013) Comparative analysis of SOA and cloud computing architectures using fact based modeling. In: Demey YT, Panetto H (eds) On the move to meaningful internet systems: OTM 2013 Workshops. Springer, Berlin, Heidelberg, pp 524–533

Portchelvi V, Venkatesan VP, Shanmugasundaram G (2012) Achieving web services composition-a survey. Softw Eng 2(5):195–202

Qi J, Xu B, Xue Y, Wang K, Sun Y (2017) Knowledge based differential evolution for cloud computing service composition. J Ambient Intell Humaniz Comput. doi:10.1007/s12652-016-0445-5

Rahmanian AA, Dastghaibyfard GH, Tahayori H (2017) Penalty-aware and cost-efficient resource management in cloud data centers. Int J Commun Syst. doi:10.1002/dac.3179

Rajabioun R (2011) Cuckoo optimization algorithm. Appl. Soft Comput 11(8):5508–5518

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

Simon B, Goldschmidt B, Kondorosi K (2013) A metamodel for the web services standards. J Grid Comput 11(4):735–752

Wang S, Sun Q, Zou H, Yang F (2013) Particle swarm optimization with skyline operator for fast cloud-based web service composition. Mobile Netw Appl 18(1):116–121

Wang D, Yang Y, Mi Z (2015) A genetic-based approach to web service composition in geo-distributed cloud environment. Comput Electric Eng 43:129–141

Wang GG, Deb S, Gandomi AH, Zhang Z, Alavi AH (2016a) Chaotic cuckoo search. Soft Comput 20(9):3349–3362

Wang H, Wang W, Sun H, Cui Z, Rahnamayan S, Zeng S (2016b) A new cuckoo search algorithm with hybrid strategies for flow shop scheduling problems. Soft Comput 18(1):116–121

Yu Q, Chen L, Li B (2015) Ant colony optimization applied to web service compositions in cloud computing. Comput Electric Eng 41:18–27

Zhao X, Shen L, Peng X, Zhao W (2015) Toward SLA-constrained service composition: an approach based on a fuzzy linguistic preference model and an evolutionary algorithm. Inf Sci 316:370–396

Zhou X, Liu Y, Li B, Li H (2016) A multiobjective discrete cuckoo search algorithm for community detection in dynamic networks. Soft Comput. doi:10.1007/s00500-016-2213-z

Zhou J, Yao X (2016) A hybrid artificial bee colony algorithm for optimal selection of QoS-based cloud manufacturing service composition. Int J Adv Manuf Technol 88(9–12):3371–3387