Computer Systems
Science and Engineering

Tech Science Press

DOI: 10.32604/csse.2023.037066
Article

Hybrid Chameleon and Honey Badger Optimization Algorithm for QoS-Based Cloud Service Composition Problem

G. Manimala* and A. Chinnasamy

Department of Computer Science & Engineering, Sri Sai Ram Engineering College, Chennai, Tamil Nadu, India
*Corresponding Author: G. Manimala. Email: manimalag25@gmail.com
Received: 21 October 2022; Accepted: 09 February 2023; Published: 26 May 2023

Abstract: Cloud computing facilitates the great potentiality of storing and managing remote access to services in terms of software as a service (SaaS). Several organizations have moved towards outsourcing over the cloud to reduce the burden on local resources. In this context, the metaheuristic optimization method is determined to be highly suitable for selecting appropriate services that comply with the requirements of the client's requests, as the services stored over the cloud are too complex and scalable. To achieve better service composition, the parameters of Quality of Service (QoS) related to each service considered to be the best resource need to be selected and optimized for attaining potential services over the cloud. Thus, the cloud service composition needs to concentrate on the selection and integration of services over the cloud to satisfy the client's requests. In this paper, a Hybrid Chameleon and Honey Badger Optimization Algorithm (HCHBOA)-based cloud service composition scheme is presented for achieving efficient services with satisfying the requirements of QoS over the cloud. This proposed HCHBOA integrated the merits of the Chameleon Search Algorithm (CSA) and Honey Badger Optimization Algorithm (HBOA) for balancing the trade-off between the rate of exploration and exploitation. It specifically used HBOA for tuning the parameters of CSA automatically so that CSA could adapt its performance depending on its incorporated tuning factors. The experimental results of the proposed HCHBOA with experimental datasets exhibited its predominance by improving the response time by 21.38%, availability by 20.93% and reliability by 19.31% with a minimized execution time of 23.18%, compared to the baseline cloud service composition schemes used for investigation.

Keywords: Cloud service composition; quality of service; chameleon search algorithm; honey badger optimization algorithm; software as a service

1 Introduction

Large stakeholders in computer technology enterprises and institutional management have already migrated to cloud computing due to the significance of low-cost access to reliable, high-performance hardware and software resources and the avoidance of upkeep costs [1,2]. The delivery of services via the cloud is developing into a model akin to that of conventional utilities like water, electricity, telephone, and gas [3,4]. Cloud computing encompasses not just the computer hardware and operating system software found in data centers but also the applications that may be used as services over the internet [5,6]. Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and SaaS are the three fundamental categories of cloud services. IaaS enables the deployment and operation of operating systems and applications by providing "processing, storage, networks, and other crucial computing resources" [7]. Applications generated using programming languages and tools can be consumed, created, or bought using PaaS. SaaS refers to applications cloud provider develops, hosts, and lets customers access online.

In cloud computing, there are primarily two types of individuals who make requests for resources and services: (1) single service request tasks, which are capable of being finished by utilizing a centralized data service, and (2) multiservice request tasks, which are capable of being finished by utilizing several different functions in a predetermined order [8,9]. Choosing the best service is relatively simple; however, forming a large-scope service with many points and cutoff points can be significantly more difficult.

Applications for services typically have three components: the "service requester," the "service provider," and the "service registry." The searches, invoke/bind, and publish processes all contribute to understanding the relationships between these three main components. To properly fulfill a complex request, different services are combined to create a composite service [10]. The four stages of the lifetime of a service composition are "definition, service selection, deployment, and execution." In the "specification" part, the "service requester" is the one who details the "service composition limits," which are meant to provide "adequate information about the user's requirements and preferences for composite service".

The restrictions are then automatically or semi-automatically dissolved into the abstract model, which defines a set of operations and information sequences interfacing with them, the exceptional performances, and QoS limitations [11]. There may be more than one application service that fulfills the standards, which is feasible and plausible. The best service must therefore be chosen as a result. After all the necessary web services have been discovered, the newly created composite service is generated. The composite service is set up after the selection section to enable user creation and operation.

The execution engine then creates and runs the composite service instance. The application of service composition as an important topic for service-oriented technology has received extensive study. In addition, the difficulty of QoS-aware service composition increases when it is assessed in the context of dynamic service scenarios with changing circumstances. The ever-changing character of service environments contributes to the emergence of two distinct difficulties. First, service selection and composition should be made during runtime because dynamic conditions require an instantaneous response to customer requests, and the availability of services cannot be predicted in advance. This is a best practice because dynamic conditions require instantaneous responses. As a result, while the problem's computational complexity is Non-deterministic Polynomial-time (NP)-hard, the execution time of service selection algorithms is strictly regulated.

As a result of these environments' dynamic nature, the second problem is the volatility of QoS conditions. During the deployment of the service composition, this problem arises when one or more of the services that make up the composition is no longer accessible or have their QoS decreased (due to network disconnections or bad network connectivity) [12]. The composition of various services from different providers is increasingly complex and difficult, and service providers grow, especially in a multi-cloud environment [13]. All service providers must measure an enormous amount of data, which consumes a lot of energy and resources [14].

Typically, a cloud user will interact with a cloud vendor to submit a job to a cloud service provider that includes functional and non-functional criteria [15]. Based on the parameters specified in the service level agreement (SLA), the broker selects the most beneficial service for the user's request [16]. A number of studies have already been done in this field, producing approaches and strong tools for customers but with drawbacks such as not looking at total energy use or assuming that the desired services are provided through a single cloud. Finding a composition method with the fewest services possible is a difficult problem [17].

Since there are so many services available, it is difficult to gauge how trustworthy they are. Service composition will only be widely accepted when users understand that the solutions they receive are built from reliable services [18]. One of the biggest problems in cloud computing is creating cloud services based on trust. Availability, responsibility, success potential, reliability, and agility makeup trust [19].

We should employ optimization methods to locate almost ideal solutions to NP-hard issues. Using their domain search space based on the natural structure that the species has, metaheuristic approaches inspired by nature struggle to obtain the "global optimal solution" [20]. Making clusters using data is one of the most efficient techniques to minimize distortion, identify related groups in large datasets, and shorten search times [21–23].

To determine which services can be relied on in the cloud, we proposed a trust assessment-based service composition approach. We implemented a hybrid optimization algorithm that combines the HCHBOA to reduce the search space required to find the best possible solution. The following objectives guide this paper:

- To fulfill users' demands, the right combination of services must be found.
- Additionally, the computation time required to complete users' requests must be reduced.
- Composing more trustworthy services is one way to increase users' trust.

The following summarizes the paper's organizational structure. Section 2 analyses and evaluates relevant research and the benefits and drawbacks of earlier approaches. Utilizing a trust-based HCHBOA optimization algorithm, In Section 3, we propose a strategy for composing cloud-based services. The outcomes of the simulations are given and discussed, and directions for further study are suggested in the final section.

2 Related Works

Numerous studies on the composition of cloud services have been conducted. This section refers to a few publications that analyze the cloud service composition and emphasize the key benefits and drawbacks.

2.1 Service Composition

Gao et al. [24] provided a way for employing evolutionary algorithms to assess confidence in a service composition. The relatively immediate solution may be found using this model's subjective probability theory-based trust evaluation model, which also provides an optimal method for finding subsequent generations to test the correctness of the solution's optimality. This strategy has the benefit of being more effective. This technique's considerably high intricacy results in an extension of the service composition's lifetime. This framework, however, increased the complexity.

Naseri et al. [25] have utilized the Particle Swarm Optimization (PSO) algorithm and developed a cloud service composition based on QoS. By asking the interface agent to determine the QoS constraints, the technique ensures service composition. This method's benefits include reducing waiting time and overall resource usage. Unfortunately, load balancing is ineffective. But load balancing is high in this framework.

Every cloud participant in Gutierrez-Garcia et al. [26] self-organizing agent-based service composition framework is represented by an agent. The following two agent-based methodologies are being used to assist the recently launched structure: "1st, resources with the agreement network protocol have been used to adaptively select services associated with service fees; 2nd, acquaintance networks with a list of common brokers and their capabilities are being used to cope with incomplete data about the locations of distributed cloud participants." The effectiveness of the service composition with insufficient data is the key benefit of this architecture. But this framework increased the probability of failure.

To solve a challenging biological optimization problem, González-Lvarez et al. [27] developed a system model of a honey bee mating optimization algorithm using MPI and OpenMP. To address the aforementioned issue, they, therefore, used multi-objective optimization. Six natural cases of varying sizes were performed to assess the effectiveness of their proposed strategy. By comparing their approach against other well-known tools in terms of parallel performance and getting great results from both angles, the evaluation findings demonstrated the program's applicability. Additionally, their research offered a method for parallelizing bio-inspired algorithms using cluster computing approaches. However, this framework increased the computational overhead.

A hybrid honey bee mating optimization algorithm is presented by Nayak et al. [28]. The classification issue in data mining has been resolved by fusing these two. According to the outcomes of experimental trials, the presented scheme offers superior stability, dependability, and accuracy compared to earlier methods like GA, PSO, and differential evolution. But this framework is not suitable for a real-time environment.

Additionally, the Ant Colony Optimized (ACO) approach for web service development is described by Alhadid et al. [29]. Using clustering and ACO methods, the presented framework provides an effective method for the best composition of semantic web services. It also makes it simple for customers to locate complementary services. A high joining capability set of online services is found using an "ant colony optimization" approach. Unfortunately, this framework has ineffective load balancing. But in this framework, the cloud load is very high, which leads to reduced operational speed.

An optimization approach based on a fuzzy is presented for cloud composition by Chahal et al. [30]. The data used to evaluate public opinion and direct trust came from analyzing four factors (security and performance) of five cloud service providers. Data generated by Nasuni, a cloud storage provider, was used to evaluate auditor trust. Combining direct trust, public scrutiny, and auditor trust

yields an estimate of the system's total reliability. Although the designed scheme is scalable and reliable, it does not handle many dynamic QoS factors.

For diagnosing the impact of trust on cloud services, Tang et al. [31] have also presented a methodology for selecting cloud services based on trust. The presented technique effectively selected service selection by combining objective and subjective trust assessments. But due to erroneous assumptions, dynamics, and dependability based upon those overwhelming ratings of service consumers.

Additionally, Messina et al. [32] have presented a reputation-based framework that can enhance the cloud service composition by taking into account charges and measurements of QoS. Although it appears that the technique has this issue, even though the service composition is enhanced by taking into account users' general perceptions, the actual improvement as a result of this composition might not always reflect the analysis of the prior clients' feedback mechanisms. This framework increased the computational cost.

2.2 Optimization Techniques

In order to reduce communication costs and financial fees, Al-Anazi et al. [33] have created a service composition algorithm. By ensuring that the cloud with the greatest range of services is always chosen before other clouds, the combinatorial optimization algorithm increases the likelihood that service requests may be fulfilled with the least overhead. The collected findings indicate that the combinatorial optimization technique keeps a small number of merged clouds while maintaining the number of services under examination, significantly affecting completion time. The findings are based on the observation that when a cloud's number of services offered is higher, there is a greater likelihood of identifying all the functions within that cloud. This framework requires additional resources to maintain the same performance.

There have been many studies done on the evaluation of trust levels in service composition. Haddad et al.'s suggested method focuses on QoS parameters instead of the customer feedback used in the previously discussed methods, which leads to unrealistic results. Additionally, the effectiveness of the suggested approach is improved by employing clustering based on trust assessments [34].

In addition, a Discrete Gbestguided Artificial Bee Colony (DGABC) algorithm for service composition is suggested by Huo et al. [35]. The suggested approach is expanded to include a time attenuation function, and service composition is defined as one of the nonlinear optimization issues. The suggested paradigm offers high-quality solutions and is more effective. But this framework increased the complexity. For the traveling salesman problem, a modified artificial bee colony approach is discussed by Choong et al. [36]. In this framework, the neighbourhood search heuristic utilized by the worker and watcher bees is chosen using the modified choice function. The Lin-Kernighan (LK) local search technique is merged with the suggested model to enhance efficiency. During various phases of the search process, the suggested model can adaptively control the weights of its intensification components. However, this framework leads to more memory consumption.

A self-adaptive ABC algorithm is developed by Xue et al. [37]. A new stochastic strategy was employed in the offered method to enhance the exploration ability of the spectator bee stage and to modify the working bee segment for optimal global capability. To avoid local minima, they also altered the startup phase. To begin the community, for example, the algorithm employs chaotic frameworks. It is possible to optimize the algorithm quite effectively. But this framework leads to more delay.

An adaptive multi-population differential artificial bee colony algorithm for many-objective service composition in cloud manufacturing has been proposed by Zhou et al. [38]. The algorithm provided a differential evolution mutation to modify the operator control parameters of the honey

bee for healthy progeny reproduction. The presented scheme is more adaptable to solving problems with various features thanks to the help of these strategies. However, this framework requires more energy.

For the bee colony optimization technique, a dynamic fuzzy-based dance mechanism is developed by Choong et al. [39]. This framework's dynamically regulated parameters outweigh the bee colony optimization linear technique's manual parameter tweaking restrictions. In terms of rewarding dance instances close to the inflection point, the provided dynamic fuzzy-based dance made a positive impact on the bee colony optimization linear algorithm. As a result, it works well for controlling the length of waggle dances. This framework, however, increased routing overhead.

A more advanced honey bee mating optimization system was reported by Solgi et al. [40]. The updated approach has been used to address problems involving many reservoirs, the optimization of natural resources, and numerous unrestricted and constrained arithmetic calculations. In comparison to other techniques, the one that is being provided is more effective and achieves an improved result with a smaller coefficient of assessment differentiation. This framework increased the probability of failure. The works mentioned in Table 1 are included with their advantages and disadvantages.

Table 1: Comparative analysis

Author's name	Methodology	Advantages	Disadvantages
Lakshmanna et al. [41]	Improved metaheuristic-driven energy-aware cluster-based routing	High packet delivery ratio (PDR)	Less throughput
Sundas et al. [42]	Modified bat algorithm with overloaded optimal virtual machine	Cost-effective framework	Increased energy consumption
Anuradha et al. [43]	The chaotic search-and-rescue-optimization-based multi-hop data transmission protocol	High PDR	Increased routing overhead
Alotaibi [44]	Tabu search algorithm	Enhanced searching process	Leads to computational overhead.
Rajendran et al. [45]	Optimal deep belief network	Reduced delay	High computational complexity
Jayapradha et al. [46]	Heap bucketization-anonymity model	High privacy with less utility loss	Increased routing overhead
Alotaibi [47]	Secured database management system architecture using intrusion detection systems	Very small FN and FP rates.	Require more resources to main stable performance

(Continued)

Table 1: Continued

Author's name	Methodology	Advantages	Disadvantages
Alsufyani et al. [48]	Intelligent data management framework for a cyber-physical system	Less complexity	Increased delay
Yan et al. [49]	Truncated singular value decomposition model for QoS	High efficiency, high accuracy.	Increased time complexity
Qi et al. [50]	Correlation graph-based approach	Compatible framework	Increased delay
Nitu et al. [51]	Machine learning-based travel recommendation system	High accuracy	High cost
Zhang et al. [52]	User recruitment algorithm	Improves the quality of sensed data	Increased routing delay

3 Proposed Methodology

We outline the cloud model that is used to compose services in this section. The problem specification is then provided, some definitions of the essential concepts and a clustering approach. In the next section, a clustering technique based on trust level is suggested for cloud service composition. The paradigm for cloud service composition is shown in Fig. 1. Loop, switch sequence, and flow are the composition paradigms for service composition that are used most frequently. They are illustrated by $*$, $\otimes$, $\rightarrow$, and $\oplus$ symbols [53]. The service composition framework is presented in Fig. 2. It is illustrated with numerical symbolization as $[[A1 \rightarrow A3] \oplus A2] \rightarrow [A4 \otimes A5] \rightarrow A6$. The proposed methodology for service composition utilizing the HCHBOA technique and trust-based clustering is presented in this section. The proposed technique's initial step involves measuring trustworthiness using QoS criteria.

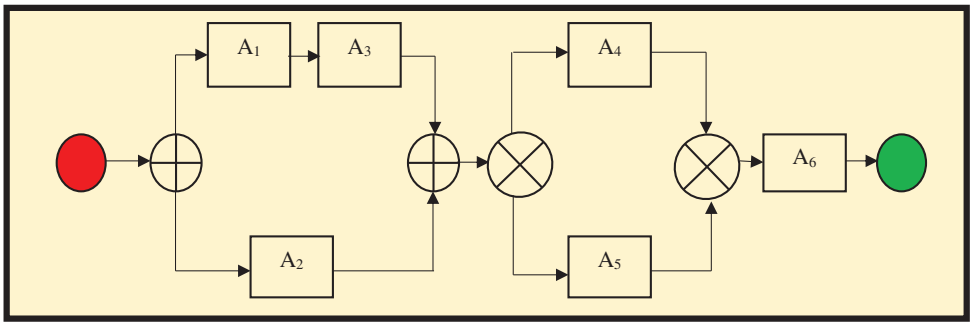

Figure 1: Service composition framework

3.1 Evaluation of Trust Degree

To determine the level of service trust, several parameters are calculated, which are presented as follows:

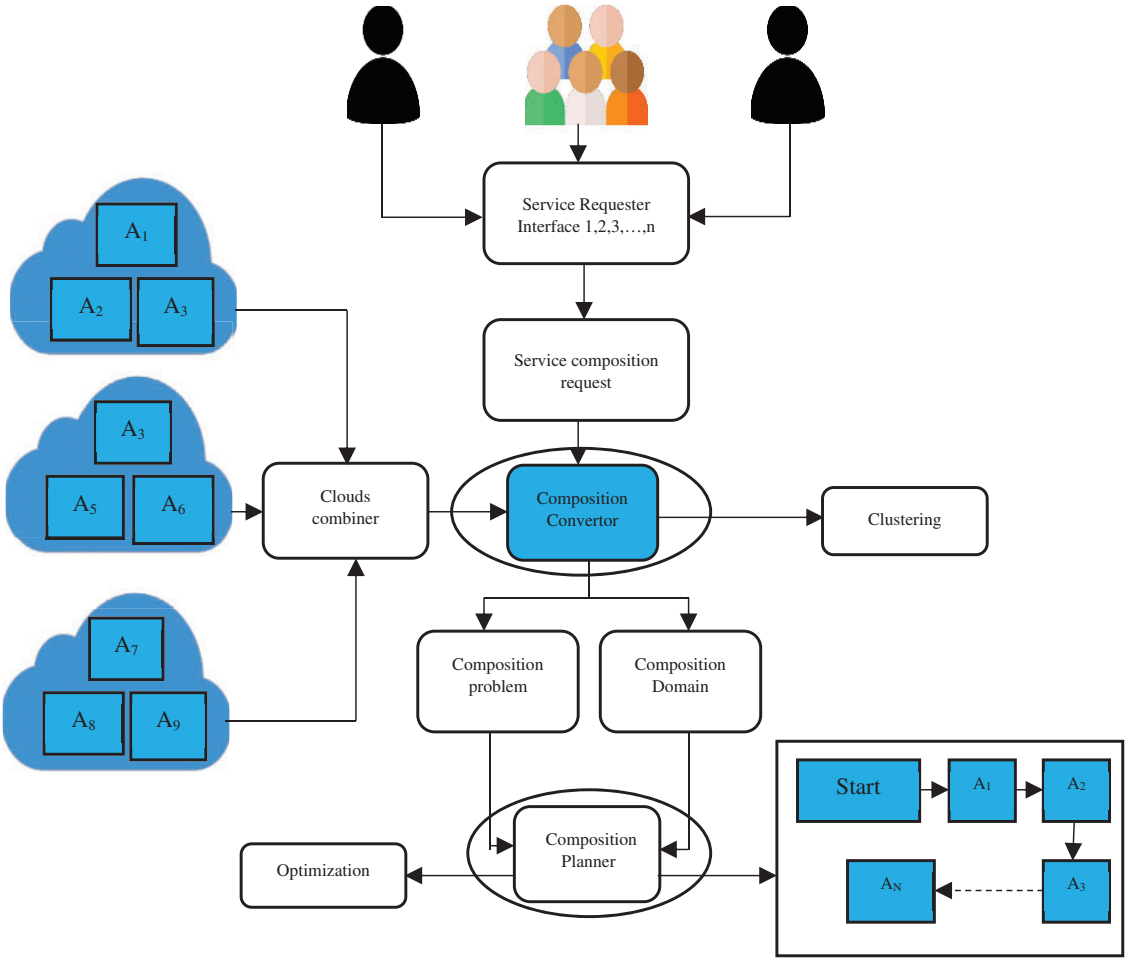

Figure 2: Illustration of a basic_service_composition framework

3.1.1 Availability

When a cloud service is available and usable when required, this is indicated by its availability degree. The service operates for the total time required or calculable to function. In Eq. (1), availability is determined qualitatively as follows:

$$Availability\ (AV_l) = \frac{B}{C} \tag{1}$$

where, $l \rightarrow$ *cloud resources* $= 1, 2, \ldots, m, \rightarrow C$ no. of jobs submitted to cloud resources, and $B \rightarrow$ No. of a job accepted by the resource l.

3.1.2 Responsibility

The ability of the service to carry out its key activities by the preset conditions throughout the predefined amount of time is known as responsibility. Eq. (2) is used to determine responsibility, as seen below:

$$Responsibility\ RS_l = \frac{D}{E} \tag{2}$$

For each $l = 1, 2, 3, \ldots m$, D $\rightarrow$ no. of jobs succumbed to l in the specified time (ms) and E $\rightarrow$ number of requests present.

3.1.3 Reliability

The foundational component of determining trust is reliability. Software or hardware can do a given job within a certain amount of time by the specified requirements. Eq. (3) can be used to compute the ratio of error messages to total messages as follows:

$$Reliability\ REL_l = \frac{J}{C} \tag{3}$$

For each $l = 1, 2, 3, \ldots m$, C $\rightarrow$ no. of jobs acknowledged by the cloud l & J $\rightarrow$ no. of jobs completed by cloud.

3.1.4 Trust Evaluation

In this section, the trust value is determined using the results of measured equations. Eq. (4) measured the trust value, which was presented as follows:

$$Trust\ (T) = Z_{AV} * AV_l + Z_{RS} * RS_l + Z_{REL} * REL_l \tag{4}$$

where Z_{AV}, Z_{RS}, $Z_{REL} \rightarrow$ trust parameter's positive weights that $Z_{AV}+, Z_{RS} + Z_{REL} = 1$. According to their importance, the weights of these trust factors are shown. For example, if $Z_{AV} = 0.2$, $Z_{RS} = 0.3$, and $Z_{REL} = 0.5$. The final trust measure should range from 0 to 1 because reliability is important. In the next phases of the suggested technique, the trust value is employed as fitness.

3.2 Clustering

Making clusters from heterogeneous data is one of the most efficient techniques to reduce distortion, identify comparable groupings within the data, and shorten search times. Given a collection of n data points, the mean squared distance between each and its greatest center should be reduced. Eq. (5) is used to calculate distance based on Euclidean distance as follows:

$$d = \sqrt{\sum_{j=1}^{m} (q_{a_{wx}} - q_{s_{wx}})^2} \tag{5}$$

where "$q_s \rightarrow$ the rate of the l^{th} attribute for the x^{th} service, $q_a \rightarrow$ the value of the w^{th} attribute for the x^{th} cluster center". The total within-cluster variance is known as the "Mean Square quantization Error" (MSE). Assessing the quality of the k-clustering using this method is one of the most dependable and effective ways to do it. It is defined in Eq. (6) as follows:

$$Perf\ (X, C) = \sum_{j=1}^{N} Min\{||X_i - C_l||^2\ l = 1, \ldots .k\} \tag{6}$$

The first step of the K-means algorithm is to randomly select K cluster centers and assign the points to the clusters. After calculating each new cluster center and assessing whether the requirements were satisfied, the cycle ends; however, if the conditions are not met, the cycle continues by giving fresh points to the clusters.

3.2.1 Qualitative Attribute Normalization

In this section, the qualitative characteristics of services are discussed. The data set used for the simulation phase reveals several ranges for subjective service qualities, such as reaction time, which

ranges from 0 to 10,000 ms for each service, and dependability level, which ranges from 0 to 100. To standardize all the qualities into a period between 0 and 1, apply Eqs. (7) and (8).

$$q^{l}_{s_{wx}} = \begin{cases} \dfrac{q^{l}_{s_{wx}} - Min\left(q^{l}_{s_x}\right)}{\left(Max\left(q^{l}_{s_{wx}}\right) - Min\left(q^{l}_{s_x}\right)\right)} 1 \,, \left(Max\left(q^{l}_{s_{wx}}\right) - Min\left(q^{l}_{s_x}\right)\right) \neq 0 \\ \left(Max\left(q^{l}_{s_{wx}}\right) - Min\left(q^{l}_{s_x}\right)\right) = 0 \end{cases} \tag{7}$$

$$q^{l}_{s_{wx}} = \begin{cases} \dfrac{Max\left(q^{l}_{s_{wx}}\right) - q^{l}_{s_{wx}}}{\left(Max\left(q^{l}_{s_{wx}}\right) - Min\left(q^{l}_{s_x}\right)\right)} 1 \,, \left(Max\left(q^{l}_{s_{wx}}\right) - Min\left(q^{l}_{s_x}\right)\right) \neq 0 \\ \left(Max\left(q^{l}_{s_{wx}}\right) - Min\left(q^{l}_{s_x}\right)\right) = 0 \end{cases} \tag{8}$$

Eq. (7) is being used to maximize the requirements for positive QoS traits like availability and reliability, whereas the negative criteria (cost and time) standards are reduced using Eq. (8). In Eqs. (7) and (8), $q^{l}_{s_{wx}}$ l^{th} attribute of the x^{th} service from the x^{th} service, Max $q^{l}_{s_{wx}}$, and Min $q^{l}_{s_x} \rightarrow$ Max. And Min. values of the l^{th} attribute for the x^{th} service.

3.3 Proposed Hybrid Optimized Solution for QoS-Based Cloud Service Composition Problem

This section presents a quick overview of the CSA and HBA.

3.3.1 Chameleon Swarm Algorithm (CSA)

Chameleons, known for their ability to change color to adapt to their surroundings, are effective hunters. Because of this quality, chameleons are excellent explorers. The developers of the CSA point out that chameleon have a unique capacity to explore, which allows them to locate their prey using this investigative function [54]. Chameleons are distinguished in that each of their two eyes may move in a manner that is independent of the other. They can keep track of their prey even when moving quickly because of their wide-angle vision. Chameleons have the ability to move their tongues extremely swiftly, which helps them in the hunt. Because of these capabilities, it is possible to assert that it is an effective hunter. One may draw parallels between fast convergence and the capacity to utilize one's tongue swiftly. Because their eyes can move in any direction and see 360 degrees, they can effectively explore the search area. There are two categories of exploratory abilities and four categories of hunting skills that are discussed in this algorithm. Desert exploration and tree exploration are two of its talents when it comes to exploring. One of their primary hunting skills is the ability to swiftly move their heads around and focus on a single target without drawing the attention of the target animal. The second talent it has is the capacity to visually lead its prey by turning its eye to the right. Because of its long tongue, it can stalk its prey from a distance without having to go too near it, which is its third talent. It can hunt by approaching its victim, the fourth talent it has. This algorithm contributes to hunting with its exploration power since it searches the whole space. The chameleon swiftly converges on its prey and has a fast convergence curve even though it dominates the whole space. Sixty-seven non-comparison tests, such as the CSA in the EC-2015 and CEC-2017 test suites, are used to evaluate the proposed system and provide a comparison.

The following Eq. (9) is used to determine the CSA's exploration:

$$Y^{i,j}_{t+1} = \begin{cases} Y^{i,j}_{t} + p_1\left(P^{i,j}_{t} - G^{j}_{t}\right) r_2 + p_2\left(G^{j}_{t} - Y^{i,j}_{t}\right) r_1, r_i \geq P_p \\ Y^{i,j}_{t} + \mu\left(\left(u^{j} - l^{j}\right) r_3 + l^{j}_{b}\right) sgn\left(rand - 0.5\right), r_i \leq P_p \end{cases} \tag{9}$$

The letter i designates the chameleon order. t stands for the total number of iterations, while j indicates the magnitude of the task being tackled by the current iteration. The position that will be in the following iteration is denoted by Y^{i}_{t+1}, while the position that is now held by Y^{i}_{t}, is denoted by the notation. P^{i}_{t}, the chameleon, is now at the most advantageous position in the t. iteration G^{j}_{t}, t identifies the position that has been in the best position during all previous iterations up to the t. iteration. The

numbers p_1 and p_2 are the ones that are used to govern who has the capacity to do reconnaissance. r_1, r_2, and r_3 are three random integers between 0 and 1 that were chosen at random. The letter P_p denotes the chance that the chameleon will chase its prey. The power to seek and find information is controlled by the value ($rand - 0.5$), which is found in the range [−1, 1]. The parameter denoted by the symbol μ is specified in the number of iterations. Fig. 3 provides a visual representation of the CSA's operational mechanism.

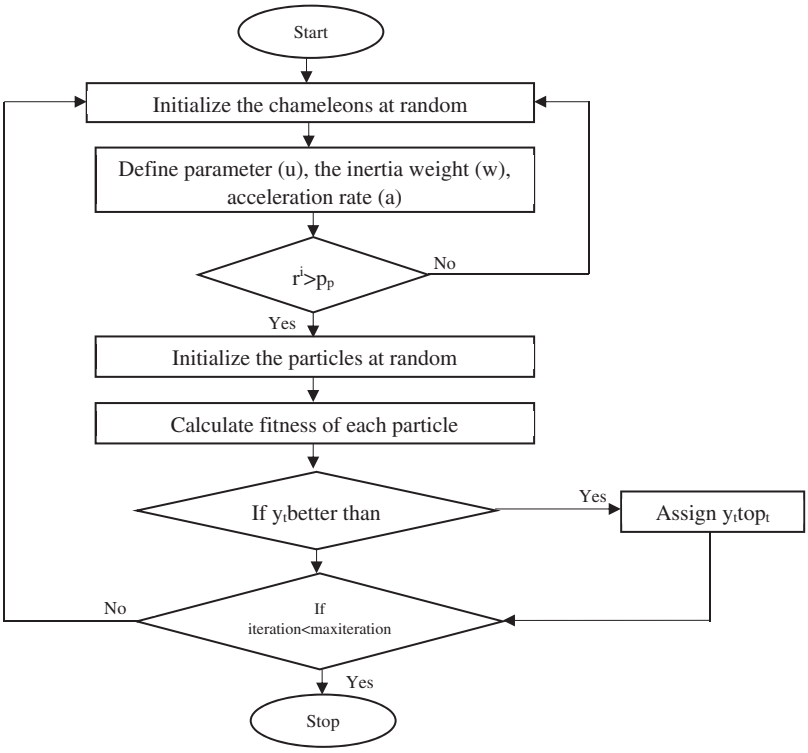

Figure 3: Flowchart of CSA

3.3.2 HBA Algorithm

The HBA [55] was introduced to computer science after its creators were motivated to create a new algorithm by observing the hunting and foraging habits of a wild animal known as the honey badger. The honey badger is a kind of animal found in areas of Africa's deserts and rainforests. It is also a species of animal found in southwest Asia and India. It is famous for its fearlessness in the natural world. It has a search method that enables it to discover and hunt 67 species, including some that are very deadly. Honey is one of the things that the honey badger enjoys eating the most. This animal can find honey using a variety of instruments. Honey badgers engage in two distinct forms of hunting and foraging in the wild. The initial part of the process is known as the digging phase. In the digging part of their hunting process, honey badgers locate their prey using their acute sense of smell to look for the animal. It approaches its prey by burrowing and scavenging in the area. The honey stage is the second step in the development process. At this point, the honey badger's sense of smell is insufficient to seek out honey on its own; therefore, it needs the assistance of a guide. The pilot bird and the honey badger use a similar search and hunting approach. The honey badger is directed during its quest by a bird. The honey badger employs this tactic by following after the guided bird. The job of looking for

birds is taken on by the guide. Who utilizes a distinct search method compared to the honey badger and successfully locates honey with the assistance of the guided bird. However, the bird is unable to access the honey. The honey badger's skill with tools comes into play at this stage, and it ultimately succeeds in obtaining honey. Because of this, the honey badger is able to make up for its deficiencies in the search regions where it is inadequate, thanks to the assistance of a guided bird. It is shown in Reference [55] that the HBA has been used to address a wide variety of engineering challenges, with successful outcomes in each case. The capacity of the honey badger to seek and use its environment uniquely may be measured using the following Eqs. (10) and (11).

$$Div_j = \frac{1}{N}\sum_{i=1}^{N} median\left(x^j\right) - x_i^j \tag{10}$$

$$Div^t = \frac{1}{N}\sum_{j=1}^{N} Div_j, t = 1, 2, .., t_{max} \tag{11}$$

Honey badgers are classified according to their order, which is denoted by i, and their size, which is denoted by j. The honey badger solution candidate is indicated by the letter *xij*. The *Divj* variety represents the typical range for dimensions. At the conclusion of each and every cycle, the dimensional diversity (D) is averaged, and that is what *Divt* represents. After calculating the variety, the next step is to apply the following Eqs. (12) and (13) to get the proportion of both exploitation and exploration:

$$Exploration\% = \frac{Div^t}{\max(Div)} \times 100 \tag{12}$$

$$Exploitation\% = \frac{Div^t - \max(Div)}{\max(Div)} \times 100 \tag{13}$$

$t_{max} \rightarrow \max(Div)$ diversity in maximum iteration. Each kind of food or prey that may be found in the HBA ecology is a potential answer to the QoS issue. In this scenario, the location of each piece of food or prey is comparable to the ideal position for LAA components. The quantity of food available at each place is intended to simulate selecting clusters. When there is a greater quantity of food or prey, a better clustering solution may be found. Fig. 4 presents the HBA flowchart in its entirety.

3.3.3 Proposed HCHBOA Algorithm

The conventional Firefly and CSA algorithms have a severe flaw that causes them to become stuck in local minima, called the trapping mechanism. As a result, in order to tackle the trapping concerns, we present a new hybrid model that we term HCHBOA. This model is a fusion of CSA and HBA. In this instance, we have chosen to use the HCHBOA as the primary engine for the construction of cloud services. The primary operator of HCHBOA is a hybrid operator, which includes elements of both CSA and HBA in its operation. We have considered the cloud composition as the multi-objective optimization function in which the two objective functions exist. One of these objective functions is reduced complexity, and the other is high performance in terms of availability, reliability, and responsibility. For the best performance, we have considered the cloud composition as the multi-objective optimization function. The initial solutions are sorted based on their fitness function, which is determined by the trust assessment, and the fitness of each cloud is compared to that of the best solution to determine which solution is the best. The whole technique for hybrid optimization will be

carried out once more until the final conditions are satisfied. Therefore, the fitness function for each iteration is presented in the following Eq. (14).

$$\text{Fitness Function P} = \mu\alpha\,(A) + \beta\left(\frac{S}{N}\right) \tag{14}$$

where $\mu\alpha\,(A)$is the function of maximum performance in terms of availability, reliability, and responsibility. S is cloud services, and N is cloud resources. μ is the main function that denotes the accuracy classification and trust degree.

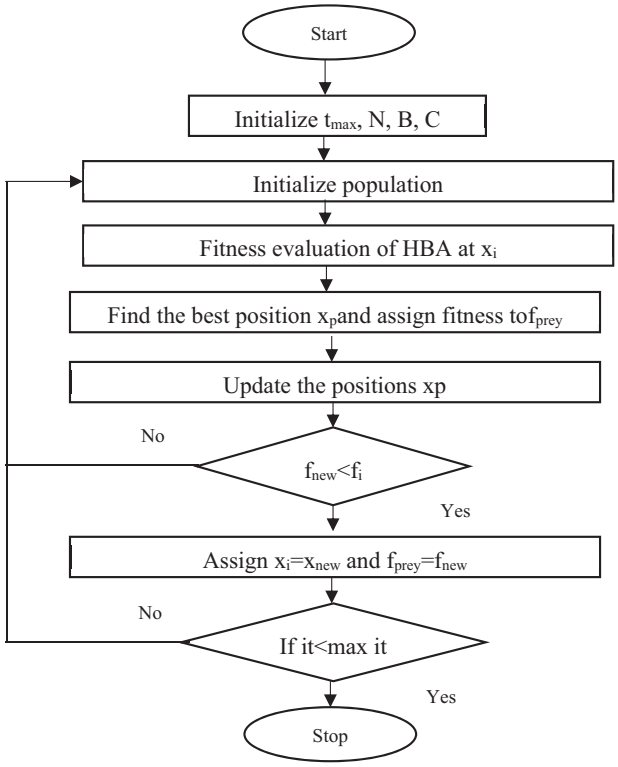

Figure 4: Flowchart of HBA

The Working Mechanism of the Proposed Hybrid Optimization Algorithm

The suggested HCHBOA algorithm's operation is described in the following manner: Sets of solutions for the initialization portion are produced randomly. The assessment will take place after creating new solutions that meet the limits of service composition. This step involves calculating and comparing the trust value of fresh, workable solutions to those offered previously. If the new number is higher, the old queen is replaced by the new one. The level of trust is higher in option 4. As a result, it gets chosen as the next queen. Fig. 5 shows the nine steps of the suggested method.

Step 1: Defining the settings for the model's input data and method (i.e., input data = QWS, 6 clusters, trust request = 0.106, n = tag, m = service)

Step 2: Trust level-based clustering services.

Step 3: Receiving the client's request (i.e., request = 1, 2, 2, 5, 4)

Step 4: The starting solutions are generated at random.

Step 5: Based on the fitness function Arranging the initial entities (i.e., solutions 1 & 2 have fitness functions of 0.3886, 0.3873, 0.3242, and 0.2485, respectively).

Step 6: Make the best one the best option.

Step 7: for each trial, determine the fitness and contrast it to the best available option.

Step 8: If the new solution has a higher fitness function, replace the existing one with the best one.

Step 9: If the conditions are met, stop the calculation; in any case, reject all previous preliminary arrangements and make new ones, then return to stage 7 and continue until the cycle is completed (the final result will be the best help service succession based on the most noteworthy trust in the proposed strategy).

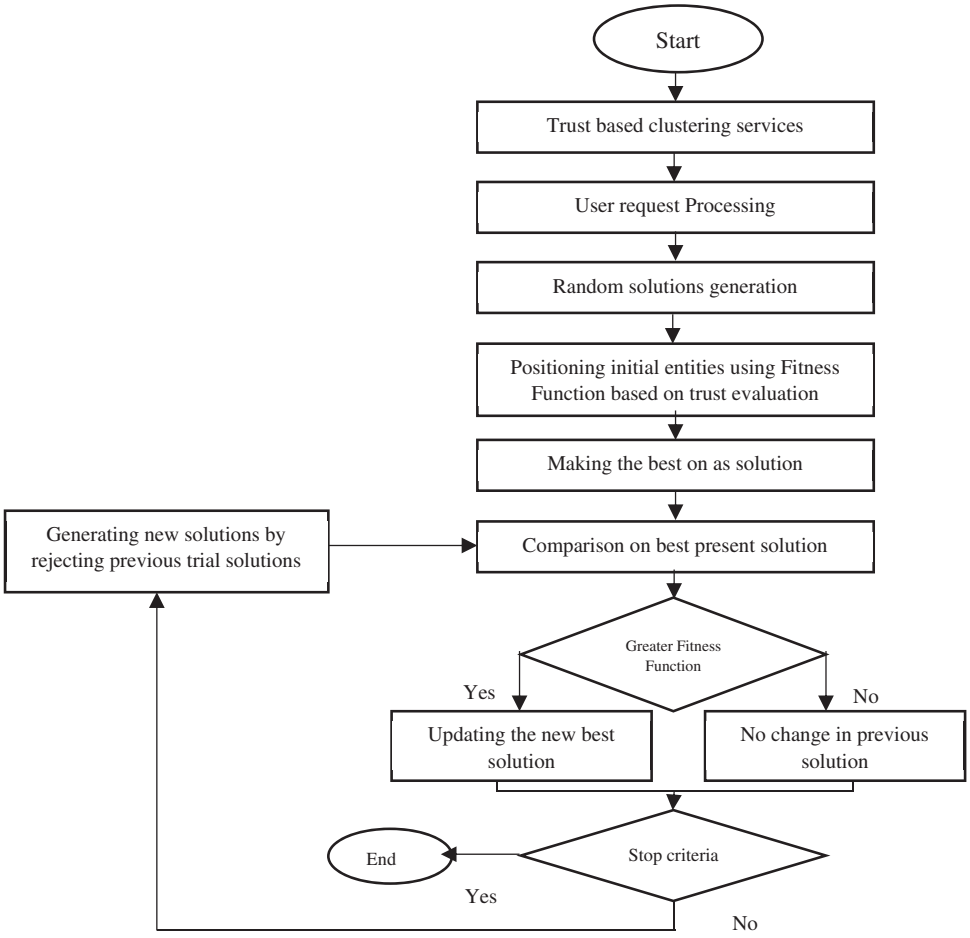

Figure 5: Flow chart of the proposed hybrid optimization

The flowchart is shown in Fig. 5 above. To select the best solution, the cloud's fitness function is compared with the best solution in every iteration. It will go through the entire evolutionary process again until it meets the final requirements. The pseudo-code for the proposed model is as follows.

Pseudo-Code of HCHBOA

```
1   Initialization
2   Get user request
3   Define all the requested services
4   Read all the services
5   Filter the clustering services based on the trust degree
6   Xbest_Value is the global best solution
7   Calculate the fitness function using Eq. (14)
7   If (Xbest_Value = fitness function)
8   Fix as the best solution
9   Elseif (Xbest_Value < fitness function)
11      No change in the previous solution
12  Elseif (Xbest_Value > fitness function)
13      Update the new solution as the best solution
14  End
15  End
16  End
```

The above hybrid operator (see Line number 22) eradicates the trapping problems. According to the proposed algorithm, HCHBOA optimization has a very simple structure. It is important to mention that the only difference between hybrid HCHBOA and other traditional algorithms, such as CSA and HBA algorithms, is that the CSA property is integrated into HBA algorithms. The computational complexity and computation time have been greatly reduced when compared with other algorithms.

To maintain security, the QoS parameters included in the proposed service compositions are means for the system to learn about the workload in terms of response time, reliability, and availability. This considerably minimizes the chance of being unable to react to client requests or exceed the service composition's maximum capacity. When the proposed framework gets the maximum service requests, it may deliver an acceptable SLA response time. Because of the combination of CSA and HBA, it has less computational and time complexity.

4 Experimental Results

The quality web service (QWS) data set assesses the suggested algorithm's effectiveness. It is a public data set [56,57] created to provide a foundation for cloud computing developers. There are 2507 real-world web services in the large dataset. Each of them has a set of nine QWS QoS criteria, including throughput, response time, success ability, availability, reliability, latency, etc. 500 000 randomly produced web services with five random boundaries (time, service URL, accessibility service name, and response time) are created in order to test the suggested algorithm in a huge situation. In this part, we contrast our approach with the DGABC [58], PSO [59], and GA [60] algorithms. We also test the responsiveness, dependability, and demand for cloud services.

Fig. 6 shows the proposed framework's performance in terms of availability. It shows that as the number of iterations increases, the proposed HCHBOA algorithms maintain performance in the range of 99.23% to 99.98%. In contrast, the other algorithms, such as DGABC, PSO, and GA, gradually lose performance. Fig. 7 shows the proposed framework's performance in terms of responsibility, responsibility, and reliability. It shows that as the number of iterations increases, the proposed HCHBOA algorithms maintain performance from 99.75% to 99.98%. In contrast, the other algorithms, such as DGABC, PSO, and GA, gradually lose performance. Fig. 8 shows the proposed framework's performance in terms of responsibility. The figure shows that as the number of iterations increases, the proposed HCHBOA algorithm maintains performance from 80.97% to 99.98%, whereas the other algorithms, such as DGABC, PSO, and GA, gradually lose performance. Based on the performance analyses, the proposed HCHBOA algorithm exhibited dominance by improving availability by 20.93%, reliability by 19.31%, and response time by 29.98% compared to the baseline cloud service composition schemes used for investigation.

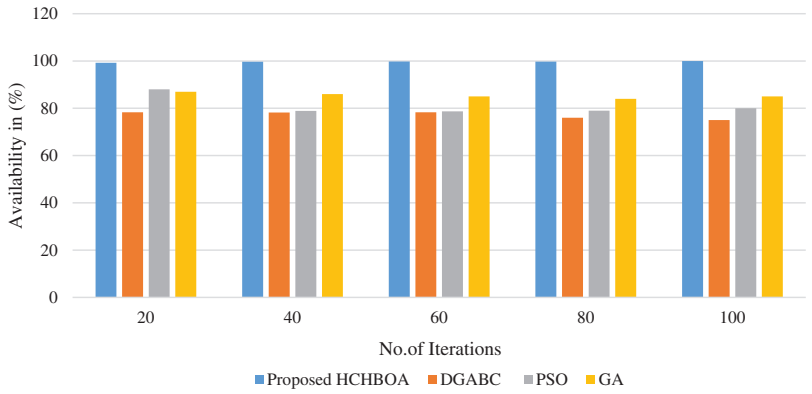

Figure 6: Availability analysis of the proposed hybrid HCHBOA algorithm with other existing frameworks

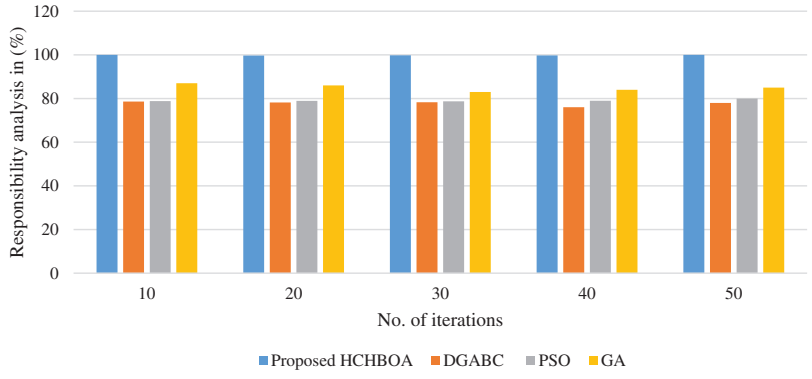

Figure 7: Responsibility analysis of the proposed hybrid HCHBOA algorithm with other existing frameworks

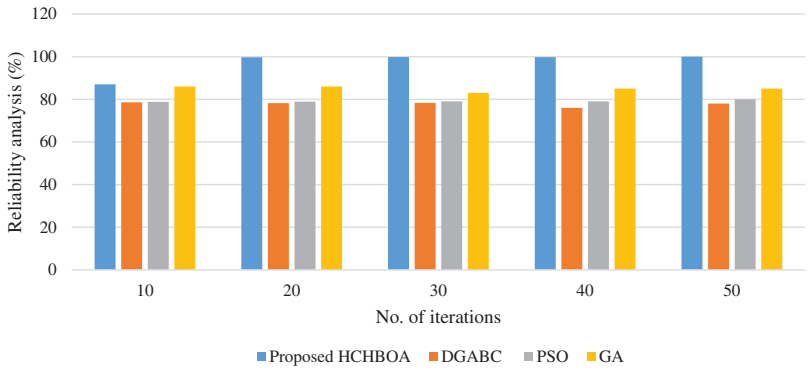

Figure 8: Responsibility analysis of the proposed hybrid HCHBOA algorithm with other existing frameworks

5 Conclusion

The reduction of expenses, which includes execution time and other pertinent factors, poses significant difficulties for cloud frameworks to satisfy users. Because of the increasing relevance of service trust issues and their selection with better and more trustworthy cloud services, we created an HCHBOA algorithm with trust-based clustering. The clustering of web services that are based on trust is where this method's procedure begins. We choose services and merge the services by the specified goals using the suggested HCHBOA optimization algorithm. The findings showed that the suggested solution operates effectively in terms of computing time and comprises higher-trusted services; however, it is ineffective for huge data sets. Future research on the following topics could be conducted to improve the method that was suggested in this study:

- Optimizing the calculation of the ideal service under the current restrictions
- Enhancing the services' filtration in large-scale services.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] M. S. Abou El-Seoud, I. A. Taj-Eddin, N. Seddiek, M. M. El-Khouly and A. Nosseir, "E-learning and students motivation: A research study on the effect of E-learning on higher education," *International Journal of Emerging Technologies in Learning*, vol. 9, no. 4, pp. 20–26, 2014.

[2] L. Rajabion, A. A. Shaltooki, M. Taghikhah, A. Ghasemi and A. Badfar, "Healthcare big data processing mechanisms: The role of cloud computing," *International Journal of Information Management*, vol. 49, no. 1, pp. 271–289, 2019.

[3] F. Seghir and A. Khababa, "A hybrid approach using genetic and fruit fly optimization algorithms for QoS-aware cloud service composition," *Journal of Intelligent Manufacturing*, vol. 29, no. 8, pp. 1773–1792, 2018.

[4] C. Huang, X. Wang and D. Wang, "Type theory based semantic verification for service composition in cloud computing environments," *Information Science*, vol. 469, no. 2, pp. 101–118, 2018.

[5] F. Jamal and R. Z. Khan, "Emerging technologies and developments in cloud computing: A systematic review," *International Journal of Emerging Trends in Engineering Research*, vol. 8, no. 3, pp. 894–905, 2020.

[6] B. Yuan, C. Zhang, X. Shao and Z. Jiang, "An effective hybrid honey bee mating optimization algorithm for balancing mixed-model two-sided assembly lines," *Computers & Operations Research*, vol. 53, no. 1, pp. 32–41, 2015.

[7] A. Jula, E. Sundararajan and Z. Othman, "Cloud computing service composition: A systematic literature review," *Expert Systems and Applications*, vol. 41, no. 8, pp. 3809–3824, 2014.

[8] S. T. Milan, L. Rajabion, H. Ranjbar and N. J. Navimipoir, "Nature inspired meta-heuristic algorithms for solving the load-balancing problem in cloud environments," *Computers & Operations Research*, vol. 110, no. 1, pp. 159–187, 2019.

[9] G. Javidi, E. Sheybani and L. Rajabion, "Moving from cloud to fog: Implications and challenges," in *9th Int. Multi-Conf. on Complexity, Informatics and Cybernetics*, Orlando, Florida, USA, pp. 53–58, 2018.

[10] S. N. Samreen, N. K. Valmik and S. M. Salve, "Introduction to cloud computing," *International Research Journal of Engineering and Technology*, vol. 5, no. 2, pp. 785–788, 2018.

[11] P. Pooja, C. Elakkiya and S. Soundharya, "E-voting in cloud using cryptography algorithm," *SSRG International Journal of Computer Science and Engineering*, vol. 28, no. 2, pp. 29–35, 2019.

[12] F. Tao, D. Zhao, Y. Hu and Z. Zhou, "Resource service composition and its optimal-selection based on particle swarm optimization in manufacturing grid system," *IEEE Transactions on Industrial Informatics*, vol. 4, no. 4, pp. 315–327, 2008.

[13] F. Chen, R. Dou, M. Li and H. A. Wu, "Flexible QoS-aware Web service composition method by multi-objective optimization in cloud manufacturing," *Computers & Industrial Engineering*, vol. 99, no. 2, pp. 423–431, 2016.

[14] C. Jatoth, G. R. Gangadharan and R. Buyya, "Optimal fitness aware cloud service composition using an adaptive genotypes evolution based genetic algorithm," *Future Generation Computer Systems*, vol. 94, no. 4, pp. 185–198, 2019.

[15] B. B. Traore, B. K. Foguem, F. Tangara and X. Desforges, "Service-oriented computing for intelligent train maintenance," *Enterprise Information Systems*, vol. 13, no. 1, pp. 63–86, 2019.

[16] S. K. Bansal, A. Bansal and M. B. Blake, "Trust-based dynamic web service composition using social network analysis," in *IEEE Int. Workshop on: Business Applications of Social Network Analysis*, Bangalore, India, pp. 1–8, 2010.

[17] B. N. Mabrouk, S. Beauche, E. Kuznetsova, N. Georgantas and V. Issarny, "QoS-aware service composition in dynamic service oriented environments," in *10th Int. Middleware Conf.*, Urbana, IL, USA, pp. 123–142, 2009.

[18] Y. Yang, B. Yang, S. Wang, F. Liu, Y. Wang *et al.,* "A dynamic ant-colony genetic algorithm for cloud service composition optimization," *International Journal of Advanced Manufacturing Technology*, vol. 102, no. 1, pp. 355–368, 2019.

[19] S. K. Gavvala, C. Jatoth, G. R. Gangadharan and R. Buyya, "QoS-aware cloud service composition using eagle strategy," *Future Generation Computer Systems*, vol. 90, no. 4, pp. 273–290, 2019.

[20] V. Anavangot and A. Kumar, "A novel approximate Lloyd-max quantizer and its analysis," in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, Brighton, UK, pp. 5107–5111, 2019.

[21] J. Qi, B. Xu, Y. Xue, K. Wang and Y. Sun, "Knowledge based differential evolution for cloud computing service composition," *Journal of Ambient Intelligence and Humanized Computing*, vol. 9, no. 3, pp. 565–574, 2018.

[22] M. Bharti, R. Kumar and S. Saxena, "Clustering-based resource discovery on internet-of-things," *International Journal of Communication Systems*, vol. 31, no. 5, pp. 3501, 2018.

[23] R. Hajizadeh and N. N. Jafari, "A method for trust evaluation in the cloud environments using a behavior graph and services grouping," *Kybernetes*, vol. 46, no. 7, pp. 1245–1261, 2017.

[24] H. Gao, J. Yan and Y. Mu, "Trust-oriented QoS-aware composite service selection based on genetic algorithms," *Concurrency and Computation: Practice and Experience*, vol. 26, no. 2, pp. 500–515, 2014.

[25] A. Naseri and N. J. Navimipour, "A new agent-based method for QoS-aware cloud service composition using particle swarm optimization algorithm," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 5, pp. 1851–1864, 2018.

[26] J. O. Gutierrez-Garcia and K. M. Sim, "Self-organizing agents for service composition in cloud computing," in *IEEE Second Int. Conf. on Cloud Computing Technology and Science*, Indianapolis, IN, USA, pp. 59–66, 2010.

[27] D. L. González-Álvarez, M. A. Vega-Rodríguez and R. L. Álvaro, "Searching for common patterns on protein sequences by means of a parallel hybrid honey-bee mating optimization algorithm," *Parallel Computing*, vol. 76, no. 1, pp. 1–17, 2018.

[28] J. Nayak and B. Naik, "A novel honey-bees mating optimization approach with higher order neural network for classification," *Journal of Classification*, vol. 35, no. 3, pp. 511–548, 2018.

[29] I. Alhadid, S. Khwaldeh and A. L. Rawajbeh, "An intelligent web service composition and resource-optimization method using k-means clustering and knapsack algorithms," *Mathematics*, vol. 9, no. 17, pp. 1–16, 2021.

[30] R. K. Chahal and S. Singh, "Fuzzy rule-based expert system for determining trustworthiness of cloud service providers," *International Journal of Fuzzy Systems*, vol. 19, no. 2, pp. 338–354, 2017.

[31] M. Tang, X. Dai, J. Liu and J. Chen, "Towards a trust evaluation middleware for cloud service selection," *Future Generation Computer Systems*, vol. 74, no. 2, pp. 302–312, 2017.

[32] F. Messina, G. Pappalardo, A. Comi, L. Fotia, D. Rosaci *et al.,* "Combining reputation and QoS measures to improve cloud service composition," *International Journal of Grid Utility Computing*, vol. 8, no. 2, pp. 142–151, 2017.

[33] H. K. Al-Anazi, C. Campbell and A. Al Faries, "A combinatorial optimization algorithm for multiple cloud service composition," *Computers and Electrical Engineering*, vol. 42, pp. 107–113, 2015.

[34] O. B. Haddad, A. Afshar and M. A. Mariño, "Honey-bees mating optimization (HBMO) algorithm: A new heuristic approach for water resources optimization," *Water Resource Management*, vol. 20, no. 5, pp. 661–680, 2006.

[35] Y. Huo, Y. Zhuang, J. Gu, S. Ni and Y. Xue, "Discrete gbest-guided artificial bee colony algorithm for cloud service composition," *Applied Intelligence*, vol. 42, no. 4, pp. 661–678, 2015.

[36] S. S. Choong, L. P. Wong and C. P. Lim, "A dynamic fuzzy-based dance mechanism for the bee colony optimization algorithm," *Computational Intelligence*, vol. 34, no. 4, pp. 999–1024, 2018.

[37] Y. Xue, J. Jiang, B. Zhao and T. Ma, "A self-adaptive artificial bee colony algorithm based on global best for global optimization," *Soft Computing*, vol. 22, no. 9, pp. 2935–2952, 2018.

[38] J. Zhou, X. Yao, Y. Lin, F. T. S. Chan and Y. Li, "An adaptive multi-population differential artificial bee colony algorithm for many-objective service composition in cloud manufacturing," *Information Science*, vol. 456, pp. 50–82, 2018.

[39] S. S. Choong, L. P. Wong and C. P. Lim, "An artificial bee colony algorithm with a modified choice function for the traveling salesman problem," *Swarm and Evolutionary Computation*, vol. 44, no. 4, pp. 622–635, 2019.

[40] M. Solgi, O. Bozorg-Haddad and H. A. Loáiciga, "The enhanced honey-bee mating optimization algorithm for water resources optimization," *Water Resource Management*, vol. 31, no. 3, pp. 885–901, 2017.

[41] K. Lakshmanna, N. Subramani, Y. Alotaibi, S. Alghamdi, O. I. Khalaf *et al.,* "Improved metaheuristic-driven energy-aware cluster-based routing scheme for IoT-assisted wireless sensor networks," *Sustainability*, vol. 14, no. 13, pp. 7712–7719, 2022.

[42] A. Sundas, S. Badotra, Y. Alotaibi, S. Alghamdi and O. I. Khalaf, "Modified bat algorithm for optimal VM's in cloud computing," *Computers, Materials and Continua*, vol. 72, no. 2, pp. 2877–2894, 2022.

[43] D. Anuradha, N. Subramani, O. I. Khalaf, Y. Alotaibi, S. Alghamdi *et al.,* "Chaotic search and rescue optimization based multi-hop data transmission protocol for underwater wireless sensor networks," *Sensors*, vol. 22, no. 9, pp. 2–20, 2022.

[44] Y. Alotaibi, "A new meta-heuristics data clustering algorithm based on tabu search," *Symmetry*, vol. 14, no. 3, pp. 623–631, 2022.

[45] S. Rajendran, O. I. Khalaf, Y. Alotaibi and S. Alghamdi, "Mapreduce-based big data classification model using feature subset selection and hyper parameter tuned deep belief network," *Scientific Reports*, vol. 11, no. 1, pp. 1–10, 2021.

[46] J. Jayapradha, M. Prakash, Y. Alotaibi, O. I. Khalaf and S. Alghamdi, "Heap bucketization anonymity—an efficient privacy-preserving data publishing model for multiple sensitive attributes," *IEEE Access*, vol. 10, pp. 28773–28791, 2022.

[47] Y. Alotaibi, "A new database intrusion detection approach based on hybrid meta-heuristics," *Computers, Materials and Continua*, vol. 66, no. 2, pp. 1879–1895, 2021.

[48] A. Alsufyani, Y. Alotaibi, A. O. Almagrabi, S. A. Alghamdi and N. Alsufyani, "Optimized intelligent data management framework for a cyber-physical system for computational applications," *Complex Intelligent Systems*, vol. 398, no. 1, pp. 1–13, 2021.

[49] C. Yan, Y. Zhang, W. Zhong, C. Zhang and B. Xin, "A truncated SVD-based ARIMA model for multiple QoS prediction in mobile edge computing," *Tsinghua Science and Technology*, vol. 27, no. 2, pp. 315–324, 2022.

[50] L. Qi, W. Lin, X. Zhang, W. Dou, X. Xu *et al.,* "A correlation graph based approach for personalized and compatible web APIs recommendation in mobile app development," *IEEE Transactions on Knowledge Data Engineering*, pp. 1, 2022. https://doi.org/10.1109/TKDE.2022.3168611

[51] P. Nitu, J. Coelho and P. Madiraju, "Improvising personalized travel recommendation system with recency effects," *Big Data Mining and Analytics*, vol. 4, no. 3, pp. 139–154, 2021.

[52] W. Zhang, Z. Li and X. Chen, "Quality-aware user recruitment based on federated learning in mobile crowd sensing," *Tsinghua Science and Technology*, vol. 26, no. 6, pp. 869–877, 2021.

[53] A. Vakili and N. J. Navimipour, "Comprehensive and systematic review of the service composition mechanisms in the cloud environments," *Journal of Network and Computer Applications*, vol. 81, no. 10, pp. 24–36, 2021.

[54] M. S. Braik, "Chameleon swarm algorithm: A bio-inspired optimizer for solving engineering design problems," *Expert Systems with Applications*, vol. 174, no. 1, pp. 1–25, 2021.

[55] F. A. Hashim, E. H. Houssein, K. Hussain, M. S. Mabrouk and W. Al-Atabany, "Honey badger algorithm: New metaheuristic algorithm for solving optimization problems," *Mathematics and Computers in Simulation*, vol. 192, no. 2, pp. 84–110, 2022.

[56] E. Al-Masri and Q. H. Mahmoud, "QoS-based discovery and ranking of web services," in *Int. Conf. on Computer Communications and Networks*, Honolulu, HI, USA, pp. 13–16, 2007.

[57] L. Purohit and S. Kumar, "A classification based web service selection approach," *IEEE Transactions on Services Computing*, vol. 14, no. 2, pp. 315–328, 2021.

[58] R. Khanam, R. R. Kumar and C. Kumar, "QoS based cloud service composition with optimal set of services using PSO," in *IEEE 4th Int. Conf. on Recent Advances in Information Technology*, Dhanbad, India, pp. 1–6, 2018.

[59] T. Hayashida, I. Nishizaki, S. Sekizaki and Y. Takamori, "Improvement of particle swarm optimization focusing on diversity of the particle swarm," in *IEEE Int. Conf. on Systems Man, and Cybernetics*, Toronto, ON, Canada, pp. 191–197, 2020.

[60] S. Subbulakshmi, K. Ramar, V. C. K. Krishna and S. Sanjeev, "Optimized QoS prediction of web service using genetic algorithm and multiple QoS aspects," in *IEEE Int. Conf. on Advances in Computing, Communications and Informatics*, Bangalore, India, pp. 922–927, 2018.