Article

QoS-Aware Cloud Service Recommendation Using Metaheuristic Approach

Soumya Snigdha Mohapatra [1,*], Rakesh Ranjan Kumar [1], Mamdouh Alenezi [2], Abu Taha Zamani [3] and Nikhat Parveen [4]

[1] Department of Computer Science and Engineering, C V Raman Global University, Mahura, Bhubaneswar 752054, India
[2] Software Engineering and Disruptive Innovation (SEDI), College of Computer and Information Sciences, Prince Sultan University, Riyadh 11586, Saudi Arabia
[3] Department of Computer Science, Northern Border University, Arar 73211, Saudi Arabia
[4] Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Guntur 522502, India
* Correspondence: soumyasnigdha.praharaj@gmail.com

Abstract: As a result of the proliferation of cloud services in recent years, several service providers now offer services that are functionally identical but have different levels of service, known as Quality of Service (QoS) characteristics. Therefore, offering a cloud assistance arrangement with optimum QoS estimates that fulfilling a customer's expectations becomes a complicated and demanding task. Several different metaheuristics are presented as potential solutions to this problem. However, most of them are unable to strike a healthy balance between exploring new territory and capitalizing on existing resources. A novel approach is suggested to balance exploration and exploitation via the use of Genetic Algorithms (GA) and the Eagle Strategy algorithm. Cloud computing provides clients with capabilities that are enabled by information technology by using services that are available on demand. To circumvent difficulties such as a delayed convergence rate or an early convergence, this technique allows for the establishment of a healthy equilibrium between exploratory and exploitative activities. The result of the experiment shows that the Eagle Strategy algorithm (ESA) and GA are better than other conventional algorithms at making a globally QoS-based Cloud Service Selection System faster.

Keywords: ESA; GA; QoS; cloud computing; selection system

check for updates

Citation: Mohapatra, S.S.; Kumar, R.R.; Alenezi, M.; Zamani, A.T.; Parveen, N. QoS-Aware Cloud Service Recommendation Using Metaheuristic Approach. *Electronics* **2022**, *11*, 3469. https://doi.org/10.3390/electronics11213469

Academic Editor: Antonio Brogi

Received: 27 September 2022
Accepted: 24 October 2022
Published: 26 October 2022

1. Introduction

Customer-facing IT-enabled capabilities are delivered as elastic and scalable services via the use of cloud computing, which allows users to access resources, platforms, and software on-demand without the need to buy and maintain [1,2]. At various price points and performance levels (QoS requirements), clients can choose from a wide range of cloud services that all have the same functionality. Since many cloud service providers cannot guarantee customer satisfaction, consumers are more likely to reject cloud services that do not meet their needs [3]. Composing several current services that complement one another to produce a composite service that fits the demands of consumers is the present need. Given that numerous concrete services might provide the same functionality, a choice must be made on which actual maintenance should be utilized for each conceptual service. QoS factors must be taken into account while selecting a facility since many providers share the same functional characteristics. As a result of technological innovation, the number of possible options for a certain subject is huge. The user of a recommendation system receives a list of options or a set of straightforward and useful outcomes. When the scope of the search is broad, it is an advantage of the time and effort saved by a recommendation system which is enormous. For most of the part, the number of Internet-enabled options is vast. To cope with this large body of knowledge, the information must be filtered and prioritized before any significant conclusion can be drawn. Recommendation systems employ dynamically produced data to build a list of desirable services or products for consumers [4].

While making recommendations; the recommendation system considers a variety of factors [5]. A significant computer technology, cloud computing has several benefits for its consumers. It is possible to access the most popular Platform as a Service and Software as a Service, which are two examples of cloud computing services and so on, using the cloud computing infrastructure. The underlying principle of the technology is to provide users with a wide range of services at a low cost [6,7]. Consequently, cloud computing reduces the need to spend a large amount of money on infrastructure or software maintenance and administration. Cloud computing ecosystem actors include Cloud Service Consumers (CSC), Cloud Service Providers (CSP), and the service itself. Those who make use of the CSP's cloud services are referred to as cloud service consumers or cloud service users. Many cloud customers rely on the server to host a wide range of services. The recommendation system model incorporates the cloud architecture and any associated nodes, as shown in Figure 1 [8].

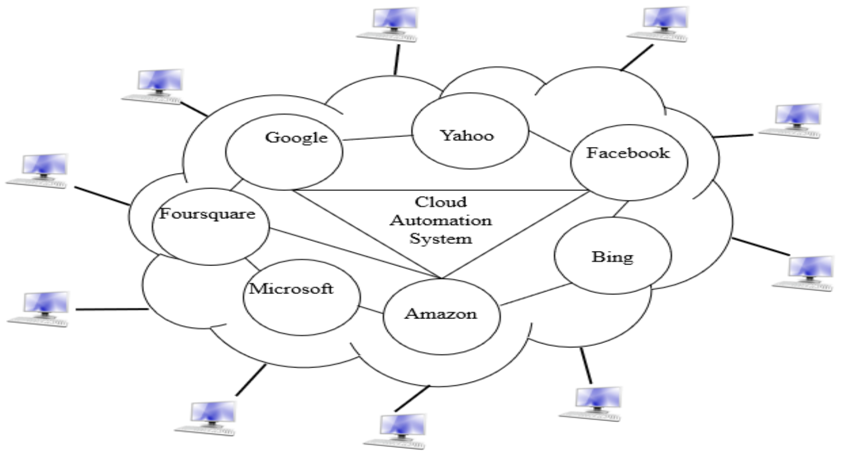


Figure 1. Cloud Recommendation System.

Cloud technology continues to advance due to the proliferation of Internet of Things (IoT) gadgets [9] and big data [10]. As a result of this networking-based design, cloud manufacturing has been adopted. The purpose of cloud manufacturing is to make better use of already-existing assets and to provide better service to cloud consumers [11–13] and the main objectives are to endorse reliable cloud services to users and to determine the best combination of cloud services from the various options available to obtain effective results with minimal waiting time. Obtaining the almost global solution requires the use of metaheuristic methods since cloud service composition is an NP hard problem. If a population has little variation, then there is a chance of premature convergence when local exploitation is used, and it is minimized to find the global solution. On the other hand, if the population is large, then there may be a chance of global exploration, but that reduces the convergence rate. Furthermore, if the population is large, then there may be the chance of global exploration but that reduces the convergence rate.

In order to address the above discussed research problem, the following research questions have been developed:

RQ1: Why is it necessary to execute the composition of cloud services that are already available and how?

RQ2: What is the optimal technique for achieving a balance between exploration and exploitation in QoS aware service composition in cloud computing?

Research Gap and Contributions

Previously, most of the services were offline. Now it has become mandatory to go for online service as the user requirement increases. Offline services cannot handle customers' vague perceptions about QoS requirements. As the complexity increases from both the users' and service providers' end, so the complexity of service requirements also increases. The main gap between the previous approaches is that the previous approaches did not make use of the work on pattern set optimization as a recommendation and the unavailability of resources in the middle-level industries. For real-time applications, we should go beyond the single service as its scope is limited. The main contributions of our paper are outlined in the following order:

1. To address the limited scope caused by the selection of a single service, we have identified the optimal combination of cloud services among the many possible possibilities, and we have proposed a methodology that can maintain a healthy balance during the foraging process.
2. To achieve desirable outcomes with a minimum amount of time spent on waiting, and select cloud services according to the quality of service (QoS) requirements of cloud service customers.

Hence, the objective can be summarized as: to recommend trustworthy cloud services to users, although our main focus is to choose the combination of cloud services that is ideal from different alternatives available and to obtain effective results with minimal wait time.

Section 2 talks about the related work on composite cloud services reflecting nature-stimulated metaheuristic methods. Section 3 deliberates the composition of the service path model with the utilization of QoS attributes. Section 4 explains the proposed methodology for optimal cloud service selection. Section 5 outlines the proposed framework and the feasibility of the proposed methodology. Section 6 represents the results, and their validations are also discussed. Finally, Section 7 deliberates the concluding remarks with the future scope.

2. Related Work

This section discusses and studies the relevant work conducted by various authors about cloud service recommendation systems using different techniques. The authors of [14] suggested that due to its effectiveness and multi-choice structure with time-varying components, Harris Hawks' Optimization (HHO) is one of the top optimization techniques. Due to its simplicity and exceptional efficiency, HHO has been used in a variety of applications. Nevertheless, the original HHO can be enhanced and modified in terms of convergence trends, and it is susceptible to local optimization under certain conditions. In this research, the author suggested a new algorithm, based on the shortcomings (under certain optimization conditions) of the different nature inspired algorithms and greatly improves the performance of the original HHO.

Alhijawi et al. [15] presented a novel Genetic-Based Recommender System based on historical rating data and semantic information. It is important to note that the study makes a distinction between evaluating and developing a list of probable suggestions. The Best List of Items Using Genetic Algorithms (BLIGA) uses a genetic algorithm to choose the best selections for the current user. Therefore, everyone represents a list of candidates that have been recommended. Three fitness functions are used in the genetic-based recommendation system to assess people hierarchically. Semantic similarity between items is assessed by the first function, which considers semantic information about the objects in question. Using the second function, it can compare the experience of a customer's satisfaction. The third function picks the best possible list of suggestions for the user based on the predicted ratings.

Chenyang Li et al. [16] presented QoS-aware Web Service Composition (QWSC) as a trendy subject in both industry and academia due to the recent surge in the number of web facilities. QWSC has employed a meta-heuristic approach to solving classical optimization problems. Due to its inherent flaws, such a method generally fails in large-scale situations.

For example, certain meta-heuristic algorithms work well in nonstop search space, whereas QWSC's search space is discrete. The author provides an approximation optimization technique based on the HHO algorithm to handle the problem of QoS-aware web service structure. Meta-heuristic algorithms created for constant difficulties can be used to solve QWSC due to a preprocessing method called Fuzzy Optimal Continuity Construction. Consequently, the local search approach of meta-heuristic algorithms is equally successful in discrete and continuous space (discrete problem). Second, it reconstructed the HHO based on the QWSC's features and combined it with the Logistic Chaotic Single-Dimensional Perturbation (LSCDP) strategy, a unique exploration method (designed to increase the algorithm's ability to escape local optimization), to create the revolutionary algorithm.

Fang Li et al. [17] discussed meeting user needs for network worth of benefit; this research presents a novel QoS assessment prototype established on the Intelligent Water Droplets (IWD) algorithm. The network QoS assessment indices generated by the model are utilized to construct a multi-objective optimization function for addressing the issue. It is then utilized to investigate the model's most critical components using mathematical simulation. The simulation results suggest that the technique is more flexible than previous approaches.

Jiang et al. [18] suggested the use of cloud service tags and informative language to extract and identify latent connections between cloud services. According to the suggested method, cloud services are divided into cluster-based descriptive text data and facility labels are used to categorize practically every other cluster of cloud services. Real-world data from programmable websites reveal that the Hierarchical Dirichlet Process (HDP-based) model and PageRank tailored procedure can leverage amorphous narrative data to deliver reliable cloud service recommendations, as shown by the study conducted.

According to Hashem Alayed et al. [19], in order to compose a web service, it is necessary to understand the demands of the end-user. Each task in this pipeline provides an abstract description of specific user demands. For workflow management, web services can be gathered from several sources. The candidate list refers to a collection of services provided by various service providers for a certain activity. For the Web Service Selection (WSS) task, the user must choose the best-fitting service based on its QoS. By using the swap concept, it can enhance the Ant Colony Optimization (ACO) method for WSS applications that are cognizant of QoS constraints. Avoiding local optimums and shortening search time are two main goals for implementing a new ACO design. Researchers think that the incorporation of several powerful solutions will assist the ACO algorithm in producing a superior solution and avoiding stagnation. The suggested method was compared to the ACO and Flying ACO (FACO) algorithms via a series of tests.

Zhang et al. [20] developed a Personalized Cloud Service Recommendation (P-CSREC). Heterogeneous information networks are critical to its success. By taking into consideration customer preferences, a similarity measure is established, and a model is developed. Frequent Pattern Growth (FP-Growth) processes are used to produce customized user suggestions to cope with the convergence problem. Clustering and the FP-Growth algorithm can be used together to increase accuracy, according to this research.

Gavvala Siva Kumar et al. [21] elaborate that although many cloud service providers are now providing functionally comparable services with different QoS characteristics, this has not always been the case in the past. Cloud facility arrangement with appropriate QoS values that meet the needs of a consumer becomes more difficult in a cloud atmosphere since it is more complicated and harder. This study proposed the Eagle Strategy to create a QoS-based cloud service composition. This approach enhances the ability to avoid problems such as premature convergence or delayed convergence by maintaining the balance between exploration and exploitation. As shown by experiment results, the hybrid of Eagle with Whale Optimization Algorithm (ESWOA) delivers global optimum solutions more effortlessly than other current algorithms. In [22], using Fuzzy Formal Concept Analysis, the author proposes a Collaborative Filtering-based recommendation system for cloud services (Fuzzy FCA). The lattice theory provides a firm mathematical grounding for fuzzy FCA.

R R Kumar et al. [23] proposes a method for evaluating the relative merits of several cloud services with respect to some quality-of-service criteria, taking into account the uncertainty inherent in such an evaluation. In this case, the Fuzzy-AHP technique is employed to specify the framework of the entire cloud service selection procedure and to determine the relative importance of various quality-of-service criteria. However, there are still restrictions of which to be aware. When making pairwise comparisons, it is important to keep in mind that human judgments might be biased and subjective. Moreover, it does not adapt well enough to the ever-changing needs of cloud customers.

F. Dahan et al. [24] suggested a hybridization of ant colony optimization (ACO) and genetic algorithms (GA) to efficiently compose the services over the cloud. The ACO automatically tunes its parameters with the help of the GA, and as a result, the ACO adjusts its performance. The main contribution of this work is to help the ACO algorithm avoid stagnation and improve its performance, which is affected by the ACO's parameters.

This research work represents a clustering-based strategy known as dynamic clustering (DCLUS) and was proposed by the authors [25]. The usage of dynamic clustering techniques in DCLUS sets it apart from static-based clustering techniques. The static various widths clustering approach is utilized in the current CLUS for the clustering of users and services. Due to the problems with static clustering, they came up with the idea of dynamic clustering as a way to improve how well data mining is used to find relationships and patterns in services and how well predictions are made.

Q. She, X. Wei, and G. Nie et al. [26] categorized and contrasted current computational intelligence techniques, examined the state of the field, and potential areas related to future research were determined. It can serve as a foundation for anybody interested in this field, including scholars and practitioners. Significantly, in the area of expert and intelligent systems, this research may support the development of intelligent systems in businesses as well as effectively support businesses in risk reduction.

3. Background Study

As a consequence of the proliferation of cloud services, several service providers now provide identical services but differentiate between them based on the functionalities, including in the Quality-of-Service parameters, which are characteristics of Quality of Service abbreviated as QoS. Offering a cloud service composition as a result with acceptable QoS values that are consistent with user expectations in a cloud-based environment becomes difficult and demanding. There are different metaheuristics that might be used to conquer this obstacle. However, most of them are unable to strike a healthy balance between exploring new territory and capitalizing on existing resources. This research work offers a combination of GA with ESA, which aims to find a healthy medium between using existing resources and discovering new ones [21].

3.1. Metaheuristic

Using metaheuristics, it is feasible to develop solutions that are close to being optimum. When users use these search tactics to handle these challenges, rather than the usual ways, they can get amazing results in a shorter amount of time (polynomial time), as opposed to an exponentially longer amount of time (exponential time). Many population-based metaheuristic algorithms obtain motivation from natural events as a common source of inspiration. These algorithms' primary emphasis is on resolving domain-specific problems, such as scheduling, planning, financial forecasting, and engineering design, amongst others.

3.2. Using Metaheuristic for Maintenance Arrangement

They are generally inspired by natural phenomena and try to search for solutions to problems by focusing on the most fruitful areas of the search space for a specific domain. Several different algorithms can be utilized for carrying out an analysis of this kind. When it comes to tackling certain problems, some algorithms are more effective than others. Stochastic approaches, such as randomization, are used to solve problems that are classified as NP-Hard. There are likely to be many local minima and local maxima in the search space; therefore, metaheuristic algorithms need to look for the solution in as many different places as possible.

3.3. Shaping of QoS-Aware Cloud MaintenanceArrangement

As a result of the complexity of service-oriented computing, many service providers have a difficult time satisfying the requirements of their clients. In some circumstances, jobs are broken up into more manageable subtasks. After the candidate services that are tasked with performing subtasks have been developed, the original request can then be completed.

Let T be the original customer's demand. Now, T = {T1, T2, T3 ... Tn} where n is the quantity of secondary assignments. The abstract service handles each secondary assignment Si, that is the assortment of applicant facilities $\{ws_{i,1}, ws_{i,2}, ws_{i,3}, \ldots, ws_{i,n_i}\}$ where n_i implies the number of aspirant facilities for that specific s_i. These contender facilities acquire various QoS estimates, but with similar functionalities.

Due to overlapping functions, QoS qualities perform a crucial position in determining the aspirant facility for each conceptual facility. QoS $(w_{si}, j) = q_1, q_2, q_3, \ldots q_n$ represents these QoS characteristics for each candidate service, where q_k represents the kth amount of QoS features, i represents the i^{th} conceptual facility, and j represents the j^{th} applicant facility for s_i. The collection of QoS elements has been indicated as that of a combined facility by the letter Q (cs), according to Table 1, which is computed using aggregation techniques to answer the research question RQ1.

Table 1. QoS accumulation tasks with different arrangements.

QoS Parameters	Series	Corresponding	Circle (Call ws_i^j Service k Times)	Conditional (Pr_i)
Availability	$\prod_{i=1}^{n}(q_{avail,i})$	$\prod_{i=1}^{m}(q_{avail,i})$	$k * \prod_{i=1}^{n}(q_{avail,i})$	$\prod_{i=1}^{n}(q_{avail,i}) * Pr_i$
Response Time	$\sum_{i=1}^{n}(q_{rt,i})$	$min(q_{rt,i})$	$k * \sum_{i=1}^{n}(q_{rt,i})$	$\sum_{i=1}^{n}(q_{rt,i}) * Pr_i$
Reliability	$\prod_{i=1}^{n}(q_{rel,i})$	$max(q_{rel,i})$	$k * \prod_{i=1}^{n}(q_{rel,i})$	$\prod_{i=1}^{n}(q_{rel,i}) * Pr_i$
Throughput	$min(q_{tp,i})$	$min(q_{tp,i})$	$k * \prod_{i=1}^{n}(q_{tp,i})$	$\prod_{i=1}^{n}(q_{tp,i}) * Pr_i$

Both favorable and unfavorable QoS [21] serve as a dividing line between the two groups. A high QoS value can be described using the positive property, whereas a low QoS value can be described using the negative attribute. Users can see how these functions are applied to a variety of composition patterns in Table 1. N is the number of service compositions and m is how many services are operating in parallel at one time. In conditional composition, it has $\sum_{i=1}^{n} Pr_i = 1$ where n is the number of options and Pr_i is the possibility of picking qualified assistance. These QoS values cover a wide range, which necessitates the normalization of all attribute values to the range [0, 1]. These two sorts of traits are handled and normalized independently. Using this method, it can normalize positive attributes by using Equation (1).

$$\mathrm{UniQ_k} = \begin{cases} \frac{Q_k - \min Q_k}{\max Q_k - \min Q_k} & \max Q_k - \min Q_k \neq 0 \\ 1 & \max Q_k - \min Q_k = 0 \end{cases} \tag{1}$$

where, $\mathrm{UniQ_k}$ is a normalized QoS value, $\mathrm{Q_k}$ is pathways, $\mathrm{minQ_k}$ is the value of the minimum pathway, and $\mathrm{maxQ_k}$ is the value of the maximum pathway. The following is the formula for normalizing negative characteristics using Equation (2).

$$\mathrm{UniQ_k} = \begin{cases} \frac{\mathrm{maxQ_k} - \mathrm{Q_k}}{\mathrm{maxQ_k} - \mathrm{minQ_k}}\ \mathrm{maxQ_k} - \mathrm{minQ_k} \neq 0 \\ 1\ \mathrm{maxQ_k} - \mathrm{minQ_k} = 0 \end{cases} \tag{2}$$

where, $\mathrm{UniQ_k}$ is a normalized QoS value, $\mathrm{Q_k}$ is pathways, $\mathrm{minQ_k}$ is the value of the minimum pathway, and $\mathrm{maxQ_k}$ is the value of the maximum pathway. Maximum and minimum QoS values for the kth dimension attribute for all composition pathways are denoted by the terms $\mathrm{maxQ_k}$ and $\mathrm{minQ_k}$. If values are equal, i.e., $\mathrm{maxQ_k} - \mathrm{minQ_k} = 0$, the normalized value would be 1, otherwise objective function is governed by the constraints such as , $\mathrm{maxQ_k} - \mathrm{minQ_k} \neq 0$.

4. Proposed Methodology

The proposed work discusses the techniques used in this research; the techniques are the genetic algorithm; the basic principle of the Genetic Algorithm; the Eagle Strategy Algorithm; and QoS Parameters. As a whole, the objectives include the study and evaluation of the previous studies on QoS-based cloud recommendation systems using ESA and GA to generate the maximum set of facilities that can be suggested to the client and to determine the best combination of cloud services from the various options available answering to research question RQ2.

4.1. Genetic Algorithm

Natural selection, the driving force behind biological evolution, is the basis for the genetic algorithm, a method for addressing optimization problems with both restricted and unconstrained parameters. Individual solutions in a population are repeatedly altered by the genetic algorithm. To produce the next generation of kids, the genetic algorithm selects parents from the present population at each step. Over several groups, the inhabitant evolves towards the best possible answer. The genetic algorithm can be used to resolve a wide range of optimization difficulties, such as those in which the objective function is not differentiable, discontinuous, severely nonlinear, or stochastic, which standard optimization approaches cannot handle. The evolutionary technique can be utilized for mixed-integer programming problems where certain components must have integer values. The genetic algorithm uses three different kinds of rules to make the next generation out of the current population. Those are:

- These individuals, known as parents, have a significant role in determining the next generation's population. The participants' scores can have a role in the selection process; however, this is not always the case;
- For the next generation's children to be produced, crossover rules mix the genes of two parents;
- To generate kids, individual parents are vulnerable to random genetic mutations.

The Fundamental Idea behind the Genetic Algorithm

GA has been used to solve a broad variety of challenges in recent decades, including optimum scheduling issues, resource allocation, and transportation, among others. GAs are a form of evolutionary computation that is based on adaptive heuristic search methods and is influenced by natural evolution [27]. GAs use biological evolution and natural selection concepts to generate several solutions to an issue. As a result, GAs can solve combinatorial optimization problems. Calculus-based techniques have a hard time coping with these issues. Although randomized at first, GAs allow users to discover the search space for better solutions using past data.

The GA's general process starts with a population of individually produced solutions that are generated at random. Each solution in the population is then assessed using a fitness function that is specified by the customer, since, in the fitness test, two individuals (parents) are selected. These individuals replicate to produce one or more offspring, after which they use a recombination operator such as crossover or mutation to produce new offspring. The procedure is repetitive until the best solution is found. The Algorithm 1 of GA's simple pseudo-code is as follows:

Algorithm 1: GA's simple pseudo-code

```
1.   Create an early random population of people.
2.   Generations = 0
3.   Assess the fitness of the individuals.
4.   while the termination condition is not met do
5.   Choose two individuals for reproduction.
6.   Generate new individuals using mutation operators and crossover.
7.   Calculate the health of the new person.
8.   Change the most appalling human being of the inhabitants into the best human being.
9.   generations = generations + 1
10.  end while
11.  Returning the population i.e., optimal.
```

A genetic algorithm is used for refining two rough set analysis processes: exploring optimum redacts and cut points for data discovering and discretization trading rules. Rough set theory is great for information description in general, while genetic algorithms are reliable for resolving combinatorial optimization problems. As a result, the realized trading rules provided by solving optimization problems are easy but explicit, which is a benefit of a smart hybrid trading system paradigm using a Genetic algorithm and rough set analysis. The Genetic algorithm is found to have the most excellent performance. It also discovers that successful regional segmentation based on model results is critical for improving a forecasting model's predictive accuracy. It develops more effective strategies by using the Genetic Algorithm Model, which results in ongoing economic benefits to the relevant investor. In this way, the model built in this study contributes to the long-term sustainability of economic development [28].

4.2. Eagle Strategy Algorithm (ESA)

The Eagle approach is a metaheuristic optimization technique created by Susah Deb and Xin-She Yang in 2010 [21]. To achieve its varied goals, it makes use of a mixture of primitive international exploration and extensive regional exploration. First, a global search is undertaken to utilize a Levy fly random walk; if an encouraging result is observed, a more efficient regional hunt is conducted using differential evolution, hill-climbing, or other creative procedures. Afterward, a new global search is conducted, followed by a new regional pursuit at a new location. Combining global and local search results in a way that is both fast and efficient is an advantage of this approach. The Algorithm 2 below contains the pseudo-code for the Eagle Strategy.

Algorithm 2: Pseudo-code for the Eagle Optimization Strategy

```
1.  Quantitative function f(x)
2.  Start of the model area
3.  While (it r < maximum number of iterations)
4.  Do International Survey by randomization
5.  Health Assessment and obtaining a capable result
6.  if (Pe < rand), do
7.  Regional development by the effective resident optimizer
8.  if (a superior answer is obtained)
9.  The recent greatest answer is modernized
10. end
11. end
12. itr = itr + 1.
13. end
```

4.3. QoS Parameters

The quality of the service parameters is a significant aspect in determining the quality of each concrete service. It comprises various dimensions based on consumer's necessity. All QoS parameters are broken down into six categories: reliability, availability, response time, scalability, throughput, and latency [29].

- **Reliability:** The probability that a task will be carried out in an appropriate manner is referred to as reliability. The ratio between the times it takes to request a web service and the total amount of time it takes to complete the request in a certain amount of time;
- **Availability:** The availability of a web service is the proportion of time during which the service is actually accessible to users. The time it takes for a user to access a service is compared to the time it takes for the user to call the facility;
- **Response time:** The period of time, measured in milliseconds, that it takes a processor to execute a request and provide a response is referred to as the response time. The time it takes to reply to a user's request as measured from the moment the user sent the request;
- **Scalability:** This refers to the capacity to continuously service the request when the number of requests increases or decreases;
- **Throughput:** The greatest number of requests that may be handled in a certain time unit is referred to as the throughput. It is expressed as the number of requests processed per minute;
- **Latency:** The latency value is computed by subtracting the response time from the request time of the web service invocation. The latency of a data service is measured in milliseconds and refers to the time it takes to either save or retrieve the data.

In our research work, we have considered availability, response time, throughput, and reliability as QoS parameters.

4.4. Cloud Service Cataloging

A cloud service [30] list is the basic stage of this proposed work, which manages a collection of accessible cloud services. The most important aspect of this stage is for the trust algorithm to be aware of the CSP's (cloud service providers) services. Firstly, the new services must be catalogued before researchers utilize them. In the same way, services that are no longer operational should be deleted from the catalog. As a result, the catalogue represents the CSP's present operational services. A TA (trust algorithm) acts as a mediator to recommend the mandatory service that the end-user has requested. The services in the catalogue are signified in the following structure and shown in Equation (3).

$$\langle \mathrm{S_{ID}, S_{typ}, Srt_{time}, Cmp_{time}, S_{srt}, S_{cmp}, S_{cnt}} \rangle \tag{3}$$

Furthermore, the catalogue maintains a unique identifier (S_{ID}) for identifying a definite service, the service nature (S_{typ}) contains the several aspect of the service rendered by the service benefactor, and the actual start time of the unallocated service is specified by start time $(S_{rt_{time}})$ that provides service to the user. When the service is going to begin, the scheduling of the start time (S_{srt}) displays as 0, indicating that it is not presented at the time and its status is set to 1 by default. When the service is finished, the status of service completion time (S_{cmp}) implies 1 to denote to the status of service that is accessible for distribution. The number of times a service is given to a user is referred to as the service count (S_{cnt}). Only current services among the start and finish times were cataloged. After the service provider's period has expired, the service becomes consistent.

4.5. Trust Degree Computation and Management

This module estimates the present cloud service's trust degree. The Trust Degree computation method computes the trust degree [31] for each period, and the calculated trust degrees of each service are maintained by the trust algorithm (TA). This is to verify that all cloud consumers always have accessibility to the most up-to-date and reliable facilities. The trust degree is calculated by six distinct factors, including hardware requirements (both memory and computation, i.e., $\gamma,\ \mu$), security (β), service appropriateness (α), availability (φ), and service response time (ρ). These trust measures oversee providing the end-user with a trustworthy, dependable, and time-saving service. The following Equation (4) is used to normalize each measure.

$$\mathrm{Norm}_x = \frac{(x - ac_l) \times (n_h - n_l)}{ac_h - ac_l} \tag{4}$$

The value of the measure to be normalized is x in the above equation, ac_l and ac_s are the attributes lowest and greatest values, respectively. The preferred range of numbers for doing normalization is n_h and n_l. The purpose of standardizing the numbers is that all faith measures fall within an equal variety, making data processing easier. The values of n_l and n_h in this example are 0 and 1. A reliable service allocation technique is used to categorize the accessible services into multiple groups. Most of this stage is spent determining trust degrees based on trust measures of accessible services. The provider can demonstrate and describe their services to the trust algorithm for the primary phase of determining trust metrics if the service is new to the trust algorithm. The TA calculates and endorses direct trust in this case. As a result, all users are believed to be authentic to use the accessible service, and trust measure calculation rigorously adheres to the trust degree calculation algorithm based on user reviews. The following section discusses trust measures.

4.6. Preference Pattern

The user's behavioral preferences vary accordingly. Some consumers have constantly preferred similar genres of films. Cloud providers who face the risks must find ways to increase their revenues as currently the industry-wide spread pay-as-you-go model benefits customers but leads to unmanageable utilization patterns. It is essential for Cloud providers to identify customers' preferences and derive the optimal set of services from this knowledge [32].

Past likes and recent likes are the first patterns of users. These are loyal customers who have a consistent interest. These individuals generally only view one type of film, and the length of that film has been constant for a long time. Recent Likes is the second user pattern, while Past Likes is the third user pattern. Interests change over time for these users. Past Dislikes: Recent Dislikes is the last category of usage pattern. The information that these people look at has a lot of different patterns, with things coming up at random times.

4.7. User Similarity Computation

There are a wide variety of similarity functions available to determine how similar two people are. It looks at Pearson correlation (PC), Jaccord similarity, and cosine similarity, which are all ways to measure similarity.

Pearson Correlation calculates the correlation between the ratings of two users to assess how similar they are. In addition to the standard Pearson correlation, there is a variation known as the Constrained Pearson correlation [33] and Equation (5) shows the mathematical formula of Pearson correlation.

$$\text{Pearson correlation} = \frac{\sum_{i=1}^{n}(a_i - \bar{a})\left(b_i - \bar{b}\right)}{\sqrt{\sum_{i=1}^{n}(a_i - \bar{a})^2}\sqrt{\sum_{i=1}^{n}\left(b_i - \bar{b}\right)^2}} \tag{5}$$

Here let a_i and b_i be the rating scores from two users, $\bar{a}$ and $\bar{b}$ denote the average ratings by the two users.

Vector-based cosine similarity is a technique for determining the degree of similarity between two individuals. It is possible to compare vectors by computing their cosine distance from each other. It is calculated with the help of the cosine similarity formula as shown in Equation (6).

$$\text{Cosine similarity} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{6}$$

where A represents the weight of each feature in vector A, and B represents the weight of each characteristic in vector B.

Jaccord similarities are shown in Equation (7), and it determines how similar two people are based on their ratings, similarity is employed. User u uses R_u, whereas user v uses R_v, and $R_{u \cap v}$ indicates the total number of cloud services utilized by both user u and user v, both of which are represented by the letters u and v, respectively.

$$J_{u,v} = \frac{R_{u \cap v}}{R_u + R_v - R_{u \cap v}} \tag{7}$$

5. Proposed Framework

Cloud computing is an unavoidable method of processing these days, and it is deemed a benefit to middle-level activities due to technological advancement. Finding a reliable service is the most difficult task. This might take up a lot of time or end in appropriate services, and much of the time, the services for which the user requested are not available. The unavailability might be for many reasons, such as resource dependencies, time constraints, loss of information and maintenance due to authorization issues. Broad categorization of services for composite service includes various theories but those are certain assumptions specific [34–37]. As a result, the cloud system now requires a cloud services proposal system. The intended work can recommend the time-preserving stability and accuracy of facilities among a pool of accessible cloud services. In the current work, a pattern-based recommendation system is presented that considers a Genetic algorithm to generate the maximum set of facilities that can be suggested to the client and overcome the problem associated with composite cloud service selection problem over different state-of-art algorithm [38]. The work considers ESA's opting for the optimized pattern set as a recommendation.

This segment defines the proposed workflow and presents the general idea of the work that operates autonomously and cooperatively as well. Figure 2 illustrates the workings of the pattern-based approval method utilizing ESA and GA. The validation for the recommended service and to generate the pattern preference list, the similarity of using a trust degree of services is counted and the QoS of the recommendation service is verified to make a final recommendation pattern for the user.

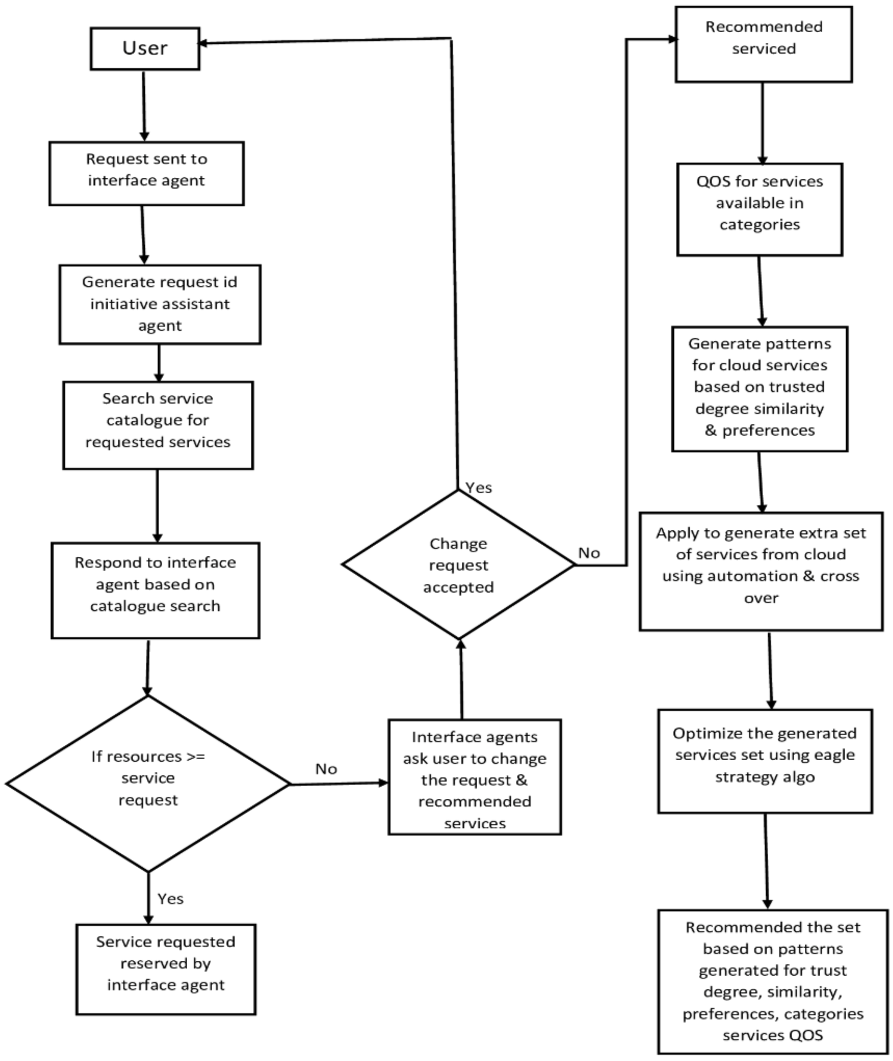

User
Request sent to interface agent
Generate request id initiative assistant agent
Search service catalogue for requested services
Respond to interface agent based on catalogue search
If resources >= service request
Yes
No
Service requested reserved by interface agent
Interface agents ask user to change the request & recommended services
Change request accepted
Yes
No
Recommended serviced
QOS for services available in categories
Generate patterns for cloud services based on trusted degree similarity & preferences
Apply to generate extra set of services from cloud using automation & cross over
Optimize the generated services set using eagle strategy algo
Recommended the set based on patterns generated for trust degree, similarity, preferences, categories services QOS

Figure 2. Proposed Framework.

Step 1: The end-user submits the source invitation to the resources that are accessible and suitable. In each input request, the interface agent is engaged and immediately acquires the appropriate information needed to process the request further.

Step 2: If the application is discovered, the system generates an application ID and then contacts an associate manager who is tasked with the responsibility of providing the application with the corresponding ID. The assistance agent searches the resource repository for resources that are appropriate for the current request.

Step 3: Then, it searches for the service catalogue for the requested service and responds accordingly to the interface agent based on the search. This is the most important step, in which the service provider maintains a catalogue of all accessible cloud services. All cloud services are given a unique identity in this catalogue so that users can distinguish between them easily.

Step 4: If available resources are greater than the service request, then the interface agent will reserve the requested service. Otherwise, the interface agent can advise users to change their request and recommend service so that resources can be allocated for the two scenarios mentioned above. As a result, the resources would be organized in the interface agent's resource repository. The procedure is continued until the client reaches the target or denies it.

Step 5: If the user does not alter the suggested service, it recommends the most reliable service accessible to the cloud user. It aids in the improvement of QoS while also saving time. The QoS metric is an important factor for determining the quality of certain services. Then patterns are generated for cloud services based on trust degree similarity and preference.

Step 6: Then, for each abstract task, ESA is applied to the real collection of concrete services. Before utilizing ESA, the actual concrete service set that is required for individual abstract tasks is decreased to increase the probability of picking a higher QoS in the composition set. ESA then creates its composition at random from a smaller set of optimum tangible services, which improves the chances of obtaining a more optimal cloud service composition.

Step 7: In the last phase, it recommends the set of services based on generated patterns for trust degree, similarity preference, and QoS services. The trust algorithm can track the degree of trust in active services over time. The consumers' similarity is calculated using a similarity function. Predicting the best cloud service for a client involves considering both the degree of trust and the degree of similarity between the users.

6. Experimental Details and Result Analysis

The experimental setup and the performance of the results are the primary focus of this section. Python programming is used for research project implementation, and it includes the hardware setups, software setups, iteration process about the execution time, the finest service providers and their best offerings, and optimal cost of research iteration. An Intel® Core™ i7 Processor is used in the work due to its superior CPU performance and acceleration of discrete-level graphics and Artificial Intelligence (AI). The Windows 10 Operating System (OS) has been used. The experimental implementation has been conducted by utilizing the regular QWS dataset.

Result 1: In terms of achieving the optimal mix of facilities, the ESA and GA procedures are superior to other approaches. Figure 3 indicates the number of iterations of the execution time. The graph shows that as the number of iterations values was increased, the execution time in seconds likewise increased at the same rate. The amount of time, in seconds, required for execution remains the same throughout a specific iteration. The execution time is 0.6 when the iteration value is around 8, and increasing the iteration value increases the execution time. Execution time is 0.68 s when the iteration value is 10, while it is 0.7 s when the iteration value is 20. Moreover, the execution time goes up by more than 0.8 s when the iteration value is 30. However, after a certain amount of time, the execution time stays the same for each iteration.

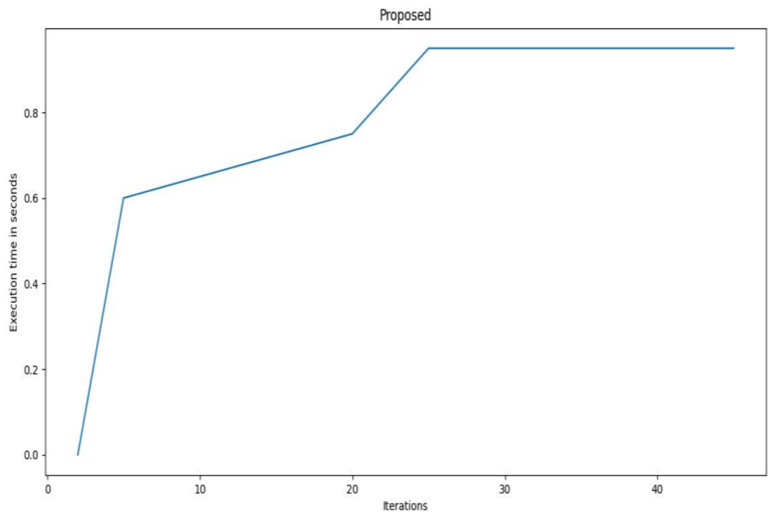
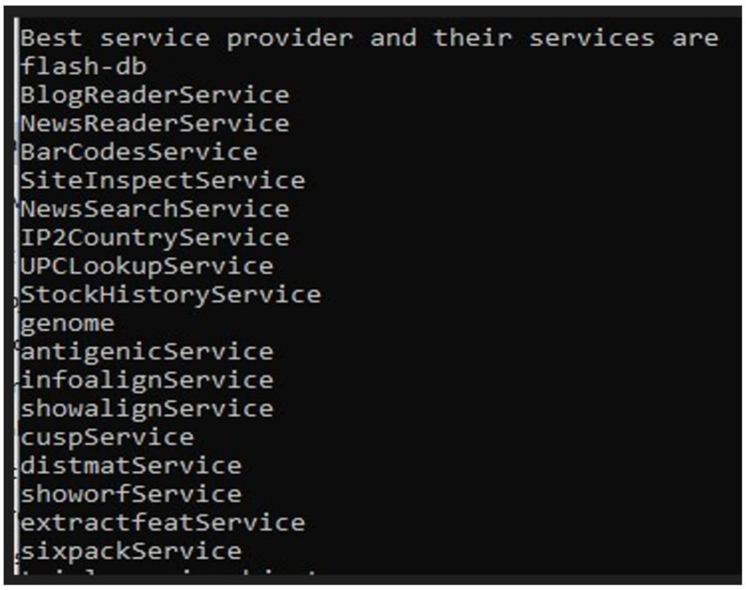


Figure 3. The iteration process about the execution time.

Result 2: The implementation process also simplifies the cloud service selection process, and Figure 4 depicts the finest service providers and their best offerings. As a result, it has two servers, which are flash-db and genome. Flash-db offers a variety of services, including Blog Reader Services, News Reader Services, and Bar Code Services, amongst others. Additionally, genome offers services such as antigenic service, info alignment service, show alignment services, and many more. These services are subject to modification in response to the requirements.



```
Best service provider and their services are
flash-db
BlogReaderService
NewsReaderService
BarCodesService
SiteInspectService
NewsSearchService
IP2CountryService
UPCLookupService
StockHistoryService
genome
antigenicService
infoalignService
showalignService
cuspService
distmatService
showorfService
extractfeatService
sixpackService
```

Figure 4. Finest service providers and their best offerings.

Result 3: Classification model parameter utilizing Eagles' Feature Selection in Combination with the Confusion Matrix for Tuning has shown in Figure 5. Figure 5 depicts the confusion matrix for tuning, which is based on eagles' features selection-based tuning. The confusion matrix shows the actual and predicted values of the model in terms of true positive, true negative, false positive, and false negative. These values are used to compute the model's accuracy, precision-recall graph, and error rate. The confusion matrix for different classes is obtained from Table 2.

Table 2. Confusion Matrix for different classes.

	Predicted Class	
Actual Class	True positive	False Negative
	False Positive	True Negative

The precision–recall graph is shown in Figure 6, which is obtained by the true positive and false positive values of the confusion matrix. In the confusion matrix, true positive value is 1, false negative value is 0, false positive value is 0.14, and true negative value is 0.86, as shown in Figure 5. The precision–recall value is computed by

$$\text{Presicion} - \text{recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} = \frac{1}{1 + 0.14} = 0.89 \quad (8)$$

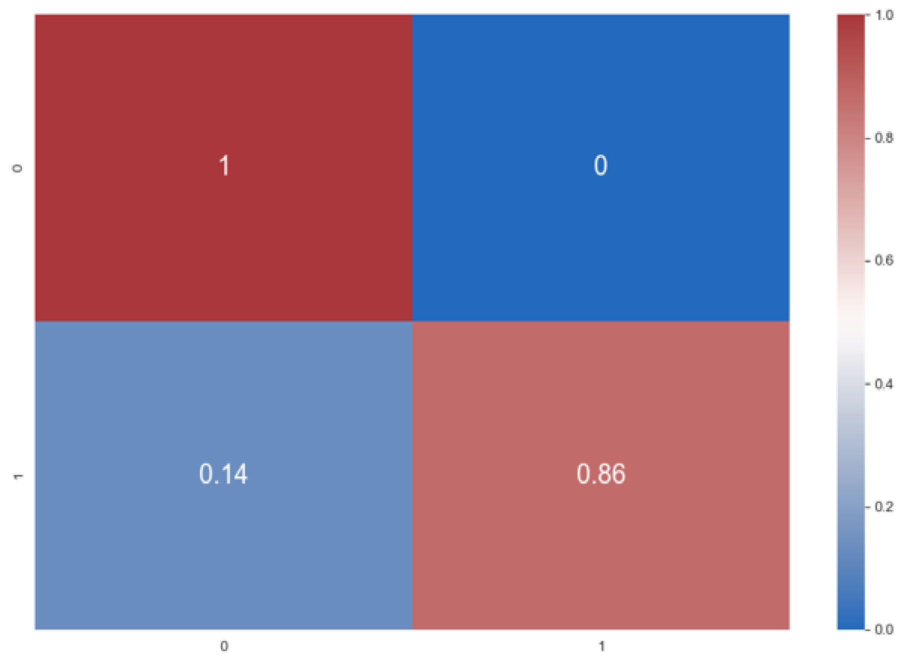


Figure 5. Eagles Feature Selection-based tuning applied to a confusion matrix for tuning.

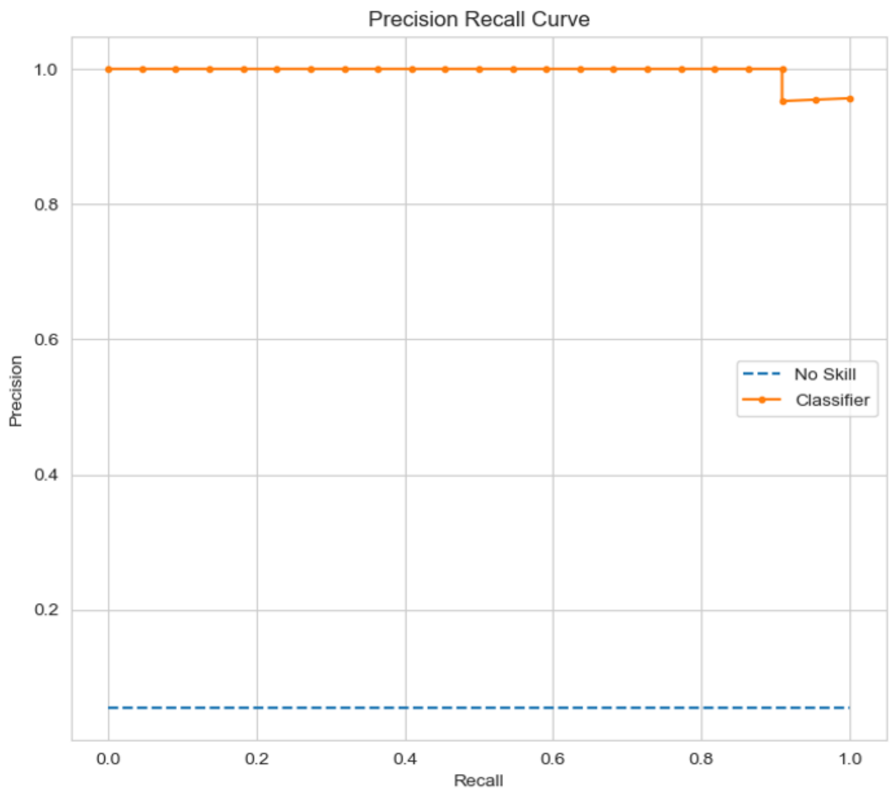


Figure 6. The precision-recall graph.

The precision–recall curve in Figure 6 was obtained using a logistic regression classification algorithm over the considered dataset. The dataset was split into train and test datasets in a 70:30 ratio using the train_test_split of the sklearn library. In this machine learning model, multiple items of web service information are categorized on the basis of four classes and fed to the machine learning algorithm to do the classifica-tion

Result 4: Figure 7 depicts the convergence curves of the proposed method as well as the algorithms that are being compared for the F1 benchmark functions. It is plotted by using a function

$$(\mathrm{f(z)} = (1.5 - \mathrm{x} + \mathrm{x} * \mathrm{y}) ** 2 + (2.25 - \mathrm{x} + \mathrm{x} * \mathrm{y} * *2) ** 2 + (2.625 - \mathrm{x} + \mathrm{x} * \mathrm{y} * * 3) ** 2) \tag{9}$$

and its range is [−4.5, 4.5]. The x-axis shows the maximum number of iterations from 0 to 500, while the *y*-axis reflects the highest score so far, from −4.5 to +4.5. All line graphs show the proposed method. It shows how users send requests to the browser and how user requests are accepted for web services. In this graph, there is no break, and it has accepted the request linearly. In the function f(z), z is equal to 1, i.e., F1.

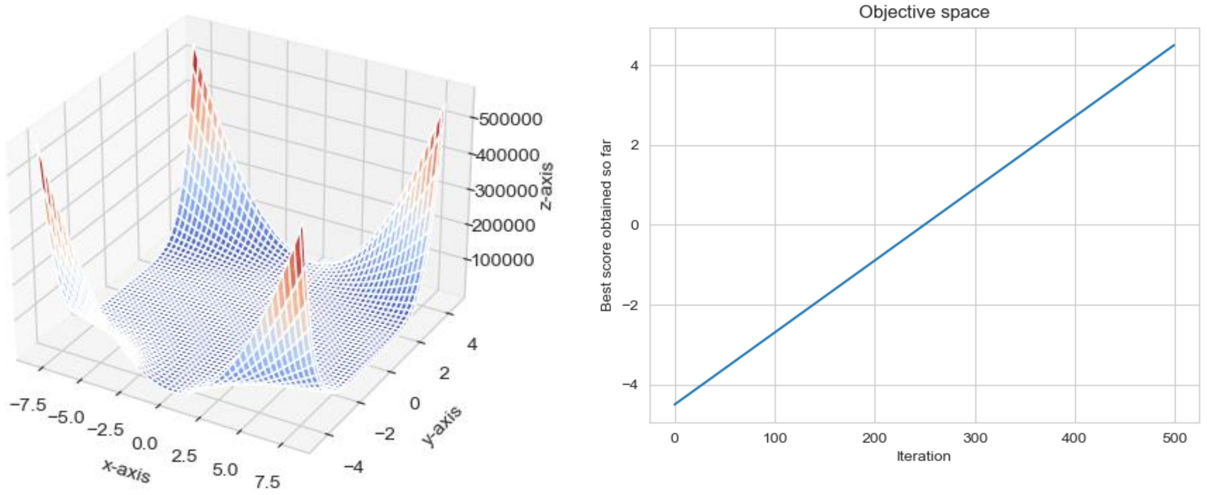


Figure 7. For function-F1 convergence curve has shown.

For the proposed method, the convergence curves are displayed in Figure 8 as well as the algorithms that are being compared for the F2 benchmark functions. The Convergence curve graph is plotted using the function

$$(\mathrm{f(z)} = \sin(\mathrm{x}) * [\sin ** (2*\mathrm{x}) ((\mathrm{x} * * 2)/3.14)]) \tag{10}$$

and its range is [–3, 3]. The *x*-axis shows the maximum number of iterations from 100 to 500, while the *y*-axis reflects the highest score so far, from −3 to 3. Users send requests to the browser, and the user's request is accepted for web services; this is shown by this graph. In this graph, there is a break. It has accepted the request linearly before the break. For a while, the request is in waiting and then it is accepted continuously. Here f(z) shows the F2 benchmark function.

The suggested method's convergence curves are shown in Figure 9 along with the algorithms that are being compared for the F3 benchmark functions. The graph which is given below is obtained by using the function

$$(\mathrm{f(z)} = \mathrm{np.sin} **(180*\mathrm{x}) + (\mathrm{x} - 1)**[1 + 10*\mathrm{np.sin} **(180*\mathrm{x} - 1)] + (\mathrm{x} - 1)**[1 + \mathrm{np.sin} **(2* 180*\mathrm{x})]) \tag{11}$$

and its range is [−10, 10]. The *x*-axis shows the maximum number of iterations in the objective space graph from 200 to 500, while the *y*-axis reflects the highest score so far, from −10 to +10. The objective space graph shows how users send requests to the browser and how user requests are accepted for web services. The graph accepted the request linearly before the break; for a while, the request of the user is in waiting; and then the request is accepted continuously. This graph shows the request is accepted in the above direction. It shows how the function is working. The F3 benchmark function is calculated by the used function in this result.

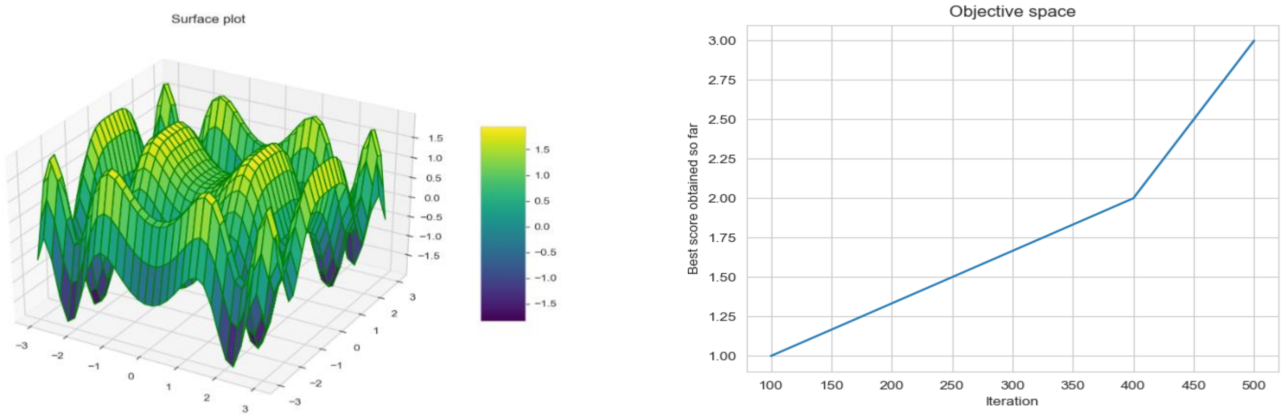


Figure 8. For function-F2 convergence curve has shown.

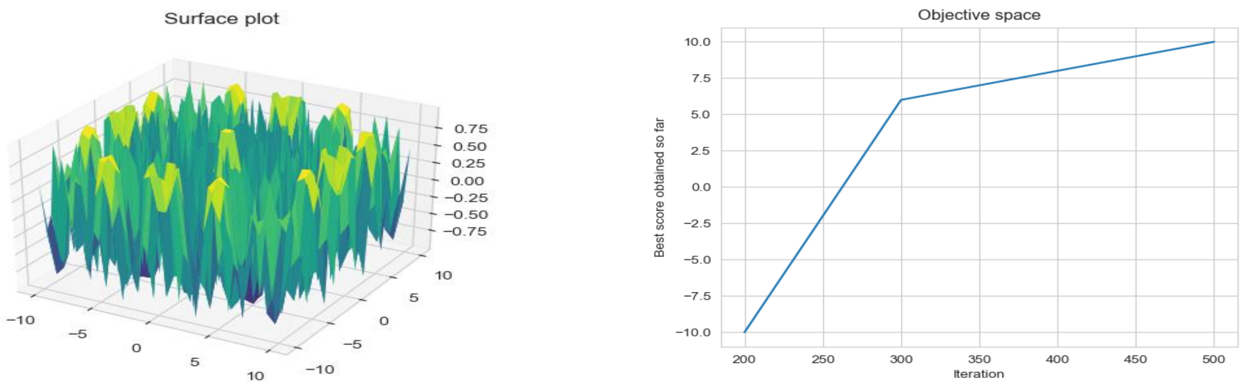


Figure 9. For function-F3 the convergence curve has shown.

Figure 10 depicts the convergence curves of the proposed method as well as the algorithms that are being compared for the F4 benchmark functions. This convergence graph was created with function.

$$(\mathrm{f(z)} = 100 * (\mathrm{y} - \mathrm{x} * * 2) ** 2 + (\mathrm{y} - 1) ** 2) \tag{12}$$

And its range is [−30, 30]. F4 benchmark function is calculated by using this function. The x-axis of objective space graph shows the number of iterations from 100 to 500, while the y-axis reflects the highest score so far from −30 to + 30. It shows how users send a request to the browser and how the user request is accepted for web services. In this graph, there is break; it accepted the request linearly before to the break. The user's request is held in a queue for a period of time before being accepted continually and it continually and it indicates that the request was granted; in the other direction, it illustrates how the function operates.

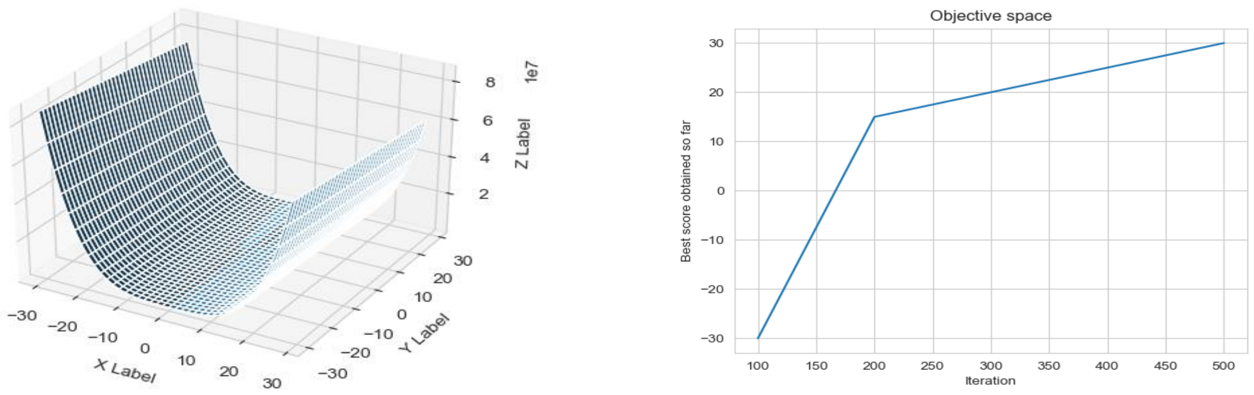


Figure 10. For function-F4 the convergence curve has shown.

Result 5: Figure 11 indicates the comparison of the correlation between the numbers of iterations. It shows that each iteration number was run 30 times to get its mean. The algorithm was put to the test against each one of the other methods. Based on Figure 11, it is observed that the suggested ESA with GA performs better than other algorithms (GA, Whale Optimization Algorithm (WOA), Discrete Guided Artificial Bee Colony (DGABC), Hybrid Genetic Algorithm (HGA), and Greedy Randomized Adaptive Search Procedure (GRASP)) with respect to the optimum fitness rating and speed of convergence reached. ESA with GA's convergence rate is the quickest among GRASP, HGA, DGABC, WOA, and GA.

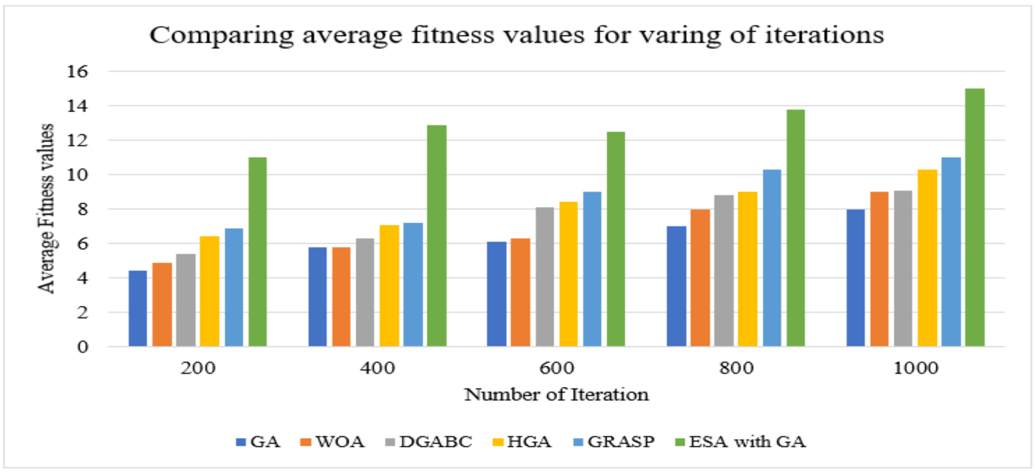


Figure 11. Comparison of the correlation between the numbers of iterations.

Result 6: Figure 12 shows a case in which a composite service is made up of 25 abstract services and each abstract service has 25, 50, 75, or 100 candidate services. Increasing the number of abstract services has no effect on the quality of the solution shown in Figure 12. Figure 12 shows that the suggested ESA with GA yields the best average fitness values of 10.09779 for 100 candidate services, whereas the different state of art algorithms, GRASP, DGABC, HGA, GA, and WOA, achieve the best results, 8.94313, 6.18779, 7.97482, 5.79482, and 5.7382, correspondingly.

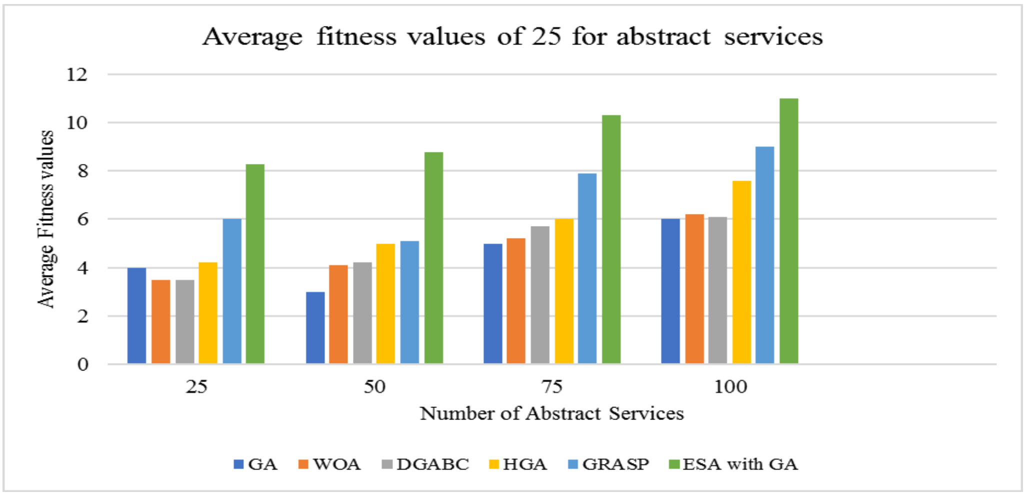


Figure 12. Best average fitness values for 25 abstract services.

Result 7: As shown in Figure 13, there are 50 abstract services in the composite service, and the potential services vary between 25 and 100 for each abstract service. It shows that while the quality of the solutions remains unchanged, with more abstract service, there is a rise in optimum fitness score. For 100 candidate services, the suggested ESA with GA has the best average fitness value of 11.9508 as compared to others.

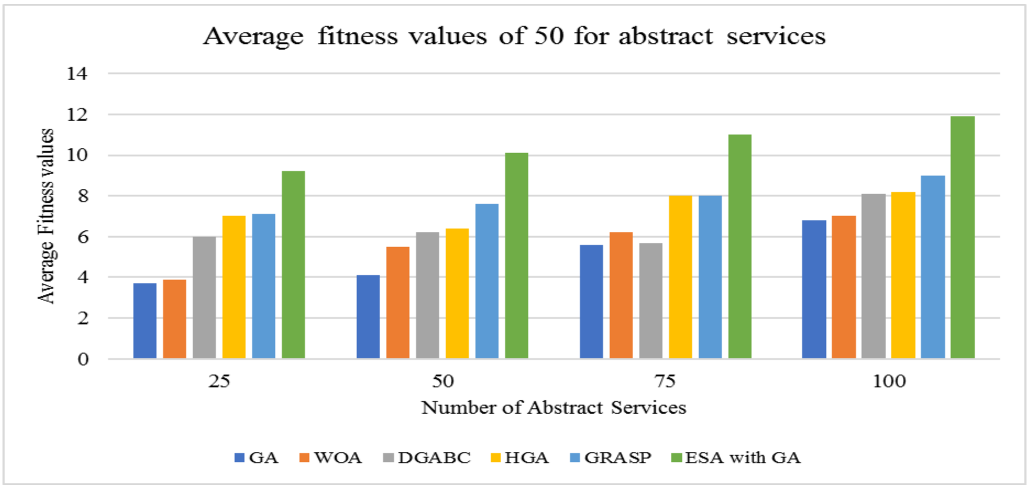


Figure 13. Best average fitness values for 50 abstract services.

Result 8: Figure 14 demonstrates the 75 abstract services that constitute the composite service, with the varying number of candidate services for each abstract service ranging between 25 and 100. We can observe that the quality of the solution remains unaffected with the growing number of abstract services and the suggested ESA with GA yields a solution of 12.82128, while the solutions of GRASP, HGA, DGABC, WOA, and GA are 11.086, 10.7631, 9.42749, 8.92924, and 7.9368, respectively, for a sample of 100 candidates.

Result 9: Figure 15 depicts a situation in which a composite service consists of 100 abstract services, with the number of candidate services between 25 and 100 for each abstract service. It finds that the average fitness values grow as the number of abstract services increases, without affecting the quality of the solution. The suggested method of ESA with GA for 100 candidate services yields a solution of 12.83428, while the next best answer has been achieved by the GRASP. Therefore, compared to other algorithms, the proposed method is more efficient.

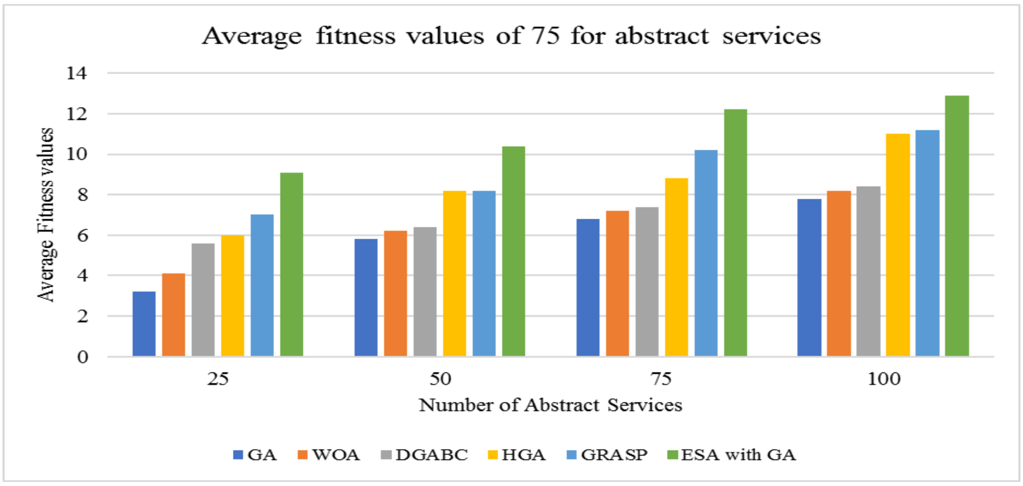


Figure 14. Best average fitness values for 75 abstract services.

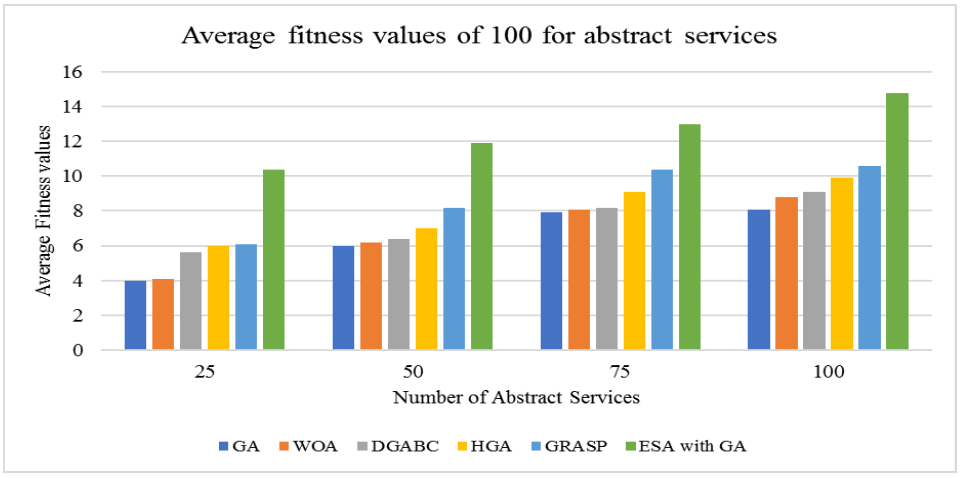


Figure 15. Best average fitness values for 100 abstract services.

Comparison Tables

The list of all benchmark functions (F), i.e., F1, F2, F3, and F4, along with a description of each, is shown in Table 3. The values of these benchmark functions are calculated as shown above.

Table 3. List of benchmarking functions.

Function	**Dim**	**Range**	f_{min}
$f_1(z) = (1.5 - x + x * y) ** 2 + (2.25 - x * y * 2) ** 2 + (2.625 - x + x * y ** 3) ** 2)$	30	$[-4.5, 4.5]$	0
$f_2(z) = (\sin(x) * \left[sin ** (2 * x)\left(\frac{x** 2}{3.14}\right)\right])$	30	$[-3, 3]$	0
$f_3(z) = np.sin ** (180 * x) + (x - 1) ** [1 + 10 * np.\sin(180 * x - 1)] + (x - 1) ** [1 + np.sin ** (2 * 180 * x)])$	30	$[-10, 10]$	0
$f_4(z) = 100 * (y - x ** 2) ** 2 + (y - 1) ** 2$	30	$[-30, 30]$	0

Table 4 summarizes the statistical data, including mean and standard deviation, for several test functions.

Table 4. Comparisons of Benchmark Functions Results.

Benchmark Function	ESA with GA		WOA	
	Mean	**σ**	**Mean**	**σ**
F1	7.504×10^5	7.865×10^5	5.151×10^{15}	2.704×10^{15}
F2	0.301	0.1423	0.5584	0.3194
F3	1.5×10^2	1.876×10^2	0.8413	0.6186
F4	4.317×10^3	2.564×10^3	40.3801	221.4131

It can be observed from Table 5 that the execution duration of abstract services varies between 25 and 100 in relation to 100 candidate services. It displays how long the time duration is for that is the execution times for GRASP, HGA, DGABC, WOA, and GA to select the best options. The minimum time taken by the technique to select the best options is considered an effective strategy and from the experimental analysis, it has been noticed that for varying numbers of services, the execution time for ESA with GA was the minimum, i.e., 41.56 s, compared to the others shown in Table 5. Hence, according to Table 4, the suggested technique for ESA with GA is more efficient than the other methods examined.

Table 5. Various stratagems' execution times are evaluated (in seconds).

Time Taken by Various Methods	No. of Services			
	25	**50**	**75**	**100**
ESA with GA	30.56	35.15	39.56	41.56
GRASP	36.40	44.56	46.14	59.14
HGA	38.20	44.59	58.12	50.12
DGABC	40.632	41.45	43.59	58.63
WOA	35.560	30.20	30.56	50.56
GA	57.351	57.58	57.35	57.35

7. Conclusions and Future Scope

Cloud consumers often have difficulties while attempting to choose the cloud solution that most effectively satisfies their requirements. As a result, a solution must be found that meets both the functional and non-functional needs while choosing the service composition. The present research has proposed a QoS-based Cloud Service Recommendation System with the use of the Eagle Strategy and a Genetic Algorithm. To circumvent difficulties such as a delayed convergence rate or an early convergence, this technique allows for the establishment of a healthy equilibrium between exploratory and exploitative activities. Experiments and evaluations demonstrate that the proposed approach has the highest average fitness values across all iteration variants, as shown in the graph, compared to other state-of-the-art methods. Here, a comparative study has been made by showing the experimental outcomes that manifest the proposed methodology obtain greater performance than the other standard algorithms. The results of the experiment show that ESA and GA are superior to other contemporary algorithms in terms of efficiently generating globally QoS-based Cloud Service Recommendation Systems. The limitations of our work can be stated as we have considered the data from a single repository, which can be enhanced further. To reduce the number of cloud-to-cloud connections, we are also looking for a variety of service repositories in future research. Cloud service interdependencies and correlations, as well as QoS metrics, will be considered in the future as part of our research. The randomness in the algorithm and number of hyper parameters can be minimized to reduce the computational complexity by using other nature inspired algorithms.

Author Contributions: S.S.M. and R.R.K. contributed to the main idea of qualitative analysis for the re-search and wrote the original manuscript. M.A., A.T.Z. and N.P. contributed significantly to improving the technical and grammatical contents of the manuscript. M.A., A.T.Z. and N.P. reviewed the manuscript and provided valuable suggestions to further refine the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data is provided in the manuscript.

Acknowledgments: The authors would like to acknowledge the support of Prince Sultan University for paying the Article Processing Charges (APC) of this publication.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Cusumano, M. Cloud computing and SaaS as new computing platforms. *Commun. ACM* **2010**, *53*, 27–29. [CrossRef]
2. Buyya, R.; Broberg, J.; Goscinski, A.M. (Eds.) *Cloud Computing: Principles and Paradigms*; John Wiley & Sons: Hoboken, NJ, USA, 2010.
3. Luo, M.; Zhang, L.J.; Lei, F. An insuanrance model for guaranteeing service assurance, integrity and qos in cloud computing. In Proceedings of the 2010 IEEE International Conference on Web Services, Miami, FL, USA, 5–10 July 2010; IEEE: Piscataway, NJ, USA, 2010; pp. 584–591.
4. Isinkaye, F.; Folajimi, Y.; Ojokoh, B. Recommendation systems: Principles, methods and evaluation. *Egypt. Inform. J.* **2015**, *16*, 261–273. [CrossRef]
5. Bobadilla, J.; Ortega, F.; Hernando, A.; Gutiérrez, A. Recommender systems survey. *Knowl. Based Syst.* **2013**, *46*, 109–132. [CrossRef]
6. Hu, P.; Dhelim, S.; Ning, H.; Qiu, T. Survey on fog computing: Architecture, key technologies, applications and open issues. *J. Netw. Comput. Appl.* **2017**, *98*, 27–42. [CrossRef]
7. Subashini, S.; Kavitha, V. A survey on security issues in service delivery models of cloud computing. *J. Netw. Comput. Appl.* **2011**, *34*, 1–11. [CrossRef]
8. Krishna, P.V.; Misra, S.; Joshi, D.; Obaidat, M.S. Learning automata based sentiment analysis for recommender system on cloud. In Proceedings of the 2013 International Conference on Computer, Information and Telecommunication Systems (CITS), Piraeus, Greece, 7–8 May 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 1–5.
9. Pallis, G. Cloud Computing: The New Frontier of Internet Computing. *IEEE Internet Comput.* **2010**, *14*, 70–73. [CrossRef]
10. Li, J.; Tao, F.; Cheng, Y.; Zhao, L. Big data in product lifecycle management. *Int. J. Adv. Manuf. Technol.* **2015**, *81*, 667–684. [CrossRef]
11. Yao, X.; Lin, Y. Emerging manufacturing paradigm shifts for the incoming industrial revolution. *Int. J. Adv. Manuf. Technol.* **2016**, *85*, 1665–1676. [CrossRef]
12. Hao, F.; Pei, Z.; Park, D.-S.; Phonexay, V.; Seo, H.-S. Mobile cloud services recommendation: A soft set-based approach. *J. Ambient Intell. Humaniz. Comput.* **2018**, *9*, 1235–1243. [CrossRef]
13. Malouche, H.; Ben Halima, Y.; Ben Ghezala, H. Trust level estimation for cloud service composition with inter-service constraints. *J. Ambient Intell. Humaniz. Comput.* **2019**, *10*, 4881–4899. [CrossRef]
14. Li, C.; Li, J.; Chen, H.; Heidari, A.A. Memetic Harris Hawks Optimization: Developments and perspectives on project scheduling and QoS-aware web service composition. *Expert Syst. Appl.* **2021**, *171*, 114529. [CrossRef]
15. Alhijawi, B.; Kilani, Y. A collaborative filtering recommender system using genetic algorithm. *Inf. Process. Manag.* **2020**, *57*, 102310. [CrossRef]
16. Li, C.; Li, J.; Chen, H. A Meta-Heuristic-Based Approach for Qos-Aware Service Composition. *IEEE Access* **2020**, *8*, 69579–69592. [CrossRef]
17. Li, F.; Wang, Y.; Li, P. The Evaluation Model of Network QoS Based on Intelligent Water Droplets Algorithm. *Discret. Dyn. Nat. Soc.* **2020**, *2020*, 1–7. [CrossRef]
18. Jiang, Y.; Tao, D.; Liu, Y.; Sun, J.; Ling, H. Cloud service recommendation based on unstructured textual information. *Futur. Gener. Comput. Syst.* **2019**, *97*, 387–396. [CrossRef]
19. Alayed, H.; Dahan, F.; Alfakih, T.; Mathkour, H.; Arafah, M. Enhancement of Ant Colony Optimization for QoS-Aware Web Service Selection. *IEEE Access* **2019**, *7*, 97041–97051. [CrossRef]
20. Zhang, C.; Li, Z.; Li, T.; Han, Y.; Wei, C.; Cheng, Y.; Peng, Y. P-CSREC: A New Approach for Personalized Cloud Service Recommendation. *IEEE Access* **2018**, *6*, 35946–35956. [CrossRef]
21. Gavvala, S.K.; Jatoth, C.; Gangadharan, G.R.; Buyya, R. QoS-aware cloud service composition using eagle strategy. *Future Gener. Comput. Syst.* **2019**, *90*, 273–290. [CrossRef]
22. Mezni, H.; Abdeljaoued, T. A cloud services recommendation system based on Fuzzy Formal Concept Analysis. *Data Knowl. Eng.* **2018**, *116*, 100–123. [CrossRef]

23. Kumar, R.R.; Shameem, M.; Kumar, C. A computational framework for ranking prediction of cloud services under fuzzy environment. *Enterp. Inf. Syst.* **2022**, *16*, 167–187. [CrossRef]
24. Dahan, F.; Binsaeedan, W.; Altaf, M.; Al-Asaly, M.S.; Hassan, M.M. An Efficient Hybrid Metaheuristic Algorithm for QoS-Aware Cloud Service Composition Problem. *IEEE Access* **2021**, *9*, 95208–95217. [CrossRef]
25. Pandharbale, P.B.; Mohanty, S.N.; Jagadev, A.K. QoS-Aware Web Services Recommendations Using Dynamic Clustering Algorithms. *Int. J. Inf. Syst. Model. Des.* **2022**, *13*, 1–16. [CrossRef]
26. She, Q.; Wei, X.; Nie, G.; Chen, D. QoS-aware cloud service composition: A systematic mapping study from the perspective of computational intelligence. *Expert Syst. Appl.* **2019**, *138*, 112804. [CrossRef]
27. Kim, K.-J.; Han, I. Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index. *Expert Syst. Appl.* **2000**, *19*, 125–132. [CrossRef]
28. ZCAN, İ.; ÇELİK, M. Developing recommendation system using genetic algorithm based alternative least squares. In Proceedings of the 2018 International Conference on Artificial Intelligence and Data Processing (IDAP), Malatya, Turkey, 28–20 September 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–5.
29. Sakkopoulos, E.; Adamopoulou, P.; Tsakalidis, A.K.; Sioutas, S.; Manolopoulos, Y. Personalized selection of web services for mobile environments: The m-scroutz solution. In Proceedings of the International Conference on Management of Emergent Digital Ecosystems, Lyon, France, 27–30 October 2009; pp. 254–260.
30. Sembiring, M.; Surendro, K. Service catalogue implementation model. In Proceedings of the 2016 4th International Conference on Information and Communication Technology (ICoICT), Bandung, Indonesia, 25–27 May 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 1–6.
31. Luo, J.; Ni, X.; Yong, J. A trust degree based access control in grid environments. *Inf. Sci.* **2009**, *179*, 2618–2628. [CrossRef]
32. Cui, Z.; Xu, X.; Xue, F.; Cai, X.; Cao, Y.; Zhang, W.; Chen, J. Personalized Recommendation System Based on Collaborative Filtering for IoT Scenarios. *IEEE Trans. Serv. Comput.* **2020**, *13*, 685–695. [CrossRef]
33. Wu, X.; Huang, Y.; Wang, S. A new similarity computation method in collaborative filtering based recommendation system. In Proceedings of the 2017 IEEE 86th Vehicular Technology Conference (VTC-Fall), Toronto, ON, Canada, 24–27 September 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 1–5.
34. Kumar, R.R.; Shameem, M.; Khanam, R.; Kumar, C. A hybrid evaluation framework for QoS based service selection and ranking in cloud environment. In Proceedings of the 2018 15th IEEE India Council International Conference (INDICON), Coimbatore, India, 16–18 December 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–6.
35. Akbar, M.A.; Shameem, M.; Mahmood, S.; Alsanad, A.; Gumaei, A. Prioritization based Taxonomy of Cloud-based Outsource Software Development Challenges: Fuzzy AHP analysis. *Appl. Soft Comput.* **2020**, *95*, 106557. [CrossRef]
36. Kumar, R.R.; Tomar, A.; Shameem, M.; Alam, N. OPTCLOUD: An Optimal Cloud Service Selection Framework Using QoS Correlation Lens. *Comput. Intell. Neurosci.* **2022**, *2022*, 2019485. [CrossRef]
37. Shameem, M.; Kumar, R.R.; Nadeem, M.; Khan, A.A. Taxonomical classification of barriers for scaling agile methods in global software development environment using fuzzy analytic hierarchy process. *Appl. Soft Comput.* **2020**, *90*, 106122. [CrossRef]
38. Kumar, M.; Sharma, S.C. PSO-based novel resource scheduling technique to improve QoS parameters in cloud computing. *Neural Comput. Appl.* **2020**, *32*, 12103–12126. [CrossRef]