Towards Service Composition based on Hybrid Bio-Inspired Cloud-based QoS Provisioning Approach.

Waleed Bahgat
Mansoura University Faculty of Computers and Information
Mahmoud A. Salam
Mansoura University
Ahmed Atwan
Mansoura University
Mahmoud Badawy (engbadawy@mans.edu.eg)
Mansoura University Faculty of Engineering https://orcid.org/0000-0002-0120-3235
Eman El-Daydamony
Mansoura university

Research

Keywords: Fruit Fly Optimization Algorithm (FOA), Fuzzy clustering, MapReduce, Parallel composition, Particle Swarm Optimization(PSO), Quality of Service (QoS), Web Service

Posted Date: December 29th, 2020

DOI: https://doi.org/10.21203/rs.3.rs-128218/v1

RESEARCH

Towards Service Composition based on Hybrid Bio-Inspired Cloud-based QoS Provisioning Approach

Waleed M. Bahgat[1], Mahmoud A.Salam[1], Ahmed Atwan[1], Mahmoud M. Badawy[2*] and Eman El-Daydamony[1]

*Correspondence: Engbadawy@mans.edu.eg
[1]Information Technology Department, Faculty of Computer and Information Sciences, Mansoura University., Mansoura, Egypt
Full list of author information is available at the end of the article

Abstract

It has recently become a critical issue to provide software development in a service-based conceptual style for business companies . As a powerful technology for service-oriented computing, the composition of web services is investigated. This offered great opportunities to improve IT industries and business processes by forming new value-added services that satisfy the user's complex requirements. Unfortunately, many challenges are facing the service composition process. These include the difficulties to satisfy the user's complex demands, maintaining the performance to be matched with the quality of service (QoS) requirements, and search space reduction for QoS missing or changeable values. Accordingly, this paper proposes a cloud-based QoS provisioning service composition (CQPC) framework to address these challenges. To prove the concept and the applicability of the CQPC framework, a Hybrid Bio-Inspired QoS provisioning (HBIQP) technique is presented for the operation of the CQPC framework modules. The solution space is reduced via utilizing skyline concepts to have faster execution time and keep only reliable and most interesting services. The CQPC framework is equipped with two proposed algorithms: (i) the modified highly accurate prediction (MHAP) algorithm to enhance the prediction of QoS values of the services participating in the composition process, (ii) the MapReduce fruit fly Particle swarm Optimization (MR-FPSO) algorithm to handle composing web services for large scale of data in the cloud environment. The experimental results demonstrate the worthiness of the HBIQP technique to meet the performance metrics more than other state-of-the-art techniques in terms of average fitness value, accuracy, and execution time.

Keywords: Fruit Fly Optimization Algorithm (FOA); Fuzzy clustering; MapReduce; Parallel composition; Particle Swarm Optimization(PSO); Quality of Service (QoS); Web Service

1 Introduction

Undoubtedly,service-oriented architecture (SOA) has significant use in enterprise information technology (IT) environments due to its technical and cost benefits. The use of SOA is recognized as one of the major trends in complex organizations for so many reasons as follows:(i) it helps the solution architects and designers to conduct better capacity planning, (ii) define better solutions for resource sharing, and (iii) adopt easier service migration solutions in terms of the deployment and maintenance [1]. With the exponential growth of web services available in the cloud

infrastructure, selecting the one that best fits the requirements of users is a challenging issue. Choosing the web service that fulfills the different needs of users is the core of service-oriented computing in the software developing environment.

With the rapid development of the Web, more services are created that provide specific functions. Recently, the convergence between the Internet of Things (IoT), big data ,and cloud computing has received significant attention in both academia and industry. The cloud will act as a transitional layer between IoT entities and applications to efficiently manage resources. The IoT consists of virtually linked objects or intelligent devices that are physically distinguishable and suitable for sensing, processing, and interaction with the circumstances. The integration of smart device services will create an application based on the IoT domain. For the operation of IoT application tasks, smart devices are equipped with sensors. These sensors collect data around the area they are deployed. The real-time data obtained from sensor devices is converted into new data sources to improve the accuracy of predictive services and to optimize network operations for efficient automation. The IoT-based cloud infrastructure prototype that allows access to the IoT is defined as sensing as a service (SaS). The SaS enables ubiquitous management of remote sensors through web services and facilitates the efficient delivery of sensor data on a demand basis to satisfy the users' requirements.

The future of Internet services has concentrated on "its capacity to build a framework has the ability for delivering the service for a significant number of users efficiently". The primary mission is to move services based on "pay as you go," from conventional form to the form of web applications so that we can supply Quality of Service (QoS). Accordingly, efficient information dissemination has become the new production factor, notably in terms of developing more web services over the network. To this end, organizations start to provide their core business in the form of services over the web. So, the concept of web services becomes very important ,and a challenge for many researchers [2]. The best service composition and service recommendation in the SOA integration context are two important aspects of applications that mainly depend on web services. Service composition is an application of web services, where a single available service may not able to guarantee user satisfaction. For this reason, many services are joined together to create a new composite service with a new value-added function. The proper creation of the composite service is a challenging issue. Additionally, several services may have the same functional attribute but differ in non-functional attributes. Hence, the service selection to participate in the workflow of composite service is another challenging issue.

In the service composition, the time and optimality dimensions are the most common factors. It is important to minimize the time to create a composite service. The execution time is highly related to the large available solution space. Besides, it is crucial to guarantee that the global composite service satisfies the users' requirements. Hence, the composition problem should be solved by taking into consideration these two factors. Composing the services is considered as an optimization problem with a large solution space of candidate services with time constraints.

Due to the Internet environment nature, the QoS of candidate services may be missed or changed along the time. The QoS values affect the optimality of composite

service. Services with unstable QoS value leads to inefficient composite service. Therefore, stable services and accurate prediction of QoS values are needed. To summarize, optimality, execution time and QoS values are essential parameters to guarantee user satisfaction. Also, achieving the scalability of available services in a certain amount of time is a composition challenge.

Accordingly, this paper addresses the following research issues:

- Selecting the most appropriate candidate service that best participates in the global end-to-end solution.
- Reducing the search space of candidate services to reduce the execution time for composite service.
- Providing an accurately predicted value for the QoS of service to cope with the changes in the network environment.
- Measuring the final solution optimality.

In this study, a novel cloud-based QoS provisioning Service composition (CQPC) framework is proposed to adopt the whole process of service composition. The CQPC framework consists of five modules as follows: (i)the normalizer module gives a certain range for all QoS attributes and historical user orders. (ii) the Clusters generator module reduces the search space and prepares a set of QoS predictive values for the next module. (iii) the QoS predictor module is responsible for predicting the QoS values that may be changed or missed because of the dynamic changes of the network. (iv)the service reduction module reduces the available candidate services to improve the composition process. Finally, (vi) the composition process is done, and a list of candidate composite services is generated through the service composer and evaluator module.

Moreover, this study presented a Hybrid Bio-Inspired QoS- provisioning (HBIQP) technique that manages the operation of the CQPC framework modules. First, the service users are clustered using the fuzzy C-Means clustering module. Then, entropy with correlation variation statistical methods is used to eliminate the unreliable services with values that have a larger deviation by using in stable service selector module. Many existing algorithms for service selection can be used [3] [4] [5]. However, after filtering solution space based on uncertainty, it is still large and has a slow execution time. The skyline service sub-module is used to reduce the solution space. The main objective is to provide faster execution time and keep only reliable and most interesting services.

To this end, some important issues should be addressed, such as the lack of monitoring of the uncertainty and unreliability of services due to the dynamic nature of the environment. QoS attributes could be changed or even becomes unavailable. The use of inaccurate values for QoS leads to inaccurate results in the composition process. For this reason, A modified highly accurate prediction (MHAP) algorithm is proposed. MHAP algorithm operates in the QoS predictor module to predict the QoS values based on the available historical information. The MapReduce fruit fly particle swarm optimization (MR-FPSO) algorithm is proposed in the service composer module to handle the large scale of data in the cloud environment. The proposed algorithms perform the composition process in an acceptable time cost with more accuracy while satisfying the users' needs.

The paper is organized as follows: in section 2 the related state-of-the-art studies and researches are reviewed. Section 3 describes the preliminary concepts of web

services. Section 4 explains various modules of the proposed CQPC framework for service composition. In section 5, the proposed HBIQ is illustrated. In section 6 the simulation results that compare the proposed HBIQP technique with the other state-of-the-art techniques are discussed. Finally, section 7 outlines conclusions and future work.

2 Literature Review

Recently, there has been extensive research on creating reliable services, selection of services ,and the composition process. This section reviews the related studies to the service computing that has been widely studied, and it explores several issues and challenges. Rehman et al. [6] and Hossain et al. [7], proposed a multi-phase process and apply multiple criteria decision making to find the value of QoS. Besides, they used the weights assigned to each QoS. However, the global end-to-end constraints provided by the user were not satisfied with the results although the time cost was acceptable. Jin et al. [8] performed the composition of services in the cloud platform using a genetic algorithm besides the correlation-aware model.

Mahmoud et al. [9] proposed a Map-Reduce Non-Dominated Sorting Genetic Algorithm (MRNSGA-II) to handle the composition problem by applying multi-objective optimization NSGA-II over Map Reduce structure beside using clustering technique. MRNSGA-II limitations were that it can't handle large scale candidate services efficiently. Besides, it is limited to search for similar services according to similar previous user requests. Seghir and Khababa [10], proposed a method that combines two evolutionary algorithms. The fruit fly optimization algorithm is combined with the genetic algorithm to take advantage of each one to find the composite service. Chen et al. [11], investigated a method that uses partial selection to select the composed services based on QoS awareness. They proved their effectiveness theoretically besides a large number of simulations.

McNabb et al. [12] presented MapReduce Particle Swarm Optimization(MRPSO). MRPSO is introduced to avoid problems arisen from large-scale parallelization. Khalid et al. [13] introduced the MRGA algorithm. They propose and evaluate the performance of GA using MapReduce (MR). They showed that their combined algorithm achieved acceptable performance while handling a huge amount of data.

Zhang et al. [14], proposed an approach that combines the PSO algorithm with Hadoop infrastructure for large-scale service composition over a distributed environment. Hossain et al. [15], proposed a method to handle the massive number of available services. They used the parallel PSO algorithm with the K-means clustering algorithm over the distributed platform, Hadoop. Unfortunately, it was inefficient in data reading. Yong et al. [16] presented a two-phase approach for service composition. The first phase applies k-means clustering for reducing the number of candidate services for selection. In the second phase, a mixed-integer programming approach is used for selecting the most appropriate service among the reduced set. Zhang et al. [17], introduced an improved version of FOA simulated with service composition to handle large-scale services available in the cloud. The disadvantage was that the execution time required was high, according to the services available.

Jin et al. [8], proposed an approach to prevent premature convergence issues. Their approach computes the fitness of individuals using a ranking method with a genetic

algorithm where the services' correlation is regarded. Ma et al. [18], presented a method for predicting the QoS values as a result of the dynamic environment of the services and network issues. They use the information of the users of services, historical information, besides the information about services itself. They aimed to find the similarity between services and service users and to obtain the similarities between users and services into a model. This model best provides predicted values for missing QoS properties based on the linear regression of different predicted values. Most of the heuristic methods are semi optimal solutions with low convergence speed and low time in large-scale data. Li [19] proposed a QoS-optimized cuckoo algorithm to improve security and service efficiency. Additionally, a multi-strategy semantic matching algorithm for improving accuracy is proposed. Farsi et al. [1] proposed a Performance-Oriented Integration Design (POID) framework that guides the software architects and designers to build better designs for the integration in SOAs. The objective of the POID framework is proved through a case study to verify the capability of the framework to achieve better accuracy in calculating the response time for the different service compositions' approaches.

Hasnain et al. [20] proposed web ranking system that is based on quality metrics of reliability and response time. Five key modules are included and 96.17% is achieved by the proposed method. TWSCO is the system proposed by Chunling et al. [21] for composing web services efficiently. The TWSCO system is designed for the trust grantee of the composition processes with a service filter, and an interface-based service. Shunshun et al. [22] implemented an algorithm to maximize adaptive brain storm (MCaBSO). MCaBSO used a dual adaptive approach to better integrate web services. A short time, good speed of convergence and scalability is achieved with the MCaBSO algorithm. Huang et al. [23] have proposed the composite stochastic Petri network model of edge servers and clusters to solve problems in large-scale networks and/or complex service processes. The main goal was to allow reliability to be achieved.

In this study, a hybrid algorithm that combines heuristic approaches particle swarm optimization (PSO) [24] and fruit fly (FOA) algorithm [25] is proposed. Also, it suggests applying this for the distributed computing platform, Hadoop. The proposed HBIQP technique aims to obtain acceptable time cost and as near-optimal to the end-to-end constraints with minimum error rate.

3 PRELIMINARIES

3.1 COMPOSITION MODEL IN THE CLOUD PLATFORM

Machines to Machines (M2M) communication manage our everyday lives. The "intelligent agents" people have touted for ages will finally materialize. Intelligent Systems' intelligent mobility initiatives provide new possibilities for optimizing the use of real-time data from various sensory resources. This has resulted in the new production factor being efficiently disseminated, particularly in improving the use of network resources, mitigating traffic congestion, and reducing the power consumption of transmission over the network. To prove a large number of tackled solutions, cloud computing is a relatively stable environment. However, the limits to network bandwidth as well as the extension into the data transfer process are the more drawbacks. Cloud computing and IoT convergence cannot meet the diverse requirements of users.

Even at the basic level, many issues and challenges such as scale, heterogeneity, physical knowledge, high complexity, privacy and security, and QoS support must be solved for successful IoT implementation. In recent years, both academia and industry have gained considerable attention for the amalgamation between the cloud and the service composition. The future of Internet services has been based on its ability to build a system or framework for a substantial number of users to efficiently deliver the service."

In this vein, seven layers are as shown in figure 1 in the amalgamation model: 1) Sensors and smart devices for the collection of IoT information comprises the perception layer. 2) The network layer makes links, sensors, and network devices to other servers. The second important aspect of this layer is that the data obtained from the perception layer is processed and transferred. 3) The Fog layer provides remote IoT scalability for a huge number of IoT devices and large volumes for low-speed applications in real-time. 4) Various sub-services to various private or public clouds may be supplied in the cloud layer. 5) The service composition layer is responsible for composing several sub-services together regarding the user's functional and non-functional requirements. 6) The application layer shall conduct end-users according to their requests with unique composite services.

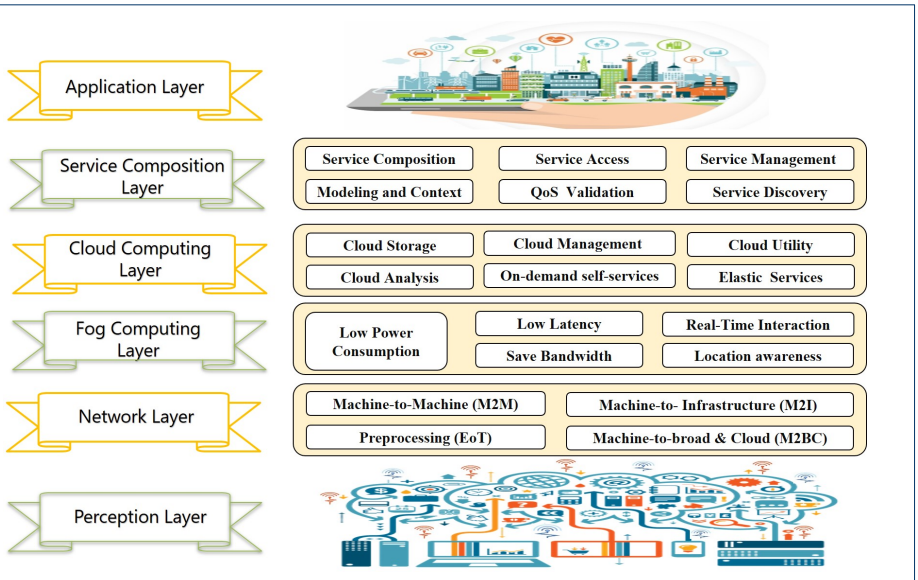

Figure 1: **Service composition and cloud architecture**

The process of forming a composite service over the cloud platform is very similar to such a traditional one. The user request represents a complex task that consists of N small tasks,where $N = \{t_1, t_2, ..., t_N\}$. For each abstract task T_i, a set of M concrete/candidate set of services is available that must satisfy the user requirement. Concrete service represented by $C_{ij} = \{CC_{i1}, CC_{i2}, ..., CC_{iM}\}$, where CC_{ij} refers to the j^{th} concrete service candidate for the i^{th} abstract task. Based on the QoS constraints, a concrete service that best fits in the global solution is selected, as shown in Figure 2.

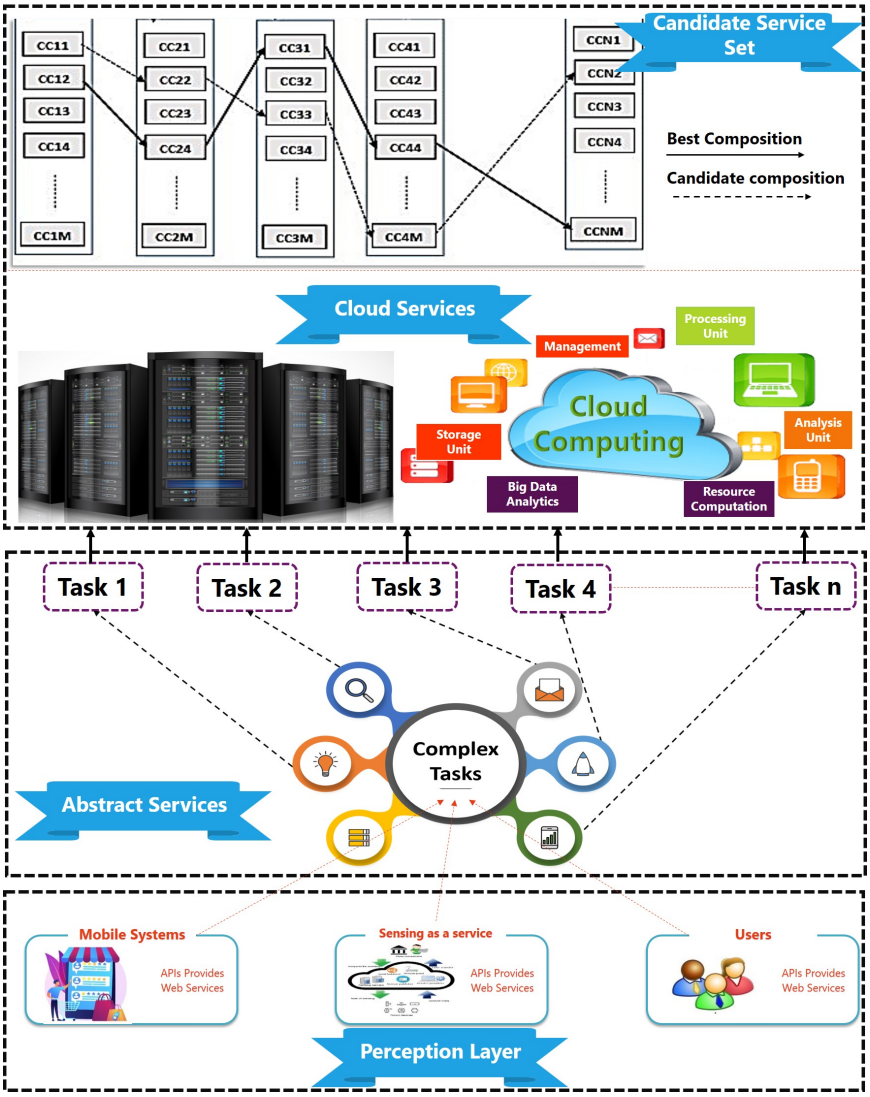

Figure 2: Composition process in the cloud

3.2 QoS-BASED MODEL

Assume S service classes are available where each class contains a set of services with the same functionality, but they are with different QoS attributes according to the service provider, such as they may differ in price, response times ,and other factors. Where $S_i = \{S_{i1}, S_{i2}, ..., S_{in}\}$ for $s \in S$. The goal is to find the best combination of services from different classes that meet the global end-to-end user preferences. There are many important used QoS non-functional attributes for describing web services over the distributed environment such as cost, response time, reliability, availability . . . etc. Table 1 reviews the brief description of important QoS parameters [26] [27].

3.3 WORKFLOW MODELS AND QOS VALUES AGGREGATION

Many workflows are available for modeling the cloud service composition task. The different composition models are based on the complexity of the user request and

Table 1: DESCRIPTION OF QOS PARAMETERS

QoS Parameter	Description
Response time	The time used by the request to be sent beside the response received.
Availability	How many requests are executed successfully /total number of requests.
Reliability	The number of failure messages / overall messages.
Throughput	How many requests per unit of time
Success	Number of responses messages/ overall number of requests
Latency	The time server needs to process a specific request.
Price	How much the user that request specific services has to pay to peruse the service.

Table 2: AGGREGATE FUNCTIONS FOR DIFFERENT MODELS

QoS attribute	Sequential	Parallel	Conditional
Response time	$\sum_{i=1}^{M} q_{t,i}{}^{j}$	$MAX_{i=1}^{M}\{q_{t,i}\}$	$\sum_{i=1}^{M} q_{t,i}{}^{j} \times pr_i$
Availability	$\prod_{i=1}^{M} q_{Av,i}^{j}$	$\prod_{i=1}^{M} q_{AV,i}^{j}$	$AVG_{i=1}^{M}\{q_{AV,i} \times pr_i\}$
Reliability	$\prod_{i=1}^{M} q_{REL,i}^{j}$	$\prod_{i=1}^{M} q_{REL,i}^{j}$	$\prod_{i=1}^{M} q_{REL,i}^{j} \times pr_i$

the posed constraints [27]. The composition models are sequential, parallel ,and conditional.

An aggregated function is used for a complex task that consists of many subtasks. This to compute each QoS attribute for the abstract task per an appropriate workflow based on various types of QoS attributes and used workflow. For each attribute, the aggregate QoS value is calculated according to the aggregated function of the most common models in Table 2 [10]. Where j is the index value of the selected concrete cloud service from the related candidate set S_i .Also, P_{ri} is the probability of i.

4 The Proposed CQPC Framework

The proposed CQPC framework consists of five modules as shown in Figure 3. First, the normalizer module is responsible for giving a certain range for all QoS attributes and historical user orders. It converts QoS values for the candidate services to normalized values ranged between 0 and 1. Normalization is essential to grantee the performance of the other CQPCs' modules.

Second, the Clusters Generator (CG) module, which is responsible for clustering data according to QoS-based services, users, and historical user orders. The CG module consists mainly of three sub-modules. These submodules are the service clustering (SC), the user clustering (UC), and the requests clustering (RC). The goal of the SC sub-module is to reduce the solution space of candidate services to reduce execution time. It improves the prediction of QoS values done by the predictor module. The UC and RC sub-modules clusters historical requests of users to determine the route of clusters to follow for any new user request.

Third, the QoS predictor (QP) module, which is responsible for predicting the QoS values that may be changed or missed because of the dynamic changes of the network. Accurate or near accurate predicted values for QoS would result in efficient composition. Many algorithms can implement this module. These algorithms include collaborative filtering, time series forecasting and ....etc. Most of these algorithms

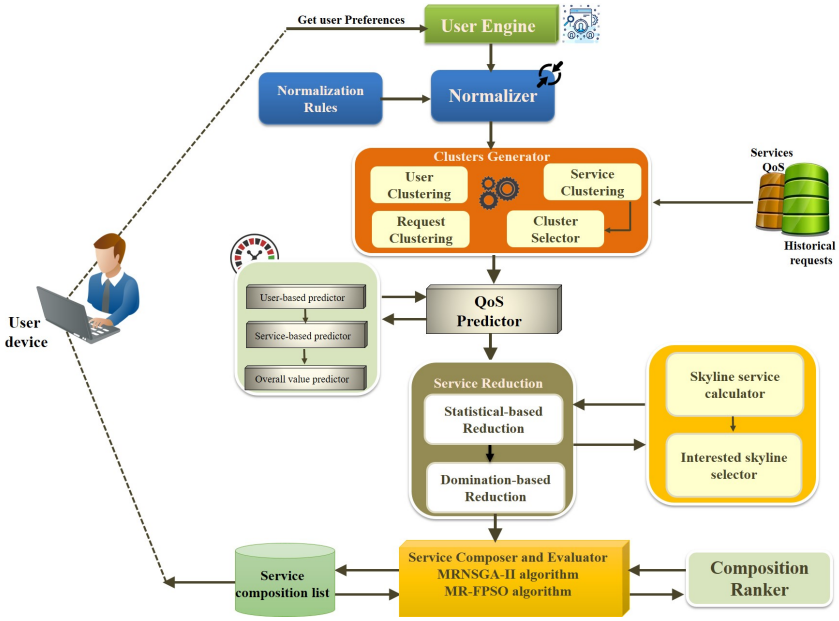

Figure 3: Hierarchal design of the CQPC proposed framework

mainly depend on the use of both user and service similarity to predict the values of QoS. Similar users are the nearest ones to a user that invoke a service with an unknown QoS value. Besides, similar services help in determining the value of QoS for a service with an unknown QoS value. Both similar users and similar services are used for making accurate predictions.

Fourth, the Service reduction (SR) module which is responsible for reducing the available candidate services to improve the composition process. This module mainly consists of two sub-modules. These sub-modules are the Statistical-based Reduction (SR) and the Domination-based Reduction (DR). The SR is responsible for removing unstable services that may affect the quality of the composition process. It employs some statistical concepts to keep only reliable and stable services. On the other hand, the DR removes redundant services for further solution space reduction.

Finally, the service composer and evaluator (SCE) module is responsible for the composition process. It generates a list of composite services. The SCE can be implemented through several algorithms. MRNSGA-II and the proposed MR-FPSO algorithm are practical examples to implement this module. The SCE is also responsible for the execution and monitoring of the service composition process. In case of failure of any composite service, it selects an appropriate alternative composite service from the generated list.

5 The Hybrid Bio-Inspired QoS-based (HBIQP) Technique

The HBIQP technique, as shown in figure 4, implements the various modules of the proposed CQPC framework. This includes the Normalizer module, CG module, QP module, SR module, and SC module. The fuzzy clustering phase implements the CG module. The user clustering sub-module is used to cluster the service users into a set of clusters to the fuzzy cluster. Also, the QoS value prediction phase implements the

QP module to predict the value of QoS. Moreover, the Stable Service Selector (SS) and Skyline Service Selector phases implement the SR and DR sub-modules. Finally, The SCE module is implemented through the proposed MR-FPSO algorithm to perform the composition process. In the next subsection, the paper discusses more details about each module.

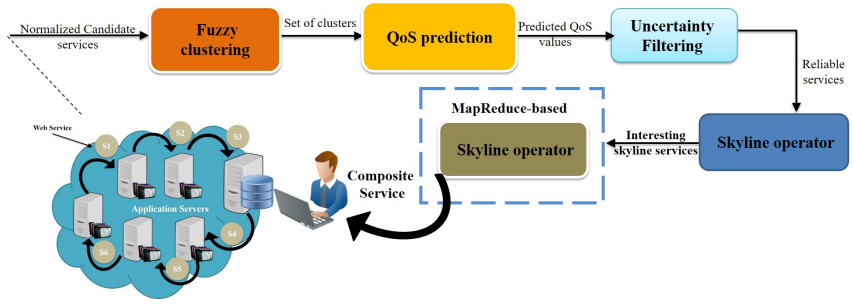

Figure 4: Cloud-vision data flow mechanism

5.1 NORMALIZER MODULE

QoS Attributes and historical requests have different scales which results in bias between attributes with low values and such with high values. To normalize all the attributes, equations (1) and (2) are used to set all values in a range from 0 to 1 [28]. The normalization module uses the available maximum and minimum values for QoS parameters.

$$f(n) = \begin{cases} \frac{q_{max} - q}{q_{max} - q_{min}} & \text{if } q_{max} - q_{min} \neq 0 \\ 1 & \text{if } q_{max} - q_{min} = 0 \end{cases} \tag{1}$$

$$f(n) = \begin{cases} \frac{q - q_{min}}{q_{max} - q_{min}} & \text{if } q_{max} - q_{min} \neq 0 \\ 1 & \text{if } q_{max} - q_{min} = 0 \end{cases} \tag{2}$$

Where q refers to the value of QoS that should be normalized, q_{min} andq_{max} refer to the minimum and maximum values for specific QoS property respectively. q_{nrm} refers to the value after normalization. Normalized QoS values for the candidate services are ready for the next following phases.

5.2 CLUSTERS GENERATOR MODULE

To implement the CG module, fuzzy clustering is used to cluster the service into a set of clusters. This gives several advantages. First, it reduces the search space of the candidate services. Second, it helps to get a set of QoS prediction values required for the next phase.

5.2.1 Fuzzy C-Means clustering (FCM) Algorithm

The Fuzzy C-Means clustering (FCM) algorithm implements the UC sub-module. FCM is used because QoS attributes are not good enough to be divided into

sharp clusters since the values may be overlapped. To determine the input and the output of the fuzzy C-means algorithm, assume there are T of users defined as $T = \{us_1, us_2, us_3, ..., us_T\}$ where us_j refers to the j^{th} user.

In addition, there are W of services of similar function $W = \{se_1, se_2, ..., se_r\}$ where se_i refers to the j^{th} web service. For each web service, there is a vector of QoS properties describing the web service se_i as an evaluation after being invoked by user us_i referred by q_{ij}. Then, for all services invoked by user us_i, the overall vector describing the QoS of services is denoted by $Q_j = \{q_{j1}, q_{j2}, q_{j3}, ..., q_{jw}\}$.

Hence, the input to the clustering algorithm is the service users and some parameters related to the clustering process. Each user is defined by a vector of QoS attributes represent the set of historical values of services invoked by the user. The output of the fuzzy C-Means algorithm is a set of centers of the clusters and the memberships of objects belong to those clusters. In the basic fuzzy C-means algorithm, the used similarity is measured through Euclidean distance. It gets distance, the similarity between the objects to be grouped and the center of that group. To get more accuracy in the clustering process, the Pearson Correlation Coefficient (PCC) algorithm is used with fuzzy C-Means instead of Euclidean distance in computations. PCC has a better accuracy since it indicates to what degree service is related and more correlated to another service. For this reason, PCC is more appropriate when dealing with quality values observed by users in the real world. In the context of PCC-based FCM, suppose that there are two service users denoted as a and b. To find the correlation and similarity between them, PCC provides the following equation:

$$similarity^{'(a,b)} = \frac{\sum_{i \in (I \bigcap I(b))} (q_{ai} - \bar{q_a}) \times (q_{bi} - \bar{q_b})}{\sqrt{\sum_{i \in I(a)} (q_{ai} - \bar{q_a})^2} \times \sqrt{\sum_{i \in I(b)} (q_{bi} - \bar{q_b})^2}} \tag{3}$$

Where q_a and q_b denote the average values that both users a and b observed for QoS values. $I(a)$ and $I(b)$ refer to the set of services invoked by users a and b respectively. High similarity means there is a small distance between users and vice versa. According to this, the final modified form of the objective function of FCM that should minimize clustering iteration is described as:

$$J = \sum_{j=1}^{N} \sum_{i=1}^{c} U_{ij}^{m} \left[\frac{1}{similarity(Q_j, C_i)} \right]^2 \tag{4}$$

FCM iterates until a termination condition is reached which is $|NDM - CDM| \leq \varepsilon$ Where, NDM is the new degree of membership and CDM is the current degree of membership.

5.3 QoS PREDICTOR MODULE

In this module, a modified highly accurate prediction (MHAP) algorithm is proposed to get an accurately predicted value of the QoS values. The proposed MHAP algorithm uses the pre-phase of fuzzy clustering. Fuzzy clustering is used to cluster similar users in the same group and use similar users to predict the values of QoS. Combining Fuzzy clustering with the HAPA algorithm leads to having a more

accurate predicted value rather than values achieved by the HAPA algorithm. An evaluation of the proposed MHAP algorithm compared to HAPA is also discussed in a later section.

5.3.1 Modified HAPA (MHAP) Algorithm

As mentioned before, HAPA [8] is an algorithm designed to accurately find predicted values from QoS values. The HAP Algorithm consists of three main steps: prediction based on similar users, on similar services ,and the overall predicted value combined from previous steps. To enhance the prediction values, the papers suggest combining fuzzy clusters which are implemented in the previous to the HAP algorithm. Services that are invoked by a user may belong to many clusters. Therefore, a user may participate in the prediction values in many clusters. A cluster is characterized by a membership matrix and cluster center matrix. To make the prediction values more accurate, the membership of each user multiplied by the centers of each cluster is considered. The predicted value for user u is represented by equation (5) as follows:

$$pre_{su}r_{u,i} = \begin{cases} x_1 + \frac{\sum_{x=1}^{k} C_x . \mu_{su,x}}{d} & \text{if } |x_1 - rp_{su}(r_{u,i}))| < |x_2 - rp_{su}(r_{u,i}))| \\ x_2 + \frac{\sum_{x=1}^{k} C_x . \mu_{su,x}}{d} & \text{Else} \end{cases} \tag{5}$$

Where the predicted value of $r_{u,i}$ denoted by $pre_{su}r_{u,i}$for a cluster of d-dimensions. Then, it is generalized for all similar users in KU. Similarly, the predicted value based on the service i is:

$$pre_{su}r_{u,i} = \begin{cases} \frac{\sum_{x=1}^{k} C_x . \mu_{si,x}}{d} & \text{if } |x_1 - rp_{si}(r_{u,i}))| < |x_2 - rp_{si}(r_{u,i}))| \\ x_2 + \frac{\sum_{x=1}^{k} C_x . \mu_{su,x}}{d} & \text{Else} \end{cases} \tag{6}$$

The proposed MHAP algorithm guarantees considering the participation of a similar user and service in all clusters with different degrees of memberships. This leads to having a more accurate predicted value. The pseudo-code of the MHAP algorithm is shown in Algorithm 1.

5.4 SERVICE REDUCTION MODULE

This module is responsible for reducing the search space of candidate services to decrease composition time. Removing redundant and unstable services reduces the available services for the composition process. This module is implemented via two phases. First, stable service selector (SSS) represents the uncertainty filtering of services. Second, Skyline Services selector (SKSS) that applies the skyline operator to find the most interesting services.

Algorithm 1 MHAP Algorithm

Input User-Clusters
Output predicted-Value

procedure User Prediction Phase
    Find the Similarity between u and similar users using PCC.
    Apply linear regression to reduce solutions into one Rough prediction $r_{u,i}$.
    Find the predicted value based on roots x_1 and x_2 and incorporating the fuzzy cluster centers using equation 5.
    Use the ku users to predict the overall value.
end procedure
procedure Service Prediction Phase
    Find the Similarity between service i and similar service using PCC.
    Apply linear regression to reduce solutions into one Rough prediction $r_{u,i}$.
    Find the predicted value based on roots x_1 and x_2 and incorporating the fuzzy cluster centers using equation 6.
    Use the KS users to predict the overall value.
end procedure
procedure Overall Prediction
    Use the decomposition of the three models to find the overall predicted value.
end procedure

Table 3: UNCERTAIN DATA EXAMPLE

User	WS1	WS2	WS3
U1	13	24	15
U2	33	28	16
U3	17	25	32
U4	32	28	33
U5	16	22	18
U6	33	26	17
U7	30	28	22
U8	15	22	30
Average	23.625	25.375	22.875

5.4.1 STABLE SERVICE SELECTOR (SSS)

The SSS is a part of the service reduction module in the proposed CQPC framework. The process of selecting candidate services needs a lot of time depending on the size of the solution space. Hence, there is a vital need to reduce the solution space. The reduction in solution space affects the selection process then the composition process according to the execution time factor. The SSS employs some statistical concepts to keep only reliable and stable services.

Table 3 presents an example of three web services with QoS response time. Considering average values mean that WS3 is the best service as it has less response time. By a closer look at the values, we note that the range of values for WS3 is distributed over a large range. It means that the values for these services are unstable. Whereas WS2 has an average response time larger than WS1 and WS3. But the range of values is closed to each other than the other two services. We conclude that WS2 is more stable in its values rather than the two other services. A vector of Entropy and correlation variation determines the degree of disordering and deviations of the values.

Correlation variation (CV) and entropy would help in filtering the services according to their uncertainty. Hence, less reliable services would be filtered. Entropy refers to the level of disordering and average uncertainty that exists in the ser-

vice. As the entropy measure increases as the uncertainty that exists in a service decreases.

To employ and calculate entropy theory, we consider the historical values of services for the real-world data to be the values to calculate the entropy using the following equation:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i) \tag{7}$$

Where X is a random variable with the range $\{X_1, X_2, X_3, ..., X_n\}$. Whereas the correlation variation refers to the degree of variations that occurred which is considered as a measure of the stability of service. Having a large correlation variation value refers to more uncertainty and instability in services. To apply the coefficient variation measure, the following equations are used:

$$CV = \frac{SD}{\bar{X}} \times 100\% \tag{8}$$

$$\bar{X} = \frac{\sum_{i-1}^{n} x_i}{n} \tag{9}$$

$$SD = \sqrt{\frac{\sum x_i - \bar{x}}{n-1}} \tag{10}$$

Such that$\bar{X}$ represents the mathematical average for random X variable within the range X_1, X_2, X_3, X_n. Both Correlation variation and entropy measures are used to filter and eliminate the less reliable services that would slow down the execution time. Besides, using both measures (not only one of them) is essential to get accurate results. Since there are some situations in which services have an equal entropy. So, CV is used to differentiate which one is more reliable than the other. As a conclusion, the process of filtering services is described as follows: firstly, entropy is applied for the service set of candidate services, R. The result of entropy is the smallest list in entropy R1 contained in R. Then, CV is used to filter list $R1$ further into a more reduced and reliable, list $R2$. This refers to the most reliable service available among candidate services. So, the result obtained by this module would reduce the search space to be used for the next module.

5.4.2 Skyline Services selector (SKSS)

After filtering solution space with entropy and CV based on uncertainty, solution space is still large and has a slow execution time. By using the skyline service submodule, solution space can be reduced to have faster execution time by keeping only reliable and most interesting services. It selects services that are not dominated by other services in terms of QoS aspects. These aspects such as response time, price,

reliability, and ... etc. Applying skyline operator separates these services that are potential from those that are not potential for the composition process. In other words, the best combination for service composition to fulfill global constraints is the one that depends on service S_i that is considered from the skyline services. Hence, it reduces a large number of redundant services and speeds up the process of service selection. This leads to improving the final optimal solution in the composite service.

The paper suggests using the Block Nested Loop skyline (BNL) algorithm for calculating skyline services [29]. To extract skyline points, BNL compares all services in terms of their QoS with each other (two-by-two). Then, it returns the skyline services that are not dominated by any other services. BNL algorithm is applied to clustered data in each map function. So, BNL is implemented for each Map function to extract skyline services in it.

Additionally, a method for determining a set of the most interesting services indicated as interesting services that best define the skyline services is investigated. The extracted skyline services might be large and consume more computations and time. Therefore, we need to identify the top-K best interesting skyline services among available skyline services. The challenge is, how to select the best interesting skyline services that provide the trade-off between different QoS available while satisfying the user-defined constraints. Also, interesting services should be small enough for computations to satisfy user requirements. To obtain top-K interesting skyline services, The cluster center is determined by the average of points or services contained. After determining a center for each group of skyline services, the top K of skyline services is chosen based on PCC distance measure. For each skyline service S in a group with a center CL, top K is chosen such that minimizing $PCC(S_i, CL_i)$.

5.5 SERVICE COMPOSER and EVALUATOR MODULE

This module is responsible for finding the best service composition that fulfills the user needs by selecting the most appropriate services from selected clusters. The composition process is performed in such a way to obtain accurate near-optimal solutions. To implement the service composer module, we propose a new hybrid bio-inspired algorithm MR-FPSO. It merges between PSO and FOA. Additionally, it is integrated with the MapReduce framework. The standard PSO algorithm suffers from falling into local optimum solutions, which results in low-quality results far from the optimal solution. To address this problem, a proposed method with modified PSO is introduced to enhance the features of PSO. The proposed algorithm MR-FPSO is implemented in the MapReduce environment to handle the huge amount of data in the cloud environment.

To improve the PSO algorithm, the fruit fly algorithm and newly updated velocity mechanism are used and integrated into PSO. This improves the position of the particle, solution, toward the location of the new solution. We utilize the advantages of FOA such as simplicity and speed. Also, the smell operator of FOA is used to update the position based on the available information for fruit flies for smell sensation.

5.5.1 FITNESS FUNCTION

Each solution is represented as a particle. We use a fitness function to evaluate the performance of each solution that represents a composite service. The main objective is to select the services that best match the user requirements among concrete services. In other words, maximizing the overall aggregated QoS value of the composite service is achieved by maximizing the fitness function provided by equation 11.

$$f_{fitness}(C_n) = \sum_{r=1}^{z} W_r q_{nrm} \tag{11}$$

5.5.2 Composition strategy for solution encoding

The paper assumes a strategy for coding using a fixed integer array. Each task is represented by a set of abstract services represented by an array of integers. Each abstract service is represented by a set of concrete services. Each array item represents the candidate service number. As shown in figure 5. The fitness function is represented by the weighted sum of the QoS parameters of the newly added service. In equation 11, the variables W_r indicates the weights of the r^{th} QoS attributes of the required composite service C_n and q_{nrm}refers to the normalized value of the r^{th} QoS attributes of the composite service.

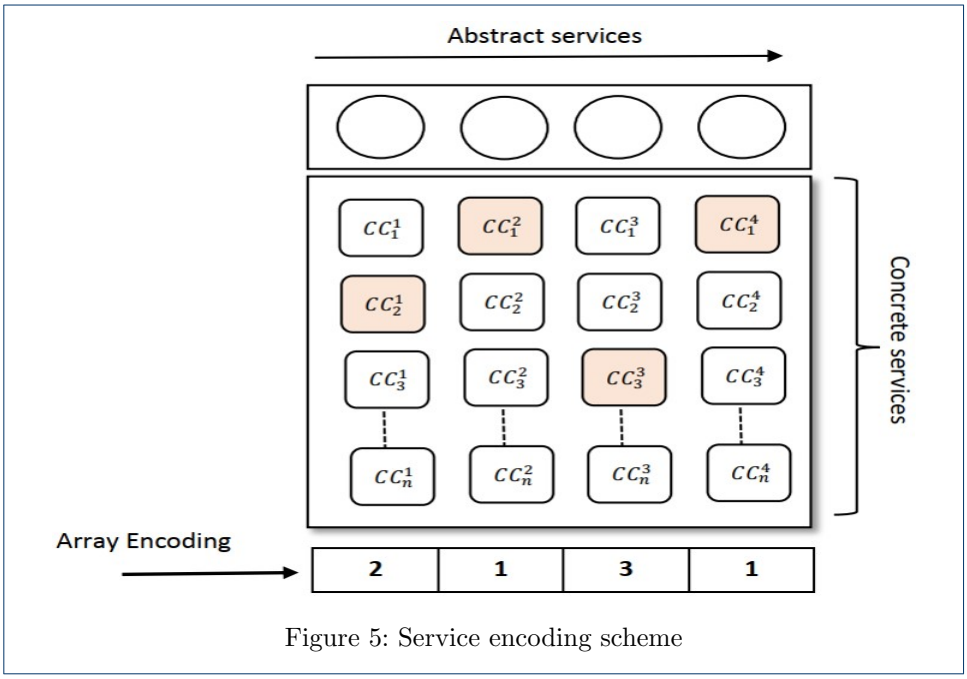

Figure 5: Service encoding scheme

5.5.3 Proposed service composition Algorithm (MR-FPSO):

The main objective of the proposed algorithm is handling large-scale data of services to create a composite service with high accuracy in an acceptable amount of time. It provides a way of parallelism. MapReduce structure is used with its two main functions Map() and Reduce(). The proposed algorithm consists of two main steps:

the preparation step, and the parallelism step. Each step is described in more detail in the following sub-sections. A summary of the overall algorithm is presented in Algorithm 2 and 3 .

Preparation step

In the first step, an initial population, i.e. swarm, is created with N particles represent some possible solutions from the solution space. For each particle in the initial population, an initial random position x_i is initialized. Then, the fitness function is calculated for each particle to evaluate such particles. The global best position is initially generated. For each particle in the initial population, a record of $< key, value >$ pair is generated for further implementation in the MapReduce structure. For every particle, the key is denoted by a constant integer number ,and the value represents information about the particle v_i, x_i, P_best, G_best, and $fitness$. $x_i = x_{i,1}, x_{i,2}, x_{i,2}, \ldots.., x_{i,n}$,$V_i = V_{i,1}, V_{i,2}, V_{i,3}, \ldots.., V_{i,n}$, and $F(*)$ refers to the fitness function evaluation. The particles are then stored in files as initial input for MapReduce.

Parallelism Step

Service data are partitioned into data blocks of disjoint subsets according to partitioning criteria. Each data block is dispatched for a Map node for parallel processing. The goal of the process of partitioning the data into blocks is efficiently fitting the partitions into memory. The workload can be distributed over many servers to increase performance. Therefore, any query is executed on each partition in a parallel way. Finally, the results are merged to get the global result. Data partitioning is employed by many schemes of partitioning such as grid-portioning [30], angular partitioning [31] and ...etc. The angular partitioning approach is more effective compared to the other approaches. It reduces the required time and fairly shares the offered workload [32].

The second step is an iterative step executed as a job in the MapReduce. At the end of each MapReduce iteration, a new swarm of updated particles is created. The new swarm replaces the previous swarm of solutions. The modifications done in each swarm is based on the functions implemented in both Map and Reduce functions.

The initial population of PSO is divided into SN subpopulations where each subpopulation is an input to the Map function. The output of each MapReduce iteration is an input to the next MapReduce iteration.

Map( ) function: After the master divides the population into subpopulations, each subpopulation is processed independently on each Map node. Each subpopulation information is updated on each Map. Algorithm 2 describes the applied steps at the Map function. There are two proposed steps to improve the PSO performance and avoid falling into the local optimum. First, newly updated velocity calculation formulas are used [33]. Second, the distance and smell operator from the FOA optimization algorithm is used to enhance the position of the particle to avoid the local optimum [34]. For the first step of newly updated formulas, it is based mainly on two cases.

Firstly, if the particle is not the one with the best position, the position of the particle is updated by an experienced best position, Pbest, and other better particles

instead of Gbest. This can be achieved by comparing the fitness value of X_i with other particles. The particles with a fitness value better than x_i are chosen in a $setBest$. One of the particles in setBest is chosen as the guide for updating X_i based on a random number T. The particle X_i with a fitness value greater than T is chosen as a parameter for updating the particle x_i using the following equations:

$$fit(x) = \begin{cases} \frac{1}{1+f(x)} & \text{if } f(x) \geq 0 \\ 1 + abs(f(x)) & \text{,else} \end{cases} \tag{12}$$

$$v_i^d(t+1) = wv_i^d(t) + c_1 r_1(pbest_i^d - x_i^d(t)) + c_2 r_2(x_j^d(t) - x_i^d(t)) \tag{13}$$

$$w = w_{max} - \frac{w_{max} - w_{min}}{itmax} iter \tag{14}$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \tag{15}$$

The second case for updating xi when the particle refers to the best position. In this case, opposition learning is applied to more widely search the solution space. To guarantee more diversity of the population a convex combination between x_i and another random particle in the population x_k. Convex combination searches the space as much as possible to let the optimal solution be found. The process of convex combination and opposition learning is provided using the following equations:

$$\hat{x_t} = (1-\lambda)_{x_i(t)} + \lambda x_k(t) \tag{16}$$

$$x_i^{new} = x_{min} + x_{max} - \hat{x_l} \tag{17}$$

The position of the particle is updated in both cases. Afterward, distance and smell operators of the FOA algorithm are applied to improve the position toward the optimal solution in short movements. The resulting position is updated using the following equations:

$$Dist_i(t+1) = \frac{1}{x_i^d(t+1)} \tag{18}$$

$$Smell_i(t+1) = \frac{1}{Dist_i(t+1)} \tag{19}$$

Then, Pbest is compared with the updated value of xi to determine whether the Pbest is to be replaced. For each subpopulation at each Map task, there is a local best position that represents the global best position for the subpopulation. The local best position for all subpopulations are merged further in the Reduce function to find the global best position for the population. The integer key and updated value of particle information are submitted at the end of the Map task.

Reduce() function:
In the Reduce function, all local best position particles are merged to determine the global best particle, which represents the best solution at the end of the iteration. At the end of the Reduce function, a new swarm with updated particles is emitted. Afterward, it is used as an input to the next iteration of the MapReduce job. The steps applied at the Reduce function is shown in Algorithm 3.

Algorithm 2 MR-FPSO(MAP function)

Input candidate services.
Output (int,$LPbestList$)

The initial population of N particles.
for each particle in N **do**
    Initialize random position x_i.
    Calculate fitness value fi for x_i.
end for
Initial $Gbest = max(f_i)$
// Split population into SN subpopulations
// Map function
Calculate fitness value for all candidates in subpopulation using Eq.10.
for i =1 to SN **do**
    Determine $setBest$.
    Calculate fitness value fi for x_i.
    if the best solution is NOT x_i **then**
        Calculate velocity v_i.
        Calculate inertial weight.
        Calculate x_i.
        Update x_i by smell concentration.
    else
        Calculate x_i.
        Update x_i by smell concentration.
    end if
    Update particle($key, value$) information, x_i, v_i, fitness(x_i), fitness.
    if fitness(x_i) ¿ fitness($Pbest_i$) **then**
        $Pbest_i$= x_i .
    end if
    if fitness($Pbest_i$) ¿ fitness($LPbest_i$) **then**
        $LPbest_i$=$Pbest_i$.
    end if
end for

Algorithm 3 MR-FPSO (Reduce function))

Input $int, LPbestList$
Output (composite service)

//Reduce function.
for each $LPbest$ in $LPbestList$ do **do**
    if fitness($LPbest$)¿fitness($Gbest$) **then**
        $Gbest$=$LPbest$.
    end if
end for
for each particle in $LPbestList$ do **do**
    Update particle ($key, value$) information; x_i, v_i, fitness.
end for

6 Experimental Results Discussion and Analysis

This section presents the experiments employed for proving the validity of the proposed CQPC framework, the evaluation of the results, and the different evaluation metrics such as fitness and execution time. The related discussion will be presented in the next subsections.

6.1 REQUIREMENTS FOR EXPERIMENTS AND PARAMETER CONFIGURATION

The experiments are performed using the parallel principle of Hadoop 2.8.5 on a Hadoop cluster. There are five nodes , one for the master and the other nodes for the data. To implement the MapReduce concept, a reduction function with four maps is used. Windows 7 with Intel Core I 7, 2,40 GHz and 8 GB is used . The effectiveness and feasibility of the proposed approach is compared to other techniques, such as MR-GA [13], MR-IDPSO [14], MRNSGA-II [9], and MRPSO [12] for the ability to handle huge amounts of data. The WSDream [35] dataset is used for MR-FPSO simulations. It comprises 5825 concrete services invoked by approximately 339 users, with a response time and throughput of two QoS parameters reflecting services invoked by different users. Five abstract services are used to manage the vast amount of data. Every one has approximately 10,000 concrete services. The mutation and the crossover are set to 0.7 and 0.21 respectively for MR-GA.

6.2 Results and Evaluation

6.2.1 Impact of The Number oF Iterations

As shown in figure 6, the average fitness value increases as the number of iterations increase with an acceptable rate rather than other techniques. This proves the feasibility of MR-FPSO to find the optimal solutions. The reason for that is using skyline operators and reliable measures such as entropy and correlation variation to reduce the solution space to have more reliable services. Another reason is the improvements in the position of particles in MapReduce jobs.

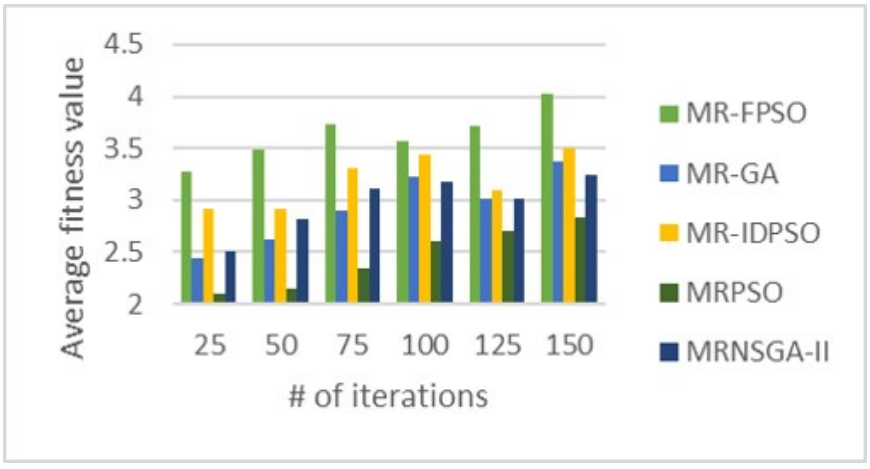

Figure 6: Feasibility evaluation with several iterations

Figure 7 shows that MR-FPSO achieves less execution time for the suggested number of iterations rather than the compared techniques.

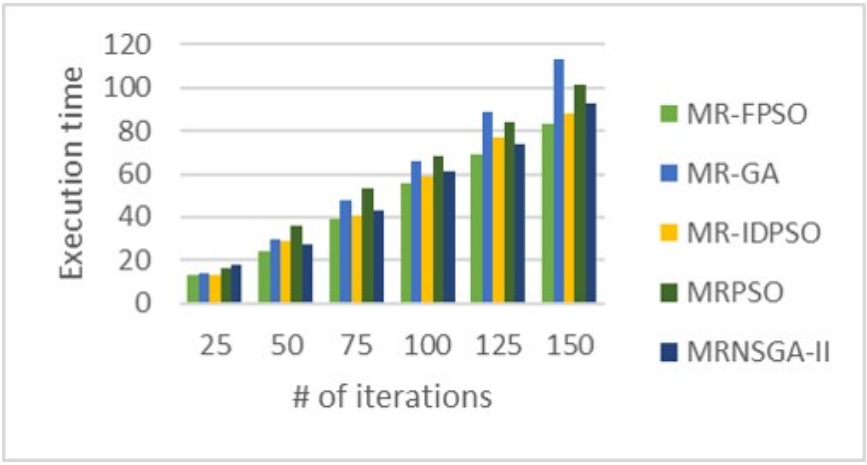

Figure 7: Execution time evaluation with the number of iterations

6.2.2 IMPACT OF NUMBER OF CONCRETE SERVICES AVAILABLE

Figure 8 and Figure 9 show that MR-FPSO outperforms other state-of-the-art techniques. The fitness value increases with the increase in the number of concrete services. Since using the service reduction module helps in reducing the candidate services hence achieving less execution time. The increase in the number of concrete services refers to increasing the number of available services to participate in the initial population. Therefore, the probability of increasing fitness values increases.

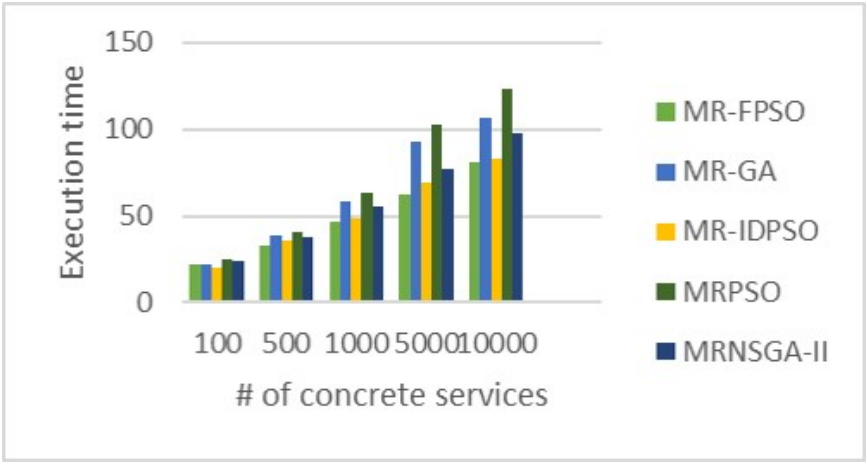

Figure 8: Execution time with the number of concrete services

Figure 10 shows that the impact of increasing the used population size over the fitness value. The more increase in population size leads to more concrete services to participate in the solution. As a result, the number of reliable services toward the optimal solution is increased. More population size results in more reliable services with good quality to be used in the context of the composition process.

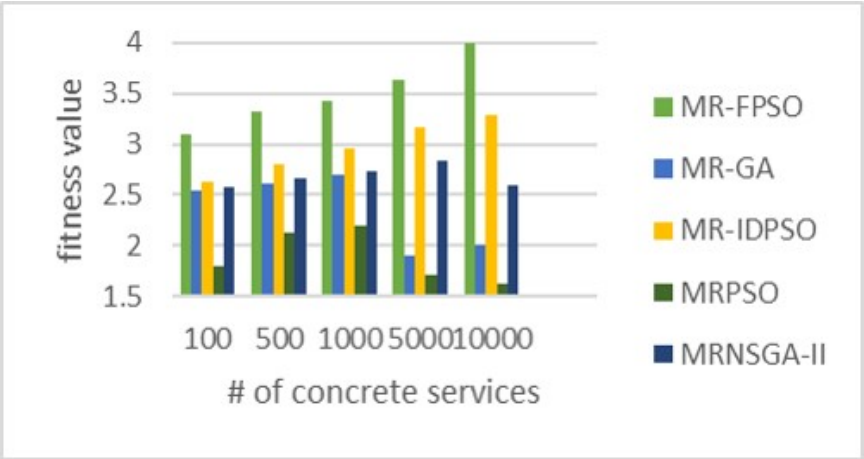

Figure 9: Fitness value with the number of concrete services

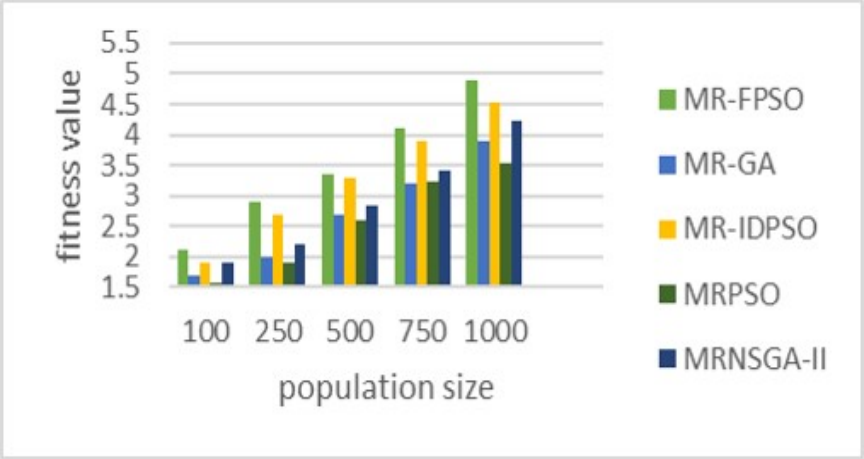

Figure 10: Fitness value with the number of population size

Figure11 shows the impact of the increase in the number of clusters for FCM clustering over the accuracy of prediction. As a result of embedding and integration of membership function matrix and cluster centers into the HAPA algorithm, more accurate prediction values are achieved. The accurate prediction results are measured using the accuracy metrics root mean squared error (RMSE) and mean absolute error (MAE) according to equations 18 and 19.

$$MAE = \frac{\sum \left| Q_{ij} - \hat{Q_{ij}} \right|}{N} \tag{20}$$

$$RMSE = \sqrt{\frac{\sum \left(Q_{ij} - \hat{Q_{ij}} \right)^2}{N}} \tag{21}$$

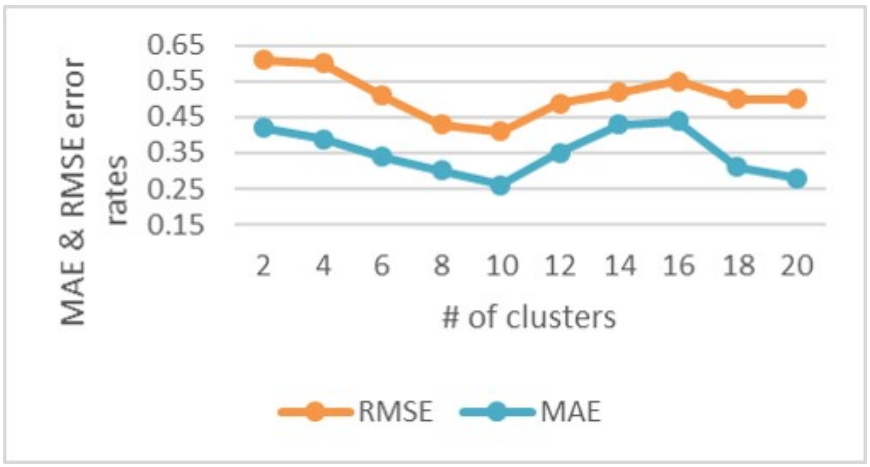

Figure 11: Number of clusters with error rates

Where Q_{ij} denotes the observed QoS attribute value by the user.While $\hat{Q}_{ij}$ denotes the predicted value of QoS and the number of values already predicted referred by N. with a threshold value of $FCM = 0.01$. The prediction error decreases when the number of clusters increases until a certain point, the prediction errors increase to an adequate level. This means that the number of clusters is determined in the experiments according to the nature of patterns to be clustered.

6.2.3 The Performance of Proposed MHAP Algorithm Against HAPA

The comparison between the MHAP algorithm and HAPA is shown in figure 12. The results show that with the increasing number of top similar users, the error rate MAE decreased. On the other hand, the increasing number of similar users over a certain threshold due to the excessive increase in the number of used similar users would produce high error rates. In all cases, the MHAP algorithm provides a less error rate rather than HAPA. The reason for that is combining the fuzzy clustering with standard HAPA. This allows similar users to be located with each other in the same cluster. This provides more accuracy rather than the HAPA algorithm.

7 Conclusion and Future Work

With the rapid proliferation of web services over the cloud environment, the service composition process involves several issues and challenges. This paper proposed a novel Cloud-based QoS-Provisioning Service Composition (CQPC) framework to address service composition. Besides, the HBIQP technique is presented to prove the concept of the proposed framework and its applicability. Also, the MHAP algorithm that integrates the HAPA algorithm over the fuzzy clustering for accurate predictions is presented. Furthermore, Skyline operator is used over the MapReduce platform and information theory concepts to reduce the solution domain. Moreover, a proposed MR-FPSO algorithm is introduced to improve the global end-to-end solution. It provides parallel service composition efficiently and uses subjective trimming operators to provide diversity for the next generations of solutions. Simulation results showed that the proposed technique efficiently addresses the challenges of

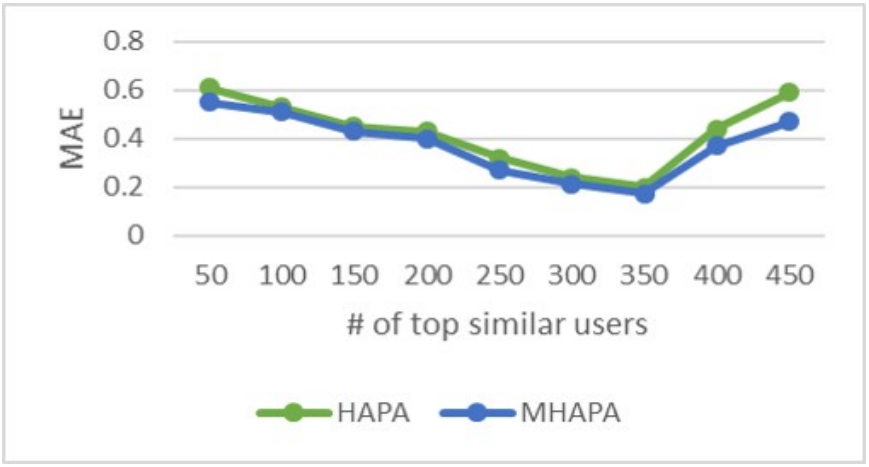

Figure 12: HAPA Vs. MHAPA

service composition over the cloud computing environment rather than compared techniques.

In the future, further enhancements for the efficiency of the proposed technique will be conducted. More efficient optimization for the service composition will be introduced considering the dynamic environment of service composition in large-scale data.

Abbreviations

Internet of Things (IoT) - Sensing as a service (SaS) - Quality of Service (QoS) - Cloud-based QoS provisioning Service composition (CQPC) framework - A Hybrid Bio-Inspired QoS- provisioning (HBIQP) technique - A modified highly accurate prediction (MHAP) algorithm - MapReduce fruit fly particle swarm optimization (MR-FPSO) algorithm - Map-Reduce Non-Dominated Sorting Genetic Algorithm (MRNSGA-II) - MapReduce Particle Swarm Optimization(MRPSO) - MapReduce (MR) - A Performance-Oriented Integration Design (POID) framework- Particle swarm optimization (PSO) Fruit fly (FOA) algorithm - Clusters Generator (CG) Service clustering (SC)- The User Clustering (UC)- The requests clustering (RC) - QoS predictor (QP)- Service Selector (SS) - Service reduction (SR) - Statistical-based Reduction (SR) - Domination-based Reduction (DR)- Service composer and evaluator (SCE)- The Fuzzy C-Means clustering (FCM) algorithm - The Pearson Correlation Coefficient (PCC) algorithm- Stable service selector (SSS) - Skyline Services selector (SKSS) Correlation variation (CV)- Block Nested Loop skyline (BNL) algorithm.

Acknowledgments

Not applicable.

Funding

Not applicable.

Ethics approval and consent to participate

Text for this section. . .

Competing interests

The authors declare that they have no competing interests.

Authors' contributions

W.B.,M.S, A.A, and E.D jointly came up with the concept of QoS-driven service composition. M.S developed the simulation model and, together with W.B, developed the proof-of-concept implementation. All authors participated in the analysis of the experiment results. W.B., M.B and E.D drafted most of the manuscript. A.A and W.B revised the manuscript in several interactions. All authors read and approved the final manuscript.

Author details

[1]Information Technology Department, Faculty of Computer and Information Sciences, Mansoura University., Mansoura, Egypt. [2]Department of computer Eng. and control systems, Faculty of Engineering,Mansoura University, Mansoura, Egypt.

References

1. Farsi, M., Badawy, M., Abdelmoneim, N., Ali, H.A., Abdulazeem, Y.: Qos-aware framework for performance enhancement of soa in enterprise it environments. IEEE Access **7**, 107699–107715 (2019)
2. Badawy, M.M., Ali, Z.H., Ali, H.A.: Qos provisioning framework for service-oriented internet of things (iot). Cluster Computing, 1–17 (2019)
3. Alrifai, M., Risse, T.: Combining global optimization with local selection for efficient qos-aware service composition. In: Proceedings of the 18th International Conference on World Wide Web, pp. 881–890 (2009). ACM
4. Lee, C., Wang, C., Kim, E., Helal, S.: Blueprint flow: a declarative service composition framework for cloud applications. IEEE Access **5**, 17634–17643 (2017)
5. Canfora, G., Di Penta, M., Esposito, R., Villani, M.L.: An approach for qos-aware service composition based on genetic algorithms. In: Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation, pp. 1069–1075 (2005). ACM
6. ur Rehman, Z., Hussain, O.K., Hussain, F.K.: Parallel cloud service selection and ranking based on qos history. International Journal of Parallel Programming **42**(5), 820–852 (2014)
7. Hossain, M.S., Hassan, M.M., Al Qurishi, M., Alghamdi, A.: Resource allocation for service composition in cloud-based video surveillance platform. In: 2012 IEEE International Conference on Multimedia and Expo Workshops, pp. 408–412 (2012). IEEE
8. Jin, H., Yao, X., Chen, Y.: Correlation-aware qos modeling and manufacturing cloud service composition. Journal of Intelligent Manufacturing **28**(8), 1947–1960 (2017)
9. Salam, M.A., Bahgat, W.M., El-Daydamony, E., Atwan, A.: A novel framework for web service composition. International Journal of Simulation–Systems, Science & Technology **20**(3) (2019)
10. Seghir, F., Khababa, A.: A hybrid approach using genetic and fruit fly optimization algorithms for qos-aware cloud service composition. Journal of Intelligent Manufacturing **29**(8), 1773–1792 (2018)
11. Chen, Y., Huang, J., Lin, C., Hu, J.: A partial selection methodology for efficient qos-aware service composition. IEEE Transactions on Services Computing **8**(3), 384–397 (2014)
12. McNabb, A.W., Monson, C.K., Seppi, K.D.: Mrpso: Mapreduce particle swarm optimization. In: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, pp. 177–177 (2007). ACM
13. Khalid, N.E.A., Fadzil, A.F.A., Manaf, M.: Adapting mapreduce framework for genetic algorithm with large population. In: 2013 IEEE Conference on Systems, Process & Control (ICSPC), pp. 36–41 (2013). IEEE
14. Zhang, Y., Jing, Z., Zhang, Y.: Mr-idpso: a novel algorithm for large-scale dynamic service composition. Tsinghua Science and Technology **20**(6), 602–612 (2015)
15. Hossain, M.S., Moniruzzaman, M., Muhammad, G., Ghoneim, A., Alamri, A.: Big data-driven service composition using parallel clustered particle swarm optimization in mobile environment. IEEE Transactions on Services Computing **9**(5), 806–817 (2016)
16. Yong, Z., Wei, L., Junzhou, L., Xiao, Z.: A novel two-phase approach for qos-aware service composition based on history records. In: 2012 Fifth IEEE International Conference on Service-Oriented Computing and Applications (SOCA), pp. 1–8 (2012). IEEE
17. Zhang, Y., Cui, G., Wang, Y., Guo, X., Zhao, S.: An optimization algorithm for service composition based on an improved foa. Tsinghua Science and Technology **20**(1), 90–99 (2015)
18. Ma, Y., Wang, S., Hung, P.C., Hsu, C.-H., Sun, Q., Yang, F.: A highly accurate prediction algorithm for unknown web service qos values. IEEE Transactions on Services Computing **9**(4), 511–523 (2015)
19. Li, B.: Efficiency optimization for communication service based on qos technology. IEEE Access **7**, 48838–48848 (2019)
20. Hasnain, M., Pasha, M.F., Ghani, I., Mehboob, B., Imran, M., Ali, A.: Benchmark dataset selection of web services technologies: A factor analysis. IEEE Access **8**, 53649–53665 (2020)
21. Hu, C., Wu, X., Li, B.: A framework for trustworthy web service composition and optimization. IEEE Access **8**, 73508–73522 (2020)
22. Peng, S., Wang, H., Yu, Q.: Multi-clusters adaptive brain storm optimization algorithm for qos-aware service composition. IEEE Access **8**, 48822–48835 (2020)
23. Huang, J., Liang, J., Ali, S.: A simulation-based optimization approach for reliability-aware service composition in edge computing. IEEE Access **8**, 50355–50366 (2020)
24. Kennedy, J.: Particle swarm optimization. Encyclopedia of machine learning, 760–766 (2010)
25. Xing, B., Gao, W.-J.: Fruit fly optimization algorithm. In: Innovative Computational Intelligence: A Rough Guide to 134 Clever Algorithms, pp. 167–170. Springer, ??? (2014)
26. Sha, M.M., Manesh, T., Ahmed, A.M.M.: A framework for evaluating the qos and cost of web services based on its functional performance. World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering **10**(1), 52–56 (2015)
27. Al-Helal, H., Gamble, R.: Introducing replaceability into web service composition. IEEE Transactions on services computing **7**(2), 198–209 (2013)
28. Ye, Z., Zhou, X., Bouguettaya, A.: Genetic algorithm based qos-aware service compositions in cloud computing. In: International Conference on Database Systems for Advanced Applications, pp. 321–334 (2011). Springer
29. Stephan, B., Donald, K., Konrad, S.: The skyline operator. In: Proc. ICDE, vol. 1 (2001)
30. Deng, K., Zhou, X., Tao, H.: Multi-source skyline query processing in road networks. In: 2007 IEEE 23rd International Conference on Data Engineering, pp. 796–805 (2007). IEEE
31. Vlachou, A., Doulkeridis, C., Kotidis, Y.: Angle-based space partitioning for efficient parallel skyline computation. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pp. 227–238 (2008). ACM
32. Doulkeridis, C., NØrvåg, K.: A survey of large-scale analytical query processing in mapreduce. The VLDB Journal—The International Journal on Very Large Data Bases **23**(3), 355–380 (2014)
33. Wang, C., Song, W.: A modified particle swarm optimization algorithm based on velocity updating mechanism.

Ain Shams Engineering Journal (2019)
34. Shefu, U., Ali Safdar, G., Epiphaniou, G.: Fruit fly optimization algorithm for network-aware web service composition in the cloud (2017)
35. QoS datasets for Web service recommendation. https://github.com/wsdream/wsdream-dataset. Accessed: 2019-04-30

Figures

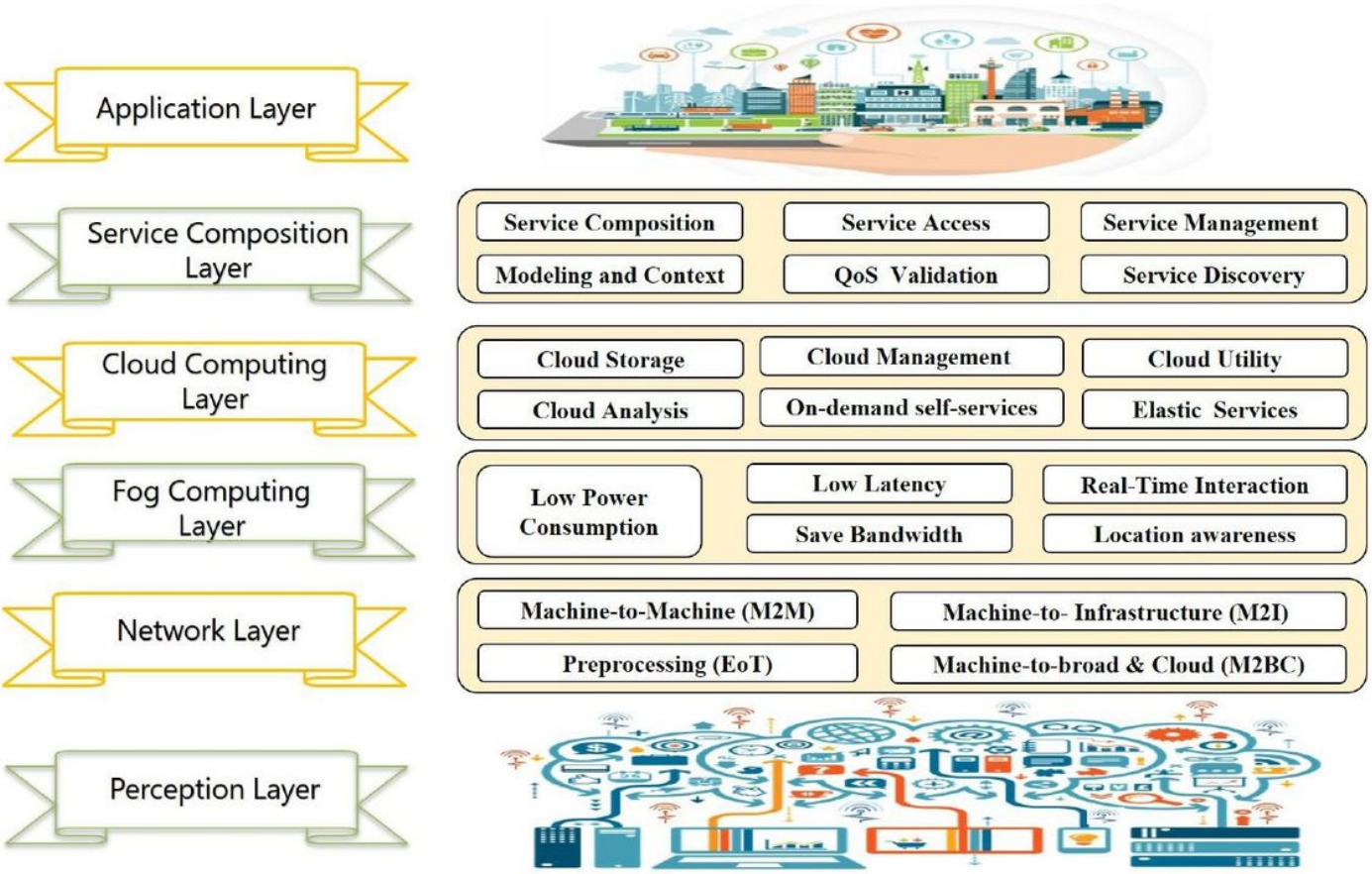

Figure 1

Service composition and cloud architecture

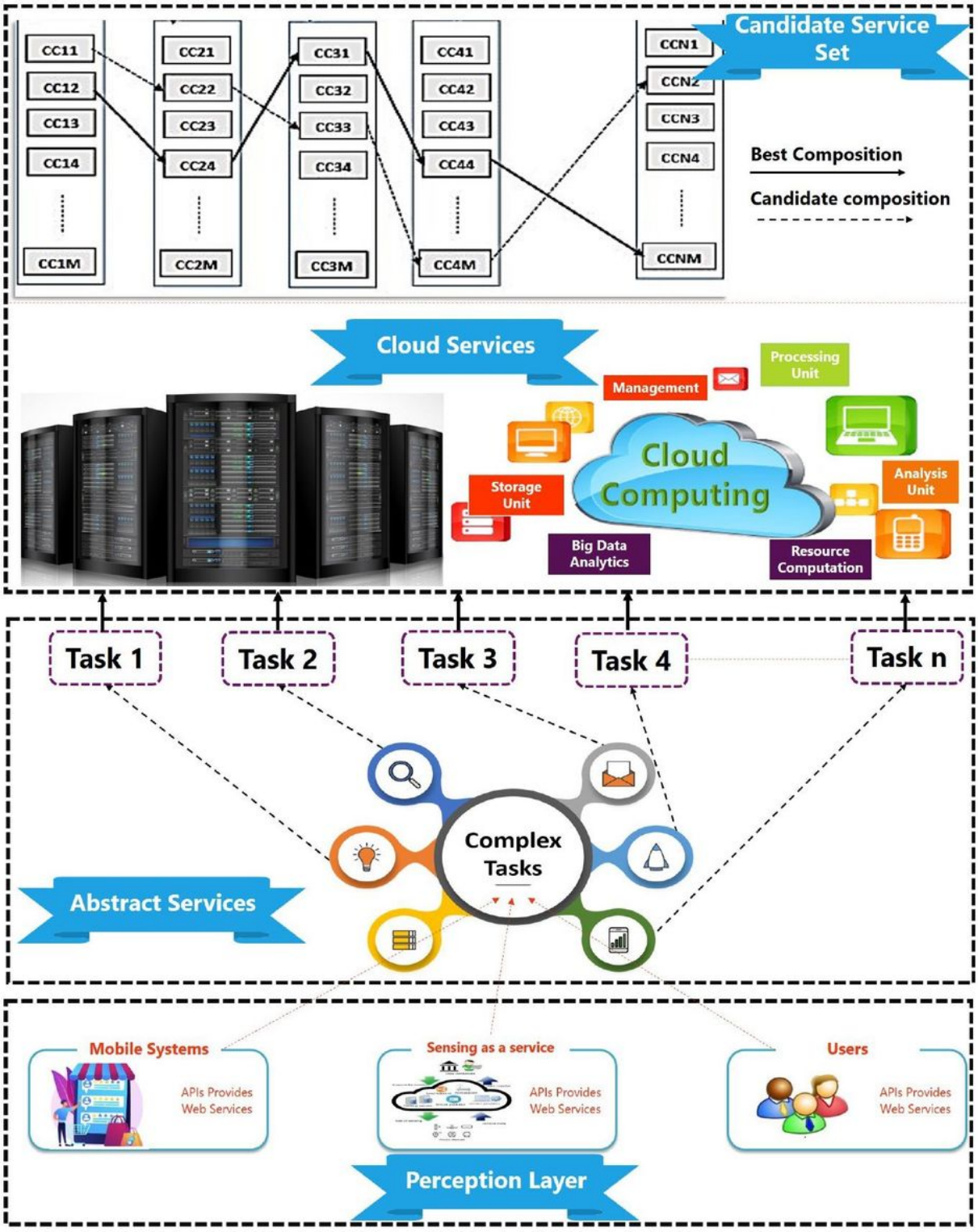

Figure 2

Composition process in the cloud

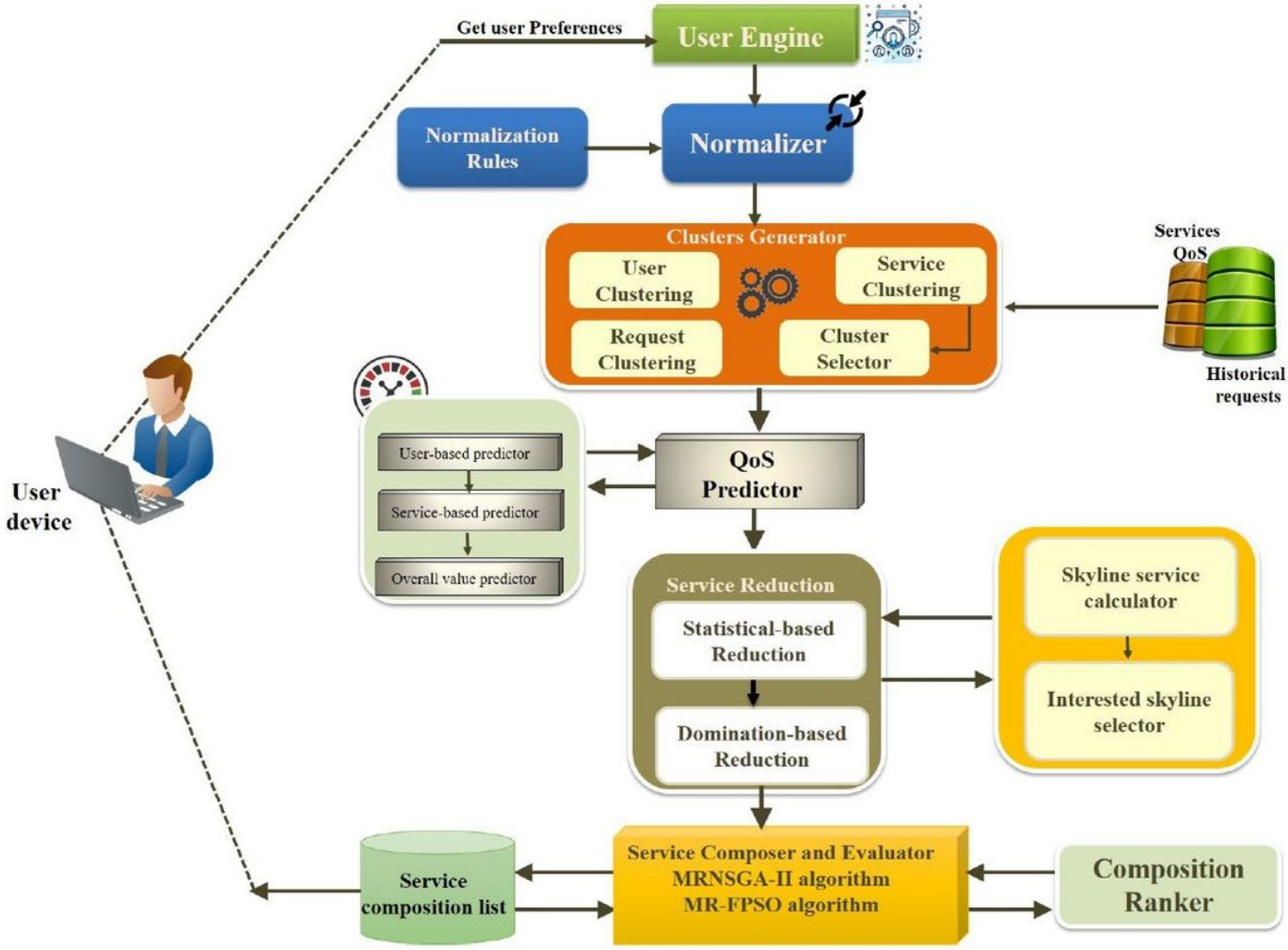

Figure 3

Hierarchal design of the CQPC proposed framework

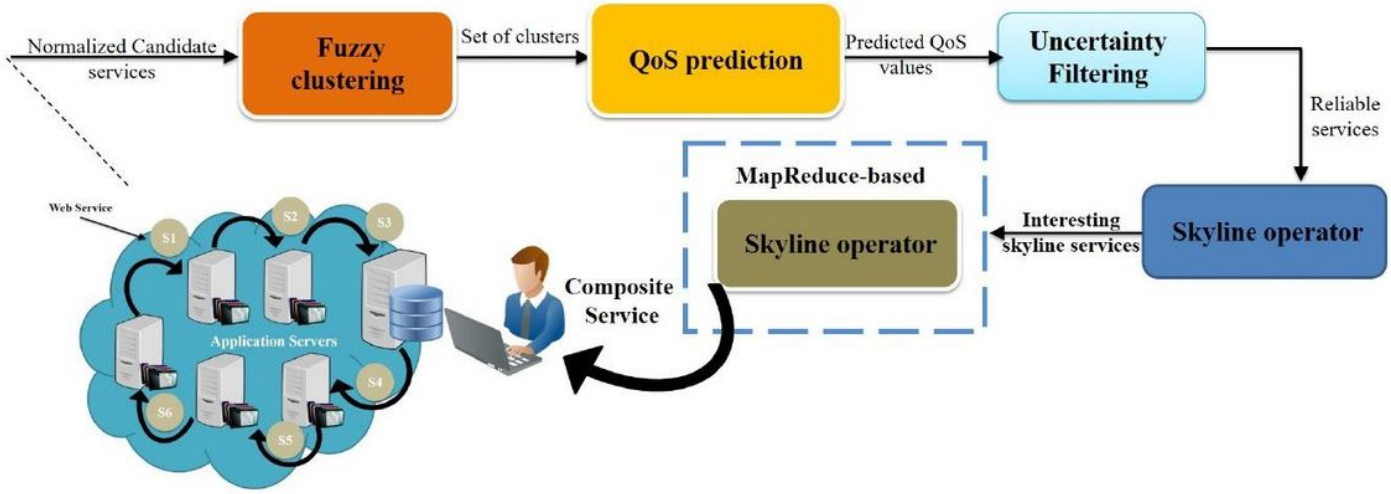

Figure 4

Cloud-vision data flow mechanism

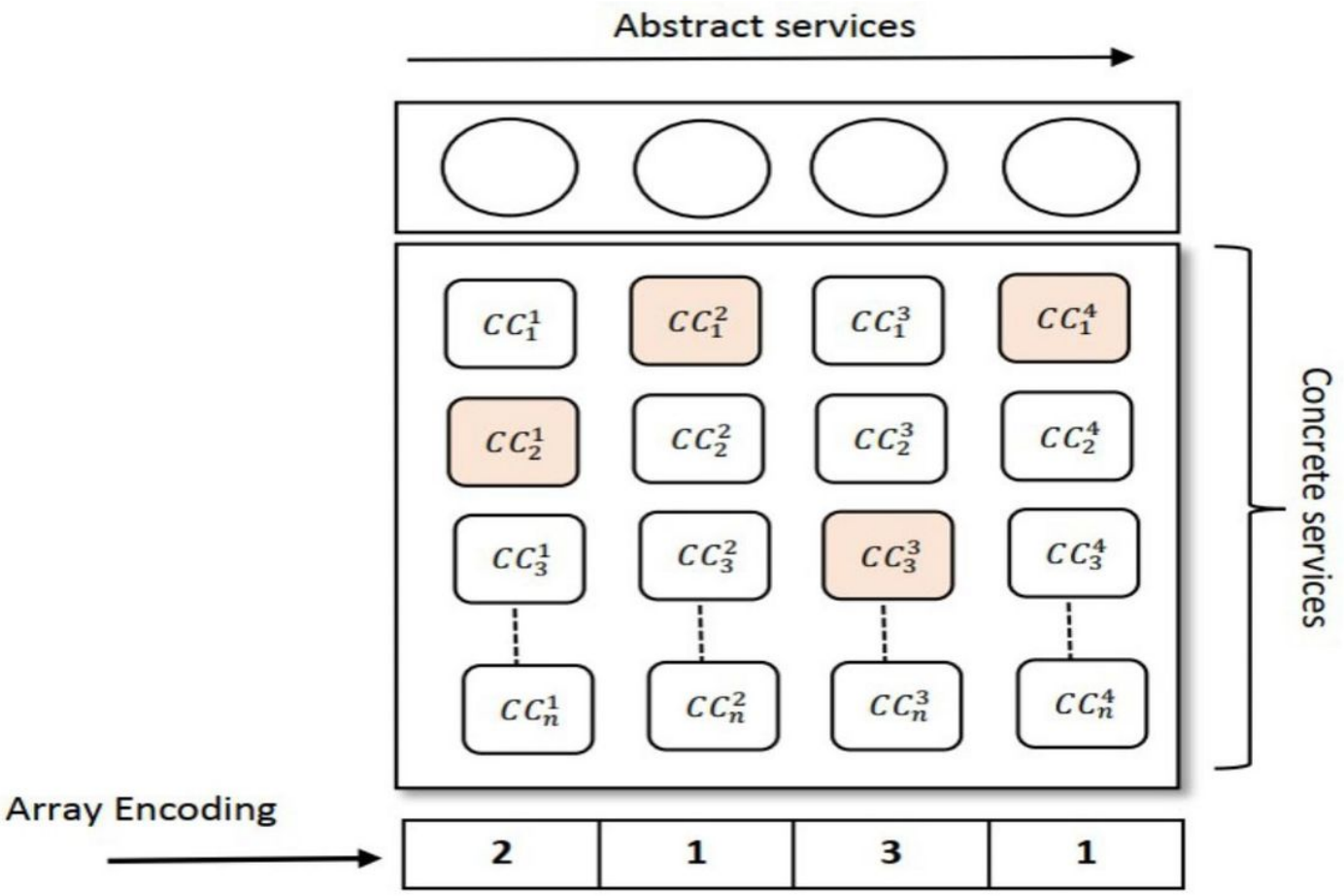

Figure 5

Service encoding scheme

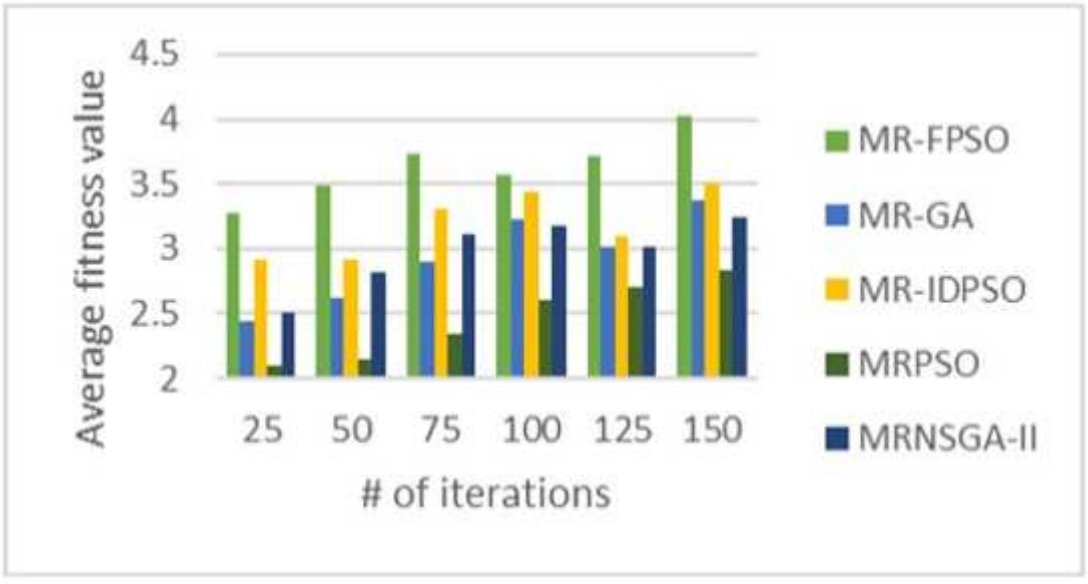

Figure 6

Feasibility evaluation with several iterations

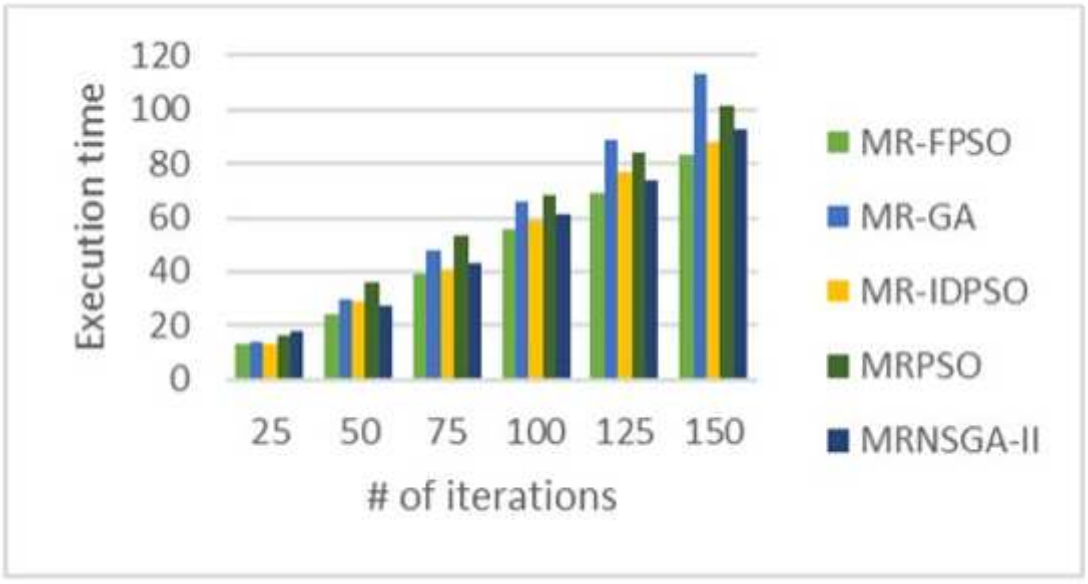

Figure 7

Execution time evaluation with the number of iterations

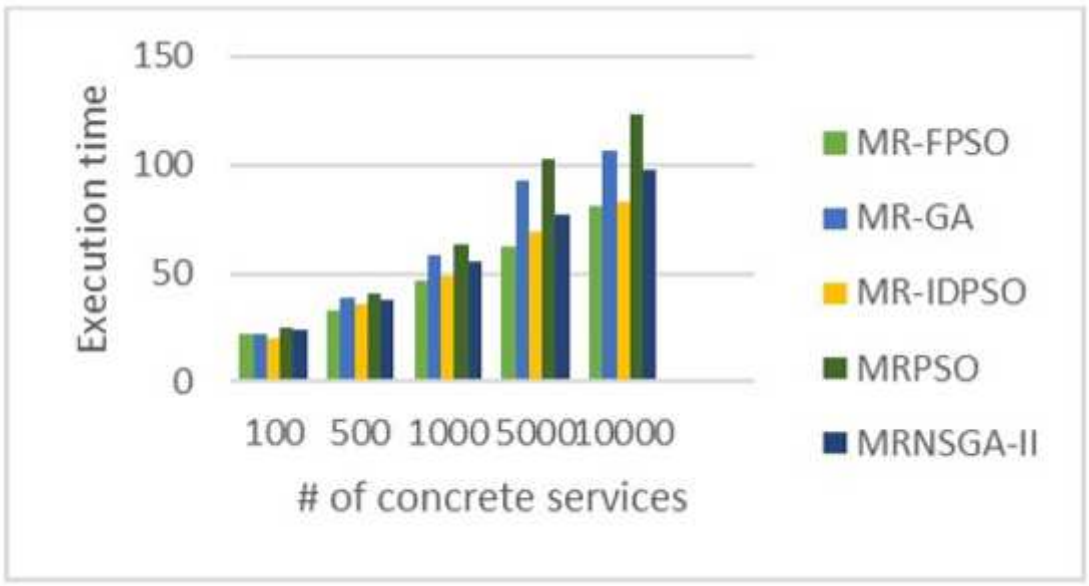

Figure 8

Execution time with the number of concrete services

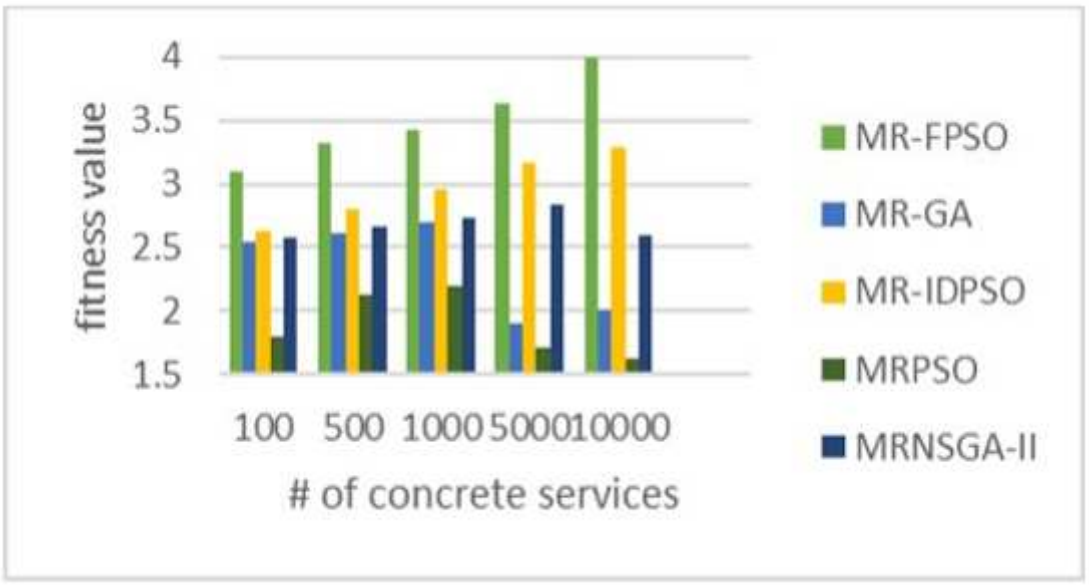

Figure 9

Fitness value with the number of concrete services

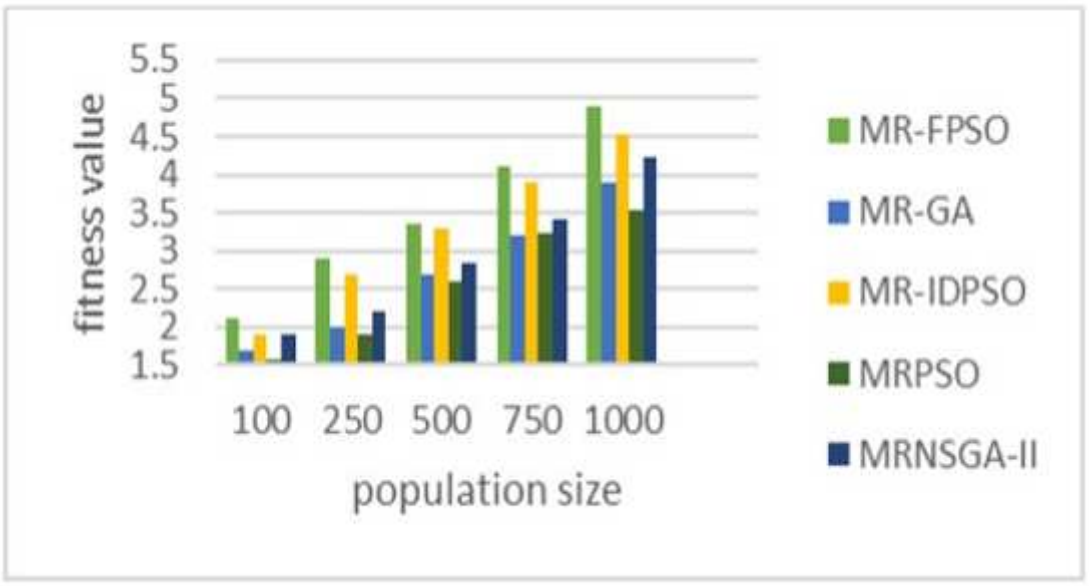

Figure 10

Fitness value with the number of population size

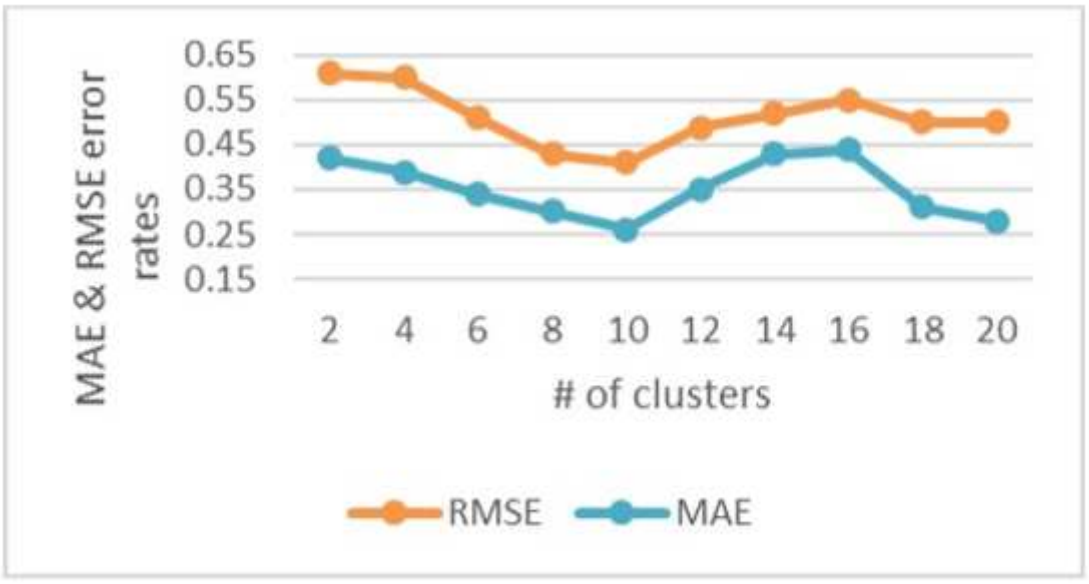

Figure 11

Number of clusters with error rates

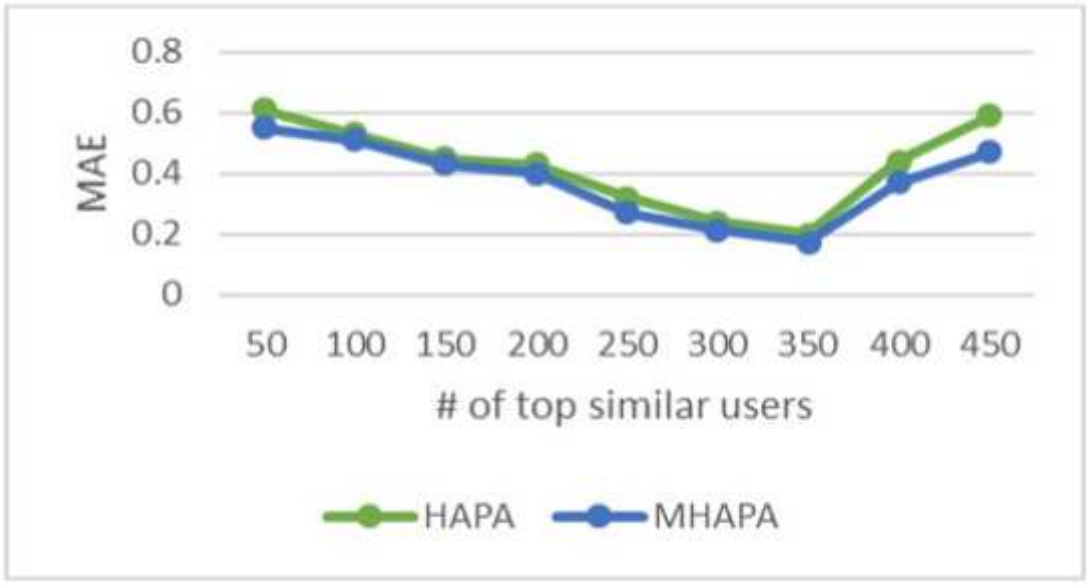

Figure 12

HAPA Vs. MHAPA