Iranian Journal of Science and Technology, Transactions of Electrical Engineering
https://doi.org/10.1007/s40998-020-00371-z

RESEARCH PAPER

Online QoS Prediction in the Cloud Environments Using Hybrid Time-Series Data Mining Approach

Amin Keshavarzi[1] · Abolfazl Toroghi Haghighat[1] · Mahdi Bohlouli[2,3,4]

Received: 25 October 2019 / Accepted: 15 August 2020

Abstract
Considering the diversity of proposed cloud computing services in federated clouds, users should be very well aware of their current required and future expected resources and values of the quality-of-service parameters to compose proper services from a pool of clouds. Various approaches and methods have been proposed to accurately address this issue and predict the quality-of-service parameters. The quality-of-service parameters are stored in the form of time series. Those works mostly discover patterns either between separate time series or inside specific time series and not both aspects together. The main research gap which is covered in this work is to make use of measuring similarities inside the current time series as well as between various time series. This work proposes a novel hybrid approach by means of time-series clustering, minimum description length, and dynamic time warping similarity to analyze user needs and provide the best-fit quality-of-service prediction solution to the users through the multi-cloud. We considered the time as one of our important factors, and the system analyzes the changes over time. Furthermore, our proposed method is a shape-based prediction that uses dynamic time warping for covering geographical time zone differences with the novel preprocessing method using statistically generated semi-real data to fulfill noisy data. The experimental results of the proposed approach show very close predictions to the real values from practices. We achieved about 0.5 mean absolute error rate on average. For this work, we used the WS-DREAM dataset which is widely used in this area.

Keywords Service composition · Quality of service · Multi-cloud · Exponentially weighted moving average · Time-series clustering · Minimum description length · Dynamic time warping

1 Introduction

Today, cloud computing is used in various domains for many different goals and applications, ranging from simulations, design, and research activities (Bohlouli and Analoui 2008), healthcare (Bohlouli et al. 2011, 2014), and support services such as customer support or human resource management. Cloud computing has the following four deployment models: public, private, hybrid, and community (Mell and Grance 2011). It may be required for the workload of cloud users to be transferred to other cloud providers for various reasons such as lack of enough resources or SLA violation prevention. In this case, the transfer may happen through interoperations between cloud providers, which is known as inter-cloud or cloud of clouds (Cases 2010). Inter-clouds can be either federations or multi-cloud (Grozev and Buyya 2014). In federated clouds, various providers agree to share and interconnect their

✉ Abolfazl Toroghi Haghighat
haghighat@qiau.ac.ir

Amin Keshavarzi
keshavarzi@miau.ac.ir

Mahdi Bohlouli
bohlouli@iasbs.ac.ir

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

[2] Department of Computer Science and Information Technology, Institute for Advanced Studies in Basic Sciences, Zanjan, Iran

[3] Research and Innovation Department, Petanux GmbH, Bonn, Germany

[4] Research Center for Basic Sciences and Modern Technologies (RBST), Institute for Advanced Studies in Basic Sciences, Zanjan, Iran

Published online: 20 September 2020

existing resources with each other in order to have higher performance and fault tolerance. In multi-clouds, it is a user or broker that chooses various services from different providers (Keshavarzi et al. 2017). The multi-cloud broker intermediates between end service users and providers, as well as helps customers to select proper services. Furthermore, the service selection (Lin et al. 2017; Garg et al. 2013) and service composition (Ye et al. 2016; Hayyolalam and Kazem 2018; Vakili and Navimipour 2017) are the two important issues in multi-cloud (Liaqat et al. 2017), that is done in multi-cloud broker. It should be noted that the proper service selection facilitates SLA violation prevention in clouds.

In multi-clouds, there may be services with similar functionalities, but different QoS features. For instance, availability, throughput, response time, and price are the main features of QoS (Menascé 2002). Some of these features such as price are on the server side, and some others such as throughput and response time are on the client side. The server-side features are typically similar to all customers. In contrast, based on the application, environment, Internet, network bandwidth, and servers' status, such as workload, client-side features may differ (Menascé 2002). Some of the QoS features in time-series data, for example sudden peaks in response time, which are caused by difficulties of the server or network congestion, have different characteristics from the common types such as stock market analysis time series. Therefore, the nature of QoS characteristics is nonlinear and dynamic, and the prediction task is complicated.

In order to offer a service selection and composition by the broker, one needs to have an accurate QoS prediction process. It should be stated that most QoS features are being contracted between customers and providers in the frame of service-level agreement (SLA), which consists of various sections being referred as service-level objectives (SLOs). Moreover, SLAs cover penalties for any SLA violation from the side of service provider. In practice, experienced values of the QoS features by customers vary from whatever they contracted with the providers in their SLAs (Amiri et al. 2018; Arianyan et al. 2017). If this is not paid attention to from the side of providers, it may result in penalties and accordingly heavy overall costs to them. Consequently, they have to have QoS monitoring and prediction to have proactive prevention activities in the case of SLA violation. This guarantees that customers will be able to receive the QoS features they agreed to in the SLAs (Ye et al. 2016). The dynamic and nonlinear properties of QoS data complicate the QoS prediction task. The QoS prediction has the following challenges:

(a) QoS features practically vary for one specific user over time. This is based on the state of servers at different time intervals, as well as workload and arrived service requests. For example, analyzing the WS-DREAM dataset #3 (Zheng et al. 2014) showed us the fact that a cloud service may have different QoS values in different time slots.
(b) During the usage of invoked service, users are faced with many fluctuations. The behavior of a system has similar patterns. The shape patterns of an interval can be repeated in the same time-series data or others.

To solve the stated challenges, different efforts have been made. Collaborative filtering (CF) (Terveen and Hill 2001) is used commonly in most of these efforts. In general, the CF is used in recommender systems (Machado et al. 2018) and uses historical data of similar services to predict the values of QoS features. Other approaches, such as time-series data analysis or combined CF and time-series analysis, are generally used in the prediction of QoS. However, in cloud computing, developing an accurate method for prediction of QoS characteristics still remains as the main challenge. The dissimilarities in the server workloads due to the different time zones and frequent patterns led to the difficulties in QoS time series. In this regard, we benefit from a hybrid approach in this work for the prediction of time-aware QoS in cloud services.

The basic idea is that the values of QoS features for one specific user can be predicted through discovering the historical and experimental data for the target or past services of this or other users. We used clustering of time-series data to realize differences inside the historical data of service usages. The similarity measures between the usage information of services over time, which exists in our historical dataset in the previously clustered ones and the service usage information, which is collected through the execution of current service for prediction, contributes to find the proper cluster. In order to proceed training of our proposed model for predicting the QoS values, we used customized minimum description length (MDL) (Rissanen 1978).

The MDL can determine the best hypothesis for a given set of data. The principle of MDL is using the pattern that exists in the data in order to compress it (Grünwald et al. 2005). The idea of using MDL in time-series data mining has been investigated in the literature (Hu et al. 2011). We use MDL in discovering repeated patterns in the monitored time series and use them in order to predict the next value of time series. To the best of our knowledge, it is the first time that this method is being used for QoS prediction. The contributions of this work are as follows:

- Proposing a hybrid method as a combination of clustering and MDL algorithms for online QoS time-series prediction.

- Applying MDL to find the best length of query window (the latest time slots of current time series).
- Finding the most similar subsequences with query window in terms of their shape and using them in QoS prediction.

In this paper, four sections are organized consisting of introduction, related work, description of the proposed model, and results. A problem definition and motivation of the research are provided in the first section. In order to investigate the shortcomings of the existing works and highlight the novelty of the proposed method, state-of-the-art and literature review are summarized in Sect. 2. The details of the proposed approach for the prediction of QoS are described in Sect. 3. The results of the proposed model are discussed in Sect. 4.

2 Related Works

Nowadays, there exist various cloud-deployed services with similar functionalities. Users compare such services with respect to the QoS features in order to make proper decisions in the composition and selection of services. This plays a vital role in a service selection, recommendation, and SLA violation detection. Thus, the key to the QoS-aware service computing is QoS prediction. In general, Web service QoS prediction methods can be categorized into (1) CF methods (Shao et al. 2007; Zheng et al. 2009, 2011; Wu et al. 2015; Zhang et al. 2014; Wu et al. 2013, 2018; Luo et al. 2015; Yu and Huang 2016; Keshavarzi et al. 2020), (2) statistical time-series (TS) analysis methods such as generalized autoregressive conditional heteroscedasticity (GARCH), autoregressive integrated moving average (ARIMA), and their hybrid approaches (Amin et al. 2012a, b), and (3) combined CF and TS approaches (Ding et al. 2018).

Breese et al. (1998) used Pearson correlation coefficient (PCC) in their proposed user-based PCC (UPCC) framework as a method in recommender systems. Later, it was adopted to QoS prediction as well (Shao et al. 2007). The UPCC predicts the values of current time slot for QoS features using other collected values of QoS features of top-k similar customers to current customer. Users' similarity is calculated using the PCC measure. This method is a well-known QoS prediction method. Lack of supporting the historical temporal QoS values in time series is considered as the main shortcoming of UPCC. Furthermore, the item-based PCC (IPCC) has been proposed in Sarwar et al. (2001) which uses the item-based prediction in recommender systems. Also, Zheng et al. (2011) have extended IPCC in their proposed approach to predict the values of QoS features. The IPCC uses the same method as UPCC, but for services instead of users. Like UPCC, IPCC also suffers from neglecting the historical temporal QoS values in the series. WSRec proposed a hybrid approach of UPCC and IPCC (Zheng et al. 2009) for QoS prediction as well. In this method, both similar users and services are used for predicting the QoS values. The drawback of WSRec is similar to that of UPCC and IPCC, and these methods are not accurate in prediction.

Wu et al. (2015) proposed a prediction method for QoS features, called CAP, which is credibility-aware and detects unreliable data offered by untrustworthy users that decrease prediction accuracy. In order to identify untrustworthy uses, Wu et al. used two-phase K-means clustering and clustered QoS values for each service and stored them for calculating untrustworthy index in the first phase. They clustered customers in the second phase according to their untrustworthy index and predicted missing values of QoS features based on the credible clustering information. The untrustworthy index defines the number of times that a user is regarded as a candidate untrustworthy user at different services. The historical QoS values of each service are clustered in the first phase using the K-means algorithm. The clustered data are stored in a matrix, and then the untrustworthy index is calculated for each customer. The K-means algorithm is reused for clustering users in the second phase based on the untrustworthy index and identifies a set of untrustworthy customers. Their evaluation shows that this method improves the prediction accuracy comparing to other approaches. Furthermore, it is robust regarding untrustworthy customers. Their method does not consider time factor in the prediction and does not use the historical data.

Zhang et al. (2014) proposed a performance prediction approach called OPred, which provides personalized performance prediction of a system using past usage experiences of various users. In order to predict Web service performance, they used a set of latent features that are the orthogonal representation of physical factors such as network distance between server and client, server's workload to realize the Web services and clients' status. They extract the latent features of services and users from previous time slots, analyze the feature change trends, and calculate the values of features for users and Web services in the current time slot. The proposed approach consists of four phases: Each user monitors and keeps local performance of Web services in the first phase. In the second phase, each user sends local performance to the performance center. The performance center combines all local performance information of all users and obtains global performance information of whole services. In the third phase, the time-specific service and user features are extracted, and by performing the time-series analysis, a performance model

is constructed. In the last phase, the overall performance of a system is predicted.

They use a matrix factorization method to fit a feature model to the client–service matrix in each time slot and learn the latent features of clients and services. Regarding the user–service matrix, the low-dimensional client-specific latent feature vectors of users and low-dimensional service-specific latent feature vectors of services are constructed using matrix factorization. The optimization of the matrix factorization in each time slice is performed by using a cost function. This function is used for evaluating the quality of approximation. Their cost function is a distance between two nonnegative matrices. The latent feature learning is done in each time slot iteratively. After the latent feature learning, the performance prediction consists of two phases: off-line and online. In the off-line step, trends of the user features are statically modeled and the service features are performed based on performance information, collected from all users.

Wu et al. (2018) proposed a generic context-sensitive matrix factorization method named CSMF to predict the QoS values. They modeled the client-to-service and environment-to-environment simultaneously and used the contextual factor in the prediction. Actually, the values of QoS features are modeled as a services function, users, and contexts data whose context is defined as the contextual information associated with the service invocation such as the user or service provider. In this method, at first, the function is estimated for the whole $user \times service \times context$ space, and then the prediction is made for the current client, service, and contextual conditions, i.e., the mixture of conceptual instances such as "$\{Germany, Google\}$." In fact, the configuration of hosts, as well as the server and network conditions, affects the values of the QoS features. Their experimental results showed that this method outperforms related methods in prediction accuracy.

Luo et al. (2015) proposed a method for the QoS prediction of cloud services, which is a hybrid approach of fuzzy neural networks and adaptive dynamic programming (ADP). They extract fuzzy rules from QoS data and learn the parameters of the fuzzy rules by using ADP. The ADP is an optimization technique that uses hybrid reinforcement learning (e.g., Q-learning) approach and dynamic programming. The ADP can use a function approximation structure, such as neural network (NN) to approximate the cost function. The ADP is suitable for learning from the dynamic environment with less information. The scalability of ADP makes it suitable for the nonlinear prediction of QoS data. Experimental results showed that this method outperforms traditional methods such as UPCC, IPCC, and UIPCC.

The time- and location-aware collaborative filtering (TLACF) method was presented in Yu and Huang (2016). Based on the locations, users and services were clustered in the proposed method. Then, the average similarity between the current service and the other services of the same cluster is calculated. Afterward, the top-k services are selected and the user average similarity is calculated based on the selected services data at the previous step. At last, the QoS prediction is made based on data from the first and second steps. The presented method in this work addresses the issue of the scalability in the CF memory-based method by means of searching for clients within smaller clusters rather than seeking in the whole database. This method combines the strengths of memory-based and model-based approaches to overcome the shortcomings of both. Their experimental results showed that this approach has higher prediction quality and scalability, as well as being easy to build and update.

Wu et al. (2013) proposed some methods, namely adjusted cosine (A-cosine), data smoothing, and fusion of similarity results (ADF). They used adjusted cosine formulation to exclude the impact of various QoS scales and calculate the service similarity. Accordingly, they use data smoothing after the similar neighbors' selection and before predicting target QoS to have higher prediction accuracy. The missing QoS features of the similar neighbors are computed by the data smoothing process. This process replaces the missing QoS values by the average of the existing QoS values in the same corresponding cluster. The final predicted QoS values are resulted by fusion of the predicted values from three various sources: (1) predicted QoS values for the same service of similar clients (similar to user-based CF), (2) predicted QoS values of similar services to the same client (similar to item-based CF), and (3) predicted QoS values of similar services to similar clients. This approach is useful, when the density of QoS data is low.

Wang et al., have proposed a hybrid approach called LASSO (Wang et al. 2016). They assume that the prediction error has a normal distribution; then, they prove that this model is not appropriate because the normal distribution cannot handle a large number of abrupt changes effectively. Next, they assume a zero-mean Laplace prior distribution on the residuals of the predicted QoS values, which corresponds to a Lasso regression problem. This leads to an effective sparse representation of temporal QoS data. The size of the selected example QoS sequences affects the computational cost of Lasso. In order to decrease the computational overhead, they used spatiotemporal information that exists in the dataset and select just a small portion of samples from their dataset, i.e., the spatially close user–service pairs with the current user–service pair. This approach consists of three steps,

including data preparation, model construction, and model selection. In the preparation phase, invalid values with zeros are handled. If the last value of a series in the dataset is invalid, that series is neglected. If any previous values are invalid, they are replaced with the first previous valid value. At the next step, the K most correlated temporal QoS sequences with the current QoS sequence are selected to train regression models. The LSR or Lasso methods are used to estimate unknown parameter w in the regression models. For a current series, several regression models can be constructed using different algorithms. After model construction, one best model is selected by applying the standard estimation error and Schwartz's Bayes criterion (SBC). They conclude from their experimental results that their approach improves the state-of-art methods about 10% in the temporal QoS prediction accuracy. They have also used the geolocation of clients and services to reduce search space and accordingly increase the prediction performance.

In connection with the statistical time-series analysis, Amin et al. (2012) have proposed integrated ARIMA with GARCH. ARIMA is the generalization of autoregressive moving average (ARMA) (Box et al. 2015). In this method, a non-stationary series should be transformed into stationary by d differences, $y'_t = y_t - y_{t-d}$, where each new value y'_t is calculated based on the difference of the original value y_t from the d previous value y_{t-d}; then, the original series is being generated by an ARIMA model. In order to model a time series using ARIMA, it must have serial dependency, normality, stationary, and invertibility (Engle 1982). GARCH (Bollerslev 1986) is a generalization autoregressive conditional heteroscedastic (ARCH) model. The ARCH (Engle 1982) model describes dynamic changes in the time-varying variance as a deterministic function of the past errors. This integration has been selected due to the low accuracy of ARIMA in the prediction of time series that is nonlinear. They predict the response time and the time between the failures of ten real-world Web services. The prediction in this work consists of three phases: (1) The ARIMA model is used to compute forecasts residuals. Residuals are the remainders after estimating and eliminating the time-series components, such as trend and seasonal ones, (2) the computed residuals are used to construct a GARCH model to fit volatility, and (3) the ARIMA–GARCH model is used to predict the future QoS values (Bollerslev 1986). They show that by integration of ARIMA and GARCH, the accuracy of QoS prediction is improved.

In another work, Amin et al. (2012) have proposed an integrated model for the autonomous use of ARIMA as a linear and self-exciting threshold ARMA (SETARMA) as a nonlinear model. Their proposed approach selects and constructs the best suitable prediction model to fit the dynamic behavior of QoS attributes. Their approach has two components: one for the model construction and another for the continuous QoS prediction. The main task of the model construction component is time-series modeling. It consists of seven phases: data preparation, model order identification, nonlinearity test, delay parameter and thresholds identification, adequate model specification, model estimation and model checking, and the best model selection. In the continuous QoS prediction, by using the best-selected model and new arrival QoS data, the future QoS values are predicted, and then the prediction error is computed. If the selected model is inaccurate, another model is selected based on the QoS data arrival and analysis of prediction error. They evaluate their work on 800 different real-world Web services and compute the response time and the time between their violations. The results show that their approach outperforms the popular existing ARIMA models and improves the prediction accuracy.

An approach for time-aware service recommendation (taSR) has been proposed by Ding et al. (2018). In this work, the first-user similarity is computed using a novel similarity enhanced CF approach that captures the time feature and addresses the data sparsity in the existing time slots. They compute the global similarity based on QoS by using an adaptation of PCC and then compute user invocation similarity using the adoption of edit distance. In the second phase, they use the similarity estimated at the previous step and choose the most similar users and the target user to fill the missing QoS values in the past and current time slots. Then, ARIMA is used to predict the values of QoS features in the future time slot. The missing values of QoS time series are filled using a CF method such that the data series have sufficient valid values for the construction of the ARIMA model. In the CF method, the client global and invocation similarity are used together. Their experimental results approve the significant performance improvement in the proposed taSR compared with existing approaches.

A complete comparison of related work is depicted in Fig. 1. As shown in this figure, our approach is a hybrid CF method and a combination of memory- and model-based methods. We used dynamic time warping (DTW) in the similarity computation (memory-based part) and k-medoid clustering algorithm for learning (the model-based part). In order to predict the QoS values, some researches utilize the location information on users and services; some of them use time; some others, both of them; and some, none. We used only time factor in the QoS prediction. Some model-based or hybrid approaches are global and some are local. In local methods, the modeling is done based on query information and running time. Our method is local.

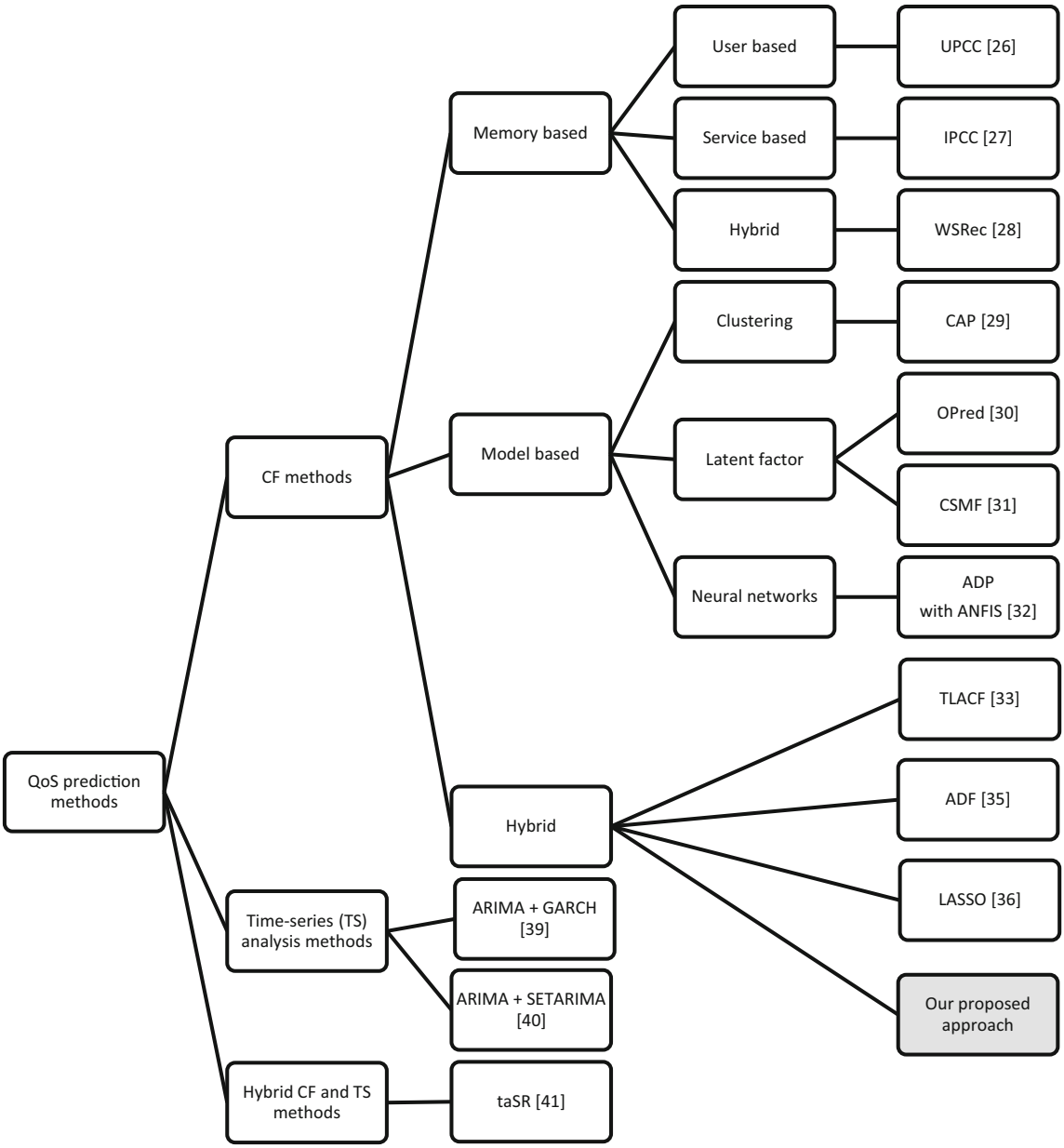

Fig. 1 Classification of the proposed approach beyond state of the art

QoS data may contain invalid values; thus, preprocessing is necessary. Some researchers carried out preprocessing, but their methods are simple and ineffective such as using mean value calculation. We use exponentially weighted moving average (EWMA) which is an effective one and more accurate method in the time-series missing values imputation. Another difference of our method in comparison with the related works is in the clustering of data. Some methods cluster users or services; and some others, both of them. Instead, in this research, we clustered time-series data. The major difference in our method in comparison with the related work is in using the current QoS time-series data for modeling. A user experiences many fluctuations in using a cloud service. When an active user uses a service, the QoS values of the previous time slots may be repeated in the near or distant future.

3 Time-Aware Cloud Service Selection Framework

The proposed hybrid framework receives the functional requirements that define what a user requires in regard to a cloud service does for it, as well as non-functional preferences such as the response time and throughput related to an active user as input and recommends the best-fit cloud

services for selection and composition by the user. Figure 2 shows further details and an architectural overview of the proposed hybrid approach. This paper focuses only on the QoS prediction component as highlighted with the blue color in the concept image (Fig. 2). The proposed approach starts with off-line procedures, which consist of service registration, monitoring, preprocessing, and QoS time-series clustering. In the registration phase, providers register their services in Universal Description, Discovery, and Integration (UDDI) registry. UDDI uses the XML-based format for the services registry. The cloud service monitoring phase collects service QoS values in different time slots and stores them in the corresponding dataset. At the next step, invalid series such as the ones with the missing values are handled at the preprocessing step. The preprocessed time series are also stored in the corresponding dataset for further processing in the next phases. In the clustering phase, time series are clustered by using a k-medoid algorithm. Accordingly, online phases consist of model construction and QoS prediction, which are accomplished as follows:

1. The active user sends his/her request to the cloud service discovery procedure. The request consists of functional requirements and non-functional preferences concerning the active user.
2. The cloud service discovery (CSD) procedure retrieves cloud service candidates from the UDDI registry that could satisfy the active user requirement. Then, CSD sends non-functional preferences and candidates services list to the cloud service selection procedure in order to construct a model.

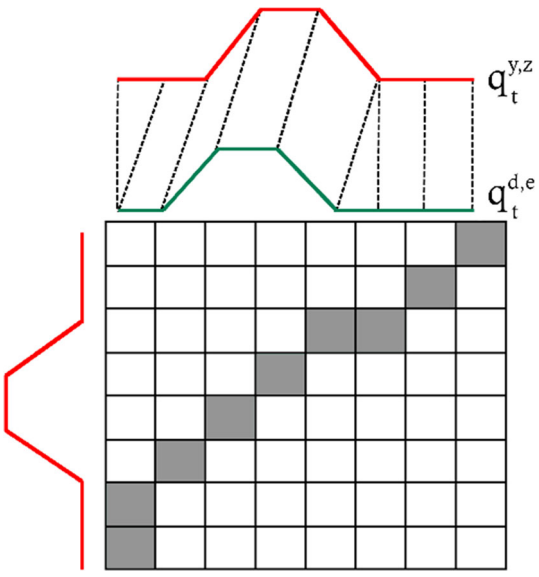

Fig. 2 An example of a warping between the two time series and the related cost matrix with the minimum-distance warp path (Engle 1982)

3. The active user's current QoS experiences are retrieved from the collected QoS dataset while the model is constructed.
4. The predictor generation procedure constructs models for each candidate service based on the retrieved QoS experience of the active user.
5. The prediction is made based on the constructed models of all candidate services. Consequently, the results are sent to the cloud service selection procedure.

3.1 Problem Definition

In the proposed work, we have a dataset of historical QoS time series related to different invoked services by different registered users in different time slots. This is known as existing historical data in our system. We may have an active user, who uses the system. We record the values of QoS features for various candidate services of the current active user up to the current time slot. The goal is predicting the next value of a QoS feature for those services of active user based on the previously monitored QoS values of the active user, as well as the available historical data of other registered users. There could be a one-step or *n*-step-ahead time-series prediction. The main focus of this work is on the one-step-ahead QoS prediction. In one-step prediction, just the next value of the time series is predicted, yet in the n-step-ahead, the next value, as well as a few more steps ahead (namely n values), is predicted. Furthermore, we concentrate on the univariate time series, which consists of a sequence of values for one variable such as the response time or throughput. All used notations in formulas are listed in Table 1.

Let MCP show the cloud providers, U for representing the registered users, S for existing cloud services, T to illustrate the time intervals, and Q for QoS values in the available time intervals. Therefore, the following definitions can be obtained:

$$MCP = \{cp_1, cp_2, \ldots, cp_z\}, \tag{1}$$

$$\begin{aligned} U &= \{u_1, u_2, \ldots, u_m\} \\ S &= \{s_1, s_2, \ldots, s_n\}, \end{aligned} \tag{2}$$

Table 1 Overview of defined notations

Notation	Definition
z	Total number of cloud providers
m	Total number of registered users in the system
n	Total number of existing cloud services
x	Total number of equal-spaced time intervals
t	One time slot

$$T = \{t_1, t_2, \ldots, t_x\}, \tag{3}$$

$$Q = \{q_t : t \in T\} = q_1, q_2, \ldots, q_x. \tag{4}$$

where q_i represents the value of QoS in ith time interval (t_i). The QoS values consist of response time and throughput. The QoS matrix for time series can be defined as:

$$M = \left(q_t^{i,j}\right)_{m \times n}. \tag{5}$$

where $i \in \{1, 2, \ldots, m\}$, $j \in \{1, 2, \ldots, n\}$ and $\left\{q_t^{i,j} : t \in T\right\}$ is a time series.

Put $R := \left\{t_i, t_{i+1}, \ldots, t_j\right\} \subseteq T$; then, a subseries of time series Q from the step i to the step j is defined as

$$P = \{q_r : r \in R\}. \tag{6}$$

If active user, u_{ac}, uses an active service, s_{ac}, the value of its QoS feature, q_t $(t \in \{1, \ldots, cur\})$ is considered. The goal is predicting the next value of QoS feature, q_{cur+1}, for active user, in which $3 \leq cur \leq x - 1$. The QoS values that an active user experiences, when he/she invokes an active service, are divided into the query window and past window. The values of query and past windows are used in the model generation of our learning phase. Based on conducted research (Hu et al. 2014), the similarity in recent time slots (current time slot and a few previous ones) has a higher impact than other time slots in all user similarity. Thus, we consider QoS experience values of the most recent time slots of the active user on the active service as query window. The past window contains the previous time slots of query windows. We first find the best length of query window, a sequence of time slots which has maximum number of similar subsequences in Q_{active} in terms of its shape. We first extract all existing shape-wise similar patterns of Q_{query} for an active user. We use Q_{query} to look for pattern similarities in service use data of the same user. The shape similarity identification process is being used for service use data of other users of the past window (Q_{past}) as well:

$$Q_{past} = \mathrm{q}_1, \mathrm{q}_2, \ldots, \mathrm{q}_\mathrm{i} \tag{7}$$

$$Q_{query} = q_{i+1}, \ldots, \mathrm{q}_\mathrm{cur} \tag{8}$$

$$Q_{active} = \mathrm{q}_1, \mathrm{q}_2, \ldots, \ \mathrm{q}_\mathrm{cur} \tag{9}$$

$$C = \{\mu_1, \mu_2, \ldots, \mu_k\} \tag{10}$$

In order to optimize searching similar patterns of the other time series, the clustering is performed. Therefore, matrix M should be clustered.

Let C represent the clusters' medoids (μ_i) and k illustrate the number of clusters. Therefore, by using the lazy learning (LL), predictor generation can be defined as follows:

$$Predictor \leftarrow LL(Q_{active}, \mathrm{length(query)}, \mu_c) \tag{11}$$

while μ_c represents the medoid that is the most similar value to the Q_{query}.

3.2 DTW

There are various distance measures among time series in the literature (Aghabozorgi et al. 2015), such as DTW, PCC, Euclidean distance (ED), cross-correlation-based distances, and probability-based distance. We use k-medoids algorithm (Kaufman and Rousseeuw 2009) and the DTW similarity measure (Berndt and Clifford 1994).

The DTW is expensive in computation, yet one of the most effective, accurate, and popular time-series similarity measures (Aghabozorgi et al. 2015). Due to the fact that we focus on the similarity measure in terms of their shape, the DTW method can perfectly convince this goal. In addition to the fixed-size assumption of time series while clustering and the subseries similarity measures, our concept consists of various length difficulties for the similarity of query window with medoids. To do this, DTW provides better and more convincing accuracy. Accordingly, DTW is more convincing and accurate than the one-to-one alignment such as the Euclidean distance measure (Berndt and Clifford 1994; Mueen et al. 2018).

The DTW similarity measure is a shape-based similarity measure for time-series data, which allows the one-to-many alignment between data points and is used for both equal and unequal time-series lengths (Berndt and Clifford 1994). This enables a significant comparison between two time series with similar shapes and locally out of phase (Dau et al. 2018). In clouds, the workload may vary as it uses the on-demand model. Different users may use similar workloads, which demonstrate similar patterns but in different time slots for such reasons as time zone locations. Such a fact can be discovered by means of DTW when it compares two time series. Because DTW allows the one-to-many alignment and captures the locally out-of-phase property of time series. Figure 2 shows further details about this point.

The DTW method first constructs the distance matrix between two time series (such as $q_t^{d,e}$ and $q_t^{y,z}$) in order to find the warping path W. Each element (i, j) in the distance matrix is the squared Euclidean distance between the ith point of $q_t^{d,e}$ and jth point of $q_t^{y,z}$. The warping path, $W = w_1, w_2, \ldots, w_K$, with the length of K, $\max(\mathrm{m},n) \leq K \leq m + n$, is a set of contiguous matrix elements that define the alignment between $q_t^{d,e}$ and $q_t^{y,z}$. The kth element of W is defined as $w_k = (i, j)$, where i is an index from time series $q_t^{d,e}$ and j is an index from time series $q_t^{y,z}$. The warping path starts from $w_1 = (1, 1)$ and ends at $W_K = (m, n)$. A path that results from the minimum possible difference between two time series should be considered as a warping path (Dau et al. 2018) (Eq. 12).

$$DTW\left(q_t^{d,e}, q_t^{y,z}\right) = \min\left\{\sqrt{\sum_{k=1}^{K} w_k}\right\} \quad (12)$$

Accordingly, the minimum-distance warp path is achieved using the dynamic programming approach. In the dynamic programming implementation of DTW, the alignment cost matrix D is constructed. The (i,j)'s element of the matrix is the minimum cumulative sum of the alignment cost up to $q_i^{d,e}$ and $q_j^{y,z}$. Consequently, the last (bottom corner) element of the matrix represents the cost for the full alignment between $q_i^{d,e}$ and $q_j^{y,z}$, which is the DTW distance between the two time series (Dau et al. 2018) (Eq. 13). To find the minimum-distance warp path, in the filling of D matrix entries, invalid indexes are not considered. The cost matrix needs to fill one column at a time from bottom to top and from left to right. In this paper, DTW is used to measure the time-series similarity and finally cluster them in order to calculate the distance between current series and clusters medoids and finally to calculate the distance between the query window and the other subseries from current series or the other series.

$$D(i,j) = \left(q_i^{d,e} - q_j^{y,z}\right)^2 + \min(D(i-1,j), D(i,j-1), D(i-1,j-1)) \quad (13)$$

3.3 Proposed Time-Aware Hybrid QoS Prediction Approach Using MDL and Clustering

The proposed time-aware hybrid approach consists of four main phases as depicted in Fig. 3: (1) preprocessing, (2) clustering, (3) model generation, and (4) prediction. The main goal in preprocessing is to clean and prepare our dataset, resolve noisy data, and regenerate the missing values of the dataset. This provides a completed dataset for the clustering phase. Lazy learning, which is used in this work, does not work accurately with noisy data, especially with the missing values, resolved in this work through the preprocessing phase. Due to a large volume of service/user data, we have used clustering to reduce the volume and process only similar to cluster data instead of the whole dataset. This phase provides scalability for our algorithms on any scale and fulfills the time-overhead drawbacks of lazy learning for large problems previously stated. First two phases are off-line and do not depend on the newly arriving data of active users/services.

3.3.1 Preprocessing

The collected values of QoS features may consist of invalid values (e.g., missing values) for any reason such as measurement errors. This data will have a great impact on the prediction performance (Kiani et al. 2020). In some literature (Yu and Huang 2016; Wang et al. 2016), the common method to calculate the missing values is the mean of its neighbors or replacing them by the nearest valid values. Neither of these methods is statistically accurate.

The missing values can be located in any position of QoS time-series data. The way we resolve invalid values is based on their place and length of continuous missing values. The method we use in this regard is described in Listing 1. The proposed approach for missing values imputation is based on EWMA. In this approach, the value of moving average window is determined based on the location and number of continuous missing values. Then, the mean value of observations near to a central missing value is calculated to impute missing value. As shown in Listing 1, in case the number of continuous invalid values would be higher than 12, the related time series is removed from the dataset (line 4); otherwise, regarding the location of missing value and its neighbor values, it is imputed using EWMA method (lines 7–11).

Input: Matrix $M_{m\times n}$

1. **for** $i = 1$ to m **do**:
2. **for** $j = 1$ to n **do**:
3. $q_t^{i,j}$=M[i,j];
4. **if** $\#missingDataPoint\left(q_t^{i,j}\right) \geq 12$ **then**:
5. $M = M - \{q_t^{i,j}\}$; /*Remove $q_t^{i,j}$ from M */
6. **else**
7. **for** each missing value in $q_t^{i,j}$ **do**:
8. *miss* ← the position of missing value
9. k ← the size of moving average window such that there are at least 2 valid values in [miss-k , miss+k] interval of $q_t^{i,j}$
10. $S_i \leftarrow \lambda\left(q_{miss-1} + (1-\lambda)q_{miss-2} + (1-\lambda)^2 q_{miss-3} + \cdots + (1-\lambda)^k q_{miss-(k+1)}\right)S_{i-(k+1)}$
/*S_t *is the value of the EWMA at any time period* t */
11. **end for**
12. **end if**
13. **end for**
14. **end for**

Listing 1. Pre-processing algorithm

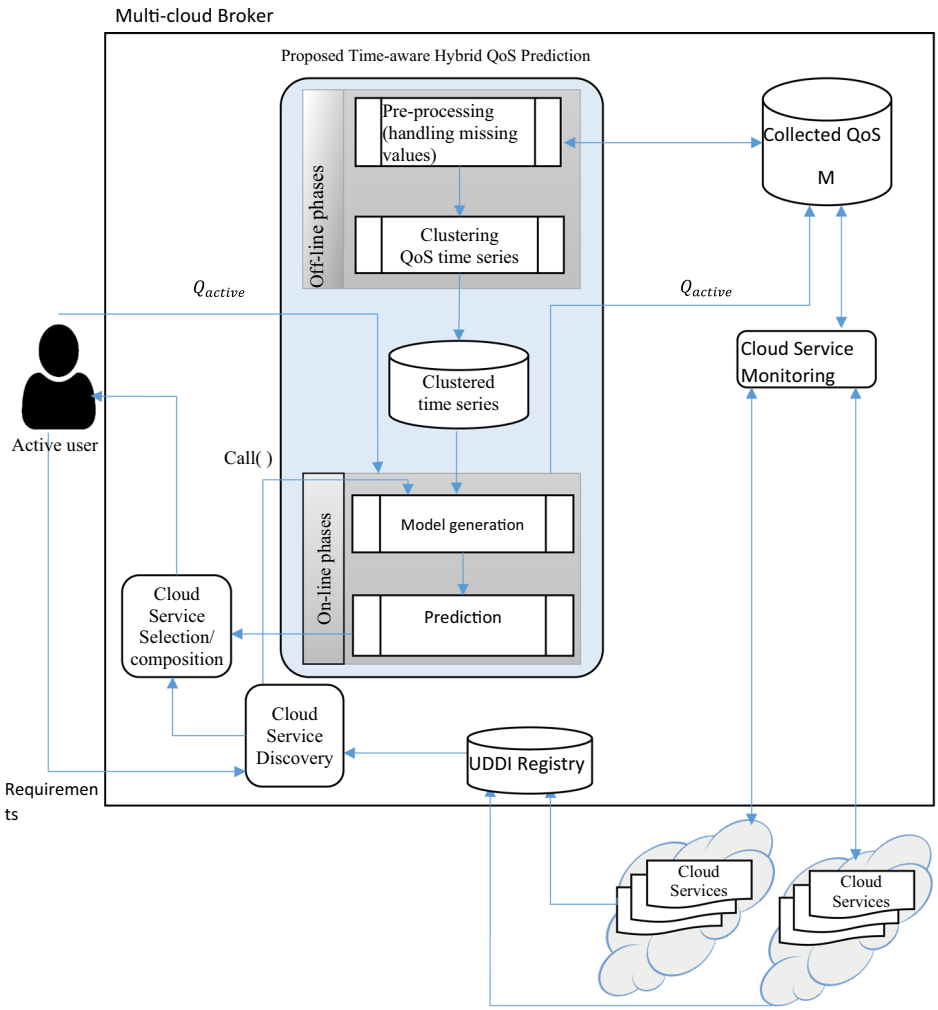

Fig. 3 Proposed time-aware service selection framework (the notations are described in Sect. 3.1)

3.3.2 Clustering

There are a number of various time-series clustering methods (Aghabozorgi et al. 2015) in the literature, such as relocation clustering, agglomerative hierarchical clustering, self-organization maps, *k*-means, fuzzy *c*-means, and *k*-medoids. We used *k*-medoids algorithm due to the fact that it is fast, simple, and efficient for clustering of same length time-series data (Aghabozorgi et al. 2015), even so, it needs to define the total number of clusters in advance. The *k*-medoids algorithm has been depicted in Listing 2. As shown in this figure, the first *k*-time series are randomly selected as initial cluster medoids (line 2). After that, for each remaining time series, the DTW distance of it to all medoids is calculated and it is assigned to the cluster whose medoid has the minimum distance to the time series (lines 4–6). In the end, new cluster medoids are determined. The algorithm converges when the medoids remain constant in two consequent iterations (lines 7–10).

The clustering phase is conducted as off-line; thus, it doesn't impose any performance overhead to the overall algorithm. Indeed, after initial clustering, new time series are assigned to the best cluster. It is obvious that after some iterations the quality of cluster is diminished. In the case of this issue, re-clustering should be done when the value of clustering evaluation has been reduced by a portion of its initial value, for example 20%.

Input: M (Series database), k (The number of clusters)

Output: clustered data

1. **Begin of algorithm**
2. Initialize cluster medoids $\mu_1,\mu_2,\ldots,\mu_k \in M$ randomly;
3. **Repeat**
4. **For** each s in M **do**:
5. $s.cluster \leftarrow \arg min_{1\le i\le k}\left(DTW(s,\mu_i)\right)$;
6. **End for**
7. **For** $i = 1$ to k **do**://Update the cluster medoids
8. $\mu_i \leftarrow medoid_{s\in M \wedge s.cluster=i}(s)$;
9. **End for**
10. **Until** no change.
11. **End algorithm**

Listing 2. k-medoids algorithm

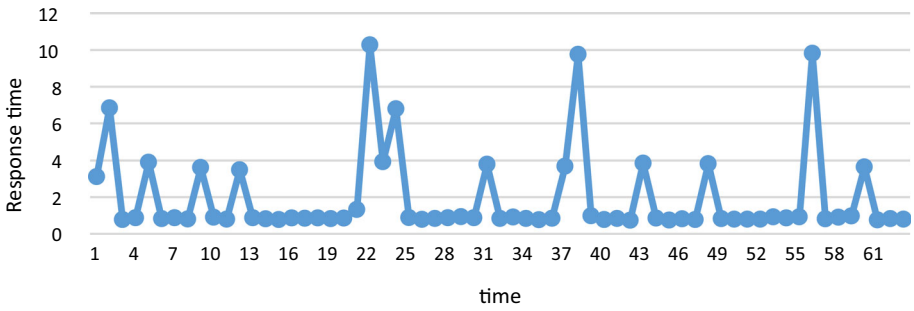

Fig. 4 An example of running proposed prediction algorithm on a QoS time series of WS-DREAM dataset

3.3.3 Model Generation

Local and global modeling can be considered for the construction of prediction models (Bontempi 2000). In the local modeling, clients' queries are the base for construction. Mutually, clients' queries do not play any role in the construction of global modeling. In general, global modeling may seem simple, but it could be affected by outliers. In contrast, local modeling may look more complex, but with higher adaptivity (Birattari et al. 1999). In the local modeling, training data are first stored in a simple manner, and the major computation is made at the running time. When the new data (query) are received, the learning process is performed based on a query. Indeed, the model construction is made at the running time and based on the query data.

The proposed method for model construction in this paper has been inspired from MDL concept (Barron et al. 1998). In MDL, if the number of bit that is needed to represent an original string S, denoted by $DL(S)$, and H_1 is a model or hypothesis for this string that the size of it in bit is denoted by $DL(H_1)$, then the size of bit required for encoding data by the H is $D(S|H_1)$. If we can find repeated pattern in original string and encode the difference, the size of remaining data can be decreased. If H_1 and H_2 are two models, and $DL(H_2) + DL(S|H_2) < DL(H_1) + DL(S|H_1)$, we prefer to use H_2 as model because it compresses the original string more than H_1 (Grünwald et al. 2005).

The MDL requires discrete data, but in our problem, the time-series data are real-valued. Thus, it is required to transform real data to discrete data by using discretization. In discretization, the real numbers are cast to a reduced cardinality version. As has been shown in Rakthanmanon et al. (2012), the cardinality reduction in real-valued time-series data does not reduce the accuracy of their classification. Various time series have different offsets and amplitudes. Thus, time-series normalization is inevitable. Without normalization, comparing time series is meaningless (Keogh and Lin 2005). We have used an adapted version of normalization that has been shown in Rakthanmanon et al. (2012) and is suitable for discrete data. The DNorm function is a discrete normalization function that normalizes a real-valued time series such as T into b-bit discrete value (Eq. 14). We have used 6 for b (64 value).

The proposed approach for model generation has been shown in Listing 3. As shown in this figure, first the best cluster (the cluster whose center has minimum distance to current time series) is determined (line 3). Then, the best length of query is determined (line 5). This part is the heart of our proposed approach that was inspired from the MDL concept. This is done by calling determine_query_length function. In this function, various lengths from min to max for query window are checked and the length that compresses active time series more than the other is selected (lines 27–49). After the determination of best query length, all subsequences are extracted from active time series as well as the medoid that is the most similar to the active time series (lines 7 and 8). In lines 9–11 of the proposed algorithm, the DTW distance of extracted subsequences is calculated.

$$DNorm(T) = round\left(\left(\frac{T - \min}{\max - \min}\right) \times \left(2^b - 1\right)\right) + 1 \tag{14}$$

3.3.4 Prediction

In the prediction phase, p subsequences that have the lowest DTW distance to the query are selected. Afterward, the mean of their next values is calculated and results in the prediction value. Lines 12 to 14 of Listing 3 are related to the prediction phase.

3.3.5 An Example

Consider Fig. 4. This figure shows a QoS time series of WS-DREAM dataset when user x invokes service y. Suppose we want to predict the value of this time series at slot 61. The values of slots 1–60 constitute active window. This window is disported to two other windows, query window and past window. The query window consists of the most recent time slots than the previous time slot which should be predicted. The length of this window can be between 3 and max, which is application based and is determined

based on trial and error. Indeed, the length of query window is determined by applying the MDL algorithm. That length which has the maximum number of matches in the past window is the best one. In this example, the best length is 4. The query window consists of time slots 57–60. This window has four matches in the past window, and based on MDL, it is the best length. After finding the best length, all subsequences in the past window, as well as the centroid which is the most similar to the active window, are extracted and sorted with respect to their distance to the query window. The predicted value is the mean of next value in the first p subsequences. Indeed, in our proposed method, the prediction is done based on frequently repeating patterns in current time series and other most similar time series.

Input: M,Q_{active}, Q_{query},C, k, cur, active user au and active service as.

Output: predicted QoS at the time interval $cur + 1$ for the active user u_a and active service s_a.

Beginning of algorithm
$SDB \longleftarrow$ All series in M;
$selected - cluster \longleftarrow \arg min_{\mu \in C} \left(DTW\left(Q_{query}, \mu\right)\right)$.
$SSDB \longleftarrow \emptyset$; //The abbreviation of sub-series data base.
$qw_{length} \longleftarrow$ determine_query_length(min, max, cur, Q_{active});
$SSDB \longleftarrow SSDB \cup$ get_sub_series (Q_{active}, qw_length, $cur - 1$);
$s \longleftarrow the\ medoid\ of selected - cluster$
$SSDB \longleftarrow SSDB \cup$ get_sub_series (s, qw_length,$|s|$);
For each ss in $SSDB$**do**:
    $ss.dtw \longleftarrow DTW(\mathrm{Q}_{\mathrm{query}}, ss[1: last - 1])$;
End for
$p - best - subseries \longleftarrow$Select the p subseries such ss from $SSDB$which have the lowest $ss.dtw$.
$predicated - QoS \longleftarrow average_{ss \in p-best-subseries}(ss[last])$;
Return $predicated - QoS$;
End algorithm
Function get_sub_series($series$, qw_length, end)
$sub - series \longleftarrow \emptyset$;
$i \longleftarrow 1$;
$j \longleftarrow qw\_length$;
While($j \neq end$) do
    $sub - series \longleftarrow sub - series \cup series[i: j]$;
    $i \longleftarrow i + 1$;
    $j \longleftarrow j + 1$;
End while
Return $sub - series$;
End function
Function determine_query_length*(minimum, maximum, current, activeSeries)*
  maxBitSave ⟵ 0;
  bestLength ⟵ minimum;
  For length=minimum **to** maximum **do:**
    query ⟵ $activeSeries\ [current - length + 1: current]$;
    dNormQuery ⟵ dNorm(query); /* see eq. 14 */
    huffmanQuery ⟵ avgHuffman(dNormQuery);
    seqDB ⟵ get_sub_series($activeSeries$, $length$, end);
    totalBitSave ⟵ 0;
    For each seq in seqDB **do:**
      dNormSeq ⟵ dNorm(Seq); /* see eq. 14 */
      huffmanSeq ⟵ avgHuffman(dNormSeq);
      diff ⟵ dNormQuery – dNormSeq;
      huffmanDiff ⟵ avgHuffman(diff);
      bitSave ⟵ huffmanSeq – huffmanDiff;
      totalBitSave ⟵ totalBitSave + bitSave;
    End for
    If totalBitSave > maxBitSave **then**
      maxBitSave ⟵ totalBitSave;
      bestLength ⟵ length;
    End if
  End for
  Return bestLength
End function

Listing 3. Proposed algorithm for model generation and prediction

4 Experiments

The experimental part of each work is associated with the evaluation results. To evaluate the performance of our proposed hybrid method, we implemented several experiments. Our first experiment focuses on the evaluation of clustering results. The second experiment aims at evaluating the impact of different parameters on the performance of the proposed approach. The third experiment results in comparing the proposed algorithm with other state-of-the-art methods. All experiments have been implemented using JDK 8.0 and NetBeans IDE 8.2. They have been deployed in Windows Server 2012. The specification of the implementation server is listed in Table 2.

4.1 Dataset

To evaluate the proposed approach by means of the real-world data, we choose WS-DREAM dataset #3[1] (Zheng et al. 2014). This dataset contains the QoS records of service invocations on the 4532 distributed Web services located across 57 countries invoked through 142 distributed computers in 22 countries from PlanetLab. The dataset covers a collection of 16 h QoS data and consists of 64 time intervals (every 15 min). This dataset is time aware and suitable for large experiments. It contains a large number of observations (491,460 time series) and also a relatively sufficient number of time slots (64 slots). In the case of time slots interval, it consists of a few minutes of intervals between two continuous time slots as a suitable case for the dynamic behavior of time series. Further, details of the dataset are given in Table 3. In this table, some statistics of WS-DREAM such as mean and standard deviation have been depicted.

The WS-DREAM dataset consists of two three-dimensional matrices, each $142 \times 4532 \times 64$ as user, service, and time. The first matrix represents response time (RT) and the second one throughput (TP). We continued our tests for RT, which is the time between sending a request and receiving the corresponding response. The details of RT matrix are given in Table 4. The coefficient of variation (CV) depicts the standard deviation of mean values to measure the dispersion. Due to the varied range of dispersion in RT that illustrates the big CV, learning process and predicting step can be difficult. The first quartile determines the middle value between the smallest and median values of the dataset. The third quartile is defined as a middle value between the median and the highest values in the dataset. These values show the dispersal of these datasets.

[1] https://github.com/wsdream/wsdream-dataset/tree/master/dataset2.

According to the boxplot of the RT matrix (Fig. 5), 50% of the RT values are between 0.186 and 1.665, and 75% of them are less than 1.665. The thick line above the top whisker shows that the RT matrix contains outliers. There is noisy data in our dataset such as zero values for RT or the inserted 20 s for RT values larger than 20(s) (Wang et al. 2016). We eliminated the series with more than 12 noisy inputs from our dataset. The noisy data were corrected using the previously stated preprocessing phase. After preprocessing, the total number of 84,017 time series was eliminated from the RT matrix. Furthermore, the total number of 89,779 time series was corrected using EWMA from the RT matrix in the preprocessing.

4.2 Evaluation Metrics

Various evaluation metrics have been used in this work to evaluate the accuracy of proposed hybrid time-aware approach. For the evaluation of clustering part, we used the cohesion (intra-cluster distance), which is the most representative and popular one (Bobadilla et al. 2018), for the evaluation of clustering (Wang et al. 2018; Fulcher 2018). This metric determines how near time series in a cluster are to the cluster medoids. The equation of this metric is as follows:

$$cohesion = \sum_{j=1}^{k} \sum_{\forall Q \in \mu_j} similarity(Q, \mu_j) \tag{15}$$

where k is the number of clusters, j is the medoid of ith cluster, Q is a time series, and *similarity* is selected metric (Pearson, Euclidean, etc.). We use *PCC* as a similarity measure, which is a type of CF similarity measure and returns $[-1, 1]$. The larger *PCC* value shows that the two time series are more similar. The case of $PCC = 1$ means that the two time series are the same. The larger cohesion value shows better clustering quality. If $q_t^{d,e}$ and $q_t^{y,z}$ are two time series, the *PCC* will be calculated as follows:

$$PCC(q_t^{d,e}, q_t^{y,z}) = \sum_{i=1}^{x} \left(\left(q_i^{d,e} - \overline{q_t^{d,e}} \right) \left(q_i^{y,z} - \overline{q_t^{y,z}} \right) \right) / \left(\sqrt{\sum_{i=1}^{x} \left(q_i^{d,e} - \overline{q_t^{d,e}} \right)^2} \sqrt{\sum_{i=1}^{x} \left(q_i^{y,z} - \overline{q_t^{y,z}} \right)^2} \right) \tag{16}$$

where x is the length of time series, $q_i^{d,e}$ and $q_i^{y,z}$ are the RT values of time series in the time slot i, and $\overline{q_t^{d,e}} = \frac{1}{x} \sum_{i=1}^{x} q_i^{d,e}$ and analogously for $\overline{q_t^{y,z}}$.

Furthermore, mean absolute error (MAE) (Zheng et al. 2011; Kahaki et al. 2018) and root-mean-square error (RMSE) (Zheng et al. 2011; Kahaki et al. 2018) are used for evaluation of QoS prediction. A smaller MAE or RMSE value means better performance. The MAE equation is as follows:

$$MAE = \sum_i \left| q_i^{d,e} - \widehat{q_i^{d,e}} \right| / N \tag{17}$$

And RMSE is defined as follows:

$$RMSE = \sqrt{\sum_i \left(q_i^{d,e} - \widehat{q_i^{d,e}} \right)^2 / N} \tag{18}$$

where $q_i^{d,e}$ denotes the real value of QoS time series, $q_t^{d,e}$, in the time slot t_i and $\widehat{q_i^{d,e}}$ is the predicted value of $q_t^{d,e}$ at t_i, and N is the total number of test cases.

4.3 Results

Experimental results are being shown from two different perspectives: (1) clustering and (2) prediction. For the prediction phase, we conducted two evaluation types, each including various experiments. The first type is for parameter tuning and the second one is to compare it with related work. In the first type, the dataset is divided into training (60%) and test data (40%). In the test phase, one time slot is being removed randomly from each time series to be predicted.

4.3.1 Clustering Results

For evaluation of the k-medoids algorithm based on the WS-DREAM dataset #3, we repeat proposed algorithm 10 times for various cluster medoids (k) on the dataset and accordingly plot the average of the cohesion metric. The number of cluster medoids is 5, 10, 25, 50, and 100. As shown in Fig. 6, by increasing cluster medoids, the quality of clustering becomes better, and the value of time-series inter-similarity increases. The cohesion is the metric that has been used to evaluate the clustering quality. In the cohesion metric, we have used PCC to calculate time-series similarity. The larger the cohesion value is, the better the quality will be. The best value of cohesion for the RT matrix is related to 100. According to Eq. 15, by increasing the number of clusters the similarity between each time series and related cluster center becomes more. The measure shows intra-similarity of clusters. As shown in this figure, when the number of clusters is 100, it shows better cluster quality. It should be noted that for k > 100, there were some empty clusters, i.e., cluster with only one member (cluster medoid).

Table 2 Specifications of the used server in our experimental evaluation

CPU	Intel Xeon
Number of processors	2
CPU speed	2.93 GHz
RAM	128 GB

4.3.2 Parameters Tuning

For understanding the impact of various parameters on the proposed approach, we have implemented several experiments. A number of experiments are conducted to evaluate the effects of various parameters in the proposed approach. In the first experiment, we investigated the impact of the cluster quality on the prediction quality. The impact of cluster quality on prediction is studied in the first experiment.

In this experiment, we ran our approach with different numbers of clusters and calculated the prediction accuracy using MAE. Figure 7 shows the achieved experimental results for this experiment. As shown in this figure, the performance of our prediction approach for various numbers of clusters is near to real numbers, when the number of selected similar subseries changes between 1 and 13 and decreases by increasing the number of selected similar subseries. When the number of selected similar subseries is 13, the difference between the best case (5) and the worst case (100) is about 0.02, which is small. Also, the mean run-time of algorithm in the second for different numbers of clusters is shown in Fig. 8. As can be seen in this figure, by increasing the number of clusters, the running time of our algorithm increases. In fact, by increasing the number of clusters, the number of comparisons increases as well, causing the run-time of algorithm to increase. The best run-time is related to $k = 5$. In view of the fact that our algorithm is online, and that time is a vital factor, and that the best MAE in the number of selected similar subseries equal to 13 is related to number of clusters = 5, we have used this number of clusters in the following experiments.

In the second experiment, we analyze the impact of the total number of selected subseries that is similar to query window on the prediction accuracy to gain its best number. We ran our algorithm on 160,977 test cases. We set other parameters as follows: The number of clusters is 5, and the position of the predicted values in time series changes between 24 and 64. As shown in Fig. 7, the best case is related to $p = 13$ and the worst case is $p = 1$. The accuracy of our algorithm is increased by more similar subseries so that the prediction error can be decreased by the growing number of selected similar subseries p.

The place of query window has been investigated in the third experiment. In this regard, we change the place of query window and test four cases: the first quartile, the second quartile, the third quartile, and the final quartile. The number of clusters is 5. The first location related to each case is 14, 25, 37, and 49, respectively. For each case, we randomly select 160,977 numbers of time series,

Table 3 WS-DREAM dataset characteristics (Zheng et al. 2014)

Statistics	Values
Number of WS invocations	30,287,611
Number of service users	142
Number of WS	4532
Number of user countries	22
Number of WS countries	57
Number of time slots	64
Interval of time slots	15 min
Mean of response time	3.165 s
Standard deviation of response time	6.12 s
Mean of throughput	9.609 kbps
Standard deviation of throughput	50.11 s

Table 4 Characteristics of the response time matrix

Statistics	Response time
Mean of all values	3.165 (s)
Standard deviation of all values	6.119
Coefficient of variation	193.335%
Num. of 20 values	2,777,983
Num. of 0 values	116,120
Total number of initial time series	491,460
Num. of time series that have 20 in all time step	4227
Num. of time series that have 0 in all time step	0
Num. of deleted time series after preprocessing	84,017
Num. of corrected time series after preprocessing	89,779
First quartile	0.186
Median	0.439
Third quartile	1.665

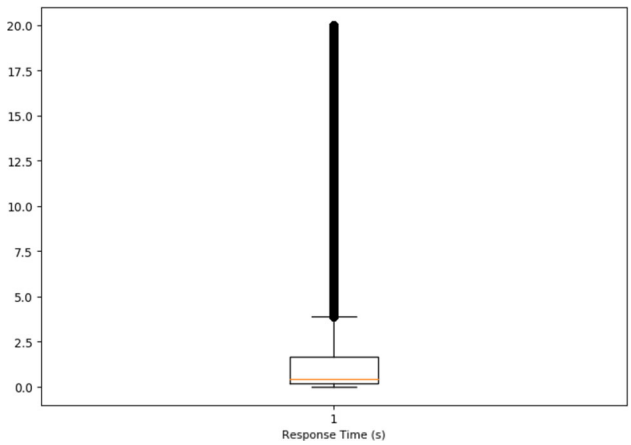

Fig. 5 Boxplot of the QoS values related to RT matrix

remove one value after query window and predict it. As shown in Fig. 9, the best case is related to third quartile. In this case, our algorithm compares query window with the previous values in the current time series and with the cluster center. Due to the shape of QoS time series in the WS-DREAM dataset and the large fluctuation in the fourth quartile, the result of this location is worse.

4.3.3 Comparison with Related Work

For the evaluation of experimental results, we considered four baselines, which used the WS-DREAM dataset. In studied baselines, the average of the last three values, UPCC (Shao et al. 2007), IPCC (Zheng et al. 2011), combined UPCC and IPCC (Zheng et al. 2009), and LASSO (Wang et al. 2016) are used for prediction. In the UPCC method, the QoS values of all other users in the current time slot are considered for prediction, whereas we use the most similar time series (paired user-service set). Mutually, IPCC uses all other service data for prediction. In our experimental trials, in order to reach a matrix density of 60%, we removed 40% of time series from data. The 40% of time series-data is removed from the dataset, so the size of density matrix is 60%. To create the test cases, 20% of QoS parameters are randomly removed from the remaining time-series data. In each test case, just one value is missing to be predicted. The number of clusters is 5. As shown in Fig. 10, proposed hybrid time-aware QoS prediction method shows better prediction accuracy in terms of MAE and RMSE, compared with the baselines. Additionally, our proposed approach and LASSO show better prediction accuracy compared with all other baseline methods. It should be stated that both of these methods and taSR are time aware and use time feature as additional information for the QoS prediction.

In the case of LASSO (Wang et al. 2016), a simple replacing method has been used in LASSO for missing value imputation which is not accurate in the case of sudden change in QoS values and affects the prediction accuracy. LASSO uses the geolocation information of time series to cluster them as well. In this case, it is possible that two different time series with the same location of user and service have different values due to factors such as load of server or the different used devices by users. Also, our approach uses intra-similarity (the similarity between different subsequence of current time series) as well as inter-similarity between different time series but LASSO only uses inter-similarity. Also, our approach outperforms the taSR (Ding et al. 2018). In this case, taSR only predicts the last value of time series using ARIMA. As stated in the literature, ARIMA is only appropriate for time series which satisfies serial dependency, stationary, normality, and invertibility assumptions (Box et al. 2015). In taSR, the

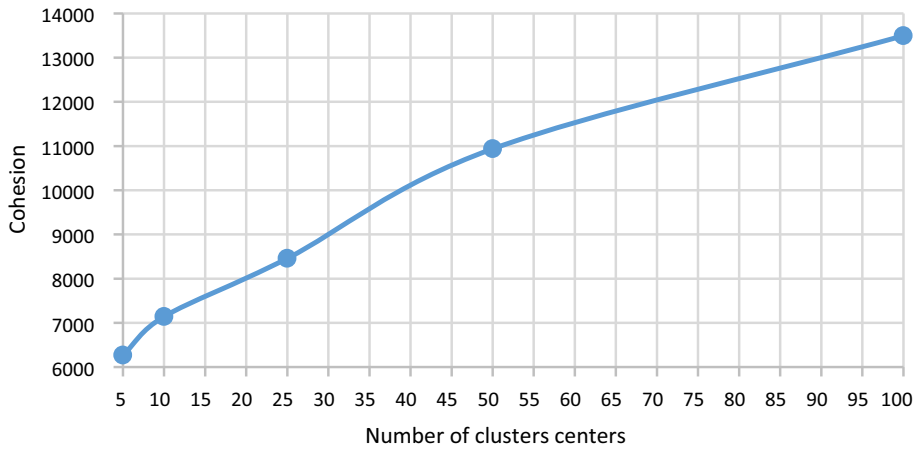

Fig. 6 Impact of number of clusters on cohesion for RT matrix

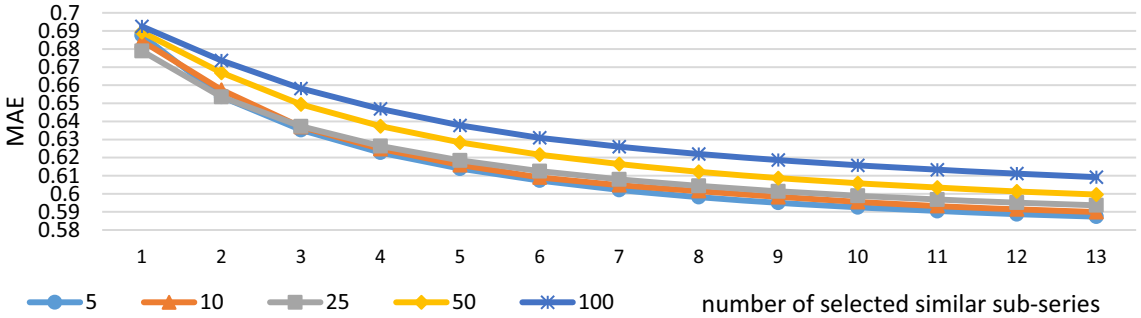

Fig. 7 Impact of the number of clusters and the number of selected similar subseries on MAE

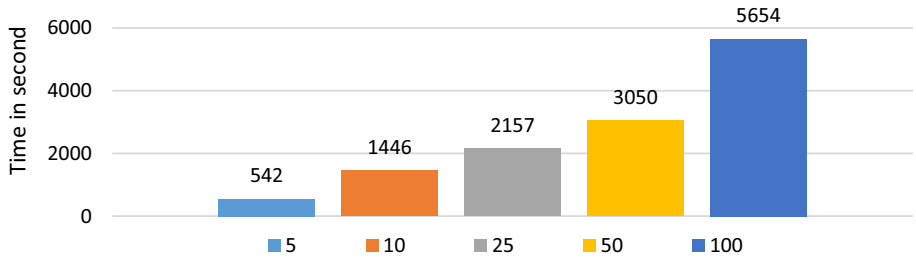

Fig. 8 Mean running time of our algorithm for different numbers of clusters

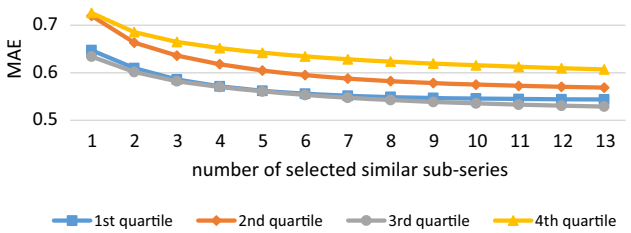

Fig. 9 The impact of query window location in QoS time series on MAE

prediction has been done without investigation of time series regarding the mentioned assumptions. Also, in the case of other time slots except the last one, a customized CF approach has been applied for prediction, which is not accurate.

As a result, we should state that the main highlight of our work comparing with the existing works is that we consider the similarity between time series themselves, in which other works consider only the similarity between user or services. Furthermore, our proposed method is shape-based and considers subsequences that are similar to the query window in current time series and time series of a medoid of a cluster that has the most similarity. Additionally, we consider time zone differences between geolocations of clients and services. The DTW alignment is one-to-many, and due to this, it uses the other time-series data in prediction even in the cases where the time zone of the other time-series users or services is different.

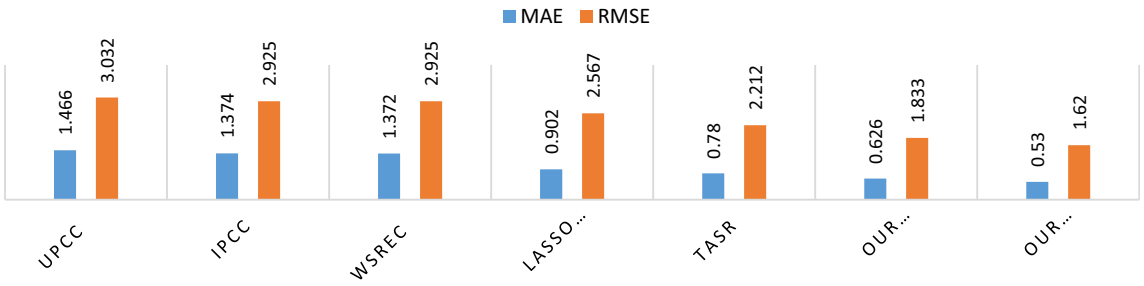

Fig. 10 Comparison of the proposed hybrid time-aware approach with the related work

5 Conclusion

The main goal of the proposed hybrid time-aware approach is to support service composition in multi-cloud by predicting the next state of the service features. In particular, we benefited from MDL, k-medoid as well as DTW methods. As a result, it can also be used in service migration and SLA violation, if service providers are aware of their future resource needs as well as available ones. The proposed approach consists of four major phases: (1) preprocessing, (2) clustering, (3) model generation, and (4) prediction. In the first phase, noisy data are removed from the dataset. The noisy data can be either missing data or invalid data. As a next phase, k-medoids are used for clustering time series of service features. The model generator uses MDL to discover similar patterns from time series for constructing the predictor. The prediction is made by providing the mean of nearest values to the querying values. The quarrying values are current and recent resources used by the current user. We used WS-DREAM dataset and have applied the proposed hybrid time-aware approach on time series of response time in the dataset. The experimental results as discussed show 0.5 mean absolute error, which is improved compared with the baseline. As contributions to this work, we tried to consider the changes over time, model them in a sort of time series and also fulfill noisy and missing data with statistically repaired semi-real data. This, of course, improves the quality of training and prediction algorithms. The use of DTW algorithm covers the time zone differences between geographical regions. This is important because, in multi-cloud, if users want to use different cloud services, the physical location of different clouds is not clear to the users and may vary. Moreover, the provided prediction is a shape-based method, which approved better results in the prediction.

As a future work, we will investigate the multivariate time-series prediction and the impact and correlation of various QoS features on each other. We plan to use FastDTW instead of DTW to reduce run time and check the quality of prediction as well as to use multi-step-ahead prediction for prediction of not only next values but also time series of future values. We are concerned about the cases that do not have enough training and query window data and plan to further develop algorithms in this regard, which is known as cold-start problem in recommender systems area.

References

Aghabozorgi S, Shirkhorshidi AS, Wah TY (2015) Time-series clustering—a decade review. Inf Syst 53:16–38

Amin A, Colman A, Grunske L (2012) An approach to forecasting QoS attributes of web services based on ARIMA and GARCH models. In: 2012 IEEE 19th International Conference on Web Services (ICWS), pp 74–81

Amin A, Grunske L, Colman A (2012) An automated approach to forecasting QoS attributes based on linear and non-linear time series modeling. In: 2012 Proceedings of the 27th IEEE/ACM international conference on Automated Software Engineering (ASE), pp 130–139

Amiri M, Mohammad-Khanli L, Mirandola R (2018) A sequential pattern mining model for application workload prediction in cloud environment. J Netw Comput Appl 105:21–62

Arianyan E, Taheri H, Khoshdel V (2017) Novel fuzzy multi objective DVFS-aware consolidation heuristics for energy and SLA efficient resource management in cloud data centers. J Netw Comput Appl 78:43–61

Barron A, Rissanen J, Yu B (1998) The minimum description length principle in coding and modeling. IEEE Trans Inf Theory 44(6):2743–2760

Berndt DJ, Clifford J (1994) Using dynamic time warping to find patterns in time series. In: KDD workshop, vol 10, pp 359–370

Birattari M, Bontempi G, Bersini H (1999) Lazy learning meets the recursive least squares algorithm. Adv Neural Inf Process Syst, pp 375–381

Bobadilla J, Bojorque R, Esteban AH, Hurtado R (2018) Recommender systems clustering using Bayesian non negative matrix factorization. IEEE Access 6:3549–3564

Bohlouli M, Analoui M (2008) Grid-hpa: predicting resource requirements of a job in the grid computing environment. World Acad Sci Eng Technol 21:747–751

Bohlouli M, Holland A, Fathi M (2011) Knowledge integration of collaborative product design using cloud computing infrastructure. In: 2011 IEEE international conference on electro/information technology (EIT), pp 1–8

Bohlouli M, Merges F, Fathi M (2014) Knowledge integration of distributed enterprises using cloud based big data analytics. In: IEEE international conference on electro/information technology, pp 612–617. https://doi.org/10.1109/eit.2014.6871835

Bollerslev T (1986) Generalized autoregressive conditional heteroskedasticity. J Econom 31(3):307–327
Bontempi G (2000) Local learning techniques for modeling, prediction and control. Ph.D. thesis, IRIDIA- Universite Libre de Br
Box GE, Jenkins GM, Reinsel GC, Ljung GM (2015) Time series analysis: forecasting and control. Wiley, Hoboken
Breese JS, Heckerman D, Kadie C (1998) Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of the fourteenth conference on Uncertainty in artificial intelligence, pp 43–52
Cases U (2010) Functional requirements for inter-cloud computing
Dau HA et al (2018) Optimizing dynamic time warping's window width for time series data mining applications. Data Min Knowl Discov 32(4):1074–1120
Ding S, Li Y, Wu D, Zhang Y, Yang S (2018) Time-aware cloud service recommendation using similarity-enhanced collaborative filtering and ARIMA model. Decis Support Syst 107:103–115
Engle RF (1982) Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. Econom J Econom Soc 987–1007
Fulcher BD (2018) Feature-based time-series analysis. In: Feature engineering for machine learning and data analytics. CRC Press, pp 87–116
Garg SK, Versteeg S, Buyya R (2013) A framework for ranking of cloud computing services. Future Gener Comput Syst 29(4):1012–1023
Grozev N, Buyya R (2014) Inter-cloud architectures and application brokering: taxonomy and survey. Softw Pract Exp 44(3):369–390
Grünwald PD, Myung IJ, Pitt MA (2005) Advances in minimum description length: theory and applications. MIT Press, Cambridge
Hayyolalam V, Kazem AAP (2018) A systematic literature review on QoS-aware service composition and selection in cloud environment. J Netw Comput Appl 110:52–74
Hu B, Rakthanmanon T, Hao Y, Evans S, Lonardi S, Keogh E (2011) Discovering the intrinsic cardinality and dimensionality of time series using MDL. In: 2011 IEEE 11th international conference on data mining, pp 1086–1091
Hu Y, Peng Q, Hu X (2014) A time-aware and data sparsity tolerant approach for web service recommendation. In: 2014 IEEE International Conference on Web Services, pp 33–40
Kahaki SM, Arshad H, Nordin MJ, Ismail W (2018) Geometric feature descriptor and dissimilarity-based registration of remotely sensed imagery. PLoS ONE 13(7):e0200676
Kaufman L, Rousseeuw PJ (2009) Finding groups in data: an introduction to cluster analysis. Wiley, Hoboken
Keogh E, Lin J (2005) Clustering of time-series subsequences is meaningless: implications for previous and future research. Knowl Inf Syst 8(2):154–177
Keshavarzi A, Haghighat AT, Bohlouli M (2017) Adaptive resource management and provisioning in the cloud computing: a survey of definitions, standards and research roadmaps. KSII Trans Internet Inf Syst 11(9):4280–4300
Keshavarzi A, Haghighat AT, Bohlouli M (2020) Enhanced time-aware QoS prediction in multi-cloud: a hybrid k-medoids and lazy learning approach (QoPC). Comput 102(4):923–949
Kiani R, Keshavarzi A, Bohlouli M (2020) Detection of thin boundaries between different types of anomalies in outlier detection using enhanced neural networks. Appl Artif Intell 34(5):345–377
Liaqat M et al (2017) Federated cloud resource management: review and discussion. J Netw Comput Appl 77:87–105
Lin F, Zeng W, Yang L, Wang Y, Lin S, Zeng J (2017) Cloud computing system risk estimation and service selection approach based on cloud focus theory. Neural Comput Appl 28(7):1863–1876
Luo X, Lv Y, Li R, Chen Y (2015) Web service QoS prediction based on adaptive dynamic programming using fuzzy neural networks for cloud services. IEEE Access 3:2260–2269
Machado GM, Maran V, Dornelles LP, Gasparini I, Thom LH, de Oliveira JPM (2018) A systematic mapping on adaptive recommender approaches for ubiquitous environments. Computing 100(2):183–209
Mell P, Grance T (2011) The NIST definition of cloud computing
Menascé DA (2002) QoS issues in web services. IEEE Internet Comput 6(6):72–75
Mueen A, Chavoshi N, Abu-El-Rub N, Hamooni H, Minnich A, MacCarthy J (2018) Speeding up dynamic time warping distance for sparse time series data. Knowl Inf Syst 54(1):237–263
Rakthanmanon T, Keogh EJ, Lonardi S, Evans S (2012) MDL-based time series clustering. Knowl Inf Syst 33(2):371–399
Rissanen J (1978) Modeling by shortest data description. Automatica 14(5):465–471
Sarwar B, Karypis G, Konstan J, Riedl J (2001) Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th international conference on World Wide Web, pp 285–295
Shao L, Zhang J, Wei Y, Zhao J, Xie B, Mei H (2007) Personalized qos prediction forweb services via collaborative filtering. In: IEEE International Conference on Web Services, 2007. ICWS 2007, pp 439–446
Terveen L, Hill W (2001) Beyond recommender systems: helping people help each other. HCI New Millenn 1(2001):487–509
Vakili A, Navimipour NJ (2017) Comprehensive and systematic review of the service composition mechanisms in the cloud environments. J Netw Comput Appl 81:24–36
Wang X, Zhu J, Zheng Z, Song W, Shen Y, Lyu MR (2016) A spatial-temporal QoS prediction approach for time-aware Web service recommendation. ACM Trans Web TWEB 10(1):7
Wang S, Wang S, Yuan H, Li Q, Geng J, Yu Y (2018) Clustering by differencing potential of data field. Computing 100(4):403–419
Wu J, Chen L, Feng Y, Zheng Z, Zhou MC, Wu Z (2013) Predicting quality of service for selection by neighborhood-based collaborative filtering. IEEE Trans Syst Man Cybern Syst 43(2):428–439
Wu C, Qiu W, Zheng Z, Wang X, Yang X (2015) QoS prediction of web services based on two-phase k-means clustering. In: 2015 IEEE International Conference on Web Services (ICWS), pp 161–168
Wu H, Yue K, Li B, Zhang B, Hsu C-H (2018) Collaborative QoS prediction with context-sensitive matrix factorization. Future Gener Comput Syst 82:669–678
Ye Z, Mistry S, Bouguettaya A, Dong H (2016) Long-term QoS-aware cloud service composition using multivariate time series analysis. IEEE Trans Serv Comput 9(3):382–393
Yu C, Huang L (2016) A Web service QoS prediction approach based on time-and location-aware collaborative filtering. Serv Oriented Comput Appl 10(2):135–149
Zhang Y, Zheng Z, Lyu MR (2014) An online performance prediction framework for service-oriented systems. IEEE Trans Syst Man Cybern Syst 44(9):1169–1181
Zheng Z, Ma H, Lyu MR, King I (2009) Wsrec: a collaborative filtering based web service recommender system. In: IEEE International Conference on Web Services, 2009. ICWS 2009, pp 437–444
Zheng Z, Ma H, Lyu MR, King I (2011) Qos-aware web service recommendation by collaborative filtering. IEEE Trans Serv Comput 4(2):140–152
Zheng Z, Zhang Y, Lyu MR (2014) Investigating QoS of real-world web services. IEEE Trans Serv Comput 7(1):32–39