electronics

MDPI

Article

Towards Dynamic Reconfiguration of a Composite Web Service: An Approach Based on QoS Prediction

Abdessalam Messiaid [1,*], Farid Mokhati [1], Rohallah Benaboud [1] and Hajer Salem [2]

[1] Department of Mathematics and Computer Science RelaCS2 Laboratory, University Larbi Ben Mhidi, Oum El Bouaghi 04000, Algeria; mokhati@yahoo.fr (F.M.); r_benaboud@yahoo.fr (R.B.)
[2] AUDENSIEL Pôle R&D, 92100 Boulogne-Billancourt, France; h.salem@audensiel.fr
* Correspondence: a_messiaid@esi.dz

Abstract: Service-oriented architecture provides the ability to combine several web services in order to fulfil a user-specific requirement. In dynamic environments, the appearance of several unforeseen events can destabilize the composite web service (CWS) and affect its quality. To deal with these issues, the composite web service must be dynamically reconfigured. Dynamic reconfiguration may be enhanced by avoiding the invocation of degraded web services by predicting QoS for the candidate web service. In this paper, we propose a dynamic reconfiguration method based on HMM (Hidden Markov Model) states to predict the imminent degradation in QoS and prevent the invocation of partner web services with degraded QoS values. PSO (Particle Swarm Optimization) and SFLA (Shuffled Frog Leaping Algorithm) are used to improve the prediction efficiency of HMM. Through extensive experiments on a real-world dataset, WS-Dream, the results demonstrate that the proposed approach can achieve better prediction accuracy. Moreover, we carried out a case study where we revealed that the proposed approach outperforms several state-of-the-art methods in terms of execution time.

Keywords: web service; dynamic reconfiguration; CWS; HMM; QoS prediction; SFLA; PSO

Citation: Messiaid, A.; Mokhati, F.; Benaboud, R.; Salem, H. Towards Dynamic Reconfiguration of a Composite Web Service: An Approach Based on QoS Prediction. *Electronics* **2021**, *10*, 1597. https://doi.org/10.3390/electronics10131597

Academic Editor: Feliz Gouveia

Received: 10 May 2021
Accepted: 28 June 2021
Published: 2 July 2021

1. Introduction

Web services, as software applications exposed on the web, are invoked by users to meet their needs. Responding to a user request often requires the composition of several web services, resulting in a composite web service (CWS). The invocation of a pre-composed web service is subject to various unexpected changes that can affect the quality of service (QoS), besides the functionality and the environment in which the application operates. Therefore, a dynamic reconfiguration of the composite web service is essential to ensure that it can adapt to various unanticipated QoS changes. However, using a reactive method for reconfiguration and service continuity takes a long time [1] and affects the performance [2], from selecting the best service to reconfiguring the entire process, hence, a proactive or predictive method for impending problems in the QoS is one of the most critical suggested solutions in the dynamic reconfiguration.

In the proactive approach, preventing the invocation of web services that are subject to imminent degradation in QoS is a major challenge; this is evident, firstly, in the case of several degraded web services during the workflow process, where it becomes difficult to apply the dynamic reconfiguration service, starting from selecting the appropriate service until the completion of the workflow. Secondly, after performing the dynamic reconfiguration of the degraded web services and replacing them with other services similar to those in the functionality, the latter may experience an imminent degradation in QoS, which requires the request of the dynamic reconfiguration service again or several times. One of these situations may impede the proper functioning of the CWS process and consequently result in a dissatisfied user.

Several efforts have been made to cover some of the stated challenges. However, enhancing the dynamic reconfiguration of a composite web service in a dynamic environment remains a major challenge. In this regard, and to address the above challenges, a prediction of imminent degradation in QoS to prevent partner web services' invocation with degraded QoS values is proposed in this paper. We propose an improved HMM-based agent to predict the imminent degradation of QoS of web services. The prediction accuracy of HMM is improved by using a hybrid meta-heuristic algorithm (SFLA–PSO). We propose to recognize the potential states that web services could be in throughout their mission using QoS historical data and the k-means clustering method. To the best of our knowledge, this is the first time that the QoS state has been used for QoS prediction. The contribution of this work is fourfold, as follows:

1. An improved dynamic reconfiguration method which avoids invoking degraded web services by predicting QoS for the candidate web service;
2. A new hybrid algorithm of the SFLA–PSO and Baum–Welch algorithms to improve the HMM initial emission matrix;
3. The QoS prediction problem is reformulated into a QoS state prediction problem, where we consider QoS states rather than QoS values and consider that a QoS state is composed of several attributes;
4. The proposed approach can predict the QoS according to each user's preferences and can predict the state of similar services without the need for their data (QoS).

The rest of the paper is organized as follows: Section 2 mentions some essential works related to the prediction of degradation in QoS; Section 3 presents an overview of the proposed approach. Section 4 details the mechanism of the proposed approach. In Section 5, we discuss the experimental results and we present a case study. Finally, the conclusion and directions for future work are presented in Section 6.

2. Related Work

State-of-the-art web service QoS prediction approaches have been widely studied in the literature. Generally, the proposed methods can be classified into two main categories: approaches for recommendation and selection services [3–9], and approaches for the monitoring and dynamic adaptation of web services [1,3,10,11]. The first category includes many studies based on the collaborative filtering (CF) algorithm to enable the recommendation and selection of the optimal web service by predicting the unknown QoS values based on historical user data [12–15]. The filtering algorithm can be divided into two kinds [16]: memory-based CF and model-based CF. A personalized online performance prediction framework based on the past usage experience from different users is proposed [14], using collaborative filtering approaches to enable the optimal web service selection. A collaborative QoS prediction approach based on the adaptive matrix factorization is proposed [3] to perform online QoS prediction for candidate services. They used a reactive adaptation when encountering a change. Song et al. [17] proposed a method for the personalized QoS prediction of dynamic web services to predict the QoS value of the requested service, where it finds the users and services similar to the target user and target service, respectively. Xiong et al. [12] proposed a deep hybrid collaborative filtering based matrix factorization approach to improve web service recommendation. Waseem et al. [18] proposed an approach for web service selection based on response time, in which the web service is predicted using the HMM model. The author restricts his work to predicting the web service's behaviour in terms of response time.

Our contribution belongs to the second category. Kahlon et al. [1] proposed a hybrid approach distributed between the service client and the service provider to manage a situation when QoS values degrade. The approach is based on the publish/subscribe mechanism, on the provider side, to communicate the monitoring data to all the clients, and on the client-side, to inform the client of the QoS degradation. It is also based on the exponentially weighted moving average (EWMA) to predict the degradation of QoS value. However, this paper lacks a discussion of the prediction accuracy problem. Aschoff et al. [19] proposed a

ParProAdapt framework that detects the need for changes in the web service composition based on the function approximation and the failure of spatial correlation techniques. In this work, the expected service operation of QoS values was modeled using (EWMA). The authors did not consider the users' context when adapting instances of the same composition. Gao et al. [20] proposed the cloud-edge based dynamic reconfiguration to service workflow. The authors considered the value and cost attributes of the service to evaluate the stability and service invocation, respectively. They used the long short-term memory (LSTM) neural network to predict the stability of services. This study is limited to the mobile environment, which is distinguished by limited resource storage and users that move often. Wang et al. [21] proposed a service composition approach based on QoS prediction and reinforcement learning. Specifically, they used a recurrent neural network to predict the QoS. Chen et al. [22] designed a framework to evaluate the survivability of SOA-based application(s) using the HMM model. The main idea of the proposed framework evolves around monitoring activities based on service logs or run time statistics provided by the service provider. However, their approach is contingent on the service provider; also, the author did not discuss the various hidden states or probabilistic insight of remote web services. Moustafa et al. [23] proposed a proactive approach to web services composition (WSC), which uses the Markov Decision Process (MDP) to model WSC process and uses Q-learning for a Reinforcement Learning technique to adapt to dynamic change in the WSC environments proactively. This approach monitors the WSC to determine proactive adaptation by analyzing the web service execution log's historical data.

Nonetheless, according to [24,25], the proactive approach can anticipate reconfiguration before the problem occurs. Several methods have been proposed in the literature that used a proactive prediction-based approach [14,26,27] for web service composition. Most of these methods predict events that may never occur, which increases the lost time and reduces service reliability [28]. So the use of statistical methods capable of predicting the imminent change in the QoS is crucial. Among these methods is the (HMM) Hidden Markov Model.

HMM has been used in many fields, including industrial applications; a few of the studies on (SOA) Service Oriented Architecture have touched upon this kind of prediction; perhaps the most important of them is proposed in [18,29]. The authors of [18] used the HMM to predict the web service's behavior in terms of response time and selected the optimal web service at run time. The authors of [29] have proposed a model for predicting the QoS-satisfied capability of composited components based on HMM. HMM showed promising results in QoS prediction and should be investigated more deeply.

The framework we propose in this paper differs from the methods mentioned above in that it supports prediction based on user preferences. It also supports predicting the state of similar services without requiring their data. More precisely, the state of the candidate services can be predicted based on the active service's QoS. It also allows the study of the gradient in the degradation of the service state.

3. The Proposed Framework Architecture

The main role of the proposed framework (see Figure 1) is to manage the dynamic reconfiguration of composite web services. This is accomplished by improving the efficiency of dynamic reconfiguration and predicting the imminent degradation of the QoS of web services. A new service predictor that prevents the potential unsatisfied change in non-functional properties is proposed. It avoids invoking web services with degraded QoS values.

For the sake of simplicity, let us consider the following definitions:

- A web service is the main component of the workflow process; it is defined by its domain and QoS attributes;
- An Agent HMM is the agent responsible for the analysis;
- QoS attributes are: response time, throughput, reliability and availability .

The framework is comprised of four components: a workflow manager, a service predictor, service improvement and service reconfiguration detailed as follows:

3.1. Workflow Manager

The Workflow Manager is the main component of the proposed framework; it receives the addresses of the services used in the composition and then transfers them to the predictor service to predict the imminent degradation of QoS. If there is an undesirable change in the QoS of the invoked web service, the workflow manager invokes the improvement service to find similar services that improve the user's preferences and then inform the reconfiguration service to replace the degraded service.

3.2. Service Predictor

The service predictor is the module responsible for predicting impending QoS changes. The proposed processing is as follows: First, the predictor assigns an agent to each web service and the agent monitors the web service by collecting QoS information; second, the QoS information is processed using Box–Cox transformation and normalisation; third, it is clustered according to its current HMM state (which reflects its QoS). Finally, if an agent detects degradation in a web service QoS, it notifies the predictor service, which informs the workflow manager to prepare a substitution of a similar web service with the help of the service reconfiguration.

3.3. Service Improvement

The service improvement module fills the internal database from an external service register using a recommendation algorithm. It selects the service that satisfies the user's preferences and saves a set of service candidates corresponding to the concrete service of each composition. We discuss the recommendation algorithm in future work.

3.4. Service Reconfiguration

The service reconfiguration module aims to optimize the user's preferences and avoid the imminent QoS change. The workflow manager may receive a notification from the predictive module or the improvement module. The notification is received from the predictive module in the case of an imminent change in QoS and is received from the improvement module when a service that better meets users' needs is available. In both situations, the workflow manager sends a message to the reconfiguration module to perform an appropriate substitution using its internal database.

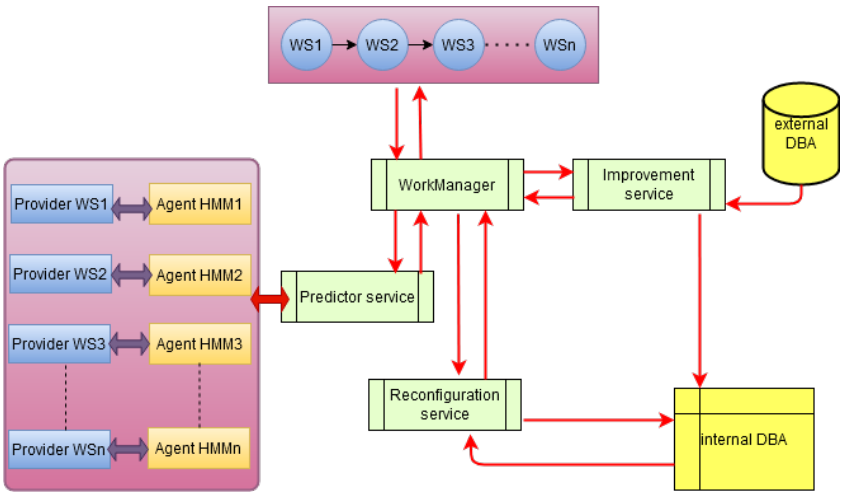


Figure 1. The proposed CWS framework architecture.

In Figure 2, using a UML sequence diagram, we illustrate an overview of our proposed approach to predict impending changes to the suggested web services and dynamic recon-

figuration of composite web services. First, once the client invokes a composite web service to perform a task, the work manager informs the predictor of the selected web services' addresses. Second, the predictor contacts each web service provider through agents to obtain historical QoS data (e.g., response time, throughput, reliability and availability) for them. The agents are responsible for analyzing and processing historical QoS data by applying our proposed approach to predict the impending changes to the suggested web services. Thirdly, historical data for each web service are grouped into clusters to obtain all possible states that may happen to the web service. Fourthly, suppose the agent predicts the existence of a change in the state of a web service; in that case, the predictor is notified and, in turn, reports to the work manager. The latter requests the service reconfiguration to replace the web service the QoS of which is degraded with another one that has a better QoS. We represent the N states $(s_1, \ldots, s_n)$ for each web service (WS), where s_1 represents the normal web service state and s_n represents the faulty state; the states between the two previous states represent the progression in degradation from best state to faulty state.

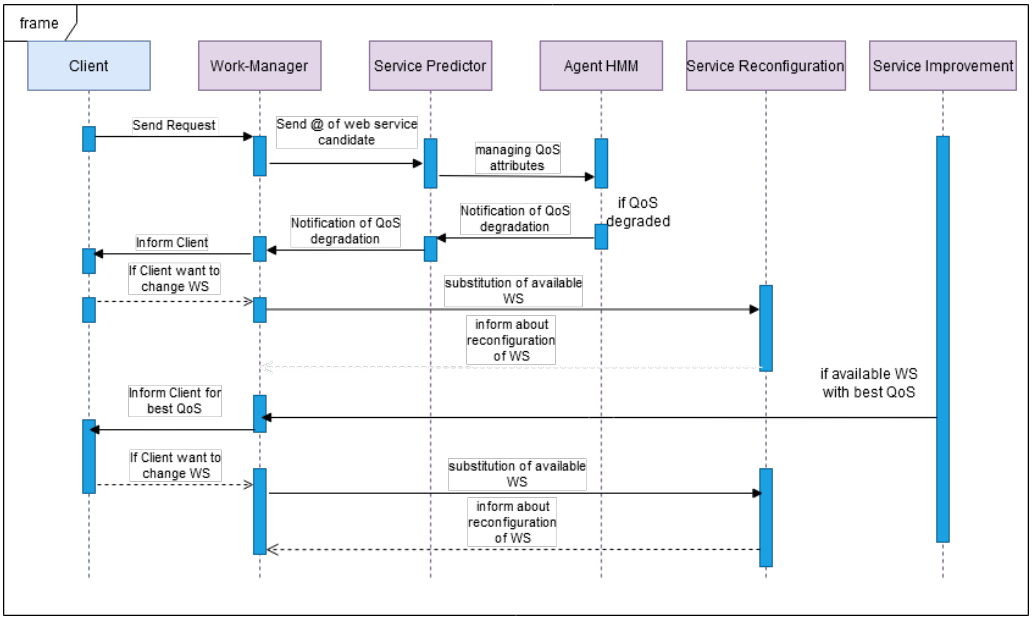


Figure 2. Framework execution process.

In all of the following, we focus on the predictor service, which contains the main contribution of our research paper.

4. A New Dynamic Reconfiguration Process

A new dynamic reconfiguration process is proposed. It is based on three main steps: First, pre-processing the QoS data to be vectorized and normalized; second, the QoS clustering service regroups the similar historical QoS data for each participating web service in the composition, to identify the possible N states that a web service could be in throughout their mission; third, an improved Hidden Markov Model is proposed to assure QoS prediction accuracy.

4.1. Pre-Processing QoS

The QoS attributes, such as response time and throughput, have different ranges; one attribute might overpower another, which negatively affects the efficiency of the clustering algorithm. Moreover, a real-world QoS dataset takes on highly skewed distributions with large variations [3,30]. Therefore, it is necessary to use a method to stabilize the data variance and make the data more normally distributed while setting it in the same range, for example [0, 1]. In our research, we used a combination of the Box–Cox transform method [31] and a normalization method [32] for QoS data. The former was chosen for its

suitability for distributions with outliers and extreme skews, and the latter was chosen for mapping a dataset at the same scale, which is presented as follows:

$$Norm(Qi) = \begin{cases} \frac{(\text{Box cox}(\text{Qi})) - \text{Qmin}}{\text{Qmax} - \text{Qmin}} & \text{if Qmax} - \text{Qmin} \neq 0 \\ 1 & \text{if Qmax} - \text{Qmin} = 0 \end{cases} \textit{for a positive QoS attributes}$$
$$Norm(Qi) = \begin{cases} \frac{\text{Qmax} - (\text{Box cox}(\text{Qi}))}{\text{Qmax} - \text{Qmin}} & \text{if Qmax} - \text{Qmin} \neq 0 \\ 1 & \text{if Qmax} - \text{Qmin} = 0 \end{cases} \textit{for a negative QoS attribute,} \quad (1)$$

where Q_{min} and Q_{max} are the minimal and the maximal QoS values respectively, and

$$Box\,\text{Cox}(Q_i) = \begin{cases} \frac{(Q_i{}^{\theta} - 1)}{\theta} & \text{if } \theta \neq 0 \\ \log(Qi) & \text{if } \theta = 0 \end{cases} \quad (2)$$

In the relationship, (2) θ is the transformation parameter.

4.2. QoS K-Means Clustering

The main idea behind this step is to identify the possible states (N states) in which web services could be throughout their mission. Similar non-functional properties (QoS) are grouped together for each web service participating in the composition, and N states are considered in this work to be discrete states inferred from the values of observed QoS, such as the response time, reliability, availability, and throughput. The K-means clustering algorithm is applied using a Euclidean distance scale to measure the similarity in clustering.

4.3. Improved Hidden Markov Model with SFLA-PSO Hybridization

The Hidden Markov Model is a time series model that is able to infer a sequence of hidden states from an observation sequence. It is defined by the set of parameters $\lambda = (A, B, \pi)$, where A is the transition probability matrix , B represents the observation model distributions and π is the set of initial probabilities. In a formal way, an HMM is a stochastic process which has the following elements:

1. A set of N-hidden states $S = \{S_1, S_2, \ldots, S_N\}$. Note that a state of the model at time t is denoted by $q_t \in S, 0 \leq t \leq T$, where T is the length of the observation sequence and q_t denotes the current state.
2. A transition probability matrix $A = \{a_{ij}\}$. a_{ij} is the probability of transition from a state S_i at time $t-1$ to another state S_j at time t

$$a_{ij} = P(q_t = S_j \mid q_{t-1} = S_i), 1 \leq i, j \leq N. \quad (3)$$

3. An observation probability matrix denoted by $B = \{b_j(x_t)\}$ defined by the probability of emitting an observation x_t at time t given S_j

$$b_j(x_t) = P(x_t \mid q_t = S_j). \quad (4)$$

4. An initial state vector $\pi = \{\pi i\}$ is defined by the probability of beginning in the state S_i $(t = 0)$;

$$\pi_i = P(q_0 = S_i), 1 \leq i \leq N. \quad (5)$$

In this work, we redefine the QoS prediction model as follows:

Definition 1 (QoS state)**.** *We consider the QoS state at time t as the hidden state of an HMM denoted by s_t and a random variable taking values in $\{S_1, \ldots, S_n\}$ (i.e., good ,medium, . . . , and bad). It represents the QoS state of web service at time instant t. The probability of transition from a state S_i at time $t-1$ to another state S_j at time t is*

$$a_{ij} = P(q_t = S_j \mid q_{t-1} = S_i), 1 \leq i, j \leq N, \quad (6)$$

where S ∈ {good, medium, . . . , and bad}.

Definition 2 (QoS observations)**.** *A sequence of observations* $X = \{x_0, x_1, \ldots, x_t, \ldots, x_T\}$*, where* x_t *is the QoS information observed at time t is defined by the f-attributes,* $x_i = \{f_1, f_2, \ldots, f_m, \ldots, f_M\}$*. For instance, for* $M = 4$*,* x_i *may be in the set {response time, reliability, availability and throughput}. The probability of emitting an observation* x_t *at time t given* S_j *is*

$$b_j(x_t) = P(x_t \mid q_t = S_j). \tag{7}$$

Definition 3 (QoS initial state vector)**.** *We assume that the initial state vector for each web service in the CWS begins in the appropriate QoS.*

Definition 4 (The HMM decoding problem for QoS prediction)**.** *Our approach predicts the next state* q_{t+1} *for each web service in the CWS, given the model* λ^* *re-estimated at time (t) corresponding to the sequence of observations* $X = \{x_0, x_1, \ldots, x_t\}$*, where* x_t *is the QoS information observed at time t. The Baum–Welch algorithm [33]—based on the forward* (α) *and backward* (β) *iterative algorithm—is used to re-estimate the model of HMM, where* λ^* *estimate the model most representative of the sequence of observations X at time t. The Baum–Welch algorithm is not discussed in this paper, for further information refer to [33]. We predict the state* q_{t+1} *as follows:*

$$q_{t+1} = \underset{1 \le j \le N}{\arg\max} \left(\sum_{i=1}^{N} \alpha_t(i) a_{ij} \right), \quad 1 \le i, j \le N, \tag{8}$$

with $\alpha_0(i) = \pi_i b_i(x_0), 0 \le t \le T$ *and N is the number of degradation states in which a service can appear.*

4.3.1. Baum–Welch Limits for HMM Parameter Learning Applied to QoS Prediction

The Baum–Welch (BW) algorithm is used for estimating the parameters of the HMM. However, the most critical limitation is that the algorithm efficiency of BW depends on the initial value of the HMM parameters, in particular, the initial value of the emission matrix [34] and the possibility of its convergence to the local optima [35]. Thus, proposing a method that improves the initial value of the emission matrix is necessary.

The hybridization of methods such as the Shuffled Frog Leaping Algorithm (SFLA) [36] and Particle Swarm Optimization (PSO) [37] with the Baum–Welch algorithm is a good alternative for achieving this goal. We chose the SFLA because it has a robust global optimization capacity [38,39], and the PSO because it is well-known for its fast convergence and robustness [37,40].

4.3.2. A New Method for Selecting the Initial Emission Matrix

Finding the best emission matrix is an optimization problem in which a fitness function must be optimized. We consider the fitness function as the sum of the product of the forward and backward variables of the BW algorithm, for which the probability of generating the sequence of QoS information $(x_0, x_1, x_2, \ldots, x_t)$ should be maximized. The fitness function is formulated in Equation (9):

$$f(\lambda) = \sum_{i=1}^{N} \alpha_t(i) \beta_t(l), \tag{9}$$

where $\alpha(i)$ and $\beta(i)$ are the forward and backward variables, respectively. $f(\lambda^*)$ are the optimal HMM parameters for the successive QoS information given at time t.

In our hybridization of SFLA and PSO methods, we assume that the initial population of the emission matrix B_0 (initial emission matrix) is created with a random function, and it is sorted in descending order according to the value of the fitness function. Then the population is divided into K sub-populations after achieving the best global solution,

and each worst solution B_0 in each sub-population learns from the best solution of their sub-population using the PSO algorithm. After that, all sub-populations are combined and split into new sub-populations. The process terminates when the final conditions are satisfied. The proposed algorithm of SFLA–PSO for an improved Baum-Welch initialization is detailed in Algorithm 1 as follows:

Algorithm 1: SFLA–PSO algorithm for an enhanced Baum–Welch initialization

Input : initial transition probability matrix: A_0;
initial state vector π_0
Initialize the coefficients c_1, c_2, r_1, r_2;
Initialize the $Nb - iteration$
Output: estimate model $\lambda = (A, B, \pi)$;

1 **begin**
2 Generate random population of P solution of initial emission matrix B_0 (b-solution);
3 Calculate the fitness function of each B_0 in P;
4 Sort the population descending order of their fitness function values;
5 Get the $gbest\_solution$ of population P;
6 **while** *Nb-iteration* **do**
7 divide the population into k sub-populations;
8 **for** *each sub-population* **do**
9 get the $best\_solution$ and the $worst\_solution$ of sub-population;
10 Update the learning rate z using
11 $z^{i+1} = c_1 r_1^i z^i + c_2 r_2^i \left(best\_solution^i - worst\_solution^i\right)$
12 Update the $worst\_solution$ of each B_0 using
13 $worst\_solution^{i+1} = worst\_solution^i + z^{i+1}$
14 **end**
15 Shuffle the sub-populations;
16 calculate the fitness function of each B_0 in P using baum-welch algorithm;
17 Sort the population in descending order of their fitness function values;
18 Update the global best solution $gbest\_solution$;
19 **end**
20 return $\lambda^* = (A, B, \pi)$.
21 **end**

The PSO adaptation for the Baum–Welch algorithm considers the following concepts: B_0 is considered as a particle in the swarm, and the PSO global best particle is selected according to the SFLA–PSO global best solution. The $best\_solution$ and the $worst\_solution$ are considered respectively as the best solution and the worst solution in each sub-population.

The proposed algorithm pursues the following steps: First, an initial value is given to the transition matrix A_0 and initial state vector π_0, as well as the initial values of the learning parameters $c1$ and $c2$ associated to the components' learning rate and the random values $r1$ and $r2$, which are in the range [0,1] (input). Second, the population P associated with the emission matrix B_0 is generated randomly. Then, the fitness function is calculated for each B_0 in P using the Baum–Welch algorithm. Next, B_0 is sorted in descending order according to their fitness function values and the global best value is gated (lines 1 to 5). In the next step, the emission matrix population is divided into k sub-populations and, for each sub-population, the worst solution is optimized by using the worst and best solutions in the sub-population (lines 6 to 14). Then, in lines 15 to 18 the grouping is performed again, calculating the fitness function for each B_0 in P, sorting B_0 in descending order according to their fitness function values and updating the global best value. By repeating this process for a defined number of iterations, the algorithm converges to the best B_0 solution.

5. Experiments and Evaluation

5.1. Dataset

Experiments were performed using data from the WSDream dataset #3 because it is suitable for large experiments where it counts a total number of 4500 web services. These web services were invoked by 142 users in 22 different regions of the world in 64 different time slots with a 15-minute interval, in which each real web service was invoked 142 × 64 times.

5.2. Experiments Settings

The proposed approach was simulated using both JADE (Java Agent DEvelopment framework) for implementation and Python for data pre-processing on a 64-bit windows 10 with core intel(R) i5-3450 processor and 4.00 GB of memory.

Blocked cross-validation was used to evaluate the prediction accuracy. We used data from 80 different users over 64 different time slots and mapped into five subsets of data arranged in chronological order. Each of these subsets of data contains a total of 1024 measurable pairwise values of response time and throughput. These five sub datasets are again divided into two parts, one containing 80% of the data as a training dataset and the other containing 20% of the data as a test dataset. To the best of our knowledge, the blocked cross-validation has never been used to analyze the WSdream dataset in the field of QoS prediction for web services.

Table 1 shows the parameters used for SFLA–PSO and Box–Cox transformation.

Table 1. Table of parameter.

SFLA–PSO		Box–Cox Transformation
$c1 = c2 = 2.0$	acceleration coefficients	θ = (0.007-, 0.05-)
$P = 200$	population size	
$k = 10$	number of sub-population	
$n = 20$	population size in each sub-population	
$Nb - iteration = 200$	terminated condition	

5.3. Evaluation Metrics

Experiments were conducted in terms of prediction accuracy to prove the effectiveness of our proposed approach. To this extent, we used two different metrics to evaluate the clustering algorithm and the QoS prediction method.

5.3.1. Clustering Algorithm Evaluation Metric

To evaluate the clustering algorithm, we used the Silhouette score [30]; this score is commonly used as a synthetic indicator to evaluate the general quality of clustering, as well as to determine the appropriate number of clusters. This index is based on the so-called silhouette coefficient, which can be expressed according to Equation (10).

$$S(\mathbf{x}) = \frac{b(\mathbf{x}) - a(\mathbf{x})}{\max(a(\mathbf{x}), b(\mathbf{x}))}, \tag{10}$$

where $a(x)$ is the average dissimilarity of the xth point to all other points in the same cluster and $b(x)$ is the average dissimilarity of the xth point with all objects in the closest cluster. This measure has a range of $[-1, 1]$. In our approach, we selected the number of clusters that returns the highest average; a score of 1 means the clustering algorithm is appropriate.

5.3.2. QoS Prediction Evaluation Metric

We evaluated the prediction accuracy of our approach using two metrics, including MAES (Mean Absolute Error State) and RMSES (Root Mean Square Error State). A smaller MAES means a higher prediction accuracy. MAES is computed according to formula (11).

$$\text{MAES} = \frac{1}{T} \sum_{i \in N} (1 - \text{Probstate}_i), \tag{11}$$

where Probstate_i is the probability that a web service is predicted to be in state i, even if it is originally in this state, T is the total number of test cases, and N is the number of states. RMSES represents the Root Mean Square Error State, and is computed according to formula (12).

$$RMSES = \sqrt{\frac{\sum_{i \in N} (1 - \text{Probstate}_i)^2}{T}}. \tag{12}$$

5.4. The Clustering Algorithm Evaluation

To evaluate the k-means clustering algorithm on the WSDream3, we ran this algorithm for a different number of clusters (k) and then plotted the average silhouette score (ten independent runs). The clustering result for the WSDream dataset is shown in Figure 3. As observed in this figure, the number of clusters that maximizes the silhouette coefficient is $K = 3$.

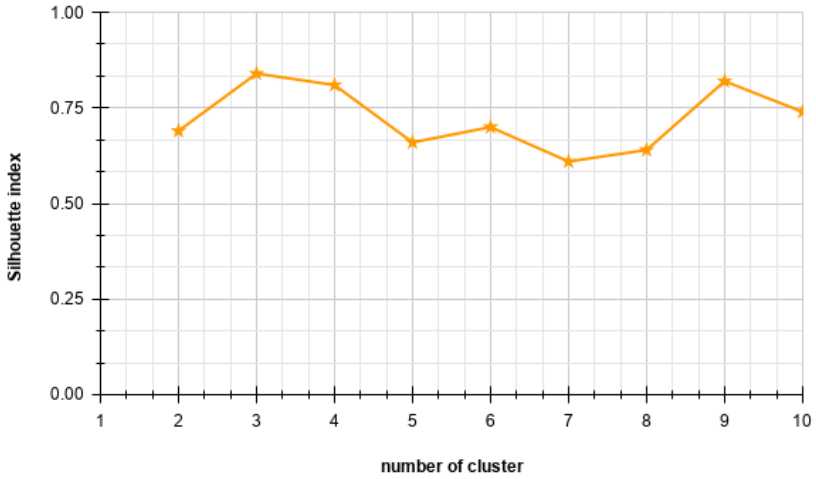


Figure 3. The number of clusters.

5.5. Evaluation of the QoS Prediction Algorithm

We conducted three kinds of experiments: the prediction algorithm evaluation, the success prediction ratio and the failure prediction ratio.

5.5.1. Accuracy Evaluation of the Prediction Algorithm

We selected 40 web services randomly and computed the average of the prediction algorithm for 10 independent runs. Results are depicted in Figure 4. As can be seen, even for highly changing web services, such as web service 6 in Figure 4a, the algorithm achieves an accuracy greater than 75%. For less changing web services, such as web service 17 in Figure 4b, the algorithm achieves an accuracy equal to 90%. Moreover, our formulation of the QoS prediction problem as a QoS state prediction problem further enhances the algorithm's prediction accuracy. From this, we conclude that the dynamic change in the QoS attributes of web services has no effect on the performance of our approach.

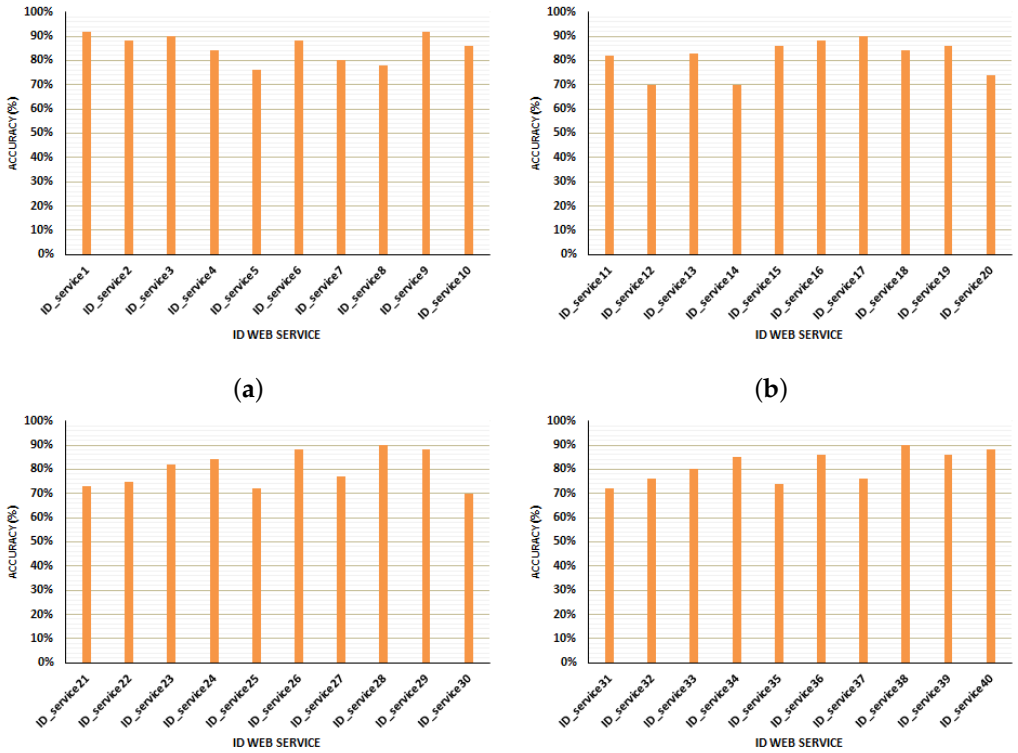


Figure 4. The prediction algorithm evaluation results in terms of accuracy. (**a**) web service 1–10; (**b**) web service 11–20; (**c**) web service 21–30; (**d**) web service 31–40.

5.5.2. Success Prediction Ratio

The success prediction ratio [41] computes the number of a successful state prediction divided by the total number of performed predictions as formulated in Equation (13). Let us remember that a state represents a web service's quality at a time instant t. We computed this ratio for forty different web services, and the obtained results are depicted in Figure 5. On average, a successful prediction ratio more than 75% was attained as shown in Figure 5a,b,d.

$$SuccessRatio = \frac{Number\ of\ successful\ predictions}{Number\ of\ predictions}. \quad (13)$$

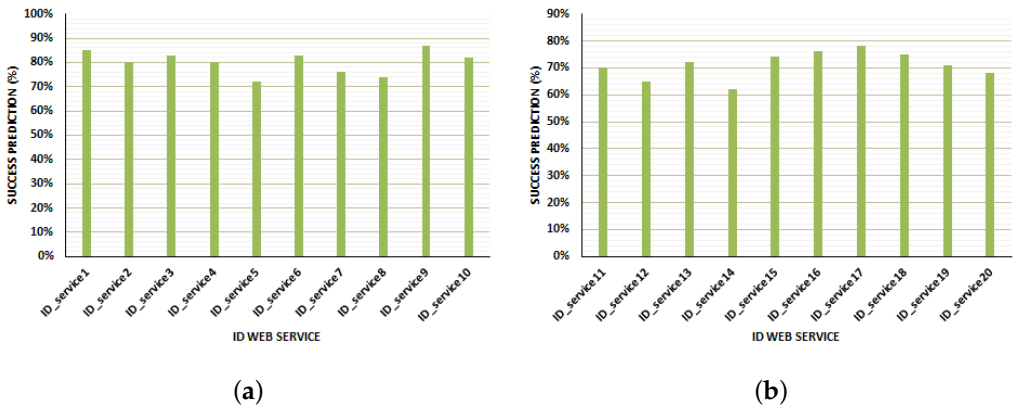
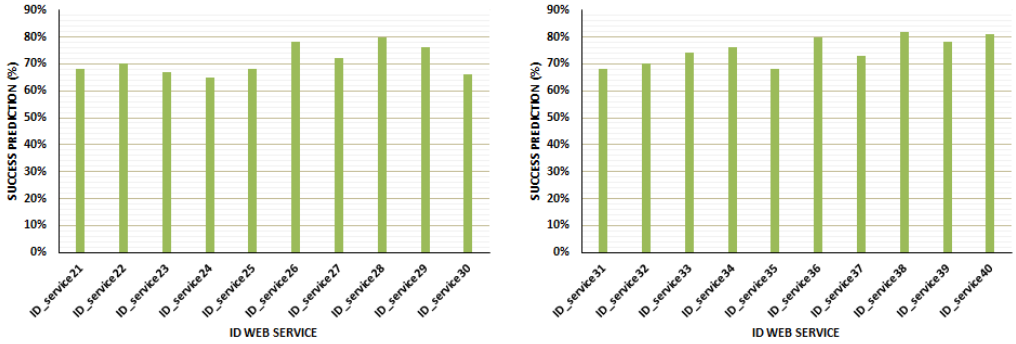


Figure 5. *Cont.*



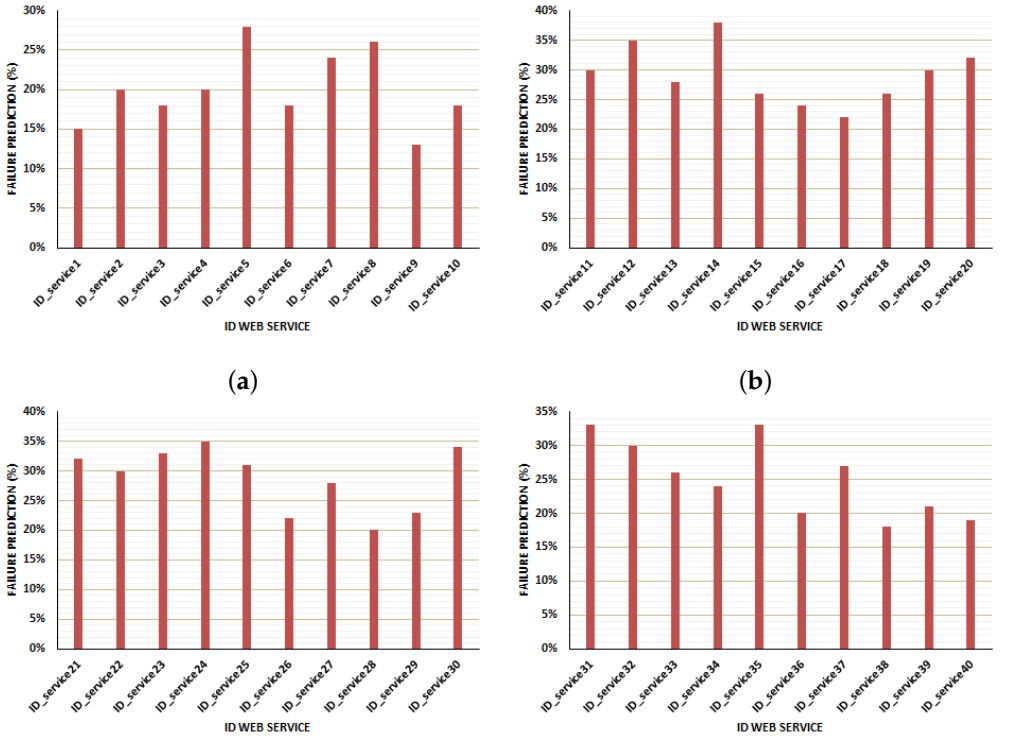
(**c**) (**d**)

Figure 5. The prediction algorithm evaluation results in terms of success prediction ratio. (**a**) web service 1–10; (**b**) web service 11–20; (**c**) web service 21–30; (**d**) web service 31–40.

5.5.3. Failure Prediction Ratio

The failure prediction ratio [41] computes the number of incorrect state predictions divided by the total number of performed predictions as formulated in Equation (14). We computed this ratio for forty different web services, and the obtained results are depicted in Figure 6. On average, a failure prediction ratio of less than 25% was attained as shown in Figure 6a,d.

$$FailureRatio = \frac{Number\ of\ failed\ predictions}{Number\ of\ predictions}. \quad (14)$$



(**a**) (**b**)

(**c**) (**d**)

Figure 6. The prediction algorithm evaluation results in terms of failure prediction ratio. (**a**) web service 1–10; (**b**) web service 11–20; (**c**) web service 21–30; (**d**) web service 31–40.

5.6. Accuracy Evaluation

To investigate the impact of the proposed PSO–SFLA hybridization for the Baum–Welch algorithm on the prediction accuracy, we compared two variants of our proposed

approach. The first variant is called the 'Improved HMM based Agent' and denotes the method proposed in Section 4.3. The second variant uses traditional HMM using the Baum–Welch algorithm and is denoted by 'HMM based Agent'. Table 2 shows the comparative results of the two discussed variants in terms of MAES and RMSES. The obtained results show that our proposed PSO–SFLA hybridization method for Baum–Welch improved the prediction accuracy on average by 7% according to MAES and by 8.8% according to RMSE. Considering the dynamics of the web service environment in the WSDream dataset, we can say that our approach produced remarkable results compared to the state-of-the-art relying on local data.

Table 2. HMM state prediction accuracy evaluation results.

	Improved HMM Based Agent		**HMM Based Agent**	
	MAES	**RMSES**	**MAES**	**RMSES**
data-set1	0.565	1.387	0.661	1.453
data-set2	0.507	1.262	0.620	1.434
data-set3	0.456	1.161	0.554	1.371
data-set4	0.416	1.123	0.435	1.131
data-set5	0.371	1.101	0.404	1.104
Average	0.463	1.206	0.534	1.298

5.7. Case Study

In this section, we evaluate the effectiveness of our prediction approach through a case study. As our interest is in QoS evaluation, we will ignore the potential issue of functional mismatch, as we randomly selected ten candidate web services for each abstract task from our dataset. Figure 7 depicts a prototype application for a COVID-19 test, ordered with a simplified workflow composed of four abstract tasks. Specifically, people who believe they have symptoms related to the COVID-19 virus start to request an online testing service that allows them to test for COVID-19 (questionnaire) and receive the results (S1). The patient can then book an appointment at one of the nearby COVID-19 (S2) testing centres, based on the patient locator service (S3). After confirmation, the ambulance or taxi service for the patient (S4) will be booked.

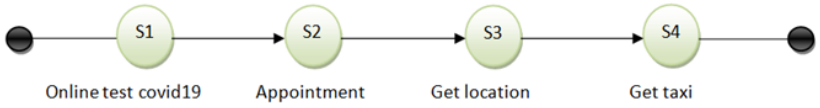


Figure 7. A prototype application for online test COVID-19.

We consider three application execution settings:

1. Ideal case: in this case, we assume that all the web services are chosen with the best QoS parameters to handle the client request. There is no degradation of QoS during the work process in all selecting web services. This is the optimal case; it acts as a benchmark.
2. Case of normal reactive approach: When the degradation occurs in a particular web service already chosen in the work process, the execution process interrupts, selecting the best candidate for the web service to be replaced with in the work process.
3. Case of our approach: We apply the IHMM-based Agent to QoS prediction. For each pre-selected web service (candidate ws), we use the prediction results in dataset 5. That is, each task dynamically chooses the best service according to current QoS predictions.

Assuming the maximum acceptable execution time is 1 s, where the execution time is defined as the time difference between a service user sending a request and receiving the corresponding response [42]. Figure 8 demonstrates the representative outcome of application execution time under the execution conditions quoted above. The execution in

the ideal situation (ideal case), without any degradation of the quality of service, requires an average execution time of 0.28 s, and this is unlikely to happen due to the dynamic nature of the web services. For the second case (reactive solution), the results show an increase in the implementation time by 0.87 s, and this is due to the dynamic environment of the web service, where when a degradation in the quality of service occurs, it is replaced by another service that may also be subjected to degradation, which makes the execution time more likely to increase with the repeated invocation of dynamic reconfiguration. In the case of our approach (third case), the results show a very noticeable improvement estimated to be 54% from 0.87 s to 0.4 s in execution time, and this refers to the optimal use of dynamic reconfiguration during and before execution.

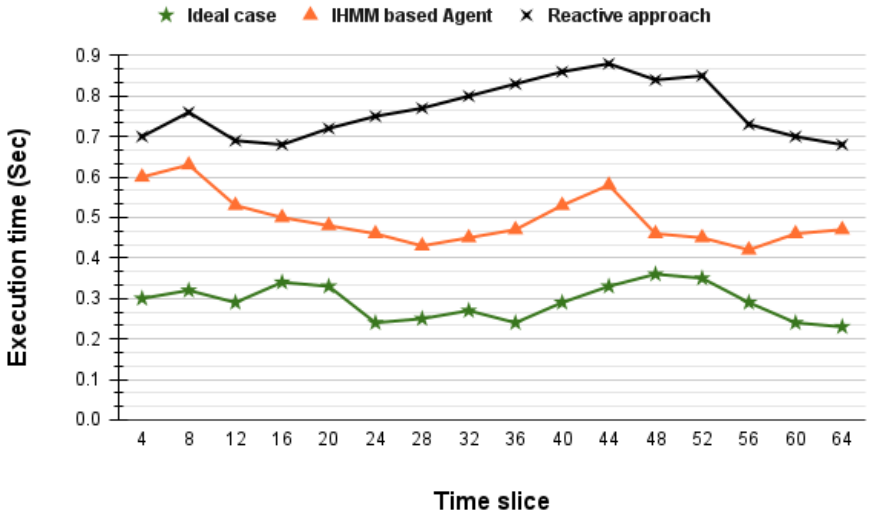


Figure 8. Comparison of three cases execution.

We compare the proposed framework and the works that are close to our approach, such as [1,3], where we repeat experiments 50 times with our proposed approach, taking into account the results found in comparative research, as shown in Figure 9. We note that our approach reduced the use of dynamic reconfiguration (adaptation) by 9% more than the results obtained in [1] despite our use of a real dataset. Our approach also outperformed that reported in [3] by 5%.

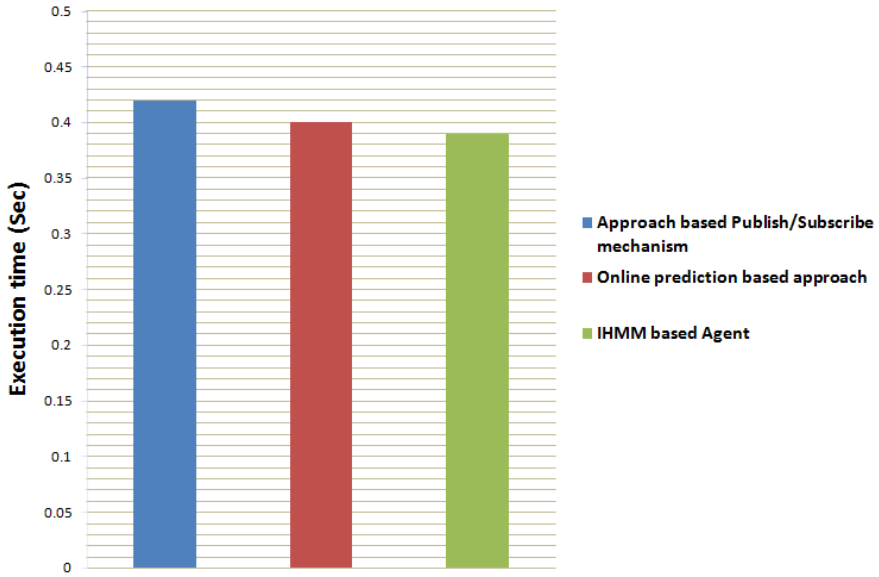


Figure 9. Execution time comparison.

6. Conclusions

In a service-oriented architecture environment, service reconfiguration provides the ability to substitute or adapt a disrupted web service or degraded QoS values during the workflow process. As the dynamism of the web services environment increases, the workflow process may be subject to multiple interruptions from the service reconfiguration due to fluctuations in the QoS values of web services. In this paper, a method for predicting the QoS status of web services is proposed. This method uses the HMM to predict the QoS status, where the prediction accuracy of HMM is improved using a hybrid meta-heuristic algorithm (SFLA-PSO). The Box–Cox transform method is also used to stabilize the data variance and make the data more normally distributed. Moreover, the K-means clustering algorithm is used to obtain the possible states of QoS. The proposed method is evaluated using block cross-validation on a real dataset and, with different evaluation metrics, the proposed method achieves better prediction results compared to state-of-the-art methods. For future work, we plan to improve the clustering algorithm with further QoS attributes (such as reliability and availability). A further study will compare the accuracy and efficiency of QoS status prediction and QoS values prediction.

Author Contributions: The major contributions of all the authors are summarized as: conceptualization, A.M.; methodology, H.S.; software, A.M.; validation, F.M. and R.B.; formal analysis, A.M.; investigation, A.M. and F.M.; resources, H.S.; writing, original draft preparation, A.M., F.M., H.S. and R.B.; writing, review, and editing, A.M., H.S., F.M. and R.B.; visualization, A.M. and H.S.; supervision, F.M. and R.B.; project administration, F.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data that support the findings of this study are available from the corresponding author upon reasonable request.

Acknowledgments: The authors would like to thank the General Directorate of Scientific Research and Technological Development, Algeria (DGRSDT), for their support and encouragement.

Conflicts of Interest: The authors declare no potential conflict of interests.

References

1. Kahlon, N.K.; Chahal, K.K.; Narang, S.B. Managing QoS Degradation of Component Web Services in a Dynamic Environment. *Int. J. Semant. Web Inf. Syst. (IJSWIS)* **2018**, *14*, 162–190. [CrossRef]
2. Yu, T.; Zhang, Y.; Lin, K.J. Efficient algorithms for web services selection with end-to-end qos constraints. *ACM Trans. Web (TWEB)* **2007**, *1*, 6-es. [CrossRef]
3. Zhu, J.; He, P.; Zheng, Z.; Lyu, M.R. Online QoS prediction for runtime service adaptation via adaptive matrix factorization. *IEEE Trans. Parallel Distrib. Syst.* **2017**, *28*, 2911–2924. [CrossRef]
4. Ryu, D.; Lee, K.; Baik, J. Location-based Web service QoS prediction via preference propagation to address cold start problem. *IEEE Trans. Serv. Comput.* **2018**, *14*, 736–746. [CrossRef]
5. Su, K.; Xiao, B.; Liu, B.; Zhang, H.; Zhang, Z. TAP: A personalized trust-aware QoS prediction approach for web service recommendation. *Knowl. Based Syst.* **2017**, *115*, 55–65. [CrossRef]
6. Zhu, X.; Jing, X.Y.; Wu, D.; He, Z.; Cao, J.; Yue, D.; Wang, L. Similarity-maintaining privacy preservation and location-aware low-rank matrix factorization for QoS prediction based web service recommendation. *IEEE Trans. Serv. Comput.* **2018**, *14*, 889–902. [CrossRef]
7. Keshavarzi, A.; Haghighat, A.T.; Bohlouli, M. Adaptive Resource Management and Provisioning in the Cloud Computing: A Survey of Definitions, Standards and Research Roadmaps. *KSII Trans. Internet Inf. Syst.* **2017**, *11*, 4280–4300.
8. Chen, X.; Zheng, Z.; Yu, Q.; Lyu, M.R. Web service recommendation via exploiting location and QoS information. *IEEE Trans. Parallel Distrib. Syst.* **2013**, *25*, 1913–1924. [CrossRef]
9. Lo, W.; Yin, J.; Deng, S.; Li, Y.; Wu, Z. An extended matrix factorization approach for qos prediction in service selection. In Proceedings of the 2012 IEEE Ninth International Conference on Services Computing, Honolulu, HI, USA, 24–29 June 2012; pp. 162–169.
10. Dai, Y.; Yang, L.; Zhang, B. QoS-driven self-healing web service composition based on performance prediction. *J. Comput. Sci. Technol.* **2009**, *24*, 250–261. [CrossRef]
11. Leitner, P.; Michlmayr, A.; Rosenberg, F.; Dustdar, S. Monitoring, prediction and prevention of sla violations in composite services. In Proceedings of the 2010 IEEE International Conference on Web Services, Miami, FL, USA, 5–10 July 2010; pp. 369–376.

12. Xiong, R.; Wang, J.; Zhang, N.; Ma, Y. Deep hybrid collaborative filtering for web service recommendation. *Expert Syst. Appl.* **2018**, *110*, 191–205. [CrossRef]
13. Shao, L.; Zhang, J.; Wei, Y.; Zhao, J.; Xie, B.; Mei, H. Personalized qos prediction forweb services via collaborative filtering. In Proceedings of the IEEE International Conference on Web Services (ICWS 2007), Salt Lake City, UT, USA, 9–13 July 2007; pp. 439–446.
14. Zhang, Y.; Zheng, Z.; Lyu, M.R. An online performance prediction framework for service-oriented systems. *IEEE Trans. Syst. Man Cybern. Syst.* **2014**, *44*, 1169–1181. [CrossRef]
15. Zhu, J.; He, P.; Zheng, Z.; Lyu, M.R. A privacy-preserving qos prediction framework for web service recommendation. In Proceedings of the 2015 IEEE International Conference on Web Services, New York, NY, USA, 27 June–2 July 2015; pp. 241–248.
16. Chen, Z.; Shen, L.; Li, F.; You, D.; Mapetu, J.P.B. Web service QoS prediction: When collaborative filtering meets data fluctuating in big-range. *World Wide Web* **2020**, *23*, 1–26. [CrossRef]
17. Song, Y.; Hu, L.; Yu, M. A novel QoS-aware prediction approach for dynamic web services. *PLoS ONE* **2018**, *13*, e0202669. [CrossRef] [PubMed]
18. Ahmed, W.; Wu, Y.; Zheng, W. Response time based optimal web service selection. *IEEE Trans. Parallel Distrib. Syst.* **2013**, *26*, 551–561. [CrossRef]
19. Aschoff, R.R.; Zisman, A.; Alexandre, P. Parallel Adaptation of Multiple Service Composition Instances. In *Engineering Adaptive Software Systems*; Springer: Singapore, 2019; pp. 115–134.
20. Gao, H.; Huang, W.; Duan, Y. The Cloud-edge-based Dynamic Reconfiguration to Service Workflow for Mobile Ecommerce Environments: A QoS Prediction Perspective. *ACM Trans. Internet Technol. (TOIT)* **2021**, *21*, 1–23.
21. Wang, H.; Li, J.; Yu, Q.; Hong, T.; Yan, J.; Zhao, W. Integrating recurrent neural networks and reinforcement learning for dynamic service composition. *Future Gener. Comput. Syst.* **2020**, *107*, 551–563. [CrossRef]
22. Chen, L.; Wang, Q.; Xu, W.; Zhang, L. Evaluating the survivability of soa systems based on hmm. In Proceedings of the 2010 IEEE International Conference on Web Services, Miami, FL, USA, 5–10 July 2010; pp. 673–675.
23. Moustafa, A.; Zhang, M. Towards proactive web service adaptation. In *International Conference on Advanced Information Systems Engineering*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 473–485.
24. Metzger, A.; Chi, C.H.; Engel, Y.; Marconi, A. Research challenges on online service quality prediction for proactive adaptation. In Proceedings of the 2012 First International Workshop on European Software Services and Systems Research-Results and Challenges (S-Cube), Zurich, Switzerland, 5 June 2012; pp. 51–57.
25. Krupitzer, C.; Roth, F.M.; VanSyckel, S.; Schiele, G.; Becker, C. A survey on engineering approaches for self-adaptive systems. *Pervasive Mob. Comput.* **2015**, *17*, 184–206. [CrossRef]
26. Zadeh, M.H.; Seyyedi, M.A. A self-healing architecture for web services based on failure prediction and a multi agent system. In Proceedings of the Fourth International Conference on the Applications of Digital Information and Web Technologies (ICADIWT 2011), Stevens Point, WI, USA, 4–6 August 2011; pp. 48–52.
27. Aschoff, R.; Zisman, A. QoS-driven proactive adaptation of service composition. In *International Conference on Service-Oriented Computing*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 421–435.
28. Soualhi, A.; Clerc, G.; Razik, H.; Guillet, F. Hidden Markov models for the prediction of impending faults. *IEEE Trans. Ind. Electron.* **2016**, *63*, 3271–3281. [CrossRef]
29. Wu, Q.; Zhang, M.; Zheng, R.; Lou, Y.; Wei, W. A QoS-satisfied prediction model for cloud-service composition based on a hidden Markov model. *Math. Probl. Eng.* **2013**, *2013*, 387083.
30. Keshavarzi, A.; Haghighat, A.T.; Bohlouli, M. Enhanced time-aware QoS prediction in multi-cloud: A hybrid k-medoids and lazy learning approach (QoPC). *Computing* **2020**, *102*, 923–949. [CrossRef]
31. Osborne, J. Improving your data transformations: Applying the Box-Cox transformation. *Pract. Assess. Res. Eval.* **2010**, *15*, 12.
32. Patro, S.; Sahu, K.K. Normalization: A preprocessing stage. *arXiv* **2015**, arXiv:1503.06462.
33. Rabiner, L.; Juang, B. An introduction to hidden Markov models. *IEEE Assp. Mag.* **1986**, *3*, 4–16. [CrossRef]
34. Zheng, H.; Wang, R.; Xu, W.; Wang, Y.; Zhu, W. Combining an HMM with a genetic algorithm for the fault diagnosis of photovoltaic inverters. *J. Power Electron.* **2017**, *17*, 1014–1026.
35. Yang, F.; Balakrishnan, S.; Wainwright, M.J. Statistical and computational guarantees for the Baum–Welch algorithm. *J. Mach. Learn. Res.* **2017**, *18*, 4528–4580.
36. Eusuff, M.; Lansey, K.; Pasha, F. Shuffled frog-leaping algorithm: A memetic meta-heuristic for discrete optimization. *Eng. Optim.* **2006**, *38*, 129–154. [CrossRef]
37. Eberhart, R.; Kennedy, J. A new optimizer using particle swarm theory. In Proceedings of the MHS'95, Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, 4–6 October 1995; pp. 39–43.
38. Boudries, F.; Sadouki, S.; Tari, A. A bio-inspired algorithm for dynamic reconfiguration with end-to-end constraints in web services composition. *Serv. Oriented Comput. Appl.* **2019**, *13*, 251–260. [CrossRef]
39. Ibrahim, G.J.; Rashid, T.A.; Akinsolu, M.O. An energy efficient service composition mechanism using a hybrid meta-heuristic algorithm in a mobile cloud environment. *J. Parallel Distrib. Comput.* **2020**, *143*, 77–87. [CrossRef]
40. Qin, S.; Sun, C.; Zhang, G.; He, X.; Tan, Y. A modified particle swarm optimization based on decomposition with different ideal points for many-objective optimization problems. *Complex Intell. Syst.* **2020**, 6263–247. [CrossRef]

41. Amirat, H.; Lagraa, N.; Fournier-Viger, P.; Ouinten, Y. MyRoute: A graph-dependency based model for real-time route prediction. *J. Commun.* **2017**, *12*, 668–676. [CrossRef]
42. Zheng, Z.; Zhang, Y.; Lyu, M.R. Investigating QoS of real-world web services. *IEEE Trans. Serv. Comput.* **2012**, *7*, 32–39. [CrossRef]