Response Time Based Optimal Web Service Selection

Waseem Ahmed, Yongwei Wu, *Member, IEEE*, and Weimin Zheng, *Member, IEEE*

Abstract—Selecting an optimal web service among a list of functionally equivalent web services still remains a challenging issue. For Internet services, the presence of low-performance servers, high latency or overall poor service quality can translate into lost sales, user frustration, and customers lost. In this paper, we propose a novel method for QoS metrification based on Hidden Markov Models (HMM), which further suggests an optimal path for the execution of user requests. The technique we show can be used to measure and predict the behavior of Web Services in terms of response time, and can thus be used to rank services quantitatively rather than just qualitatively. We demonstrate the feasibility and usefulness of our methodology by drawing experiments on real world data. The results have shown how our proposed method can help the user to automatically select the most reliable Web Service taking into account several metrics, among them, system predictability and response time variability. Later ROC curve shows a 12 percent improvement in prediction accuracy using HMM.

Index Terms—Hidden states, modeling and prediction, optimal path, web service composition

1 Introduction

The Internet made the world a smaller place. Companies from all around the world may now compete over different service offerings not only with their local adversaries, but do now under a global scale. Escalating the competition and lead in industry segment can often be a matter of offering and, perhaps even most importantly, assuring the good quality of the services offered. In the Web this should be no different; controlling quality for Web Services (WS) is done by enforcing Quality of Service (QoS) policies and assuring needed quality conditions are always met.

On the user's side, the increased number of services means more and more offerings to choose from. Unfortunately, due to the explosive growth in the number of WSs available in the world, selecting the best WS to solve a given task has become a quite challenging task. Currently, users cast their choice based on the reviews and experiences of other users. User-created ranks are often the first resource for finding reliability information regarding a particular service, often given in terms of response time, throughput, availability, security and reliability.

Dynamically composing web services requires the service consumer to discover services that satisfy functional and non-functional requirements [4]. In a dynamic environment, non-functional requirement such as WS's reliability in terms of response time is unlikely to be congruous with that provided by venders in the service level agreement [SLA] [5]. [1], [2], [3] have considered the reliability parameters of WSs either as constant or suggested venders to provide probabilistic details of the WS flow. Similarly, QoS attributes modeled as probability distribution if considered as constant or user defined function values is also not sufficient [5]. Analyzing QoS parameters of WSs considering constant probabilistic values as baseline does not reflect precise results. Similarly, user defined function values are also not sufficient to predict future behavior of component web services.

There is no standard way, however, for the users to weigh their options directly and individually, for themselves. This paper aims to fill this gap providing a standard way to measure and predict WS behavior in terms of response time using HMM. Reliability of service oriented architecture (SOA) based systems heavily depend on various underlying technologies for instance web services, computing environment (CPU, Disk, and Network) and unpredictable internet [21]. In this paper we have specifically focused on predicting web service's behavior in terms of response time (RT). For other factors such as CPU, disk or network one can find solutions in [1], [2], [3]. In HMM, the number of hidden states to be used is unknown. Usually, based on domain knowledge there is only some guess about it. For example, in case of web servers, network load balancing distributes incoming users' requests among multiple web servers to handle more traffic and faster response. In this case multiple web servers can be different hidden states responding to users' requests randomly. Generally, remote web service is composed of various hidden states as shown in Table 1. These states are hidden from the users' and respond to their requests randomly based on their execution. However, there are two things to consider:

1. Web services are owned and hosted by other organizations. So users have no way to analyze them directly.

• *The authors are with the Department of Computer Science and Technology, Tsinghua National Laboratory for Information Science and Technology (TNLIST), Tsinghua University, Beijing 100084, China, and also with the Research Institute of Tsinghua University in Shenzhen, Shenzhen 518057, China. E-mail: wuyw@tsinghua.edu.cn.*

Manuscript received 31 July 2013; revised 18 Oct. 2013; accepted 25 Nov. 2013. Date of publication 12 Dec. 2013; date of current version 9 Jan. 2015.
Recommended for acceptance by X. Li.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2013.310

TABLE 1
Possible Hidden States

#	*Hidden States*
1.	Database
2.	Bad node in server clustering
3.	Mutated coding
4.	Calling another WS

a. This list *can have more hidden states based on the architecture of the application.*

2. These hidden states can neither be discovered or guaranteed with traditional exhaustive testing [3] nor can be relied on service providers' exposed parameters defined in SLA.

Therefore, it is more challenging to analyze or predict behavior of hidden states with respect to response time. To tackle this challenge, in this paper we present a novel approach. It first computes the behavior of internal structure of WS using the HMM. Later it combines the status of underlying hidden states to compute the overall behavior of each component web service. Approach defined here can also be utilized to find an optimal path to accomplish users' requests. This can be achieved by building a directed graph among various hidden states. In comparison to existing strategies, our contribution in this paper can be summarized as follows:

1. Predicted web service's behavior by predicting the status of underlying hidden states in terms of Response Time (RT).
2. Selected optimal WSs and an optimal path at runtime for executing user request by identifying the status of underlying hidden states.

The remainder of the paper is organized as follows: Section 2 introduces related work, Section 3 describes in depth details of our proposed model, Section 4 presents our experiments and results and finally Section 5 concludes the paper.

2 Related Work

Analyzing QoS parameters of web services in dynamic and unpredictable environment is an important and challenging research area. Many researchers have analyzed these attributes and proposed different frameworks. In this section we have presented a review of existing methods or frameworks proposed by different researchers.

V. Grassi [1], L. Baresi [2], Z. Zibin [6], R. Perrone [7], and M. Cristescu [8] have produced certain methods to tackle this challenge. They have focused on predicting reliability of various factors involved in building enterprise application, nonetheless, considered reliability of remote web service as constants. V. Grassi [1] analyzed both ways to predict reliability i.e., by considering services offered by software component and services offered by hardware devices. However, for remote web services, he assumed that the vender will provide probabilistic details about the flow of executing user requests. To achieve this he also suggested modifications in WSDL. In our approach, however we compute probabilistic details with the help of HMM. Z. Zibin and R. Michael [6], have employed past failure data of real web services to find reliability for current web services. To achieve desired goals, they have calculated similarity parameters among real web services and service users with those of current web services and current users respectively. At first this approach did not consider the effect of environment and secondly it did not give any information about internal structure failure probability of remote web services.

Huiyuan Zheng [5] suggested a probabilistic model to analyze QoS attributes of component services with dynamic probabilities. Their main contribution was to compare the efficiency and accuracy of their algorithm with simulation models. However, they did not consider combining various hidden patterns with integrations patterns to compute overall behavior of WS integration.

D. Zhong [9], has suggested a CSPN model to predict reliability and the degree of trustworthiness of WSC. His main contribution was to define a model for transforming BPEL process into CSPN model. Yet this model deals more with design time problems and does not reflect the impact of problems that occur at runtime. Joyce EI and Maude [10], Sami [11], Li [12] have considered the transactional properties of web services to define a strategy for reliable web service composition. They have studied in detail transactional dependency among different type of web services. Later, they have suggested web service selection algorithms based on users' preferences. These models also tried to solve design time issues during service composition. Kaouthar and Zahi [13] have proposed a flexible architecture for dynamic web service composition related to user requirements. Their main contribution was to ensure availability of appropriate web service at runtime. However, they did not define any QoS metrification to find the appropriate web services among functionally equivalent web services. Tao [14] has suggested an efficient algorithm for selecting appropriate web services based on user's provided weights. They have mapped web service composition with the Knapsack problem and then calculated the optimal path for executing user's requests. However, in an unpredictable environment user defined parameters for calculating WSs behavior are not sufficient. Yilei Zhang and Zibin Zheng [15] have proposed a model-based QoS prediction framework called WSPred. It was a time-aware personalized QoS prediction approach that analyzes latent features of users, service and time by performing tensor factorization. Similarly, S. Maheswari and G.R. Karpagam [16] have proposed a framework that considered seven QoS attributes i.e., response time, execution Time, throughput, scalability, reputation, accessibility, and availability for better web service selection. Overall these models consider probability as constant value or base it on the user's defined function values except Huiyuan Zheng [5]. However, Huiyuan Zheng [5] did not consider predicting WS's behavior during the nth time interval in the future.

The HMM has already been used in different papers for analyzing quality factors of distributed computing systems.

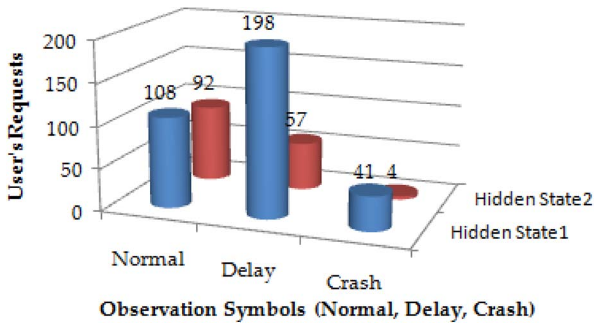


Fig. 1. Hidden states' observation patterns in terms of number of requests.

Nonetheless, they have their own issues, constraints and shortcomings. For instance, Leilei Chen [17] has designed a framework to evaluate survivability of a SOA based application(s) using the HMM. The main idea of their 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; in addition the author did not discuss the various hidden states or probabilistic insight of remote web services. G. Rahnavard [18] has used HMM to detect anomalies in web services i.e., the author has designed a framework to detect intrusion in WS.

However this strategy cannot be used to gauge QoS attributes of overall WS. Flex Selfner [19] has proposed the use of HMMs to categorize and distinguish error patterns leading to failures. This author also suggested a mechanism for predicting the future occurrence of failures or errors. WS assessment model [20] has used this model to assess failures during certain time in the future. In short HMM has already been successfully used to analyze various aspects in distributed computing systems.

QoS parameters of any web service sometimes depend on its internal complexity i.e., apart from unpredictable internet. Such as, how the programmer has written his algorithm to accomplish users' requests or how the architect has defined the architecture to entertain users' requests. Therefore, estimating behavior of individual component service with respect to response time (RT) in turn requires ensuring behavior of all possible hidden states within component service [21]. Therefore, this paper provides comprehensive information for selecting an optimal WS and predicting it's behavior in terms of RT.

3 System Model and Parameters

HMM is a powerful statistical tool for modeling generative sequences that can be characterized by an underlying process generating observable sequences [22]. Word hidden specifies that internal structure of the underlying system is hidden from the observer. Observer does not know in which state system may be in, but has only probabilistic insight where it should be. In HMM, one does not know how many hidden states to use. Usually, based on domain knowledge there is only some guess about hidden states. We have discussed in detail about WS hidden states below in Sections 3.2.1 and 3.2.2. Later training algorithm will find out how to connect these hidden states. HMM can solve three fundamental issues i.e., Evaluation, Decoding, Training. More details can be found in [22].

Using HMM to measure and predict WS behavior with respect to response time, our model consists of a two step process. First step will require us to train the model to find optimal HMM parameters i.e., A, B & π, such that model best fits the training sequence. Training sequence in our model can be exploited by recording and labeling response time of a web service at regular intervals of time. Baum-Welch algorithm a particular case of expectation-maximization (EM) can be used to train the model. It iteratively improves the basic model which provides convergence to local optima, whereas second step, first requires us to compute current state of the system. Then based on current state, future behavior of the system is predicted. This can be computed using VITERBI algorithm.Based on above two steps, for selecting an optimal WS and an optimal path for executing user requests our strategy can be further divided into following steps:

1. Building a directed graph among hidden states of component web services used in composition.
2. Analyzing the current status of each vertex of directed graph i.e., underlying hidden states.
3. Predicting hidden states' behavior in terms of response time during nth time interval t.
4. Finally, selecting optimal web services used in composition based on hidden states' behavior.

3.1 Exemplary Scenario

To analyze the behavioral pattern of hidden states, we have selected a weather forecast WS with best rank. More than 500 threads are used in parallel in a distributed environment. As in HMM, one does not know how many hidden state to use, so we have supposed that target WS is running on a clustser of webserver, containing 2 webservers. We found that 9 percent of the overall result was with observation symbol ''C'' as shown in Fig. 1. Later, when we further scrutinized the result by training the model, we have found that 8.2 percent from 9 percent failures were caused by web server1 whereas only 0.8 percent failures were originated by web server2. This shows that probability of receiving failure when web server1 executes the results is more than web server2. The analysis shows that at runtime behavioral patterns of the hidden states can be utilized for selecting optimal web services among the list of functionally equivalent web services. Hidden states with observation symbol A (as shown in Table 3) of different WSs can be connected at runtime to process user's request.

In the next step we will explain the process of building a directed graph among the hidden states of different web services used in composition.

3.2 Directed Graph Among Hidden States

The key idea to select optimal WS for completing user requests is to exploit behavioral patterns of underlying hidden states. This can be done by recognizing hidden states' emission probabilities and combining them with integration patterns. As per our knowledge, previously different hidden states have never been considered with the integration patterns of WSC to estimate its behavior in terms of RT. The basic structure of hidden states and corresponding observations in terms of RT is necessary to understand before building a directed graph.

3.2.1 Hidden States Analysis

Generally atomic web service is composed of different hidden states that are invisible to consumers as shown in Table 1. Each state is responsible to output certain results during time t. The probability of the response to certain requests depends on execution of these hidden states. When a consumer of a web service accesses a remote web service, sometimes he receives an unprecedented delay even under best operational conditions. We believe that this exceptional delay is because of unreliable hidden states responding to users' request at that particular time. Delay represents the state where system receives the results after long time, however, crash/error represents the state where WS crashes or user receives no response from WS.

Predicting WS behavior in our model is about evaluating hidden states' replies in terms of response time during the nth time interval. Generally, to build a highly available system, venders normally use different processing units to implement WS. In our model these processing units are termed as hidden states. For example, in case of web servers, network load balancing distributes incoming users' requests among multiple web servers to handle more traffic and faster response. Depending on a consumer's requirement and nature of application, response time patterns (shown in Table 2) for different observable units (shown in Table 3) can be redefined.

3.2.2 Response Time Analysis with HMM

Hidden states can entertain user's requests randomly and produce results anytime. As these hidden states can also be accessed from other hidden states while service invocation, hence the model is of ergodic type. There may be/exist certain hidden states that may produce results in similar patterns of time intervals of having different configurations. For instance, hidden states 2 and 4 (defined in Table 1) can have similar response time patterns. Thus for a given time interval we can define feature vectors including values defined in Table 1. Because of differences in implementation of hidden states (e.g., *for hidden state 4 in Table 1, communication delay or latency for calling another service inside target web service will also be involved in over all response time*) clear identification among feature values is required which in machine learning is referred to as Feature Normalization [18]. These features may be categorized in terms of memory requirements, network requirements, software service requirements etc. This will help to define the initial values of the probability of success or failure of hidden states.

TABLE 2
Response Time Pattern

Label	*Response Time (sec)*	*Remarks*
A	Response time < = 5	Normal
B	Response Time > 5	Delay
C	Not replied so long or Crashed	Error

a. *Defined by architect and can be re-defined based on the nature of application*

TABLE 3
Observation Units

#	*Observation Symbols*	*Labels*
1.	Normal Execution	A
2.	Delay in Response	B
3.	Error / Crash	C

Fig. 2 shows general implementation of a WS used in examplory scenario. User receives different observation symbols A, B, and C while he invokes a web service during a certain time interval t. Here observation symbols represent response time (as defined in Table 2) of output values. Depending on hidden states i.e., Web Server 1 and Web Server 2, the application users receive the results from a web service with either A, B, or C. In our approach when a user receives an unprecedented delay or error i.e., with response time pattern C, underlying hidden state is considered as an unreliable state S_{urel} as shown in Fig. 2. By initializing HMM parameters it can be ensured that the model transits to an unreliable state once the user receives exceptional delay or error. We have further explained this concept in the *"Training the model"* section below. For now we can define some basic parameters of HMM in terms of component web service as:

1. States: Number of hidden states S within a component web service.
2. Observations: Distinct output observations i.e V= (Normal (A), Delay (B), and Error/Crash (C)) such that output observation at time t is O_t where sequence of observation is $O = O_1, O_2, \ldots, O_n$ Here sequence of observation represents various response times generated by remote web service depending upon execution of relevant hidden state.
3. $A_{i,j}$ represents the transitional probability from hidden state S_i following S_j.
4. B_{sj} represents the probability of hidden state generating output being produced from an hidden state S_j.
5. π represents the initial probability distribution values of underling hidden states.

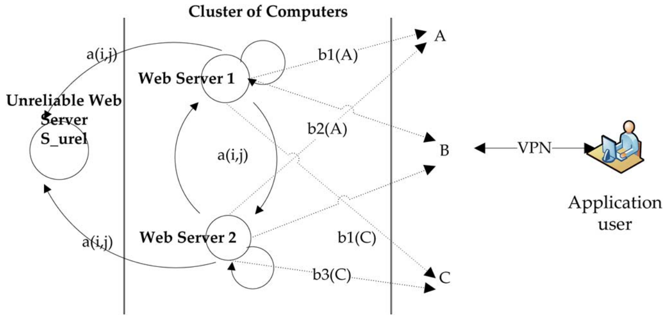


Fig. 2. Hidden states and corresponding observation symbol.

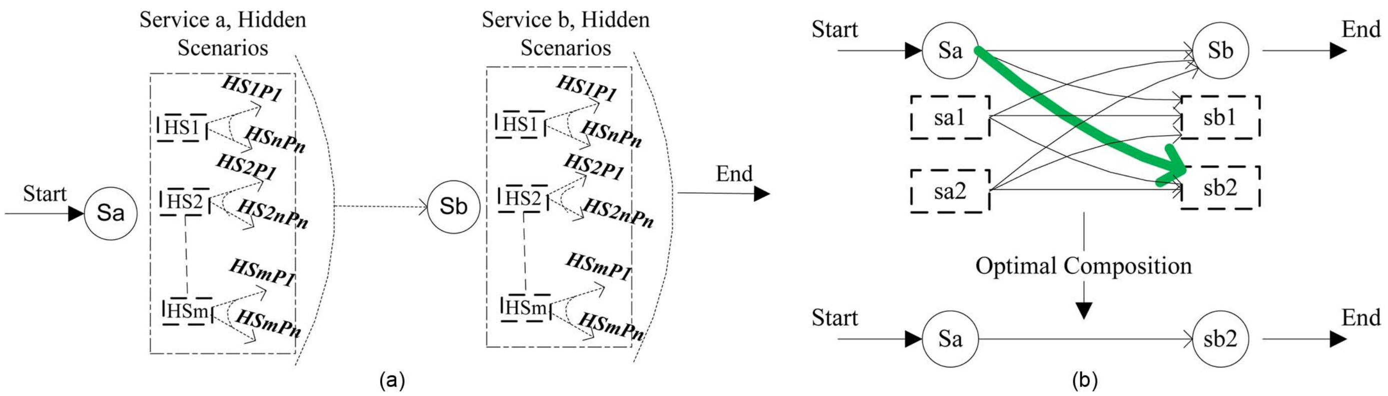


Fig. 3. Example of a simple case in composite web service (CWS) with various hidden scenarios. (a) Basic composition (simple case). (b) Optimal composition of web services.

As per definition of HMM we have:

$$\lambda = (A, B, \pi). \quad (1)$$

With the help of training algorithm of the HMM we can identify model parameters λ in the light of the analysis above. To train the model, observations must represent the actual execution time of the hidden states. Below we have outlined the mechanism for extracting execution time of the hidden states.

3.2.3 Web Service Execution Time

Execution time of any hidden state at time t must be required to compute λ in "(1)". This will further help to figure out unreliable hidden states. Generally, response time of a web service observed on the client side during any specific time t is composed of two things.

$$Response\ time\ rt = Latency + Execution\ Time\ of\ WS. \quad (2)$$

Based on the instability of the internet, users receive different quality of service in terms of response time. Therefore, it is necessary to subtract latency from round-trip time to obtain the exact execution time of corresponding hidden states. Latency can be calculated by using any NMS (network management system) tool. A common method to calculate latency is to *"send an echo request at the same time when web service was being invoked by the application i.e., PING to the destination IP and record the time it takes to receive answer. Later this time can be subtracted from round-trip time to obtain actual execution time"*.

3.2.4 Basic Concept of Directed Graph

Fig. 2 indicates that A, B and C are the discrete emissions representing Normal, Delay and Crash respectively. These emissions are based on the execution of different hidden states as per the defined range (mentioned in Table 2). Nonetheless, in reality emissions (A, B, and C) for different web services having similar functionalities consist of vector of probabilities during time interval t as shown in Fig. 3a, e.g., For hidden state1, HS1P1 represents hidden state1 with probability-1 and HSnPn represents hidden staten with probability-n. To build a directed graph among hidden states, following is the basic definition used in our model.

Definition 1. *Directed Graph: Let HS be the set of hidden states for each WS and U be the set of observations produced by different hidden states, P be the set of probabilities for each observation, Then we can define the parameters for a directed graph $G = (V, E)$ as:*

$V = HS$, set of the hidden states of each candidate web services registered for composition are considered as vertices.

$\mathrm{E} = \{\mathrm{HS_i} -> \mathrm{HS_j}, \mathrm{P_i}\}$, edge represents the link from a hidden state $\mathrm{HS_i}$ of $\mathrm{S_i}$ with observation symbol $\mathrm{U_i}$ having probability $\mathrm{P_i}$ to hidden state $\mathrm{HS_j}$ of $\mathrm{S_j}$, $\exists \mathrm{S_i}$, $\mathrm{S_j} \subseteq \mathrm{S}$ and $\forall\ \mathrm{HS_i} \in \mathrm{S_i} -> \mathrm{HS_j} \in \mathrm{S_j}$.

In the next section we will discuss how the hidden states of atomic web services can be combined in a process of composition with a directed graph to select optimal path during any time interval t.

3.2.5 Composition Model

Selecting the optimal path in a composed web service in turns requires examining QoS (response time) attributes of the each component WS. This can be examined by analyzing probabilistic behavior of their hidden states. In our approach estimating probabilistic behavior of hidden states (vertices) of component web services can be combined together to select an optimal path. Fig. 4 shows the detailed process of web service composition in terms of hidden states. Let us consider a simple case with two web services "a" and "b" consisting of hidden states 3 and 2 respectively. Then they can be connected with each other in the form of a directed graph as shown in Fig. 4. Any hidden state i.e., $\mathrm{HS_1}$, $\mathrm{HS_2}$ or $\mathrm{HS_3}$ of the component web service "a" can execute users' requests during certain time interval t

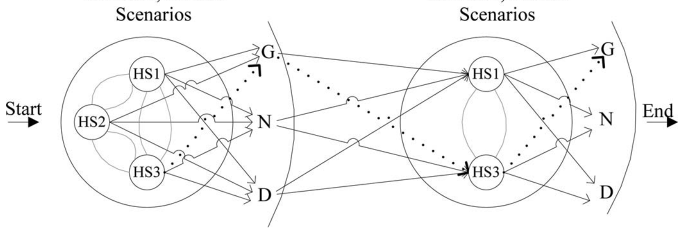


Fig. 4. Directed graph of composition in terms of hidden states.

# of users / Service	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	⋯
WS responses	A	A	A	A	B	A	B	B	B	A	B	A	A	A	A	C	B	A	A	B	B	C	⋯

Fig. 5. Example of a simple case in composite web service (CWS) with various hidden scenarios.

and give response to the end user as defined above in the "Response Time Analysis with HMM" section. The response of hidden states is categorized by observation symbols U as defined above. The dotted line in Fig. 4 shows that probabilistic value of service "a" i.e., its hidden state HS_3 has produced results with observation symbol A which was used as input of service "b". In service b hidden state HS_2 has produced results with observation symbol A, and finally user receives reply with a minimum time. So our main objective can be achieved using this model. Therefore, at first current state of the system is estimated using training sequence as discussed in Section 3.4. Later, future behavior of each hidden state of the WS is predicted.

3.3 Analyzing Current Status

The current status of underlying hidden states (mentioned in Table 1) is analyzed using HMM. Each hidden state is analyzed in terms of QoS attribute i.e., response time. State of WS during time interval t can be considered as vector of probabilities that WS is in hidden state executing a certain request R during time interval t having pattern of response time $O = O_1, O_2, \ldots, O_n$. The current state of WS can lead us to predict state of WS during the nth time interval under various operational conditions. State of WS during time interval t can be computed with the help of HMM i.e.,

$$\delta_t(i) = \overset{Max}{HS_1, \ldots, HS_{n-1}}\, P(HS_1, \ldots, HS_{n-1}, HS_n = i, O_1, \ldots, O_n/\lambda). \quad (3)$$

Here $\delta_t(i)$ represents the state of WS i.e., it represent maximum probability (computing maximum over all possible hidden states sequences) that the model went through hidden states $HS_1, \ldots, HS_{n-1}$ and the system is in state i at hidden state n. i.e., $HS_n = i$ while observing $O = O_1, \ldots, O_n$. Equation "(3)" in HMM is known as VITERBI algorithm, which can help users to build relations among hidden states S_i and response time pattern generated by each hidden state S_i during time interval t_i. VITERBI algorithm can further be used to calculate the probability of a current status of any specific hidden state HS_i.Let Q_j^k be the probability of observations from $k+1$ to n such that $HS_k = j$, Then recursively we can write:

$$Q_k^j = \sum P(Q_{k+1} | HS_{k+1} = i) P_{j,i} Q_k^{j+1}. \quad (4)$$

Here $P_{j,i}$ represents the marginal probability distribution of hidden states described by the homogenous Markov Chain process such that $P_{j,i} = P\ (HS_k = i,\ HS_{k+1} = j)$. If $P_{O_{k=i}}^{HS_k = j}$ is the probability of the observations $(O = O_1, O_2, \ldots, O_k)$ such that $HS_k = j$, then the probability of the current status of any specific hidden state can be calculated as:

$$P(HS_k = j) = P_k^j Q_k^j / P(O). \quad (5)$$

Where $P(O)$ is the probability of observations and can be recursively calculated as

$$P(O) = \sum_{H_n = j} P_{O_n = i}^{H_n = j}. \quad (6)$$

3.4 Predicting Future Behavior of Component Service

To find the probabilistic behavior of each hidden state of a WS for defined response time pattern (as shown in Table 2) against different user requests, we can calculate P_{ur} (n). That is, probability of hidden state HS_k in the kth time interval is unreliable or hidden state HS_k producing results with delay during kth time interval. This probability is calculated by "First Passage Time Distribution". Let T_k be the time in the kth time interval, also known as "First Passage Time", when hidden state HS_k produced delayed results, then

$$T_k = \min(n = \ > k > \ = 0 : \ HS_k = HS_{ur}).$$

Here HS_k represents the hidden state at time k during the nth time interval. Probability distribution among hidden states can be computed as below:

$$\sum_{i=0}^{n} P(T_k < = n | HS_j = i) P(HS_j = i) \quad \text{s.t. } j = 0 \quad (7)$$

where $P(HS_j = i)$ is the probability of response time patterns that WS is in hidden state j at current time as computed in "(3)". $P(T_k < = n | HS_j = i)$ is the probability of going through hidden state during the nth time interval starting from $j = 0$ which can be computed recursively. Equation (7) represents probability distribution that hidden state HS_{ur} produces unreliable results during time interval k which can be further scrutinized using dynamic programming to efficiently compute for various time intervals.

3.4.1 Training the Model

In prediction models, training is generally done through historical data. To obtain actual execution time of a web services as a training sequence we have randomly selected a web service with best rank. Then, we developed an interface in .Net to access this WS in a distributed environment. Response time of more than 500 threads is then recorded during parallel execution in a distributed environment. Equation (2) was then used to extract actual execution time of the target web service.

To train the model a training sequence as described above is used. This sequence is first labeled as an observation symbol as shown in Fig. 5 i.e., A, B, and C (as defined in Table 3) based on the criteria as defined in Table 2. When an error is produced during service invocation observed by any thread in distributed environment, we represent it as observation symbol C. In our framework such responses are modeled and represented by hidden state S_{urel} as shown in Fig. 2. For example, when data with uncertain result is obtained i.e. with observation symbol C, the underlying state is considered as an unreliable state. Each column in Fig. 5 represents the execution time of each WS invocation, received by different users, when a web service with higher rank was invoked in parallel. Then by initializing HMM parameters in "(1)" i.e., initial, transition and emission probabilities, such that states representing as unreliable, are the only states that produce results with "C", it can be ensured that the model transits to the unreliable state when uncertainty appears in the training sequence. Baum-Welch algorithm a particular case of expectation-maximization (EM) is used to train the model. It iteratively improves the basic model which provides convergence to local optima. After training the model, current and future state is predicted using VITERBI algorithm as discussed above.

3.5 Optimal WS Selection

Selecting an optimal WS among list of functionally equivalent WSs we refer to definition 1.

In our model we exploited pool of WSs having similar functionalities and selected a better WS at runtime to perform certain task. As per definition1 in Section 3.4.2 each component web service S_i consists of n number of hidden states HS_i that correspond to user requests on invocations. This can help to build a directed graph $G = (V, E)$ among the hidden states of different component web services as shown in Fig. 4. Each hidden state has an edge from hidden state of web service S_i to hidden states of web service S_k with the observation symbol U having probability P_i. Therefore it can be observed that during anytime interval t the edge from hidden state HS_i can be categorized into various feasible solutions when invoked by service users as shown in Fig. 3b). The process of selecting the best feasible solution produced by underlying hidden state among several feasible solutions is shown in *Algorithm 1.*

```
Input:   DIRECTED_GRAPH(V,E)
// V = Set of hidden states   group by web services and E
represents edges from each Hidden State group by obser-
vation symbol i.e. (Good /Normal /Bad)
Output: OPTIMAL_PATH

1. for each $v_s$ in V
2.     for each $v_{hs}$ in $V_s$
3.         for each e (start, $v_{hs}$) in E
4.             CALCULATE_MIN
5.             CREATE_PATH $P_i$
6.         If e ($v_{hs}$, End) true
7              FIND_MIN_PATH $P_i$
8              ADD MIN_PATH $P_i$ to Graph G
9          else
10                 Initialize_Parameter
11 For each $P_i$ in Graph G
12     For each $P_i$ in $Group_i$
13         SUM_QoS
14         FIND_MIN_PATH
15         OP=Build_OPTIMAL_PATH
16 IF ( $P_i$ in Graph G ends) true;
17 Return OP;
```

Algorithm 1 indicates that application at client side exploits HMM parameters for each WS to build a directed graph among hidden states. Later, it returns the optimal path by calculating MIN of QoS values i.e. response time. Finally, system uses this optimal path to execute user's requests efficiently and reliably.

4 Experiments and Results

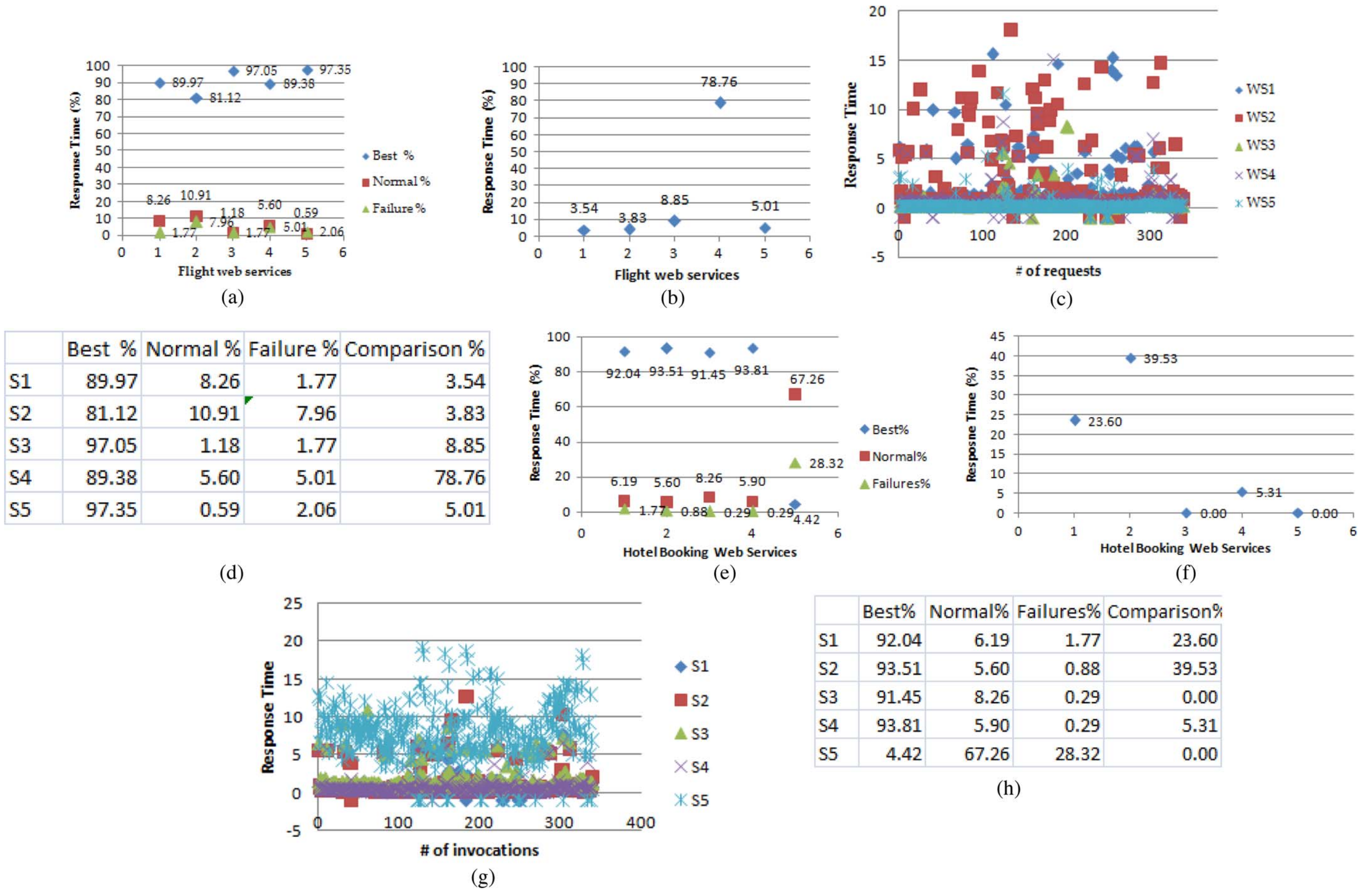
Selecting web services for composition at runtime with better QoS (i.e., response time), it is necessary to have a pool of web services having similar functionalities. In our experiment we have randomly selected two groups of services to integrate, each group consists of 5 web services having similar functionalities. Response time observations against all these WSs have special patterns (as shown in Table 2) linked with the execution of each hidden state as discussed in Section A-2. Our case study is based on a real scenario where an application facilitates end users for flight and hotel reservation simultaneously as shown in Fig. 6. Response time observations are categorized using Table 2 as shown in Fig. 6. It is apparent from Figs. 6a, 6b, 6c, and 6d that more than 80 percent of user's requests responded with label A by all flight web services. Nonetheless, all these web services encountered failure during different time intervals as shown in Figs. 6a and 6c. Fig. 6d shows that the WS3 and WS5 have produced best results with respect to response time, during simultaneous invocation. However when data was compared in terms of minimum of QoS attribute (response time), we noticed WS4 generated results in less time among selected WSs as shown in Fig. 6b. Percentage of results with observation symbol A is a lot better than having some exceptions as shown in Fig. 6e. However, there still exist a small percentage of exceptions for each flight WS. Consequently, selecting any WS for composition can make the composition unreliable.

As discussed in Section 3.2.1 and exemplary scenario section, these exists some hidden states. We believe that these exceptions are because of those unreliable hidden states responding to users' request at that particular time. To further scrutinize results of WSs shown in Fig. 6 for optimal WS selection, we analyzed internal structure of these WSs using HMM. Figs. 6e, 6f, 6g, and 6h shows the QoS details of hotel booking web services. To deal with exceptions discussed above, we have exploited more than 300 simultaneous requests to:

1. Adjust the model parameters to analyze current state of the internal structure of web services.
2. Build a directed graph using concept of hidden states to select optimal path for executing the user request.
3. Predict the future behavior of these web services during nth time interval to select better web service for optimal composition.

4.1 Adjusting the Model Parameters

Before predicting behavior of the hidden states, it is necessary to train the model to get estimated transition and emission probabilities. Detailed process of adjusting



	Best %	Normal %	Failure %	Comparison %
S1	89.97	8.26	1.77	3.54
S2	81.12	10.91	7.96	3.83
S3	97.05	1.18	1.77	8.85
S4	89.38	5.60	5.01	78.76
S5	97.35	0.59	2.06	5.01

	Best%	Normal%	Failures%	Comparison%
S1	92.04	6.19	1.77	23.60
S2	93.51	5.60	0.88	39.53
S3	91.45	8.26	0.29	0.00
S4	93.81	5.90	0.29	5.31
S5	4.42	67.26	28.32	0.00

Fig. 6. Status of various flight and hotel booking web services. (a) Respose time patterns of Flight booking WSs (Percentage percent wise). (b) Comparisons of minimum results received in terms of response time of each flight booking WS. (c) Response time pattern of flight booking WSs. (d) Response time pattern of each flight booking web service. (e) Response time patterns of hotel booking web service. (f) Comparison of minimum results received in terms of response time for each hotel booking web service. (g) Response time pattern for 5 different hotel booking web services (S1, S2, S3, S4, S5). (h) Response time pattern of each hotel booking web service.

HMM parameters is discussed in Section 3.4.1. These estimated values are then used in ''(3)'' to compute most probable hidden states sequences. Purpose of training the model is to find optimal HMM parameters i.e., A, B and π, such that model best fits the training sequence. Baum-Welch algorithm that iteratively improves the basic model is used. Trained HMM is then used to analyze the current status of hidden states in terms of response time. Finally, response time is predicted during anytime interval Δt.

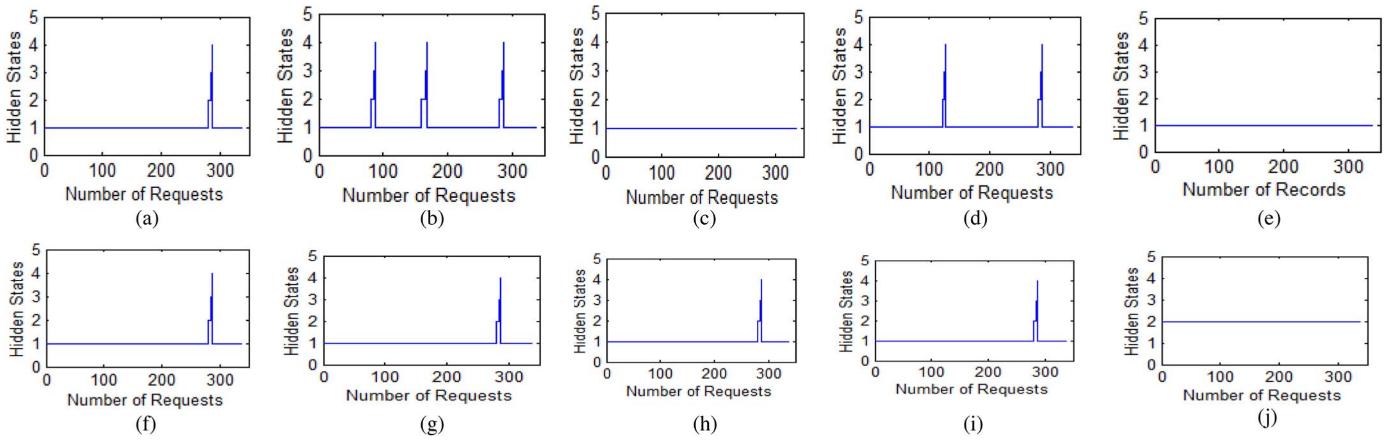


Fig. 7. Current state of hidden states of various flight and hotel web services. (a) Hidden states pattern for Flight WS1. (b) Hidden states pattern for Flight WS2. (c) Hidden states pattern for Flight WS3. (d) Hidden states pattern for Flight WS4. (e) Hidden states pattern for Flight WS5. (f) Hidden states pattern for Hotel Booking WS1. (g) Hidden states pattern for Hotel Booking WS2. (h) Hidden states pattern for Hotel Booking WS3. (i) Hidden states pattern for Hotel Booking WS4. (j) Hidden states pattern for Hotel Booking WS5.

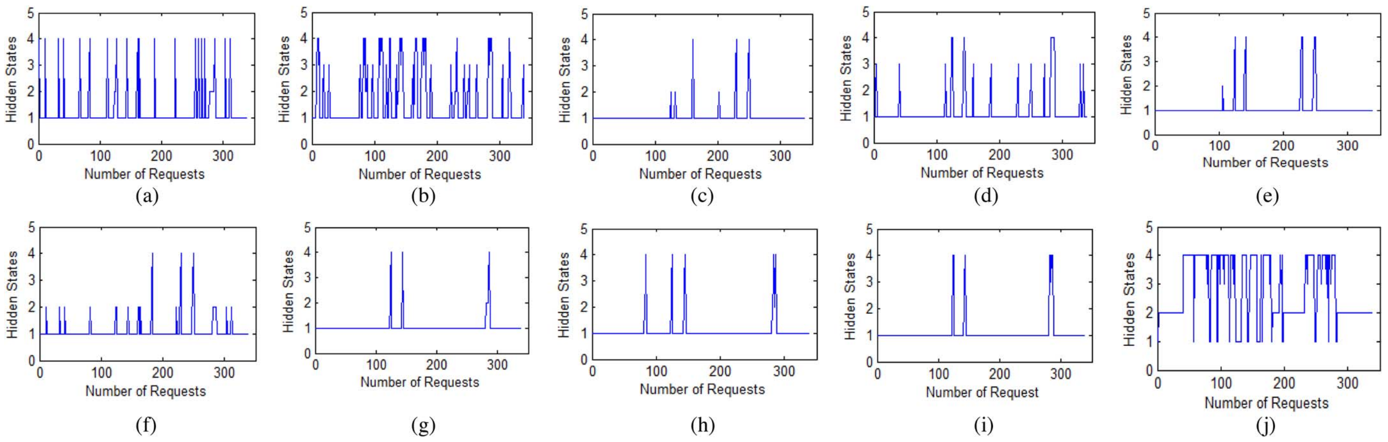


Fig. 8. Predicted values of hidden states of various flight and hotel web services. (a) Predicted hidden states pattern for flight WS1. (b) Predicted hidden states pattern for flight WS2. (c) Predicted hidden states pattern for flight WS3. (d) Predicted hidden states pattern for flight WS4. (e) Predicted hidden states pattern for flight WS5. (f) Predicted hidden states pattern for hotel WS1. (g) Predicted hidden states pattern for hotel WS2. (h) Predicted hidden states pattern for hotel WS3. (i) Predicted hidden states pattern for hotel WS4. (j) Predicted hidden states pattern for hotel WS5.

4.2 Analyzing Current State

The state of component web service during time interval t is a vector of probabilities that system is in the hidden state HS_i when observation symbol U is observed. VITERBI algorithm is used to calculate the most probable hidden state sequence (as discussed in Section 3.2) that has generated the training sequence as shown in Fig. 7. Figs. 7a, 7b, 7c, 7d, and 7e shows the current value of various hidden states (as defined in Table 1) of Flight web services. It is apparent from Figs. 6d and 6h and Figs. 7a, 7b, 7c, 7d, 7e, 7f, 7g, 7h, 7i, 7j that most of the results were produced by State1. However, each web service encountered errors which resulted when the internal system transited to the other hidden states. Predicting these transitions among hidden states for any WS can help to gauge the behavior of that WS. This will further help to select optimal WS at runtime when there is a chance of transition among hidden states of one WS.

4.3 Predicting Future Behavior and Selecting

4.3.1 Optimal Path

As HMM is normally used to recognize patterns, therefore to predict behavior of the hidden states, idea is to classify suspicious response time patterns i.e., patterns with observation symbols C. This classification will indicate upcoming suspicious patterns. As per proposed technique, response time in a training sequence is divided into equal lengths slots. These time slots having observations symbol ''C'' are termed as ''unreliable''.

First the model is trained using training sequences (as shown in Fig. 5). Then based on trained HMM current status of the hidden state's behavior is analyzed using VITERBI algorithm. Later, based on current state, future behavior of hidden states is predicted by calculating ''first passage time distribution'' into unreliable state. Based on minimum QoS attribute analysis users can select $WS4$ considering it as best among all in group1 and $WS2$ from

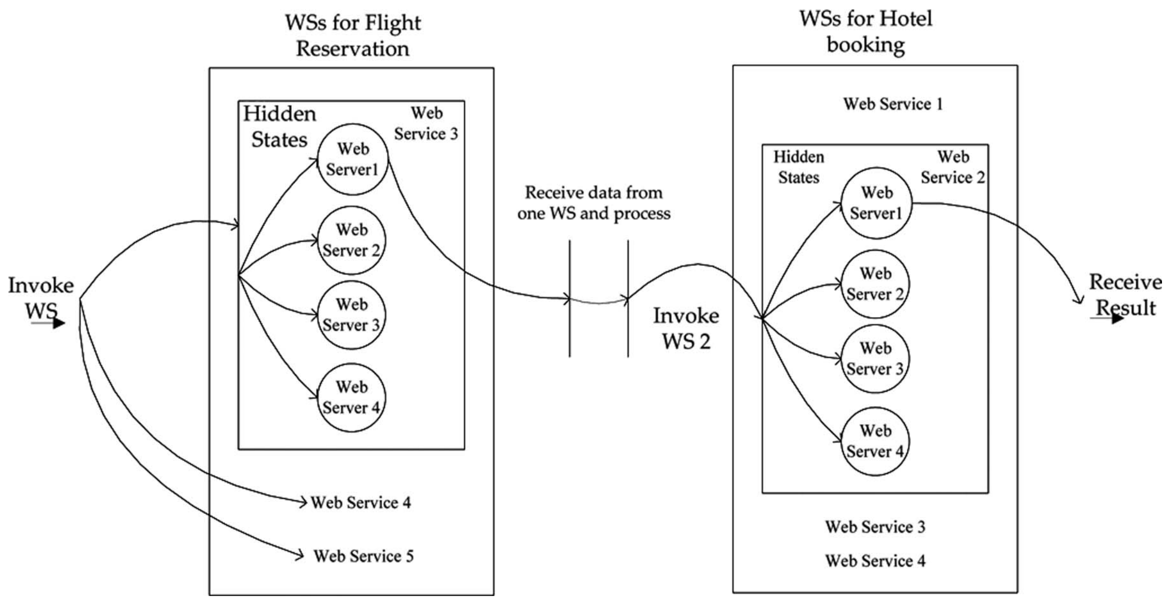


Fig. 9. Directed graph among hidden states of different WSs in an online flight and hotel reservation system.

TABLE 4
Comparison of Prediction Accuracy

Models >	Prposed Solution		User Collobarative Model		Execution Time Prob. Model	
Cut of value	Sensitivity	Specificity	Sensitivity	Specificity	Sensitivity	Specificity
>10	0.98993289	1	0.9909	0.972454	0.9909	0.7493
>9	0.94594595	1	0.9899	0.934311	0.9899	0.6812
>8	0.96405229	1	0.96405	0.901893	0.96405	0.66914
>7	0.8653	0.9921	0.9459	0.850201	0.4529	0.6729
>6	0.69093656	0.95	0.6653	0.896209	0.3294	0.6962
>5	0.50342466	1	0.3456	0.819419	0.3456	0.39992
>4	0.34567901	1	0.2193	0.793656	0.2193	0.31956
>3	0.219324	0.9183	0.1382	0.7678212	0.19425	0.193562
>2	0.08196721	0.67647059	0.0818	0.654139	0.00312	0.12492
>1	0.012	0.568	0.0129	0.316721	0.0001	0.013

group2 to integrate in one application. Nonetheless, overall behavior of the CWS cannot be ensured. Because there still exist some hidden states in $WS4$ and $WS2$ of group1 and group2 respectively (as shown in Figs. 7d and 7g), that can cause slow response or sometime failure making overall composition as unreliable. For instance in Fig. 8 user's request between 120-130 and 180-190 in $WS4$ of group1 and 115-130 in $WS2$ of group2 transited from hidden state 1 to other hidden states causing slow response or failure. Eventually, selecting web services at runtime from each group based only on minimum of QoS attribute still not sufficient. Therefore, behavior of hidden states along with minimum of QoS attribute can ensure reliable composition in terms of response time. Figs. 8a, 8b, 8c, 8d, 8e, 8f, 8g, 8h, 8i, and 8j represents the predicted behavior of underlying hidden states for both types of web services i.e flight and hotel reservations. It can be observed in prediction results that the probability of transiting from hidden state 1 to other hidden states of a web service have increased significantly that may cause response from web services as either B or C (as defined in Table 2).

However, probability of time of transition among hidden states of the web services having similar functionalities is different from each other. This can be useful in building a directed graph among hidden states of web service in group1 to the hidden states of web service in group2 at time t. As per results in Fig. 8, a directed graph is developed among 12 hidden states of 3 web services $WS3$, $WS4$, and $WS5$ from group1 and 16 hidden states of 4 web services $WS1$, $WS2$, $WS3$, and $WS4$ of group2 (shown in Fig. 9). For simpilicity we have shown only two web services with their corresponding hidden states. When user connects with the system, then based on prediction results system finds the WS, probabilistic behavior of whose hidden states are at their best level during that interval and then connect with it. Later algorithm1 is exploited to select an optimal path with minimum probabilistic value for executing user requests in the most efficient and reliable way.

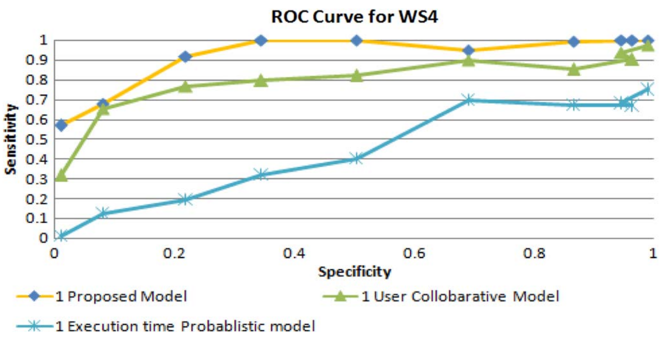


Fig. 10. ROC curve for best WS among given pool of web services.

4.4 Comparison of Prediction Accuracy

For comparing the accuracy of our results and also comparing the model with existing methodologies we have used receiver operating characteristic (ROC) graph. ROC technique is basically used for visualizing, organizing and selecting classifiers based on their performance. For proof of concept, we have selected the best WS among the pool of services to compare prediction accuracy results of our model with existing methodologies such as user collaborative approach [6], execution time probabilistic approach [7]. Table 4 shows the sensitivity & specificity of different prediction approaches employing certain cut of values to response time of WS4 of group1. We defined maximum tolerable response value and then sliced the results on different points of possible response values. In Fig. 10 ROC graph shows that our prediction methodology obtains better predictions accuracy.

5 Conclusion

In this paper, we at first propose a probabilistic model for predicting response time of web service and then selected an optimal web service at runtime from the list of functionally equivalent web services. To know the probabilistic insight of WSs we have used HMM. In our model we have assumed that WS is deployed on a cluster of web servers and sometime the delay or crash during WS invocation is because the bad node in sever clustering responds to users' requests.

With the help of HMM we have predicted the probabilistic behavior of these web servers and then selected the WS based on their probabilistic value. Experiment shows that the proposed model is more general and detailed in comparison to existing models. This not only predicts the overall behavior of composite web service but it further provides the solution to complete user requests in the most efficient and reliable way.

ACKNOWLEDGMENT

The authors thank F. Teklemical for her support & proofreading of our paper. This work was supported in part by Natural Science Foundation of China (61373145, 61170210), National High-Tech R&D (863) Program of China (2013AA01A213, 2012AA012600), and National Basic Research (973) Program of China (2011CB302505).

REFERENCES

[1] V. Grassi, "Architecture-Based Reliability Prediction for Service-Oriented Computing," in *Architecting Dependable Systems III*. Berlin, Germany: Springer-Verlag, 2005, pp. 279-299.
[2] V. Cortellessa and V. Grassi, "Reliability Modeling and Analysis of Service-Oriented Architectures," in *Test and Analysis of Web Services*. Berlin, Germany: Springer-Verlag, 2007, pp. 339-362.
[3] G. Stefano, C. Ghezzi, R. Mirandola, and G. Tamburrelli, "Quality Prediction of Service Compositions through Probabilistic Model Checking," in *Proc. 4th Int'l Conf. Quality Software-Architect., Models Architect.*, 2008, pp. 119-134.
[4] D.A. Menasce, "Composing Web Services: A QoS View," *IEEE Internet Comput.*, vol. 8, no. 6, pp. 80-90, Nov. 2004.
[5] H. Zheng, J. Yang, W. Zhao, and A. Bouguettaya, "QoS Analysis for Web Service Compositions Based on Probabilistic QoS," in *Service-Oriented Computing*. Berlin, Germany: Springer-Verlag, 2011, pp. 47-61.
[6] Z. Zibin and R.L. Michael, "Collaborative Reliability Prediction of Service-Oriented Systems," in *Proc. 32nd ACM/IEEE Int'l Conf. Softw. Eng.*, Cape Town, Africa, 2010, vol. 1, pp. 35-44.
[7] R. Perrone, R. Macedo, G. Lima, and V. Lima, "An Approach for Estimating Execution Time Probability Distributions of Component-Based Real-Time Systems," *J. Universal Comput. Sci.*, vol. 15, no. 11, pp. 2142-2165, 2009.
[8] M. Cristescu and L. Ciovica, "Estimation of the Reliability of Distributed Applications," *Inf. Econ.*, vol. 14, no. 4, pp. 19-29, 2010.
[9] D. Zhong, Z. Qi, and X. Xu, "Reliability Prediction and Sensitivity Analysis of WS Composition," in *Petri Net: Theory and Applications*, V. Kordic, Ed. Rijeka, Croatia: Intech, 2008, pp. 459-470.
[10] J. El Haddad, M. Manouvrier, G. Ramirez, and M. Rukoz, "QoS-Driven Selection of Web Services for Transactional Composition," in *Proc. IEEE ICWS*, 2008, pp. 653-660.
[11] B. Sami, G. Claude, and P. Olivier, "Transactional Patterns for Reliable Web Services Compositions," in *Proc. 6th Int'l Conf. Web Eng.*, Palo Alto, CA, USA, 2006, pp. 137-144.
[12] L. Li, L. Chengfei, and W. Junhu, "Deriving Transactional Properties of Composite Web Services," in *Proc. IEEE ICWS*, 2007, pp. 631-638.
[13] K. Boumhamdi and Z. Jarir, "A Flexible Approach to Compose Web Services in Dynamic Environment," *Int'l J. Digit. Soc.*, vol. 1, no. 2, pp. 157-163, 2010.
[14] Y. Tao, Z. Yue, and L. Kwei-Jay, "Efficient Algorithms for Web Services Selection with End-to-End QoS Constraints," *ACM Trans. Web*, vol. 1, no. 1, p. 6, May 2007.
[15] Z. Yilei, Z. Zibin, and M.R. Lyu, "WSPred: A Time-Aware Personalized QoS Prediction Framework for Web Services," in *Proc. IEEE 22nd ISSRE*, 2011, pp. 210-219.
[16] S. Maheswari, "QoS Based Efficient Web Service Selection," *Eur. J. Sci. Res.*, vol. 66, pp. 428-440, 2011.
[17] C. Leilei, Q. Wang, W. Xu, and L. Zhang, "Evaluating the Survivability of SOA Systems Based on HMM," in *Proc. IEEE Int'l Conf. Web Serv.*, 2010, pp. 673-675.
[18] G. Rahnavard, M.S.A. Najjar, and S. Taherifar, "A Method to Evaluate Web Services Anomaly Detection Using Hidden Markov Models," in *Proc. ICCAIE*, 2010, pp. 261-265.
[19] F. Salfner, "Predicting Failures with Hidden Markov Models," in *Proc. 5th Eur. Dependable Comput. Conf.*, 2005, pp. 41-46.
[20] M. Zaki, A. Ihsan, and B. Athman, "Web Services Reputation Assessment Using a Hidden Markov Model," in *Proc. 7th Int'l Joint Conf. Serv.-Oriented Comput.*, 2009, pp. 576-591.
[21] W. Ahmed and Y.W. Wu, "A Survey on Reliability in Distributed Systems," *J. Comput. Syst. Sci.*, vol. 79, no. 8, pp. 1243-1255, Dec. 2013.
[22] P. Blunsom, *Hidden Markov Models*, Retrieved May 19, 2006, 2004. [Online]. Available: www.cs.mu.oz.au/460/2004/materials/hmm-tutorial.pdf

Waseem Ahmed received the MSc degree from the Computer Science Department of Quaid-e-Azam University, Islamabad, in 2002. In the time frame of 2002 to 2010 he assumed various roles within the Software Industry in Pakistan. Since September 2010, he is following a doctoral track in Computer Science within the Parallel and Distributed Systems Group, Tsinghua University. His research interests are in the areas of performance evaluation of large-scale distributed systems, in particular Service Oriented Archtiecture.

Yongwei Wu received the PhD degree in applied mathematics from the Chinese Academy of Sciences, in 2002. He is currently a professor in Computer Science and Technology at Tsinghua University, China. His research interests include distributed processing, virtualization and cloud computing. Dr. Wu has published over 80 research publications and has received two Best Paper Awards. He is currently on the editorial board of the International Journal of Networked and Distributed Computing and Communication of China Computer Federation. He is a member of the IEEE.

Weimin Zheng received the BS and MS degrees, in 1970 and 1982, respectively from Tsinghua University, China, where he is currently a professor of Computer Science and Technology. He is the research director of the Institute of High Performance Computing at Tsinghua University and the managing director of the Chinese Computer Society. His research interests include computer architecture, operating system, storage networks, and distributed computing. He is a member of the IEEE.