Online Reliability Prediction via Motifs-Based Dynamic Bayesian Networks for Service-Oriented Systems

Hongbing Wang, *Member, IEEE*, Lei Wang, Qi Yu, *Member, IEEE*, Zibin Zheng, *Member, IEEE*, Athman Bouguettaya, *Fellow, IEEE*, and Michael R. Lyu, *Fellow, IEEE*

Abstract—A service-oriented System of Systems (SoS) considers a system as a service and constructs a robust and value-added SoS by outsourcing external component systems through service composition techniques. Online reliability prediction for the component systems for the purpose of assuring the overall Quality of Service (QoS) is often a major challenge in coping with a loosely coupled SoS operating under dynamic and uncertain running environments. It is also a prerequisite for guaranteeing runtime QoS of a SoS through optimal service selection for reliable system construction. We propose a novel online reliability time series prediction approach for the component systems in a service-oriented SoS. We utilize Probabilistic Graphical Models (PGMs) to yield near-future, time series predictions. We assess the approach via invocation records collected from widely used real Web services. Experimental results have confirmed the effectiveness of the approach.

Index Terms—Online reliability prediction, time series, service-oriented computing, system of systems

1 Introduction

A System of Systems (or SoS) pools individual, possibly heterogeneous, systems together, to create a new, value-added, and more complex system [1], [2]. The concept of SoS and the associated challenges have attracted significant attention in recent times [3]. As a relatively new computing paradigm, Service-Oriented Architecture (SOA) has provided a principled mechanism for constructing a SoS by dynamically integrating its component systems through service composition [4], [5], [6], [7].

A major step in constructing a service-oriented SoS is the realization of each component system as a Web service via some popular frameworks (e.g., Jersey[1] and Flask[2]). The functionality of each component system may be simple or relatively complex. From the perspective of users, each component system can be an atomic or a composite service and provides one or multiple Web-based invocation interfaces for SoS integration. The SoS is constructed finally by dynamically integrating different Web services by means of a service composition methodology based on Web technologies.

A service-oriented SoS usually operates in complicated and highly dynamic environments where the component systems are much more loosely coupled than simple in-house software systems. It is also different from traditional network-based systems as a set of novel challenges arises in a service-oriented SoS environment. These include (1) More effective execution and analysis technologies are needed to make independent and heterogeneous systems working cooperatively for achieving a common goal; (2) an SoS requires higher capabilities and performance than traditional systems to interact among themselves; and (3) reliability has become the focus of an SoS because failures in individual systems within the SoS may even lead to cascading failures [3]. Hence, the question of how to guarantee the performance of the constructed system is of significant importance for a service-oriented SoS.

Proactive Fault Management (or PFM) aims to ensure runtime quality of systems by failure avoidance and offers an effective mechanism for enhancing the reliability of a

- *H. Wang is with the School of Computer Science and Engineering and Key Laboratory of Computer Network and Information Integration, Southeast University, SIPAILOU 2, Nanjing 210096, China. E-mail: hbw@seu.edu.cn.*
- *L. Wang is with the School of Computer Science and Engineering and Key Laboratory of Computer Network and Information Integration, Southeast University, SIPAILOU 2, Nanjing 210096, China, and with the Department of Management Science and Engineering, Nanjing Forestry University, Nanjing, Jiangsu 210037, China. E-mail: leiwang@seu.edu.cn.*
- *Q. Yu is with the College of Computing and Information Sciences, Rochester Institute of Technology, 102 Lomb Memorial Drive, Rochester, NY 14623-5608. E-mail: qi.yu@rit.edu.*
- *Z. Zheng is with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510275, China, and the Key Laboratory of Machine Intelligence and Advanced Computing, Sun Yat-sen University, Ministry of Education, Guangzhou 510275, China. E-mail: zhzibin@mail.sysu.edu.cn.*
- *M.R. Lyu is with the Shenzhen Research Institute, and the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin HSB 101, Hong Kong, China, and with the School of Computer Science, National University of Defence Technology, Hunan 410073, China. E-mail: lyu@cse.cuhk.edu.hk.*
- *A. Bouguettaya is with the School of Information Technologies, The University of Sydney, NSW 2006, Australia. E-mail: athman.bouguettaya@sydney.edu.au.*

Manuscript received 7 Jan. 2014; revised 17 Sept. 2016; accepted 30 Sept. 2016. Date of publication 5 Oct. 2016; date of current version 20 June 2017. Recommended for acceptance by M. Dwyer.
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/TSE.2016.2615615

1. https://jersey.java.net/
2. http://flask.pocoo.org/

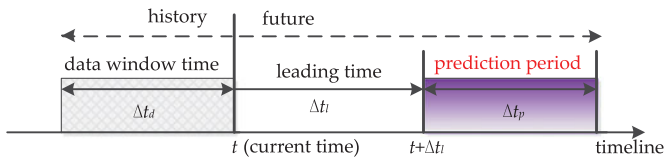


Fig. 1. Schematic view of online reliability prediction.

software system [8]. Specially, in the IEEE standard glossary, reliability is defined as "*the ability of a system or component to perform its required functions under stated conditions for a specified period of time*" [9]. Invariably, while integrating a service-oriented SoS, each component system has replaceable ones with similar (or even identical) functionalities. However, the Quality of Service (QoS) provided by each component system can be quite different and vary with time. To achieve PFM and avoid potential software and hardware failures and performance anomalies at the level of the SoS, each component system needs to be selected optimally while constructing the SoS. A key requirement in this regard is the successful deployment of a sustainable and stable SoS through optimal service selection. Further, the SoS requires accurate, real time, online prediction of the reliability of each invoked component system.

As illustrated in Fig. 1, online reliability prediction seeks to estimate the component system's reliability in the "near future" (i.e., the prediction time period of Δt_p) based on currently observed system execution parameters and historical records. Δt_l is defined as the *leading time* and starts from t and ends when the component system is invoked. The length of Δt_l should be greater than the time required to construct an optimal composite system. Δt_p is the *prediction period*, which corresponds to a future invocation time period. Δt_d is the *data window time* for compiling historic records.

In a service-oriented SoS, the prediction period Δt_p for each invoked component system is determined by the corresponding execution period. However, since the duration of invocation is usually uncertain, the length of Δt_p varies with one user's requirement to another.

To deal with the communication link instabilities caused by the needs of various users, reliability prediction for the component systems in a service-oriented SoS needs to capture changes in reliability during a variable prediction period. Assume that Δt_p is long enough for most users' requirements. The key idea behind reliability prediction with respect to the component systems is to predict the *reliability time series* (i.e., the reliability values at multiple continuously time points) within Δt_p. In practice, to avoid prediction difficulties caused by an overly long prediction period setting, the length of Δt_p should cover most users' usage habits and should be set in accordance with the particular application.

To the best of our knowledge, this is the first work on *online reliability time series prediction* for the component systems in a service-oriented SoS. Specifically, the aim is to meet the following three challenges:

1) The QoS parameters of a component system often change irregularly over time, which makes it difficult to identify time-dependent regularities in system reliability. The patterns of system parameters may also change with time.
2) Reliability prediction may cover a time interval containing multiple time points (in the form of a time series). Due to the dynamic nature of QoS parameters, it is difficult to assure high prediction accuracy over multiple time points in the near future on the basis of the updated system running state.
3) With the exception of response time, throughput, and reliability, the performance parameters of a component system (e.g., CPU usage, memory usage, and system load) are usually difficult to measure solely from client-side evaluations.

None of the existing approaches can systematically address the above challenges [10], [11]. For example, in collaborative filtering (CF)-based Web service reliability prediction (e.g., [12], [13], [14]), the prediction results come from the user-service failure probability matrix. The failure probability (which can be used to calculate reliability) is usually evaluated during the historical time period (i.e., data window time in Fig. 1). However, to deal with the complex and dynamic running environment, we still need to predict the online reliability time series based on the system parameters during the data window time. Likewise, with reference to online failure prediction methodologies for traditional computer systems (i.e., Failure Tracking, Symptom Monitoring, and Detected Error Reporting) [8], [11], we may assert the following. First, *failure tracking* based online failure prediction captures the time regularities of historic failures. As for a traditional non-networked computer system, the system performance may gradually decline during the execution life cycle. The occurrence time of failures will follow certain statistical rules. However, due to the highly variable QoS caused by uncertain network, service usages, and internal working states, this is often infeasible for component systems in SoS. Second, *Symptom Monitoring* and *Detected Error Reporting* require analyses of the system's internal performance parameters or log files. This renders them inapplicable to a component system of a service-oriented SoS; since the parameter information of remotely located component systems is usually not available. Moreover, the fact that the running environment of a component system is typically more dynamic makes the variation regularity of parameters more difficult to be discovered.

In contrast to existing online failure prediction issues, which focus on predicting system failures (the occurrence times and types of failures), our work aims to predict the reliability time series (i.e., the reliability values of multiple and continuous time points). Not all influencing factors for *the occurrence of system failures* and the *dynamic changes of reliability time series* are the same.

The reliability of the component system is affected often by several major factors, including (1) the unstable communication links between the component systems and the client; (2) the internal working status of the component systems, such as exceptional memory usage, CPU load, disk I/O, and unusual function calls; and (3) the loading capacity of the component systems under the corresponding throughput.

The above three factors significantly affect the response times and throughputs of the component systems. In particular, when the three factors of a component system vary, response time and throughput will change accordingly. More importantly, parameters such as response time and

throughput can be easily obtained via client-side evaluation of the target component system, which allows one to analyze a component system's reliability based on its response time and throughput.

This paper presents a novel online reliability time series prediction approach for the component systems in a service-oriented SoS. The approach serves the purpose of dynamic component selection for SoS construction by addressing the key challenges. The major contribution of this paper arises from the use of Probabilistic Graphical Models (*PGMs*) for analyzing historical and the most recent system parameters (including response time, throughput, and reliability). Relevant historical parameters are preprocessed and divided into time series of equal length. The concept of *motifs* is then adopted to describe the patterns of the historical parameter time series. First-order Markov chain rules are employed to capture the causal relationships between different time series of system parameters. These relationships are represented as Conditional Probability Tables (*CPTs*), which are used together with PGMs to make an online reliability time series prediction based upon the updated system parameters. Experiments conducted over real-world Web services have confirmed the effectiveness of the proposed approach.

The application of PGMs for online reliability time series prediction was first presented in a shortened form as a conference paper [15]. In that paper, we presented a comprehensive framework, referred to as *PGMs-RTSOP* (or *ROP* for short), which exploited PGMs to achieve accurate and robust Reliability Time Series Online Prediction for an SoS. The remainder of the present paper is organized as follows. We give an overview of related works in Section 2. We briefly introduce background knowledge and basic definitions in Section 3. We describe the proposed prediction methodology in Section 4. We present a case study in Section 5 and experimental results in Section 6. We conclude by identifying some important future directions in Section 7.

2 Related Works

In this section, we review previous works that are particularly relevant to the proposed approach, including reliability prediction for service-oriented systems and methodologies for online prediction.

2.1 Reliability Prediction

Service selection is a widely investigated topic amongst researchers studying the integration of distributed services and the construction of service-oriented systems. The QoS-MOS framework was proposed to support service selection for composing a system [16]. The MOSES tool can support runtime assurance of QoS (which involves reliability as well as performance) through service selection and coordination pattern selection mechanisms [17] under the prevailing distributed, dynamic and complex operating environment [18].

Reliability prediction has become an important issue in optimal service selection. Personalized reliability prediction usually utilizes Collaborative Filtering (CF) to predict the QoS (including reliability) of Web services on the basis of a limited number of client-side reliability evaluations [10], [12], [19], [20], [21], [22]. Clustering (CLUS)-based reliability prediction considers user-, service- and environment-specific parameters to predict the reliability of an atomic Web service in a specific time window by means of k-means clustering [23]. The overall system level reliability can be computed by aggregating prediction results on the component services on the basis of the composition (or workflow) structure [12], [24]. In such reliability prediction exercises, the historically average reliability is used to evaluate the performance of each Web service. The reliability of previously unknown Web services is then predicted.

Statistical time series models (e.g., Autoregressive Integrated Moving Average, ARIMA, and novel improved models) have also been used to predict future QoS and reliability of Web services [25], [26]. Using statistical time series models, predictions are made on the basis of trend statistics at the next time series point [27]. Each subsequent prediction depends on the previous predicted result. More nearby time points will enhance prediction accuracy.

It has been observed that the arrival times of failures in an atomic Web service follow an Erlangian distribution, i.e., the failures depend upon the running states (e.g., idle and active states) of services [13], [28].

In a highly dynamic execution environment, the QoS of each service varies with time. Effective service selection should be able to avoid interruptions during the operation of service-based applications [29]. Such reliability prediction has already been used to assure the software runtime quality [30], [31], [32].

Reliability prediction for a component system selection for a service-oriented SoS should be able to forecast the reliability during the execution period, if the component system has been invoked. Different from most existing works, this paper investigates an online reliability time series prediction methodology capable of providing some early information on future system reliability time series with a view to supporting more effective component system selection.

2.2 Prediction Methodologies

A variety of online prediction methodologies are available in literature on traditional computer systems. An example is *Online Failure Prediction* that aims to identify whether a failure will occur in the *near future* based on an assessment of the current running state of the system. This has become an important research issue in view of the increasing mobility of devices, changing execution environments, frequent updates and upgrades, online repairs and improvements, addition and removal of system components, and system/network complexity. Existing online failure prediction approaches can be grouped into the three categories [8], [11] described below.

(1) *Failure Tracking* based failure prediction aims to make inferences about upcoming failures by studying the occurrence of previous failures as well as the types of failures that have occurred. Most approaches rely on estimations of the probability distribution. For example, *Bayesian Predictors* [33] adopt a Bayesian framework to estimate the probability of failures on the basis of data related to previous failure occurrences. The work in [34] assumes that failures satisfy a Bernoulli experiment and collects statistics on time-between-failures (TBF) to estimate the probability $P(f_t)$ for a failure f that occurs at time t. *Co-occurrence* is another approach [35], which analyzes the temporal and

spatial compression for all events to examine correlations between different types of failures.

(2) *Symptom Monitoring* analyzes monitored data to detect symptoms (i.e., side effects of errors) that point to an upcoming failure. It can dynamically predict failures according to the current system state. Typical approaches for symptom monitoring include *Function Approximation*, *Classifiers*, *System Models*, and *Time Series Analyses*. *Function Approximation* discovers how historical parameters change with time and fit a function to predict the future system running state [36], [37], [38]. *Classifiers* divide historical parameters into a number of classes and assign the future state to the most similar class [39], [40], [41], [42], [43]. *System Models* model the system's running state together with failure occurrence so as to predict future system failures [40], [42], [44], [45], [46], [47], [48], [49]. *Time Series Analyses* make predictions by analyzing the characteristics of the time series on the basis of the system's historical system parameters [50], [51], [52], [53].

(3) *Detected Error Reporting* uses error reports (collected via some logging mechanism) as input to model chronological information on failure occurrence, conversion of failure state, and similarities between failures. Typical technologies include genetic algorithms [54], [55], Discrete Fourier Transform (DFT) [56], hidden semi-Markov Models (HSMM) [57], and alliance queue [58]. Rule-based Approaches are adopted [54], [55] to predict online failures using system log files. Episode rules (failure rules) are determined using data mining followed by manual selection. Semi-Markov chains and HSMM are used to model failure-prone system states [57]. Other approaches, such as pairwise alignment, are used to compute similarities between sequences by means of biological sequence analyses [58].

Many existing online failure prediction approaches estimate the probability of failure occurrence by conducting performance evaluation regularly (e.g., note gradual declines) of traditional computer systems. To predict the online reliability time series for component systems in SoS, most existing approaches perform reliability prediction by analyzing system running states through observed system QoS parameters. Different from the traditional computer system, the QoS of a service-oriented system changes irregularly. More determinants make the causal relation for evaluating complex regularities in the running states of the system. Although probability-based models can be used to capture the evaluation regularities, the complex causal relations for evaluating the running states makes existing online failure prediction methodologies fall short of being able to meet new prediction challenges. In this paper, we will combine probability and the complex causal relations based on probabilistic graphical models to model the evaluation regularities in the system's running status and predict the online reliability time series for the component systems of service-oriented SoS.

3 Preliminaries

Section 3.1 presents the reifying application to motivate the discussion of the proposed ROP approach. Section 3.2 briefly overviews the Dynamic Bayesian Networks (DBN) along with the 2-Time-slice Bayesian Network (or 2-TBN) which is a cornerstone of the approach being proposed in this paper. Section 3.3 clarifies the notations and basic definitions of major concepts used in this paper.

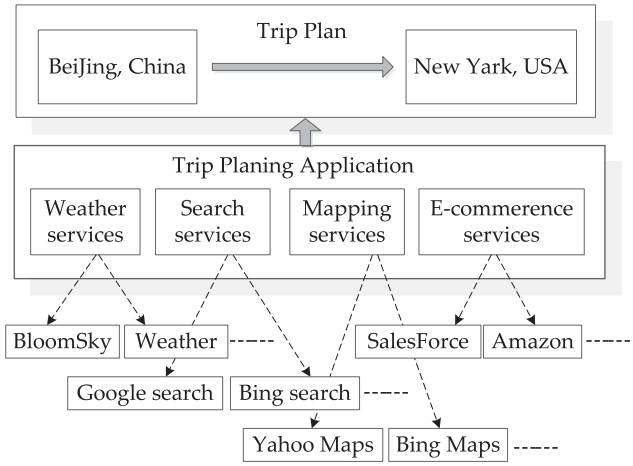


Fig. 2. Trip planning application for international travelers.

3.1 Reifying Through Application

In this section, we will examine how our ROP approach can be applied to a service-oriented SoS. First, we present an example for the application of a service-oriented SoS, a travel planning system, known as TripPlanner. Second, we examine how ROP can be applied to the trip planning application and support service selection for the service composition execution engine.

Let us consider a travel planning system, TripPlanner, which aims to provide services to international travelers. A service composition execution engine in TripPlanner selects optimal component systems from massive candidates and integrates the individual component systems. Reliable applications for TripPlanner are dynamically constructed by the service composition execution engine to fulfill the needs of travelers. TripPlanner can serve as a representative application scenario for a service-oriented SoS.

As illustrated in Fig. 2, suppose that a user, say George, is planning to travel from Beijing, China to New York, USA on a weekend morning. When turning to TripPlanner for help, George wants to get weather forecasting information and information on feature spots in and around New York to plan his touring route. When George has determined his feature spots, he may want to use a mapping service to select the hotel(s) and public transports in New York. He may also want an E-commerce platform to book the hotel and air tickets. TripPlanner will construct a trip planning application for George. Each requirement can be fulfilled by integrating a public Web service.

There may be a large number of publicly available candidate Web services that meet each of the requirements. For example, BloomSky and Weather Web services may be used for Weather forecasting. Google search and Bing search are helpful for information retrieval. Yahoo Maps and Bing Maps can provide mapping services, while SalesForce and Amazon are well-known E-commerce services. To guarantee the execution quality of the constructed system, TripPlanner should optimally select the Web services with desired online reliability. The results from the selected Web

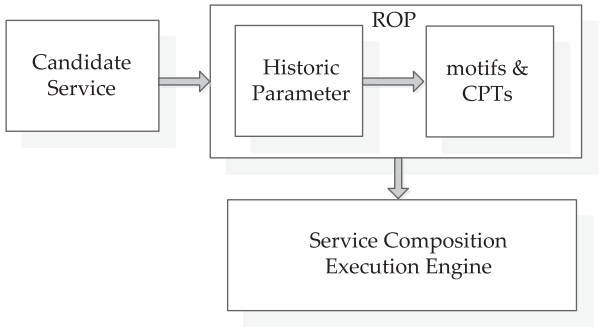


Fig. 3. Application framework of ROP.

services will be composed by the service composition execution engine. A reliable TripPlanner application will be constructed for George.

The ROP method for predicting the online reliability time series can serve the purpose of selecting reliable candidate Web services for a TripPlanner application. As illustrated in Fig. 3, ROP can work for the service composition execution engine through the following steps. First, we collect the historic parameters (including response time, throughput, and reliability) for each of the candidate Web services for TripPlanner. The historic parameters are then used to identify the time series motifs and construct the CPTs for the time series evaluation regularity. Online reliability prediction is provided based on the system parameters time series motifs collected in real time and the CPTs. The prediction results for each of the candidate services are gathered and used by the service composition execution engine for service selection.

ROP is capable to handle large-scale service composition applications. On the one hand, the ROP prediction can be focused on some critical component systems. Through this method, we can guarantee the reliability of the critical component systems. On the other hand, a broker node can also be deployed in the Internet. For example, we can use a centralized broker to collect the ROP prediction results by limiting the number of the couples from client side to identify candidate component systems. For all of the client sides and the candidate component systems, a sparse matrix can be used to record the prediction value for each time point of the online reliability time series. Collaborative filtering can be used to calculate the missing value in the sparse matrix (cf. [10]). In this way, the ROP prediction can be expanded through the broker. For each client-side node of the application, we only need to deploy ROP prediction on some critical component systems. The broker node will provide the prediction results to the service composition execution engine.

3.2 Dynamic Bayesian Networks

As a class of probabilistic graphical models, DBNs are well-known in the context of template-based representations and reasoning [59]. When a dynamic system characterized by the template variables $\chi = \{\chi^{(0)}, \chi^{(1)}, \ldots, \chi^{(t)}, \chi^{(t+1)}, \ldots\}$ satisfies the independence assumption of a first-order Markov chain process, we can represent the process by DBNs. As illustrated in Fig. 4a, the independence assumption states that, the variables in $\chi^{(t+1)}$ is only determined by $\chi^{(t)}$, it cannot directly depend on variables in $\chi^{(t')}$, for all $t \geq 0$ and $t' < t$.

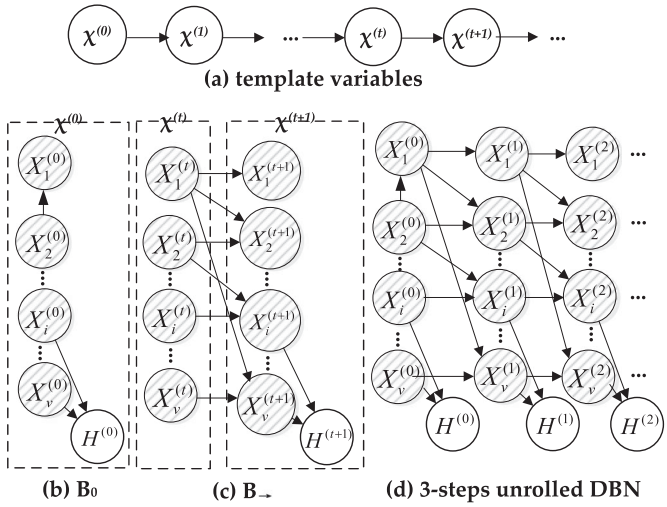


Fig. 4. Dynamic Bayesian networks.

Generally, a DBN is a two-tuple $< B_0, B_{\rightarrow} >$, where B_0 is a Bayesian network over $\chi^{(0)}$ representing the initial state distribution, and $B_{\rightarrow}$ represents the transition model for the first-order Markov process. Specially, the transition model is a Conditional Probability Distribution (or CPD), and can be represented by a conditional Bayesian network, also called a 2-Time-slice Bayesian Network. Fig. 4c gives a 2-TBN model as an example, where $X_i^{(t)} \in \chi^{(t)}$, for any integer $i \in [1, v]$, represent observed variables, $H^{(t)} \in \chi^{(t)}$ represents hidden variable and is determined by the observed variables. 2-TBN describes the transition model for template variables evolving from $\chi^{(t)}$ to $\chi^{(t+1)}$, for all $t > 0$.

In practice, a Conditional Probability Table is typically used to represent the conditional probability distribution of jointly random variables for 2-TBN [60]. Each template variable of $\chi^{(t+1)}$ in the 2-TBN has a CPT that represents all possible values of the conditional probability for each value of template variable $X_i^{(t+1)}$.

The DBN model can be used to reason upon the subsequent template variables for the dynamic system. As illustrated in Fig. 4d, we can unroll a DBN over any desired time span by B_0 and $B_{\rightarrow}$, where

- the structure and CPDs for $X_i^{(0)}$ are the same as for those in B_0, and
- the structure and CPDs of $X_i^{(j)}$ for $j > 0$ are the same as those for $X_i^{(t+1)}$ in $B_{\rightarrow}$.

3.3 Notations and Definitions

This section provides basic definitions related to reliability for a service-oriented SoS as well as the definition of our prediction objective. The major notations of our online reliability time series prediction are listed in Table 1.

The reliability of a service-oriented SoS describes the probability of successful message delivery between the two endpoints (the component system and the client) in the presence of component-system/network failures and component system performance anomalies. As defined in ANSI/IEEE STD-729-1991 [9], *software reliability* is widely used to describe the reliability of software systems, component-based systems, Web services, etc., [10], [61], [62]. As a

TABLE 1
Notations of Online Reliability Time Series Prediction

Categories	Notations	Descriptions
Number	g	The number of time series in a continuously historic system parameter
	n	The number of time points within a time series
	k	The number of motifs for the set of a type of system parameter time series
	n_f	The number of invocations with failure responses or performance anomalies
	n_i	The total number of invocations
	η	The total number of the time points in a continuously system parameter, and $\eta = g \times n$
Time	Δt_d	Data window time, i.e., the time span for the component system's running state on which a prediction depends
	Δt_l	Leading time, i.e., the time span for which the period for prediction should skip ahead
	Δt_p	Prediction period, i.e., the time span for the reliability time series to be predicted
	t	Current time
	T	The up-to-date time span
	u	A time span
	u_i	The ith ($i \in [1, \eta]$) time span for evaluating a system parameter value
Time Series	$\overrightarrow{p}_r$	The online reliability time series to be predicted
	$\overrightarrow{\chi}$	The set for the time series template variables
	$\overrightarrow{\chi}^{(T)}$	The time series for $\overrightarrow{\chi}$ in time span T
	$\overrightarrow{R}$	The set for the reliability time series with the evolution of time
	$\overrightarrow{Q}$	The set for a QoS parameter time series with the evolution of time
	$\overrightarrow{D}_{rt}$	The set for the time series of response time parameter during data window time
	$\overrightarrow{D}_t$	The set for the time series of throughput parameter during data window time
	$\overrightarrow{P}_{rt}$	The set for the time series of response time parameter during prediction period
	$\overrightarrow{P}_t$	The set for the time series of throughput parameter during prediction period
	$\overrightarrow{P}_r$	The set for the time series of reliability parameter during prediction period
	$\overrightarrow{O}_{rt}$	The real-time observed up-to-date response time parameter time series
	$\overrightarrow{O}_t$	The real-time observed up-to-date throughput parameter time series
	$\overrightarrow{O}_r$	The real-time observed up-to-date reliability parameter time series
	$\overrightarrow{D}_{rt}^{(i)}$	The ith ($i \in [1, g]$) time series in $\overrightarrow{D}_{rt}$
	$\overrightarrow{r}_r$	The observed real reliability time series during the prediction period
Motifs	$\widehat{Q}$	The set of the motifs for a QoS parameter Q's time series
	$\widehat{D}_{rt}$	The set of the motifs for the response time parameter time series during data window time
	$\widehat{D}_t$	The set of the motifs for the throughput parameter time series during data window time
	$\widehat{P}_{rt}$	The set of the motifs for the response time parameter time series during prediction period
	$\widehat{P}_t$	The set of the motifs for the throughput parameter time series during prediction period
	$\widehat{P}_r$	The set of the motifs for the reliability parameter time series during prediction period
	$\widehat{D}_{rt}(i)$	The ith ($i \in [1, k]$) motif in $\widehat{D}_{rt}$
	A	The set for the motifs in a Λ
CPT	B	The set for the conditional probability table for the right side nodes in the m_DBNs
	$CPT(\widehat{P}_{rt})$	The conditional probability table for $\widehat{P}_{rt}$
	$CPT(\widehat{P}_t)$	The conditional probability table for $\widehat{P}_t$
	$CPT(\widehat{P}_r)$	The conditional probability table for $\widehat{P}_r$
Models	B_0	The Bayesian Network for the initial template variables set $\chi^{(0)}$
	$B_{\rightarrow}$	The two-time slice Bayesian Network describing the transition model for the template variables from $\chi^{(t)}$ to $\chi^{(t+1)}$
	$\widehat{B}_{obs}$	Bayesian Network for the real-time observed system parameters time series which are represented by their motifs
	$\widehat{B}_{\rightarrow}$	The two-time slice Bayesian Network describing the transition model for the time series motifs from Δt_d to Δt_p
	Λ	The simplified probability representation for an m_DBNs model
Parameter Values	$\overrightarrow{p}_r^{(i)}$	The value for the ith ($i \in [1, n]$) time point in $\overrightarrow{p}_r$
	$\overrightarrow{D}_{rt}^{(i)}(j)$	The value of jth ($j \in [1, n]$) time point in the time series $\overrightarrow{D}_{rt}^{(i)}$
	$\overline{D}_{rt}(j)$	The average value of the jth ($j \in [1, n]$) time point for the time series of $\widehat{D}_{rt}$
	$\overrightarrow{O}_{rt}(j)$	The value for the jth ($j \in [1, n]$) time point in the observed up-to-date response time parameter time series
	$r(u)$	The reliability value of a component system evaluated by a client during time span u
	$rt(i)$	ith ($i \in [1, \eta]$) value of the response time parameter
	$t(i)$	ith ($i \in [1, \eta]$) value of the through parameter
	$r(i)$	ith ($i \in [1, \eta]$) value of the reliability parameter
	$rt(t_j)$	The returned response time parameter for the jth time of invocation
	$t(t_j)$	The value of throughput for the jth time of invocation
	$\overrightarrow{r}_r^{(i)}$	The value for the ith ($i \in [1, n]$) time point in $\overrightarrow{r}_r$

TABLE 1
(*Continued*)

Categories	Notations	Descriptions
	$rt(t_j)^s$	The returned response time parameter for the sth thread's jth time of invocation
	$t(t_j)^s$	The value of throughput for the sth thread's jth time of invocation
Miscellaneous	χ	The template variables set with the evolution of time
	$\chi^{(t)}$	The template variables set in time t
	γ	Failure-rate, and $\gamma = n_f/n_i$
	X_i	The set for the ith ($i \in [1, v]$) template variable with the evolution of time
	χ^2_{rt}	chi-square statics for the observed up-to-date response time parameter time series
	χ^2_t	chi-square statics for the observed up-to-date throughput parameter time series

type of complex software system, service-oriented SoS constructs the system by dynamically integrating the component systems through a service composition technique. Accordingly, the definition of software reliability also applies to a service-oriented SoS.

Moreover, reliability can be measured in different forms, such as Mean Time To Failure (MTTF), Mean Time Between Failures (MTBF), hazard rate (the system has survived until time t) and failure-rate (or failure probability) [63]. As for the uncertain running environment of a component system, the time interval between successive failures appears usually with no obvious regularity, which makes the metrics of MTTF, MTBF, and hazard rate inapplicable for the systems referred to in this paper. Therefore, just like [10], we use failure-rate as a basis for the component system's reliability measurement.

The failure-rate reflects the probability that an invocation may fail under the evaluation period [61]. Similar to [10], [12], we calculate in this paper failure-rates during continuous invocation evaluations under a fixed time period for an observed component system. For each evaluation period, the exponential reliability function is then used to calculate the component system's normalized reliability based on its failure-rate [63]. Different from the traditional computer systems, the performance of component systems changes irregularly. The failure responses or the performance anomalies for each invocation make the component system deviate from the requirement of the users. Consequently, failures for the component system invocations include (1) the failure response, and (2) the performance anomalies (e.g., time-outs). The failure-rate is calculated by counting the number of fail invocations under a fixed total invocation time and a fixed time period. The reliability of a component system of a service-oriented SoS is defined as follows [10], [12]:

Definition 1 (Reliability). *Assume that the twofold endpoints consist of a component system S and a client node C. Let u represent a time period, t_u be any point of time during u. The reliability of S is evaluated according to the failure-rate, when S is continuously invoked by C during period u. We formally define the reliability of S at time t_u evaluated by C as*

$$r(t_u) = r(u) = e^{-\gamma \times len(u)}, \tag{1}$$

where $len(u)$ represents the length of time period u, $\gamma = {}^{n_f}/_{n_i}$ is the failure-rate of equal length time-interval client-side invocation evaluations for S, n_i represents the number of total invocation times, and n_f denotes the number of invocations that meet with failure responses or performance anomalies, which deviate from the user (or application system) requirement.

Given the above reliability definition, this paper seeks to predict the time series of component system S's reliability relative to client C. The time series covers the time duration of prediction period Δt_p. The prediction is based on the system's running state (reflected by the QoS parameters) during data window of time. We can now define an online reliability time series as follows:

Definition 2 (Online Reliability Time Series). *Let Δt_p represent the prediction period (see Fig. 1), we divide Δt_p into n sub time durations with equal length, i.e., $u_1, u_2, \ldots, u_n$. The online reliability time series to be predicted ($\overrightarrow{p}_r$) for a component system S relative to client C is defined as a vector describing the reliability values in multiple time points, i.e.,*

$$\overrightarrow{p}_r = (\overrightarrow{p}_r^{(1)}, \overrightarrow{p}_r^{(2)}, \ldots, \overrightarrow{p}_r^{(n)}),$$

where n represents the number of time points within a time series; $\overrightarrow{p}_r^{(i)}$, for any integer $i \in [1, n]$, represents the ith time point's reliability value, where $\overrightarrow{p}_r^{(i)} = r(u_i)$.

4 Time Series Prediction

To cope with the three prediction challenges for a component system in a service-oriented SoS (see Section 1), we will study the conditional dependency for the dynamically waving system runtime parameters of the component systems. We use the system parameter time series to characterize the system running status and make online reliability prediction by modeling the causal relations [59] between the system running status during the data window time (i.e., the time period of Δt_d) and the system reliability in prediction period (i.e., Δt_p).

First, we propose a motifs-based Dynamic Bayesian Networks (or m_DBNs) model to represent the casual relations for the evolution of system parameters time series from data window time to prediction period. Second, we give the prediction steps of our ROP approach using the m_DBNs model.

4.1 Model

The typical *PGM* model supports temporal processes, such as the DBNs model (retrieve Section 3.2), which evolve through continuous time [59]. Recently, DBNs have been widely used to represent the conditional dependency among time intervals and predict the near-time future.

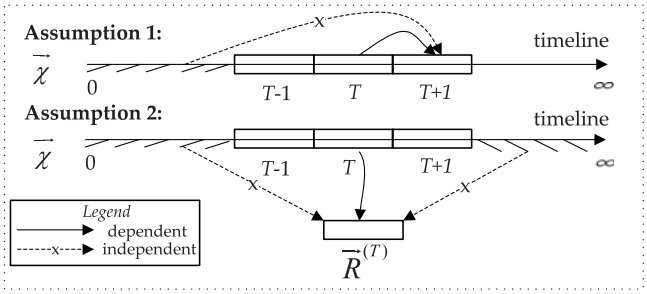


Fig. 5. Independence assumptions.

As stated in the description of challenge 1 (see Section 1), the changes from adjacent time points of historical system parameters may not exhibit obvious causality due to the dynamics of the SoS runtime environment. This means that the uncertain network status, internal working status, and system load lead to random waves in component system runtime parameters.

The regulation of changes in the system parameters' time series can reflect the system's running state for the corresponding time duration. Therefore, we look inside neighboring system parameters time series to discover regularities. In particular, to evaluate the real-time performance, application-level metrics (i.e., response time and throughput) or hardware-level metrics (e.g., instructions execution rate and cache access behavior) have been used to describe the running states of Web-based systems [64]. To predict the online reliability time series, we use the combination of time series for response time and throughput parameters to reflect the running states for a component system. Intuitively, when the waiting time for an invoked component system has significantly exceeded its normal response time, it is reasonable to believe that the component system may be in the state of low reliability. The issue on what kind of dependencies between the performance parameters (e.g., response time) and reliability may be derived from statistical learning.

Consider that a component system is working under a certain running state. It will be transferred to the next running state if it is influenced by a specific event (e.g., fluctuations of system load or network performance). Nonetheless, it is always uncertain for the event type to have occurred for the current component system. Therefore, the component system's running state in the future is affected only by the current running state and the event itself. It is independent of the previous running state. Hence, we can say that the time series related to the system parameters will evolve in a first-order Markovian chain manner, i.e., the near future system parameters time series will depend only on the up-to-date time series. Earlier time series are irrelevant for predicting the future. If the system's running state can be reflected by some system parameters, for the above evolving characteristic, the time series associated with the parameters can describe only the current running state; it cannot reflect the system running state during other time durations.

As illustrated in Fig. 5, the entire axis is divided into time spans of equal length, i.e., $0, \ldots, T-1, T, T+1, \ldots$, where T is the current time span. The templates variables, χ, are divided as time series, i.e., $\overrightarrow{\chi}$. We can use the time series in $\overrightarrow{\chi}$ to reflect the system runtime status for the corresponding time period. For example, the system parameters' time series $\overrightarrow{\chi}^{(T)}$ represents the system running state under time span T. We use the following two *independence assumptions* to describe the dynamic evolution for the component system's system parameters time series [59]:

1) Markovian Process Assumption: The first-order Markovian chain evolution nature of system parameters time series makes the future system parameters time series (i.e., $\overrightarrow{\chi}^{(T+1)}$) to be determined only by the current system parameters time series $\overrightarrow{\chi}^{(T)}$. It is independent of the previous system parameters time series from 0 to $T-1$.
2) Conditional Independence Assumption: Let $\overrightarrow{R}$ be the reliability time series, the current reliability time series $\overrightarrow{R}^{(T)}$ is determined only by the current running state of the component system. The current system running state is reflected by $\overrightarrow{\chi}^{(T)}$, that is to say, $\overrightarrow{R}^{(T)}$ is in dependence of $\overrightarrow{\chi}^{(T)}$. It is also independent of the other time span's system parameters time series (i.e., the system parameters time series fall in the ranges 0 to $T-1$, and $T+1$ to ∞).

As for prediction challenge 2, to achieve online time series prediction and a high prediction accuracy, we propose an augmentation of the traditional DBNs, *m_DBNs*, which uses the above two independence assumptions to model the evolution regularity between the historic system parameters time series (i.e., the system parameter time series during the data window time Δt_d and the time series during prediction period Δt_p). As shown in Fig. 6, m_DBNs combines the traditional DBNs model with time series motifs [65], [66], [67]. The parameters are represented by the patterns exhibited by the time series (henceforth referred to as *motifs*) discovered from historical data to facilitate the prediction of future time series.

In sum, m_DBNs models the casual relationships on a component system running parameter time series' temporal evolution regularities. The time series motifs of response time and throughput are used to describe the running states of the component system for a specified time period [64]. As for a component system, the following casual relationships are assumed and used for online reliability time series prediction based on the two independence assumptions [59]. First, the component system's running state during prediction period is only determined by the running state during its data window time. Second, the component system's running state only determines its reliability time series during the same time period.

More specifically, we divide the system parameter from a long time period into multiple time series of equal length. The recurring time series are referred to as motifs in the *m_DBNs* model. As a result, the possible values for each node in the *m_DBNs* model are the corresponding parameter motifs. We define motifs as follows:

Definition 3 (motifs). *Let* $\overrightarrow{Q} = \{\overrightarrow{Q}^{(1)}, \overrightarrow{Q}^{(2)}, \ldots, \overrightarrow{Q}^{(T)}\}$ *be the time series which are divided from a long-term continuously QoS parameter Q of a component system with the time spans of* $0, 1, \ldots, T$. *After a clustering algorithm (e.g., k-means) on* $\overrightarrow{Q}$, *motifs for the parameter is defined as the centroids of the*

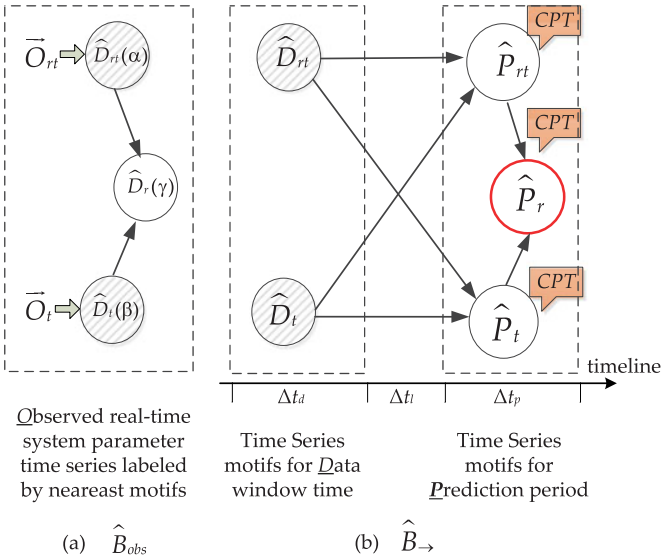


Fig. 6. Motifs-based dynamic Bayesian networks.

resultant time series clusters, i.e., $\widehat{Q} = \{\widehat{Q}(1), \widehat{Q}(2), \ldots, \widehat{Q}(k)\}$, *where* k *represents the number of motifs.*

To better describe the system parameter time series during the prediction period, we separately identify the motifs for each parameter time series within a given data window time and prediction period. For any QoS parameter, Q, we separately collect time series within a Data window time ($\overrightarrow{D}_Q$) and a Prediction period ($\overrightarrow{P}_Q$) during the historical system parameter evaluation. The motifs are calculated as $\widehat{D}_Q$ and $\widehat{P}_Q$, respectively. As stated above, the first-order Markov chain rule and motifs are adopted in the DBNs model. The *m_DBNs* model is fundamentally different from existing DBNs. The variables for template transition in m_DBNs are represented by time series motifs. Each motif can be regarded as a step within the evolution trajectory, which contains multiple continuous DBN template variables. We define the *m_DBNs* model as follows:

Definition 4 (m_DBNs). *A motifs-based Dynamic Bayesian Networks is a pair of* $< \widehat{B}_{obs}, \widehat{B}_{\rightarrow} >$, *where the values for each node are the corresponding parameter motifs, where*

- $\widehat{B}_{obs}$ *is a Bayesian Network of the real-time observed system parameter motifs (during time span* T*), representing the dependence rules for the system parameters motifs under time span* T.
- $\widehat{B}_{\rightarrow}$ *is a 2-time-slice Bayesian network for the system parameter time series motifs transformation from the data window time to the prediction period, where each time-slice is composed of a time series motif.*
- *The Conditional Probability Tables for each of the prediction period nodes in* $\widehat{B}_{\rightarrow}$ *are the Conditional Probability Distribution under the joint distribution of their parent nodes.*
- *As for the online prediction of desired time series span, the distribution of the system parameter time series motifs over* T *and* T*'s prediction period is defined in terms of one-step unrolled Bayesian Networks, where* $\widehat{B}_{obs}$ *is the distribution of the initial system parameter time series motifs, and* $\widehat{B}_{\rightarrow}$ *describes the unrolling rule.*

As for prediction challenge 3, the nodes in the *m_DBNs* corresponding to the component system's QoS parameters, including *Response Time*, *Throughput*, and *Reliability*, are evaluated through client side observation.

In particular, the same response time parameter for a component system under different running statuses may lead to different throughput values. Consequently, the response time and throughput are independent in a single time-slice Bayesian network for the dynamic variation of service load and uncertain network status. Since both are easy to measure and can be used to reflect the component system runtime status, they are denoted as shaded nodes.

On the other hand, the joint distribution of response time and throughput always significantly impacts the system runtime status, as well as its *Reliability*. Hence, the latter is usually derived on the basis of both response time and throughput; it can be regarded as a hidden variable (denoted as an unshaded node). According to the second independence assumption presented above, reliability depends only on current system parameters time series of response time and throughput.

In $\widehat{B}_{obs}$ (see Fig. 6a), the realtime Observed response time and throughput parameters time series ($\overrightarrow{O}_{rt}$ and $\overrightarrow{O}_t$) are labeled separately by the nearest motifs of response time and throughput parameter time series for the data window time. We denote these as $\widehat{D}_{rt}(\alpha)$ and $\widehat{D}_t(\beta)$, respectively. The combination of $\widehat{D}_{rt}(\alpha), \widehat{D}_t(\beta)$ can determine the reliability motif $\widehat{D}_r(\gamma)$ under the observation time span.

In addition, dependencies across the time series boundaries (i.e., from Δt_d to Δt_p) follow the first-order Markov chain rule whereas dependencies within the same time series follow the causal relationships among response time, throughput, and reliability, as mentioned above.

In $\widehat{B}_{\rightarrow}$ (see Fig. 6b), the following dependency rules are modeled. First, according to the first independence assumption, the component system's running status in the prediction period is determined only by the running state during the period on data window time. The combination of the time series motifs within Data window time for response time ($\widehat{D}_{rt}$) and throughput ($\widehat{D}_t$) has causal relationships with the component system's running states in the Prediction period, i.e., with the motifs of $\widehat{P}_{rt}$ for response time and the $\widehat{P}_t$ for throughput, respectively. Second, according to the second independence assumption, similar to $\widehat{B}_{obs}$, $\widehat{P}_r$ is determined by the combination of $\widehat{P}_{rt}$ and $\widehat{P}_t$.

In the m_DBNs model, the CPTs for the nodes over the prediction period can be used to describe the temporal evaluation regularity of the motifs for the historical in $\widehat{B}_{\rightarrow}$. The CPTs for $\widehat{P}_{rt}$, $\widehat{P}_t$, and $\widehat{P}_r$ are constructed on the basis of the statistics of the historical evaluation process for the historical parameters.

Fig. 7 gives a brief demonstration of the CPT for $\widehat{P}_{rt}$. Assuming that the motifs number k is set to 2, let the motifs that have been identified for $\widehat{D}_{rt}$, $\widehat{D}_t$, and $\widehat{P}_{rt}$ in the historic system parameter be: $\widehat{D}_{rt} = \{\widehat{D}_{rt}(1), \widehat{D}_{rt}(2)\}$, $\widehat{D}_t = \{\widehat{D}_t(1), \widehat{D}_t(2)\}$, and $\widehat{P}_{rt} = \{\widehat{P}_{rt}(1), \widehat{P}_{rt}(2)\}$, respectively. The CPT for $\widehat{P}_{rt}$ is a table describing the conditional probability from the joint distribution of $\widehat{P}_{rt}$'s parents nodes to $\widehat{P}_{rt}$. Specially, each

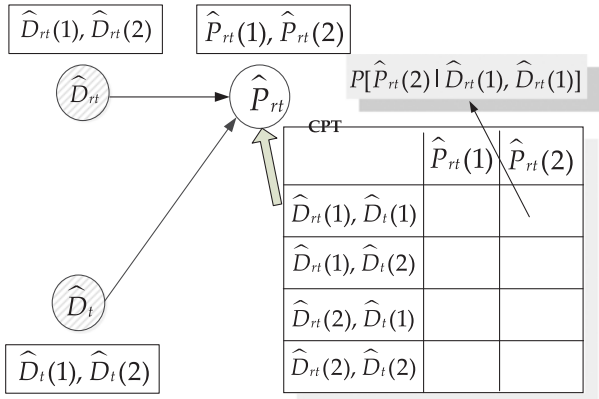


Fig. 7. Brief demonstration of conditional probability table (CPT).

possible combination of the values of $\widehat{D}_{rt}$ and $\widehat{D}_t$ represents a conditional item. The probability from the conditional item to each value of $\widehat{P}_{rt}$ describing the distribution of the conditional probability arises within historic temporal evaluations.

Prediction of the online reliability time series is executed by unrolling the m_DBNs based on the CPTs for the nodes over prediction period and the motifs in $\widehat{B}_{obs}$. The proposed m_DBNs model can support the uncertain execution environment for predicting the online reliability time series associated with the component systems. The result obtained by using a one-step-ahead time series prediction is a reliability time series. It is fundamentally different from the traditional statistical time series prediction models (e.g., ARIMA model [27]), as each step-ahead prediction result only contains one time point.

For practical purpose, we can simplify the m_DBNs model probability representation by a two-tuple, i.e., $\Lambda = < A, B >$, where

- A represents the featured patterns of system parameters time series identified from historical system parameters, for both data window time Δt_d and prediction period Δt_p. Actually, the patterns of system parameters time series can be represented by types-types of system parameters motifs, i.e.,

$$A = \{\widehat{D}_{rt}, \widehat{D}_t, \widehat{P}_{rt}, \widehat{P}_t, \widehat{P}_r\}.$$

- B is the set of CPTs for the nodes over prediction period, each CPT describes the system parameters motifs' transition probability from the parent nodes to the target node, i.e.,

$$B = \{CPT(\widehat{P}_{rt}), CPT(\widehat{P}_t), CPT(\widehat{P}_r)\},$$

where $CPT(\cdot)$ represents a CPT.

The m_DBNs with the graphical model (cf. Fig. 6) and the simplified probability representation Λ will be used for our online reliability time series prediction.

4.2 Approach

The proposed *ROP* using m_DBNs for online reliability time series prediction consists of five main steps. First, in *Data Preparation*, we preprocess the historical system parameters. Second, in the stage of *Motifs Discovery*, we discover motifs from the historical time series of system parameters. Third, in *Time Series Representation*, we use the motifs to represent the time series of the system parameters by labeling them using the most similar motifs. Fourth, in *Conditional Probability Table Construction*, we derive the causal relationships from historical data and construct the *CPTs*. Finally, in *Reliability Time Series Prediction*, we make predictions according to the currently observed up-to-date (i.e., real-time) time series of the system parameters.

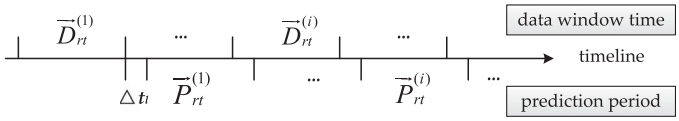


Fig. 8. Response time parameter time series.

4.2.1 Data Preparation

Data preparation refers to the stage of preprocessing the continuously collected historical parameters of response time, throughput and reliability for training the m_DBNs model. We divide each of the historical parameters into two different groups of time series—the data window time and prediction period, respectively. As for the nodes in $\widehat{B}_{\rightarrow}$, the motifs for each parameter, they can be discovered from two categories of the time series, which are different for the left column and the right column nodes in $\widehat{B}_{\rightarrow}$.

To enable the m_DBNs to model continuous time series with time evolutions (i.e., $0,\ldots,T-1,T$), we divide the time series for adjacent nodes in m_DBNs into the same length. Hence, the lengths of the data window time of Δt_d and the prediction period of Δt_p are the same.

As an example for the data preparation of response time parameter (see Fig. 8), we continuously divide the time series from the historic parameter. In this way, we capture the continuous evolutions for the historic response time parameter time series. Each data window time series of response time parameter has a time series for its prediction period. We have the following two types of time series.

First, we divide the historic response time parameters into multiple continuous time series for the data window time as $\overrightarrow{D}_{rt}$. The time span for each of the time series is Δt_p. Letting T be time span for the last time series of the response time parameter, the time series in the data window time for response time parameter can be represented as

$$\overrightarrow{D}_{rt} = \{\overrightarrow{D}_{rt}^{(1)}, \ldots, \overrightarrow{D}_{rt}^{(i)}, \ldots, \overrightarrow{D}_{rt}^{(T)}\}, \quad (2)$$

where $\overrightarrow{D}_{rt}^{(i)}$ represents the ith time series in $\overrightarrow{D}_{rt}$, and it is a vector containing multiple response time parameter data points for the component system. We further assume that n is the number of time points in time series $\overrightarrow{D}_{rt}^{(i)}$. Let $\overrightarrow{D}_{rt}^{(i)}(j)$ represent the jth data point in the time series of $\overrightarrow{D}_{rt}^{(i)}$, then we can get $\overrightarrow{D}_{rt}^{(i)}$ as

$$\overrightarrow{D}_{rt}^{(i)} = (\overrightarrow{D}_{rt}^{(i)}(1), \ldots, \overrightarrow{D}_{rt}^{(i)}(j), \ldots, \overrightarrow{D}_{rt}^{(i)}(n)). \quad (3)$$

Second, to obtain the time series for the prediction period, we re-divide the time period for response time parameter and then obtain the *near future* time series. We choose the time period after Δt_l for each $\overrightarrow{D}_{rt}^{(i)}$, which is

defined as $\overrightarrow{P}_{rt}^{(i)}$. In this way, we also divide the response time parameter into time series data. These also contain n data points in each time series. It is important to note that, during the historical execution parameter of the component system, when the response time parameter time series is $\overrightarrow{D}_{rt}^{(i)}$, the corresponding time series during the prediction period will be $\overrightarrow{P}_{rt}^{(i)}$.

The same process can be applied to preprocess and obtain the other parameters' time series (i.e., $\overrightarrow{D}_t$, $\overrightarrow{P}_t$, and $\overrightarrow{P}_r$) for the historical parameters.

4.2.2 Motifs Discovery

To identify the featured patterns of time series (or motifs) from the parameters of response time, throughput and reliability of a component system, we group similar time series in each preprocessed historic parameter by means of a K-means clustering algorithm.

In the following two steps, we use the response time parameter as an example to illustrate motifs discovery and the same process can be applied to other parameters.

First, we discover the motifs for the data window time of response time parameter time series $\overrightarrow{D}_{rt}$. We use the euclidean distance to calculate the distance between two time series. Hence, the distance between two time series $\overrightarrow{D}_{rt}^{(a)}$ and $\overrightarrow{D}_{rt}^{(b)}$ is

$$dist[\overrightarrow{D}_{rt}^{(a)}, \overrightarrow{D}_{rt}^{(b)}] = \sqrt{\sum_{j=1}^{n} [\overrightarrow{D}_{rt}^{(a)}(j) - \overrightarrow{D}_{rt}^{(b)}(j)]^2}, \quad (4)$$

where $\overrightarrow{D}_{rt}^{(a)}(j)$ and $\overrightarrow{D}_{rt}^{(b)}(j)$ represent the value of jth time point in $\overrightarrow{D}_{rt}^{(a)}$ and $\overrightarrow{D}_{rt}^{(b)}$, respectively, and n is the total number of time points within the time series. Once the clusters have been formed by the K-means clustering algorithm using the above distance function, the motifs for the data window time parameter time series of response time are calculated as the centroids of the clusters. These may be formalized as

$$\widehat{D}_{rt} = \{\widehat{D}_{rt}(1), \widehat{D}_{rt}(2), \ldots, \widehat{D}_{rt}(k)\}, \quad (5)$$

where k represents the number of the motifs, which is the centroids number for the K-means algorithm.

Second, we will use the motifs discovery method presented earlier for $\overrightarrow{D}_{rt}$ to discover motifs in $\overrightarrow{P}_{rt}$ for the prediction period parameter time series. As mentioned in Section 4.1, for the simplified probability representation Λ of m_DBNs model, the tuple A for the patterns of the system parameter time series can be identified via motifs discovery. As for the response time parameter, the historical response time parameter time series for the data window time is divided as $\overrightarrow{D}_{rt}$. The motifs for $\overrightarrow{D}_{rt}$ can be identified as $\widehat{D}_{rt}$. The motifs for response time parameter time series during the prediction period and the other parameters (throughput and reliability) are identified using a method similar to motifs discovery.

4.2.3 Time Series Representation

In this step, we label system parameters time series by the discovered motifs. In particular, each time series of a system parameter is labeled by the nearest motif discovered in the previous step. This allows the time series system parameters to be represented by the respective motifs so that their causal relationships can be derived. Again, we will use response time parameter as an example and the same process applies to other parameters.

Let k represent the motifs number for each system parameter time series. For some integer $\varpi \in [1, k]$, we label the response time parameter time series to each data window time $\overrightarrow{D}_{rt}^{(i)}$ by $\widehat{D}_{rt}(\varpi)$ as

$$\overrightarrow{D}_{rt}^{(i)} \leftarrow \widehat{D}_{rt}(\varpi), \quad (6)$$

where $\widehat{D}_{rt}(\varpi) \in \widehat{D}_{rt}$ is the nearest motif for $\overrightarrow{D}_{rt}^{(i)}$. For any integer $w \in [1, k]$, $\widehat{D}_{rt}(\varpi)$ can be determined by the following equation:

$$\widehat{D}_{rt}(\varpi) = \underset{\widehat{D}_{rt}(w)}{\arg\min} \{dist[\overrightarrow{D}_{rt}^{(i)}, \widehat{D}_{rt}(w)]\}. \quad (7)$$

Accordingly, each system parameter's time series for prediction period is labeled according to the corresponding motifs.

In practice, the nearest motifs for $\overrightarrow{D}_{rt}^{(i)}$ may be not unique. As an example, let integer $\varpi' \in [1, k]$, and $\varpi < \varpi'$, the following equation may hold:

$$\widehat{D}_{rt}(\varpi') = \widehat{D}_{rt}(\varpi).$$

The motifs with different serial numbers (i.e., ϖ' and ϖ) together may constitute the nearest motif for the time series of $\overrightarrow{D}_{rt}^{(i)}$. Ambiguities can arise in the time series representation for $\overrightarrow{D}_{rt}^{(i)}$. The ambiguities can be eliminated by improving the precision of the values for the response time parameter. However, if an ambiguity does arise, the motif with the least serial number can be used to represent the time series. As for the example, $\widehat{D}_{rt}(\varpi)$ will be used to represent $\overrightarrow{D}_{rt}^{(i)}$.

4.2.4 Conditional Probability Table Construction

In the proposed *m_DBNs*, the evaluation rules of the component system's running states are reflected by $\widehat{B}_{\rightarrow}$. The CPTs in $\widehat{B}_{\rightarrow}$ capture the causal relations for the system parameter time series.

As defined in the simplified m_DBNs probability representation of Λ, the CPTs included in tuple B are used to describe the temporal evaluation regularity for the target nodes in $\widehat{B}_{\rightarrow}$. To predict the online reliability time series of a component system, the CPTs need to be constructed via statistics on the evolution process of the historical system parameter time series. We use the CPT for the prediction period response time motifs $\widehat{P}_{rt}$ as an example. The same process can be applied to other CPTs.

Returning to the example presented in Fig. 7, to construct the CPT for $\widehat{P}_{rt}$, we need the statistics on the historical time series evaluation regularity from the joint distribution of response time and throughput in the data window time to the response time in prediction period. We also assume that the motifs number k is set to 2. As an example, we choose

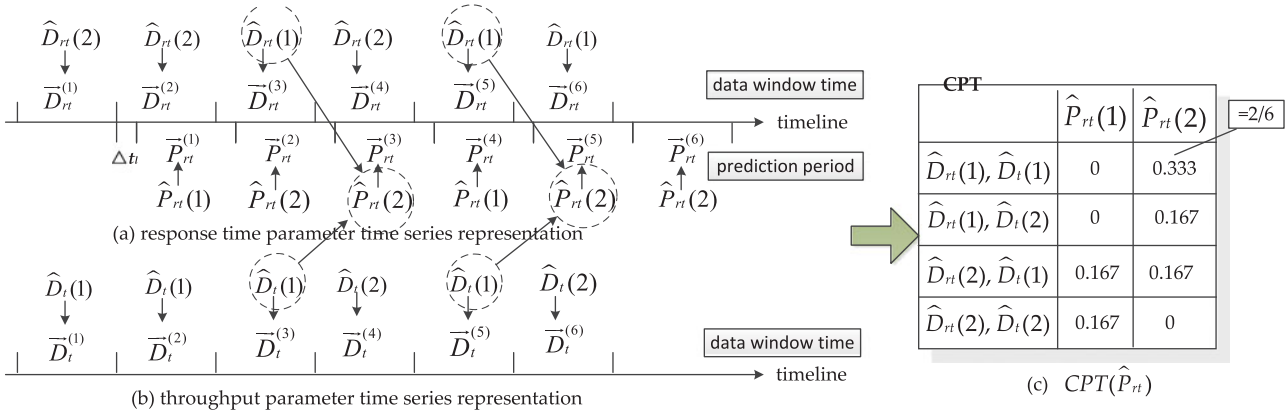


Fig. 9. Construction process of CPT.

six groups of continuously system parameter time series to demonstrate the construction process of $CPT(\widehat{P}_{rt})$.

The time series representations of the historical system parameter time series of $\overrightarrow{D}_{rt}$, $\overrightarrow{D}_t$, and $\overrightarrow{P}_{rt}$ are shown in Figs. 9a and 9b, respectively. Each group of the historical system parameter time series has a series number with the range of $[1, 6]$, represents a 2-TBN transformation of the system parameter time series. For example, the 3th group time series represents that, with the time series of $\overrightarrow{D}_{rt}^{(3)}$ and $\overrightarrow{D}_t^{(3)}$ as the data window time system parameter, the response time parameter time series would be transformed as $\overrightarrow{P}_{rt}^{(3)}$ during the prediction period.

As for the CPT, each row in $CPT(\widehat{P}_{rt})$ corresponds to one possible combination of values (represented by motifs) taken by its dependent nodes, i.e., $\widehat{D}_{rt}$ and $\widehat{D}_t$. Each column denotes one of the k motifs of $\widehat{P}_{rt}$. We analyze the historical parameters' chains represented and gather the necessary statistics to fill each cell of the CPT. Letting g represent the total group number of the time series, we identify a time series serial number set I, for each $i \in I$, $i \leq g$, if each ith group of system parameter time series can be represented by the motifs as

$$\begin{cases} \overrightarrow{D}_{rt}^{(i)} \leftarrow \hat{D}_{rt}(\mu) \\ \overrightarrow{D}_{t}^{(i)} \leftarrow \hat{D}_{t}(\nu) \\ \overrightarrow{P}_{rt}^{(i)} \leftarrow \hat{P}_{rt}(\psi). \end{cases} \tag{8}$$

We will fill the cell in $CPT(\widehat{P}_{rt})$ at the intersection of row $\widehat{D}_{rt}(\mu), \widehat{D}_t(\nu)$ and column $\widehat{P}_{rt}(\psi)$ by the value of $|I|/g$.

As demonstrated in Fig. 9, since the time series serial numbers of 3 and 5 satisfy the dependence rule in the CPT with the row of $\widehat{D}_{rt}(1), \widehat{D}_t(1)$ and the column of $\widehat{P}_{rt}(2)$, then $I = \{3, 5\}$, and $|I| = 2$. The cell at the intersection of row $\widehat{D}_{rt}(1), \widehat{D}_t(1)$ and column $\widehat{P}_{rt}(2)$ will be $|I|/g = 2/6$.

In sum, the simplified probability representation $\Lambda = \{A, B\}$ for a component system can be constructed via the *Motifs Discovery* and *Conditional Probability Table Construction* steps. The construction process is based on the long-term observed continuously historical system parameters of response time, throughput, and reliability. Λ is used with the real-time observed up-to-date response time and throughput time series for online reliability time series prediction. The algorithm for constructing Λ is named ROP-SPRC and is summarized in Algorithm 1.

Algorithm 1. ROP-SPRC

Input: The training set $\{\overrightarrow{D}_{rt}^{(i)}, \overrightarrow{D}_t^{(i)}, \overrightarrow{P}_{rt}^{(i)}, \overrightarrow{P}_t^{(i)}, \overrightarrow{P}_r^{(i)}\}$, for $i \in [1, g]$, and the motifs number k
Output: $\Lambda = \{A, B\}$
1: Find the motifs $\widehat{D}_{rt}$ for $\overrightarrow{D}_{rt} = \{\overrightarrow{D}_{rt}^{(i)}\}_{i=1}^{g}$ by the k-means clustering;
2: Find the motifs for $\overrightarrow{D}_t$, $\overrightarrow{P}_{rt}$, $\overrightarrow{P}_t$, $\overrightarrow{P}_r$;
3: $A \leftarrow \{\widehat{D}_{rt}, \widehat{D}_t, \widehat{P}_{rt}, \widehat{P}_t, \widehat{P}_r\}$;
4: Represent each $\overrightarrow{P}_{rt}^{(i)}$ by solving (6);
5: Represent other system parameter time series by similar method;
6: Initialize the CPT table structure for $CPT(\widehat{P}_{rt})$;
7: **for** Each cell in $CPT(\widehat{P}_{rt})$ **do**
8: **for all** $i \in [1, g]$ **do**
9: $I \leftarrow i$, when the dependence rule for the cell satisfies (8);
10: Fill the cell by $|I|/g$;
11: **end for**
12: **end for**
13: Construct $CPT(\widehat{P}_t), CPT(\widehat{P}_r)$ by similar method;
14: $B \leftarrow \{CPT(\widehat{P}_{rt}), CPT(\widehat{P}_t), CPT(\widehat{P}_r)\}$;
15: **return** $\{A, B\}$;

4.2.5 Reliability Time Series Prediction

The online reliability time series for a component system is predicted based on the one-step unrolling of the corresponding Bayesian Networks. To unroll the Bayesian Networks, the real-time observed up-to-date system parameters time series $\overrightarrow{O}_{rt}$ and $\overrightarrow{O}_t$ are labeled by the nearest motifs identified for data window time, i.e., $\widehat{D}_{rt}$ and $\widehat{D}_t$, respectively. The results are substituted into the transformation model $\hat{B}_{\rightarrow}$. According to the maximal conditional probability distribution, the CPTs in Λ are used finally to find the most possible online reliability time series as the prediction result for the component system. In particular, the prediction is carried out through the following three steps:

1) $\overrightarrow{O}_{rt}$ and $\overrightarrow{O}_t$ are labeled by their nearest motifs we had previously discovered for response time and throughput for data window time. We assume that they result in $\hat{D}_{rt}(\alpha)$ and $\hat{D}_t(\beta)$, respectively.

2) $\hat{D}_{rt}(\alpha), \hat{D}_t(\beta)$ will be the conditional item for $CPT(\widehat{P}_{rt})$ and $CPT(\widehat{P}_t)$. Hence, the prediction results for the parameters of response time and throughput will be the motifs holding the maximal probability by the conditional item. We assume them to result in $\widehat{P}_{rt}(\theta)$ and $\widehat{P}_t(\vartheta)$, respectively.

3) Finally, $\widehat{P}_{rt}(\theta), \widehat{P}_t(\vartheta)$ will be the conditional item of $CPT(\widehat{P}_r)$, and the prediction result for online reliability time series of $\overrightarrow{p}_r$ will be

$$\overrightarrow{p}_r = \widehat{P}_r(\xi), \tag{9}$$

where $\widehat{P}_r(\xi)$ is the motif of the reliability time series identified for prediction period, which holds a maximal probability by the $CPT(\widehat{P}_r)$'s conditional item with the property combination of $\widehat{P}_{rt}(\theta), \widehat{P}_t(\vartheta)$, i.e.,

$$\widehat{P}_r(\xi) = \underset{\widehat{P}_r(w),\ w\in[1,k]}{\arg\max} \{P[\widehat{P}_r(w)|\widehat{P}_{rt}(\theta), \widehat{P}_t(\vartheta)]\}. \tag{10}$$

During the above steps, while substituting the conditional items, the maximal probability for the objective motifs in the corresponding CPTs may not turn out to be unique. An example can be found in Fig. 9c, at row $\widehat{D}_{rt}(2), \widehat{D}_t(1)$. All transformation probabilities are equal to 0.167 with the conditional item. This poses a dilemma during the prediction of the objective motif. We can always avoid this problem by increasing the size of model training set. But, when conflict arises, we will select the motif having the least label number as the prediction result in order to side step this kind of conflict completely.

Moreover, the system may come across abnormal running states due to the conditions of service attack, hardware failures, etc. Our online prediction approach should be able to deal with such conditions by adopting specific mechanisms, because they may lack samples reflecting system's outlier state in the training set which is used to train model Λ.

To help our approach predict outlier states, the following two techniques are added:

- Specific motifs (i.e., worst-case [68] motifs) are added into the tuple A of Λ for $\widehat{D}_{rt}$ and $\widehat{D}_t$. During Online Reliability Time Series Prediction, when the real-time system parameters' nearest motifs match the worst-case, the prediction result will be a fixed vector with the least reliability values for each time point, e.g., $\overrightarrow{p}_r = (0.14, 0.14, \ldots, 0.14)$;
- Outlier detection techniques are also applicable and can help in finding abnormal parameter time series (e.g., we can use the chi-square statistic to capture the multivariate outlier [69]). Let $\overline{D}_{rt}(j)$ be the jth time point's average value for all the time series of response time parameter in data window time. According to (2) and (3), we have

$$\overline{D}_{rt}(j) = \frac{1}{g}\sum\nolimits_{i=1}^{g} [\overrightarrow{D}_{rt}^{(i)}(j)],$$

where $\overrightarrow{D}_{rt}^{(i)}(j)$ represents the jth time point for the ith time series of the data window time response time parameter, and g represents the total number of time series for $\overrightarrow{D}_{rt}$ in the training set.

TABLE 2
Motifs for Reliability Parameter Time Series in Prediction Period

Labels	Values
$\widehat{P}_r(1)$	(0.82, 0.67, 0.82, 0.55, 0.67, 0.82, 1, 0.82, 0.82, 1)
$\widehat{P}_r(2)$	(0.82, 0.82, 1, 0.82, 0.67, 0.55, 1, 0.45, 0.82, 0.82)

We define $\overrightarrow{O}_{rt}(j)$ as the value for the jth time point of the observed real-time response time parameter time series, so the chi-square statistic χ^2 for real-time response time parameter is defined as

$$\chi^2_{rt} = \sum\nolimits_{j=1}^{n} \frac{[\overrightarrow{O}_{rt}(j) - \overline{D}_{rt}(j)]^2}{\overline{D}_{rt}(j)},$$

where χ^2_{rt} represents the chi-square statistic for real-time response time parameter time series, n represents the number of time points within a time series. Similarly, we can obtain the chi-square statistic χ^2_t for real-time throughput parameter time series.

During the process of online reliability time series prediction, we will first calculate the chi-square statistic. If χ^2_{rt} or χ^2_t is bigger than a threshold value δ, then the system running state will be regarded as anomaly. Consequently, the reliability prediction result will hold least reliability values for each time point, e.g., $\overrightarrow{p}_r = (0.14, 0.14, \ldots, 0.14)$. Otherwise, the reliability prediction will be regularly handled by m_DBNs model.

5 Case Study

In this section, we present a case study to illustrate how the proposed ROP can be used to make online reliability time series prediction. We collect historical system parameters of response time, throughput, and reliability by invoking the Microsoft Bing search Web service for 24 hours. To calculate the reliability of each time point in the time series, the time interval needed for each client side service invocation is 200 ms. Every 10 continuous returned observations of a QoS parameter are calculated as a time point.

We assume that the leading time of when users will request for a search service is $\Delta t_l = 4$ s with the average operating time being 20 s. Consequently, the length of each time series is set as $\Delta t_p = 20$ s, while there are $10\{= 20\,\mathrm{s}/(200\ \mathrm{ms} \times 10)\}$ time points in each time series.

5.1 Model Training

To train the m_DBN model probability representation Λ, we divide the above three groups of parameters into $4,320(= 24 \times 60 \times 3)$ continuous time series for data window time and prediction period, respectively, by Data Preparation (see Section 4.2.1). More details concerning data processing will be presented in Section 6.1.

We train the m_DBNs model probability representation by the preprocessed historical system parameters using the process described in Algorithm 1. During the process of Motifs Discovery (see Section 4.2.2), when the motifs number k is set to 2, the motifs for the reliability time series in prediction period (i.e., $\widehat{P}_r$) show significant differences. As can be seen from Table 2, for $\widehat{P}_r(1)$, the reliability values of

TABLE 3
Motifs for the Parameters Time Series within the Data Window Time

Labels	Values
$\widehat{D}_{rt}(1)$	(618.7, 714.1, 711.6, 724.6, 714.2, 711.1, 716.9, 875.6, 606.8, 528.9)
$\widehat{D}_{rt}(2)$	(702.5, 532.4, 474.1, 812, 877.1, 1059.8, 795.3, 793.7, 679.2, 518.7)
$\widehat{D}_{t}(1)$	(18,078.9, 24,419, 25,605.1, 15,820.2, 16,756.5, 15,625.8, 14,934.3, 16,255.3, 19,847.7, 22,343.7)
$\widehat{D}_{t}(2)$	(17,715, 18,679.4, 18,501.8, 18,727.2, 18,579.9, 18,246.9, 16,842.1, 20,251.3, 18,267.9, 20,497.6)

the front half part time points are lower than the latter part. For $\widehat{P}_r(2)$, it is just the opposite. The Web services providing the above different prediction results will easily found to be suitable for different users usage habit, so we set the number of motifs as $k = 2$ in our case study.

Note that k in ROP is used to specify the number of motifs in the k-means clustering algorithm. The value of k impacts the performance and scalability for the application of ROP. To scale to larger datasets and guarantee the performance, there have been studies on how to determine the k value. For example, we can first select a set of data points with a reasonable size and provide an approximate range of k values according to the application requirement. We then determine the best value of k by comparing the performance of the results obtained from different k [70].

As can be seen from Table 3, we obtain the motifs for each parameter time series of m_DBNs for the data window time (i.e., $\widehat{D}_{rt}$ and $\widehat{D}_t$) by the process of Motifs Discovery.

Each group of the 4,320 time series for data window time is represented by its nearest motifs label and the same process is conducted for the parameters time series in prediction period. As illustrated in Fig. 10, each cell in the CPT for $CPT(\widehat{P}_{rt})$ is calculated by the statistics on the conditional probability from the historical system parameter. Each column in $CPT(\widehat{P}_{rt})$ is a motif in $\widehat{P}_{rt}$. For the probability of $\widehat{P}_{rt}(1)$ or $\widehat{P}_{rt}(2)$, the conditional item for the conditional probability is the joint distribution of the data window time motifs, one from response time parameter and the other from throughput parameter.

Specifically, since the representation satisfying

$$[\widehat{D}_{rt}(1), \widehat{D}_t(1)] \rightarrow \widehat{P}_{rt}(1)$$

appears 456 times in the evolution of historic parameters, i.e., $|I| = 456$, and $g = 4{,}320$, the value of the cell at the intersection of the row $\widehat{D}_{rt}(1), \widehat{D}_t(1)$ and column $\widehat{P}_{rt}(1)$ is: $0.1056 (= 456/4{,}320)$. The other CPTs are computed in the same way. The resulting $CPT(\widehat{P}_{rt})$, $CPT(\widehat{P}_t)$, and $CPT(\widehat{P}_r)$ are presented in Fig. 10. The CPTs as well as the motifs that consist of the trained m_DBNs model probability representation Λ. These will be used for online reliability prediction.

5.2 Online Prediction

The goal of online prediction is to predict the online time series for system reliability by the trained m_DBNs model and the observed real-time up-to-date system parameters time series. In our case study, the up-to-date time series for system parameters of response time $\overrightarrow{O}_{rt}$ and throughput $\overrightarrow{O}_t$ are first collected and then preprocessed. The results are shown in Table 4.

By (4), the distance from the up-to-date response time and throughput parameters time series to their motifs are:

- $dist[\overrightarrow{O}_{rt}, \widehat{D}_{rt}(1)] = 870.01$,
- $dist[\overrightarrow{O}_{rt}, \widehat{D}_{rt}(2)] = 844.05$,
- $dist[\overrightarrow{O}_{t}, \widehat{D}_{t}(1)] = 36{,}920.50$,
- $dist[\overrightarrow{O}_{t}, \widehat{D}_{t}(2)] = 31{,}435.46$.

Hence, $\overrightarrow{O}_{rt}$ and $\overrightarrow{O}_t$ are represented by their nearest motifs as $\widehat{D}_{rt}(2)$ and $\widehat{D}_t(2)$, separately. The combination of $\widehat{D}_{rt}(2), \widehat{D}_t(2)$ are the conditional items substituted into

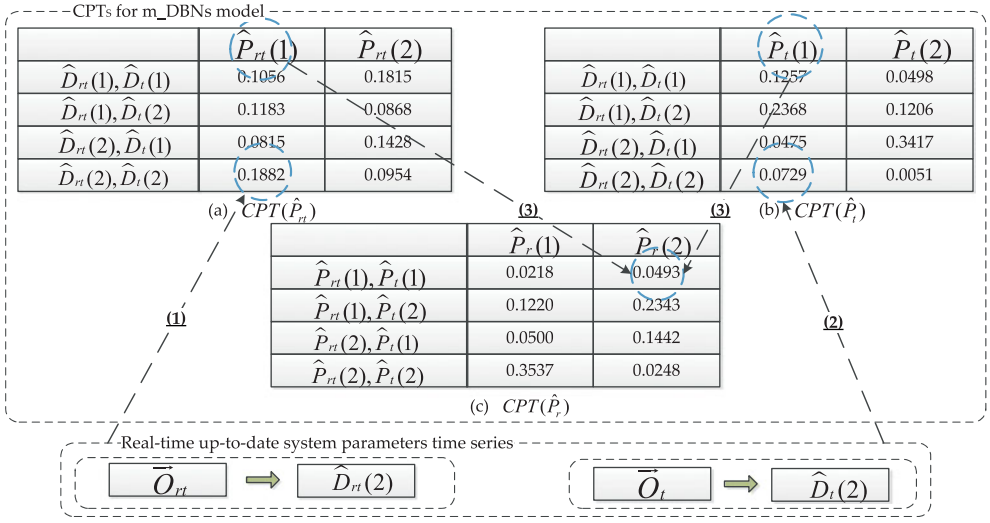


Fig. 10. A concrete example for ROP prediction.

TABLE 4
The Real-Time Up-to-Date System Parameters Time Series

Parameters	Parameter Values
$\overrightarrow{O}_{rt}$	(692, 581, 401, 806, 1,013, 1,013, 418, 1,142, 943, 951)
$\overrightarrow{O}_{t}$	(33,109, 10,694, 9,989, 7,715, 30,484, 10,717, 31,443, 22,151, 10,603, 16,514)

$CPT(\widehat{P}_{rt})$ and $CPT(\widehat{P}_t)$, separately, to obtain the maximal probability. As shown in Fig. 10, the biggest values in rows $\widehat{D}_{rt}(2), \widehat{D}_t(2)$ in $CPT(\widehat{P}_{rt})$ and $CPT(\widehat{P}_t)$ are 0.1882 and 0.0779, respectively. Therefore, we obtain the predicted motifs for parameter response time and throughput through steps (1) and (2). Finally, the online reliability time series is predicted as $\widehat{P}_r(2)$ through step (3). From Table 2, we can obtain immediately the predicted online reliability time series values as $\overrightarrow{p}_r = (0.82, 0.82, 1, 0.82, 0.67, 0.55, 1, 0.45, 0.82, 0.82)$. The reliability values from time points 1-4 are 0.82 or 1. According to (1), the 0.82 represents one failure (or performance anomaly) returned during 10 times of invocations, and the reliability value of 1 represents no failure. The values for time points 5, 6 and 8 are $0.67, 0.55$ and 0.45, respectively. They indicate that there are 2, 3, and 4 times of failures (or performance anomalies) returned during the 10 times of invocations, respectively. This prediction result suggests that, when a user's usage habit is only one time search and s/he seldom prefers subsequent operations (e.g., change keywords for re-search or move over the next page's hyperlink, etc.), then the search system with the predicted reliability values in front of the time points is suitable for this user. Otherwise, if the user prefers operating frequently, then the *Service Composition Execution Engine* should consider for other candidate search Web services, and select more suitable ones.

6 Experiments

We conducted a set of experiments to assess the effectiveness of the proposed ROP. As stated previously in Section 1, each component in a service-oriented SoS is realized as a Web service. The ROP prediction for Web services can also be applied for the component systems in a service-oriented SoS. We focus on evaluating the proposed ROP approach on general Web services for experimentation purposes. The ROP prediction requires a historical dataset that continuously records system parameters, including response time, throughput, and reliability, to train the m_DBNs model. However, there is no sizable Web service dataset that provides continuous observations on these system parameters in the public domain. As a result, we build our own dataset by invoking a set of real Web services and recording the required system parameters.

6.1 Dataset

To build our dataset, we downloaded the WSDL files of Web services, including: (1) well-known popular Web services, such as *Bing, SalesForce, PayPal, Google Search, and Amazon*; (2) Web services from WebserviceX service repository [3]; and (3) popular Web services published in China: *Weather, QQ Online*, and *DomesticAirline*. In the event, a total of 100 Web services were collected from different industries and regions. All these services provide usable WSDL files and support Java Remote Method Invocation (RMI). We then converted the WSDL files into java classes and generate java test files using Axis2.[4] Finally, service invocation requests for a randomly selected API of each Web service were sent out every 200 ms from our PC client (installed with Windows 7 OS and Intel(R) Core(TM) i7 2600 CPU, 4 GB RAM, Seagate 1 TB HDD). Next, the response times, the sizes of returned data (bit), and the returned types of HTTP message were collected.

As can been seen from the above procedure, the response time parameter is directly determined from the service invocations. Let the data size for the returned message from a remote Web service be *ret_size*, the response time parameter be rt. We represented the throughput as the data size successfully transmitted within a unit time from the Web service, i.e., $\frac{ret\_size}{rt \times 1{,}000}$ (kbps). We set an upper limit for the response time of a service invocation as 1,000 ms. If the response time went beyond the limit, it was considered a *Timeout* error. In the event, we collected the system parameters at two different time spans: 24 hours and 1 month, which resulted in two different datasets: 24 H and 1 M.

We preprocessed the collected system parameters as follows. We defined the time interval for each continuous 10 returned messages as the period for each system parameter evaluation. Since a service request is sent every 200 ms, let $u = \{u_1, u_2, \ldots, u_i, \ldots\}$ represented the time slide for neighboring system parameters, then $len(u_i) = 2$ s. The ith parameter for the continuous historical system parameter was calculated as

$$rt(i) = rt(u_i) = \frac{1}{10}\sum\nolimits_{j=1}^{10} rt(t_j), \tag{11}$$

$$t(i) = t(u_i) = \frac{1}{10}\sum\nolimits_{j=1}^{10} t(t_j), \tag{12}$$

$$r(i) = r(u_i) = e^{-\gamma \times len(u_i)}, \tag{13}$$

where $rt(i)$, $t(i)$, and $r(i)$ represent the ith response time, throughput, and reliability parameter in the continuous historical system parameter, respectively; u_i represents the time period for the evaluation of $rt(i)$, $t(i)$, and $r(i)$; $rt(t_j)$ and $t(t_j)$ denote the returned response time and throughput parameter for the jth invocation during u_i, separately; γ is the failure-rate of the invocations during u_i.

To obtain the time series for data window time and prediction period, we set the length for time period $\Delta t_p = 20$ s, and $\Delta t_l = 4$ s. So the predicted online reliability time series contained 10 time points. We built the historic time series parameters for the data window time and prediction period according to Section 4.2.1. More specifically, the time series for data window time: the preprocessed response time, throughput and reliability system parameters for each Web service were divided into time series as $\overrightarrow{D}_{rt}$, $\overrightarrow{D}_t$, and $\overrightarrow{D}_r$. Finally, we got $4{,}320 (= 24 \times 60 \times 3)$ continuous time series in the 24 H dataset and $129{,}600 (= 30 \times 24 \times 60 \times 3)$ continuous time series in the 1 M dataset for each Web service,

3. http://www.webservicex.net/ws/default.aspx

4. http://axis.apache.org/axis2/java/core/

respectively. To generate the time series for prediction period, we moved right twelve points (i.e., the time span of $\Delta t_d + \Delta t_l$, which is 24 s based on the time points at time Δt_d, and each set of 10 continuous points as a time series. These time series are represented as $\overrightarrow{P}_{rt}$, $\overrightarrow{P}_t$, and $\overrightarrow{P}_r$, respectively.

6.2 Approaches Subjected to Comparison

As noted in Section 2, many prediction approaches rely on Bayesian rules (or Markov models) for primary modeling notation. Moreover, modeling approaches such as Regression, Classification, System Model, and Collaborative Filtering are also widely adopted. These notations have also become the potential online reliability time series prediction approaches. Accordingly, we implemented four different reliability time series online prediction approaches based on existing works and compared the proposed *ROP* with four independent approaches to justify the effectiveness of the proposed approach. Specifically, the four approaches under comparison included

- *Average Value of Historical Reliability (AVHR)*: The reliability prediction result for a Web service based on collaborative filtering approaches (e.g., [10], [12], [19], [20], [21]) is the average value (during the observed time period) from multiple evaluating clients. In order to study the effectiveness if this method is applied for online reliability prediction (i.e., predict the near future reliability), we use the historic average reliability value as the prediction results and name this approach as *AVHR*, i.e., the 10 points of the predicted time series $\overrightarrow{p}_r$ are all $\frac{1}{\eta}\sum_{i=1}^{\eta} r(i)$, where $r(i)$ represents the ith reliability value, and there are all together η reliability values in the historical system parameter.
- *Regression (Reg)*: Regression is a widely used, classical method for prediction. In literature [37], [50], regression models based on curve fitting are used for online failure prediction. Inspired by this observation, a regression method based on mean least square error is proposed for comparison. The least square fitting function is defined as

$$y = \overrightarrow{p}_r = f(\overrightarrow{O}_r) = a_0 + a_1\overrightarrow{O}_r + a_2(\overrightarrow{O}_r)^2,$$

  where $\overrightarrow{O}_r$ represents the real-time observed up-to-date reliability time series, and $\overrightarrow{p}_r$ represents the online reliability time series to be predicted. The historical reliability time series divided for data window time and prediction period will form $\overrightarrow{P}_r^{(i)} = f(\overrightarrow{D}_r^{(i)})$. The coefficients a_0, a_1 and a_2 in the fitting function y are calculated according to the historical reliability time series. Next the fitting function is used to predict the reliability time series of $\overrightarrow{p}_r$.
- *Similarity-based Prediction (SP)*: Instance Models (e.g., [44]) for online failure prediction adopt distance-based similarity to find the most similar historic parameters to the real-time parameters. Hence, when the real-time reliability time series parameter is similar to some historic reliability time series, the most similar historic reliability time series' near future time series represent the prediction result. The *SP* approach has been designed under the above assumption. Let the real-time observed up-to-date reliability time series be $\overrightarrow{O}_r$, $\overrightarrow{P}_r^{(i)}$ will be the prediction result, where $\overrightarrow{D}_r^{(i)}$ is the nearest data window time reliability time series to $\overrightarrow{O}_r$.
- *Bayes' Rules (BR)*: The *BR* approach is designed using the Bayes' Rule for prediction (e.g., [33], [57]). We cluster the preprocessed historical reliability time series and identify the motifs by the proposed motifs discovery approach. We label each historic reliability time series as recommended in (6). We estimate the probability of the causal relations by the motifs in the historical reliability time series evolution from $\widehat{D}_r$ to $\widehat{P}_r$. Let g represent the total number of reliability time series in the historical parameter, and for any integer $w \in [1, g]$, assume the nearest motif for the real-time observed up-to-date reliability time series is $\hat{D}_r(\gamma)$, then the motif of $\hat{P}_r(\xi)$ will be the prediction result, which makes $P[\hat{P}_r(\xi)|\hat{D}_r(\gamma)]$ obtain the maximal probability, i.e,

$$\overrightarrow{p}_r = \widehat{P}_r(\xi) = \underset{\widehat{P}_r(w),\, w\in[1,g]}{\arg\max}\ \{P[\widehat{P}_r(w)|\widehat{D}_r(\gamma)]\}.$$

6.3 Metrics

We employ the widely adopted Mean Absolute Error (*MAE)* and Root Mean Square Error (*RMSE*) to evaluate the prediction accuracy of different prediction approaches.

Let $\overrightarrow{p}_r$ be the prediction result and $\overrightarrow{r}_r$ be the observed real reliability time series during the prediction period. The *MAE* and *RMSE* are then defined as

$$MAE = \frac{\sum_{s=1}^{N}\sum_{i=1}^{10}|\overrightarrow{p}_r^{(i)} - \overrightarrow{r}_r^{(i)}|_s}{10 \times N}, \tag{14}$$

$$RMSE = \sqrt{\frac{\sum_{s=1}^{N}\sum_{i=1}^{10}(\overrightarrow{p}_r^{(i)} - \overrightarrow{r}_r^{(i)})^2{}_{|s}}{10 \times N}}, \tag{15}$$

where $\overrightarrow{p}_r^{(i)}$ and $\overrightarrow{r}_r^{(i)}$ are the values for the ith time point in the time series of $\overrightarrow{p}_r$ and $\overrightarrow{r}_r$, respectively. N is the total times of predictions. According to (14) and (15), smaller values of *MAE* and *RMSE* indicate better prediction accuracy.

6.4 Performance Comparison

We use both the 24 H dataset and 1 M dataset to train the prediction models for each Web service. We set the number of motifs as *k = 20* and *k = 25* in *ROP* and *BR*, respectively, for each Web service, and compare the averaged *MAE* and *RMSE* of different prediction methods for 10, 50, 100, 200, 300 and 400 times of predictions. The experimental results for 24 H and 1 M datasets are shown in Tables 5 and 6, respectively. Because *AVHR*, *Reg* and *SP* methods are independent of the number of motifs, the values are the same in row.

As can be seen from the results, both for *BR* and the proposed *ROP*, the prediction accuracy has increased slightly with the increment of prediction times. The prediction

TABLE 5
Prediction Accuracy Comparison for the 24 H Training Dataset

Metrics	Methods	The motifs number $k = 20$						The motifs number $k = 25$					
		$N = 10$	$N = 50$	$N = 100$	$N = 200$	$N = 300$	$N = 400$	$N = 10$	$N = 50$	$N = 100$	$N = 200$	$N = 300$	$N = 400$
MAE	ROP	**0.033**	**0.031**	**0.028**	**0.027**	**0.026**	**0.027**	**0.026**	**0.024**	**0.022**	**0.020**	**0.020**	**0.018**
	AVHR	0.097	0.087	0.095	0.088	0.085	0.099	0.097	0.087	0.095	0.088	0.085	0.099
	Reg	0.056	0.053	0.049	0.048	0.045	0.045	0.056	0.053	0.049	0.048	0.045	0.045
	SP	0.084	0.072	0.074	0.074	0.071	0.061	0.084	0.072	0.074	0.074	0.071	0.061
	BR	0.069	0.060	0.062	0.064	0.058	0.057	0.062	0.055	0.053	0.051	0.045	0.038
RMSE	ROP	**0.068**	**0.065**	**0.065**	**0.061**	**0.057**	**0.057**	**0.052**	**0.052**	**0.050**	**0.050**	**0.045**	**0.041**
	AVHR	0.201	0.183	0.197	0.187	0.178	0.208	0.201	0.183	0.197	0.187	0.178	0.208
	Reg	0.103	0.097	0.095	0.094	0.092	0.093	0.103	0.097	0.095	0.094	0.092	0.093
	SP	0.170	0.162	0.150	0.182	0.144	0.132	0.170	0.162	0.150	0.182	0.144	0.132
	BR	0.135	0.113	0.122	0.109	0.115	0.110	0.113	0.107	0.104	0.105	0.092	0.083

accuracies of *AVHR*, *Reg* and *SP* change differently as the number of predictions increases. In addition, changes in *ROP* and *BR* are much more gentle, whereas those of *AVHR* and *SP* show obvious fluctuations. This observation demonstrates the robustness of prediction performance of *ROP* and *BR*. This is mainly due to that the dependency between adjacent motifs has certain patterns as the system parameters change, which makes the proposed motifs-based DBNs model more suitable to carry out online reliability time series prediction.

6.5 Motifs Number k

To study the impact of the number of motifs, we vary k from 5 to 30. The experiments are conducted using both the 24 H and 1 M datasets. Each prediction method is executed 50 and 200 times for each Web service. The *MAE* and *RMSE* values were analyzed and compared with the other four prediction methods. Also, in the *BR* method, the number of motifs was found to be the same as with our *ROP* method.

As can seen in Fig. 11, the value of motifs number exhibits a significant impact on both *ROP* and *BR*. The larger value of k results in a smaller *MAE* and *RMSE*. When $k \geq 20$, improvements in prediction accuracy slow down for both the 24 H and 1 M datasets. Since the *AVHR*, *Reg* and *SP* do not exploit motifs, their *MAE* and *RMSE* values remain constant over different k values. When $k \geq 20$, the prediction accuracy of *ROP* outperforms all other four approaches significantly.

In sum, as the number of motifs increases, the prediction accuracy also improves. However, when the number of motifs reaches a certain limit ($k \geq 25$), no further improvement can be achieved. In addition, comparing the results obtained from the 24 H and 1 M dataset, for the same k value, a larger training dataset shows a higher prediction accuracy.

6.6 Training Set Scale

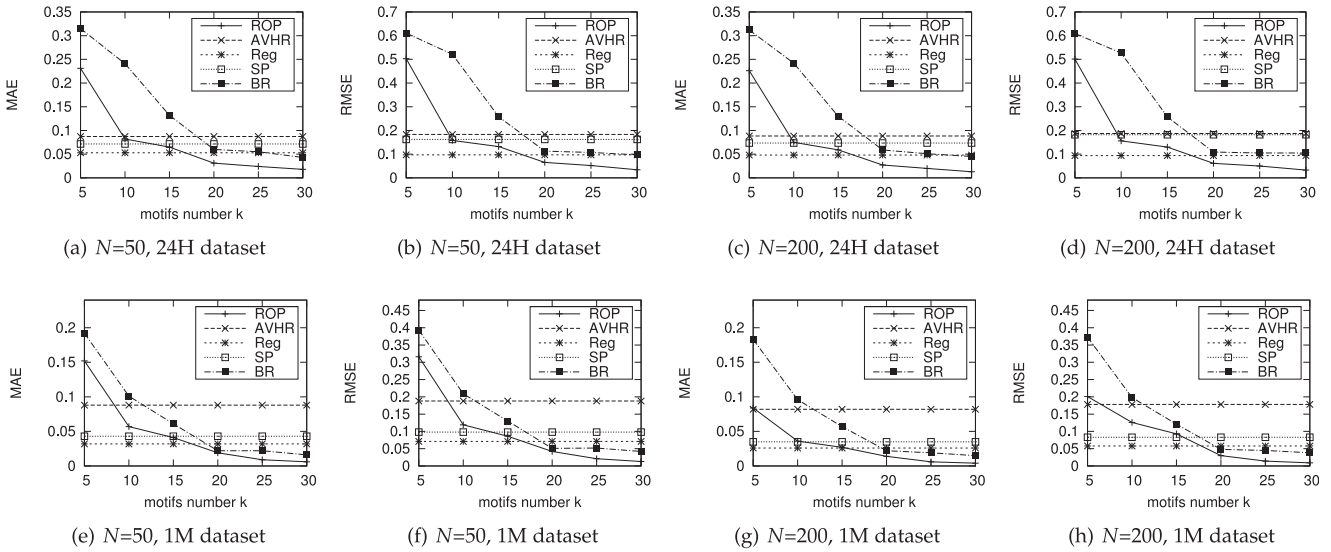
To study the effect of training dataset scale on prediction accuracy, we extracted 1,500, 2,000, 2,500, 3,000, 3,500, and 4,000 time series respectively from each parameter's 4,320 time series in 24 H dataset, and built the prediction model for each method of *ROP*, *AVHR*, *Reg*, *SP* and *BR*, in which the numbers of motifs in *ROP* and *BR* were set as 20 and 25, respectively. For each prediction method, we executed the routine 100 times and 200 times, respectively, for each Web service. The results are compared in Figs. 12a, 12b, 12c, and 12d.

Moreover, we separately selected the datasets nearby 5, 10, 15, 20, 25, and 30 days in the 1 M dataset, and constructed the *m_DBNs* model. The motifs numbers were all set as $k = 25$. We used the above six *m_DBNs* models for *ROP* prediction, and the number of prediction times were all set as $N = 200$. The *MAE* and *RMSE* compared with other four approaches are analyzed in Figs. 12e and 12f.

We also divided the 1 M dataset as day 1-5, day 6-10, day 11-15, ... , day 26-30, and got six groups of datasets, and constructed the *m_DBNs* models, separately. We also set the motifs number as $k = 25$, and the number of prediction times as $N = 200$. The comparison result related to *ROP*

TABLE 6
Prediction Accuracy Comparison for the 1 M Training Dataset

Metrics	Methods	The motifs number $k = 20$						The motifs number $k = 25$					
		$N = 10$	$N = 50$	$N = 100$	$N = 200$	$N = 300$	$N = 400$	$N = 10$	$N = 50$	$N = 100$	$N = 200$	$N = 300$	$N = 400$
MAE	ROP	**0.023**	**0.019**	**0.016**	**0.014**	**0.013**	**0.012**	**0.013**	**0.009**	**0.007**	**0.006**	**0.006**	**0.006**
	AVHR	0.092	0.088	0.081	0.082	0.081	0.080	0.092	0.088	0.081	0.082	0.081	0.080
	Reg	0.038	0.032	0.029	0.026	0.024	0.023	0.038	0.032	0.029	0.026	0.024	0.023
	SP	0.052	0.043	0.039	0.035	0.031	0.031	0.052	0.043	0.039	0.035	0.031	0.031
	BR	0.028	0.022	0.021	0.022	0.017	0.015	0.028	0.022	0.021	0.019	0.017	0.015
RMSE	ROP	**0.049**	**0.042**	**0.037**	**0.030**	**0.029**	**0.025**	**0.028**	**0.021**	**0.017**	**0.014**	**0.013**	**0.013**
	AVHR	0.191	0.188	0.177	0.178	0.176	0.172	0.191	0.188	0.177	0.178	0.176	0.172
	Reg	0.085	0.071	0.063	0.058	0.055	0.054	0.085	0.071	0.063	0.058	0.055	0.054
	SP	0.105	0.098	0.090	0.083	0.077	0.076	0.105	0.098	0.090	0.083	0.077	0.076
	BR	0.064	0.051	0.049	0.048	0.041	0.039	0.065	0.052	0.050	0.045	0.042	0.038



(a) N=50, 24H dataset (b) N=50, 24H dataset (c) N=200, 24H dataset (d) N=200, 24H dataset

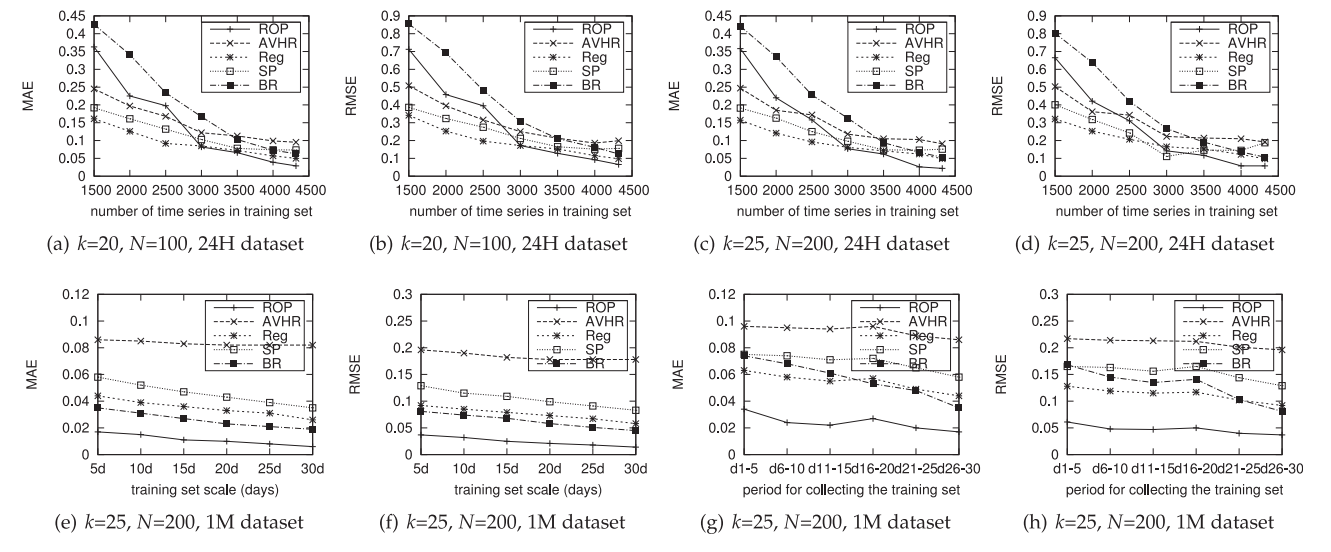
(e) N=50, 1M dataset (f) N=50, 1M dataset (g) N=200, 1M dataset (h) N=200, 1M dataset

Fig. 11. Prediction accuracy as impacted by the motifs number.

prediction and other approaches using the above six datasets, respectively, are analyzed in Figs. 12g and 12h.

The following observations can be made from Fig. 12: (1) When the number of time series is small, the prediction accuracy of *ROP* is smaller than other four methods. However, when the size reaches 3,000, its accuracy will be higher, which indicates that *ROP* heavily relies on the scale of training dataset and needs an adequate scale to ensure the accuracy of the prediction model; (2) When the scale of the training dataset increases, for taking different values of k and N, in addition to the *SP* and *BR*, other prediction methods show a significantly increasing trend in its prediction accuracy rate. There is some fluctuation in the accuracy of *SP* and *BR* with the training set size increasing, which further proves the lack of robustness of the methods; (3) The increasing trend of prediction accuracy slows down when the number of time series in the training dataset reaches 3,500. When we further enlarge the training dataset scale, the prediction accuracy continues to grow, but slowly; and (4) A more nearby dataset will result in higher prediction accuracy. Therefore, a larger and more nearby training dataset is more effective in increasing the prediction accuracy. The running statuses of Web services may be different during various periods, which may impact the seasonal characteristics of system parameters (e.g., system overload during a specific festival). Furthermore, while considering the computational complexity for model training and updating, tradeoff decisions need to be made between accuracy and running time. We can also consider using multiple m_DBNs models trained by the datasets under different time duration. The model with higher prediction accuracy can be selected for the corresponding time duration.

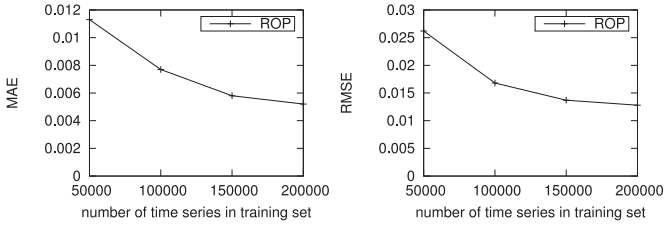
To further indicate the effectiveness of ROP in terms of scalability and performance, we also extended the datasets and generated synthetic datasets for reliability prediction experiments. To better simulate the real-life Web service QoS data, the randomly selected time series data from the 1 M dataset (altogether, we extracted a couple of time series containing the time series for data window time and prediction period) are added into the original 24 H dataset. As a result, the original 24 H dataset was extended to have 50,000, 100,000, 150,000, and 200,000 time series, respectively.



(a) k=20, N=100, 24H dataset (b) k=20, N=100, 24H dataset (c) k=25, N=200, 24H dataset (d) k=25, N=200, 24H dataset

(e) k=25, N=200, 1M dataset (f) k=25, N=200, 1M dataset (g) k=25, N=200, 1M dataset (h) k=25, N=200, 1M dataset

Fig. 12. Impact of training dataset scale.



(a) k=25, N=200, MAE (b) k=25, N=200, RMSE

Fig. 13. Simulation experiments on synthetic datasets.

We separately constructed the m_DBNs models for each of the 100 Web services based on the above four groups of synthetic datasets, for which all the motifs numbers were all set as $k = 25$. The prediction models were used for online reliability time series prediction for the Web services. The number of prediction times for each Web service was set as $N = 200$. The prediction results are compared with the observed real reliability time series data. The MAE and RMSE are shown in Fig. 13.

As can be seen from the results, (1) when the numbers of time series are 50,000 and 100,000, the *MAE* and *RMSE* are close to the results obtained by using the data for 15 days and 30 days in the 1 M dataset as the training sets (cf., Figs. 12e and 12f), which contain 64,800 and 129,600 time series, respectively. This indicates that the synthetic datasets approximate real data well and the prediction accuracy is relative to the scale of training set; (2) The prediction results also indicate high prediction accuracy with the synthetic datasets. The prediction accuracies increase with the increase of training set size. This provides further evidence that a larger training set will result in better prediction performance of ROP. The method of extending the training set by adding randomly selected time series from the dataset that is collected during different time duration is effective.

During the model training process, to ensure the trained model is ready for use, some strategies are also needed to judge whether sufficient data has been included in the training dataset. A more in-depth discussion can be found in [70]. In sum, a common way to judge if the training dataset contains sufficient data is to check whether the prediction accuracy of the trained model meets the application requirement. In general, under the premise of meeting the time efficiency requirement for model training process, a large training set leads to a more accurate model. To reduce the computational cost, random sampling can be used to reduce the scale of training data [70].

6.7 Prediction Term

As a short-term prediction problem, the online reliability time series prediction needs to cover an effective prediction term in the "near future". The prediction term of *ROP* consists of two time periods: the leading time Δt_l and the prediction period Δt_p. Here the length of Δt_l is always determined by the time for constructing a reliable composite system and the time when the user would actually invoke the component system. The length of Δt_p is determined by the service execution time invoked by the user.

To study the impact of varying the length of the prediction term on prediction accuracy, we perform two groups of experiments: (1) fix the value of Δt_p as 20 s and we vary the value of Δt_l from 2 to 6 s, (2) fix the value of Δt_l as 4 s and vary the value of Δt_p from 10 to 50 s. We used both the 24 H and 1 M datasets, and preprocessed the two datasets using the above settings, respectively. We set the motifs number as $k = 25$, the number of predictions N as 200 times. The *MAE* and *RMSE* are analyzed and compared with the other prediction methods as shown in Figs. 14a, 14b, 14c, 14d, 14e, 14f, 14g, and 14h.

Furthermore, to find the inflection point of prediction term length, (1) when $\Delta t_p = 20$ s, we further varied the value of Δt_l from 10, 20, 30, 40, 50, 55, 60 to 65 s; (2) when $\Delta t_l = 4$ s, we varied the value of Δt_p from 70, 80, 90, 100, 110, 115, 120 to 125 s. The *MAE* and *RMSE* for the prediction results from *ROP* under 24 H and 1 M dataset, are analyzed as Figs. 14i, 14j, 14k, and 14l.

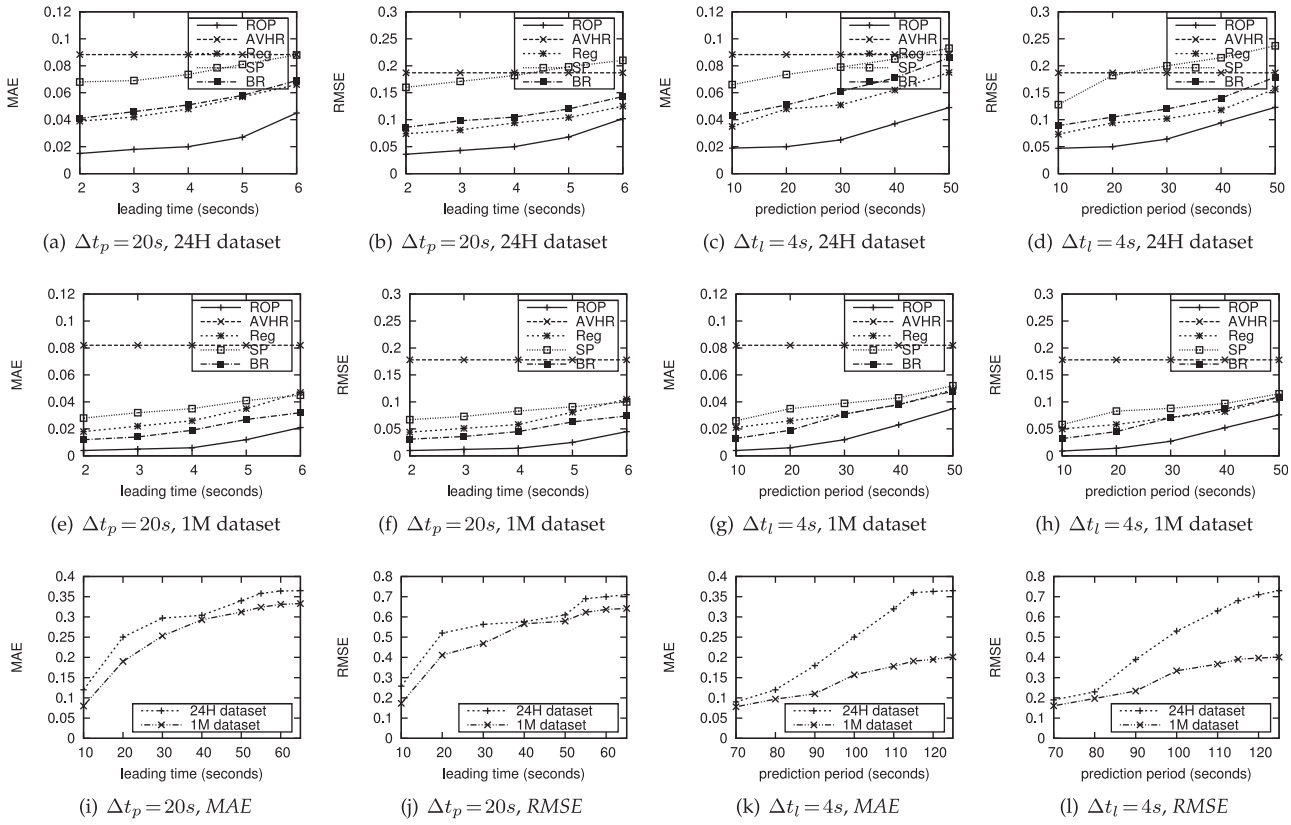
As can be seen from Fig. 14, (1) the length of prediction term exhibits a significant impact on the prediction accuracy of *ROP* and the compared methods (besides *AVHR*). The larger value of prediction term for both Δt_l and Δt_p results in a bigger *MAE* and *RMSE*; (2) Compared with other prediction methods, the *MAE* and *RMSE* values from *ROP* are always the smallest under our prediction term settings; and (3) When $\Delta t_p = 20$ s and $\Delta t_l = 55$ s, a further extension of Δt_l will have little impact on prediction accuracy. The same is found to occur when $\Delta t_l = 4$ s and $\Delta t_p = 115$ s, and Δt_p is extended. The above results indicate that *ROP* fits better for a short-term prediction. It can also provide a higher prediction accuracy than other methods of competition.

6.8 Services Invocation Granularity

While invoking the Web services, some APIs needed input parameters and the Web services performed with different execution granularities in response to different inputs. However, the calculation approach of reliability put forward in (13) is dependent on the upper limit of the response time. It will result in a larger invoking granularity that will lead to lower reliability for a component system. In the above experiments, we invoked the appropriate Web service and collected the QoS parameters by binding randomly selected API, while the input parameter for each invocation was fixed. Hence, the prediction results from the above QoS evaluation methods may be taken to be invalid; invocation granularities from different users are unknown.

To simulate an environment with varying granularity during service invocation, we selected 30 Web services from our previously evaluated Web services collection. For each Web service, we partitioned its APIs with various possible inputs into 20 equivalence classes according to the services execution granularity. In the event, we created 20 threads in our evaluation PC client at the same time. We then set a stochastic granularity invocation request for each thread by selecting an input parameter for the invoked API from a random equivalence. The 20 threads continuously send their same invocation requests at the same time (every 200 ms). The response times, response data sizes and response HTTP types were received and saved separately. The upper limit for response time was set as 1,000 ms. The invocations are conducted continuously for 24 hours.

With such QoS observation approach, our purpose is to provide an integrated evaluation to the Web services, i.e., for an unknown invoking granularity, the response time, throughput, and reliability are approximately equal to the



(a) $\Delta t_p = 20s$, 24H dataset (b) $\Delta t_p = 20s$, 24H dataset (c) $\Delta t_l = 4s$, 24H dataset (d) $\Delta t_l = 4s$, 24H dataset

(e) $\Delta t_p = 20s$, 1M dataset (f) $\Delta t_p = 20s$, 1M dataset (g) $\Delta t_l = 4s$, 1M dataset (h) $\Delta t_l = 4s$, 1M dataset

(i) $\Delta t_p = 20s$, *MAE* (j) $\Delta t_p = 20s$, *RMSE* (k) $\Delta t_l = 4s$, *MAE* (l) $\Delta t_l = 4s$, *RMSE*

Fig. 14. Impact of the length of prediction term.

average invocation granularity. Hence, (11) and (12) can be written as

$$rt(i) = rt(u_i) = \frac{1}{10 \times 20} \sum_{j=1}^{10} \sum_{s=1}^{20} rt(t_j)^s, \qquad (16)$$

$$t(i) = t(u_i) = \frac{1}{10 \times 20} \sum_{j=1}^{10} \sum_{s=1}^{20} t(t_j)^s, \qquad (17)$$

where $rt(t_j)^s$ and $t(t_j)^s$ represent the returned response time and throughput parameter for sth thread's jth invocation during time slice u_i, respectively. Moreover, the failure-rate γ for the reliability in (13) was calculated by the whole 20 threads each with 10 invocations during time period u_i.

The other preprocessing for this dataset are all the same as the approaches in Section 6.1. We will use this new dataset to perform the following experiments.

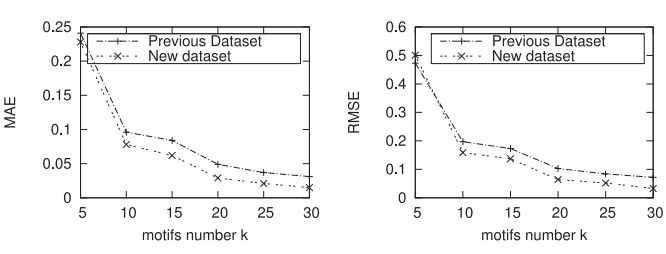
During this group of experiments, we vary the motifs number from 5 to 30, and the *m_DBNs* models are trained by the new dataset for each of our observed Web services, separately. To study the prediction accuracy of *ROP* under the new dataset, we simulated 200 rounds of the unknown granularity user invocations by selecting stochastic granularity invoking request for each Web service. The real-time up-to-date response time and throughput parameters time series are observed from the simulated random granularity's invocations for each time of prediction. The *MAE* and *RMSE* are separately computed between the real-time observed and predicted reliability time series.

We also use the previous 24 H dataset (constructed in Section 6.1 for these 30 Web services) to train the *m_DBNs* models for each Web service, and the models are used to predict the reliability time series by the simulated random invoking granularity's real-time system parameters. The comparisons for different dataset are shown in Fig. 15.

As can be seen from the results, the models trained by the new dataset show higher prediction accuracy; this provides more evidence than the previous ones. This also indicates that: (1) Different granularities of invocations affect the reliability of Web services mainly because of our reliability calculation approaches for a timeout limit, which is always determined by the user's preferences; (2) When $k \geq 20$, the improvement of prediction accuracy also slows down under the two different datasets; and (3) Different service QoS evaluation methods are suitable for different types of Web services APIs. The new QoS evaluation method is more effective if the API performs well at different granularity or returns different response data sizes under different invoking input parameters. More specially, a hybrid evaluation of QoS can be applied for large scale Web services.

6.9 Computational Complexity

The computational complexity of *ROP* can be divided into *model training* (collecting the QoS parameters and train the



(a) MAE (b) RMSE

Fig. 15. Impact of services invocation granularity.

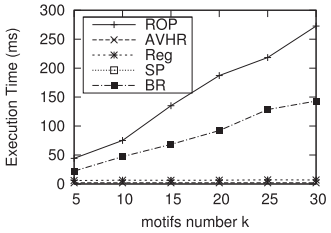


Fig. 16. Execution time comparison.

m_DBNs to construct the prediction model) and *online prediction* (performing predictions based on the model and the real-time up-to-date system parameters). Model training needs an off-line computing environment and it depends on a continued time of the QoS evaluation. Once an m_DBNs model has been trained, the computational complexity of online prediction is mainly affected by the motifs number, and it is independent from the scale of training set.

To study the impact of motifs number to the computational complexity of online prediction, we used the 24 H dataset, and varied the motifs number k from 5 to 30. We separately constructed the m_DBNs model for each Web service. We used MySQL Database to manage the CPTs. Each CPT was saved in a database table. To simplify the database operation, each record in the table was provided with an identification (*ID*) column, with the *ID* representing the CPT conditional item's motifs combination. For example, in the row of $\widehat{D}_{rt}(i), \widehat{D}_t(j)$ (cf. Fig. 10), the *ID* could be calculated as

$$ID = (i-1) \times k + j,$$

where k is the motifs number.

We also constructed the prediction models for the other four prediction approaches. We set the number of predictions for each Web service as $N = 20$, and compared the average execution time for each online prediction.

As can be seen from Fig. 16, the average execution time for each *ROP* prediction increases with the increment of motifs number. As the average execution time for *SP* method ≈ 17 second, for any k, the curve for *SP* is beyond the range of this figure. It is obviously bigger than that found with other methods. The average execution time of *AVHR*, *Reg*, and *BR* is lower than that with the *ROP* method. This indicates that trade-off decisions need to be made between accuracy and running time while selecting different prediction methods.

It is worth noting that, the process of constructing the m_DBNs model is offline. An ROP prediction with a trained m_DBNs model is online, and the computational complexity is under linear time, with the increment of k value. The computational complexity of ROP prediction mainly included computing the similarity between real-time parameters and their motifs, and searching in CPTs. The similarities in computing operations require constant times. Clearly, *ROP* is highly suitable for almost real-time prediction.

6.10 Discussion

In the short-term, *ROP* can predict the online reliability time series for popular Web services such as Bing, SalesForce, PayPal, Google Search and Amazon. Our experimental results have demonstrated the effectiveness of *ROP* in solving the three prediction challenges (see Section 1) for the component systems in a service-oriented SoS.

With respect to prediction challenge 1, the Markov chain rule was adopted to predict the future on the basis of the up-to-date running status of the system. From Figs. 11, 12, 13, and 15, it can be seen that a proper value of motifs number k and training set scale will lead to higher prediction accuracy for *ROP*. When k = 20, 25, 30 and the training set scale is more than 3,500 time series, the prediction accuracy of *ROP* performs better than other comparable approaches. This points to the effectiveness of first-order Markov chain rule in solving the challenge of the component system's irregularly changed QoS parameters.

Regarding prediction challenge 2, we proposed the motifs-based *DBNs* model (named as *m_DBNs*). Figs. 11 and 15 show that, when value of k is further enlarged, the trend of growing prediction accuracy gets arrested. This is because the motifs in *m_DBNs* model basically reflect the patterns of time series, when $k \geq 20$. However, our experimental results have shown that *ROP* is associated with a higher prediction accuracy than the *BR* approach; this further illustrates that the causal relations among the nodes in *m_DBNs* model are correct and effective.

With respect to prediction challenge 3, the QoS parameters of response time, throughput and reliability were evaluated through client side service (or component system) invocations and the corresponding responses. This provided us the estimates of the observed parameters for online reliability time series prediction. We have seen that the use of our QoS evaluation method and different service invocation granularity may lead to higher running complexity and QoS values of different server side jobs. In case of unknown (or unpredictable) size service invocation granularity, the training set with average invocation granularity constructed under stochastic granularity service calls has yielded higher prediction accuracies.

Overall, the results of the experiments have confirmed the merit of the proposed *ROP*. The prediction results show high prediction accuracy and robustness. We conclude that the impacts on the effectiveness of ROP are on the following: (1) the scale of the training dataset, (2) the motifs number k, (3) the length of leading time and the prediction period, and (4) the setting for service (or component system) invocation granularity while collecting the training dataset.

7 Conclusion and Future Work

In this paper, we have presented an online reliability time series prediction approach, referred to as *ROP* for the component systems in a service-oriented SoS. The proposed approach integrates time series motifs into the traditional dynamic Bayesian Networks, yielding an *m_DBNs* model, that is capable of dealing with the three prediction challenges arising from (1) uncertain component systems' runtime environment, (2) the need to predict the time series for near future, and (3) limited observed variables. The Markovian process assumption and the conditional independence assumption are adopted to describe the dynamic evolution for the component systems, and key system parameters, including *response time*, *throughput*, and *reliability*, which are represented as motifs to enable the training

and prediction using the *m_DBNs* model. We have also conducted experiments on real-world Web services to evaluate the effectiveness of the proposed approach. Four other reliability prediction approaches have also been implemented for comparison purposes. The experimental results have demonstrated the high prediction accuracy and the robust prediction performance of *ROP*.

The proposed ROP can provide the client/service couple's online reliability time series prediction. In practice, each client monitors and predicts its own focused SoS components. As an extension, a centralized prediction broker can be used to gather all user/service couples' reliability time series prediction results under a unified clock. At each time point, the prediction results will form a sparse user/service matrix. A collaborative filtering approach can then be adopted to predict the missing values for each time point's sparse matrix. The proposed online reliability time series prediction approach can be an instrument in serving SoS's optimal service selection for composite system construction under a complicated and changing environment.

We have also identified some limitations of the proposed *ROP*, which have pointed to important future research directions. One limitation is that the observed input parameters may change while applying *ROP* in certain other scenarios. The *m_DBNs* model may not work well anymore, and we will need to analyze the causal relations and modify its structure. Besides, although the experimental results have shown high prediction accuracy from *ROP*, it may still be possible to further optimize the structure of *m_DBNs*.

Moreover, once *ROP* is integrated into a service composition execution engine for the purpose of selecting component systems with higher runtime reliabilities, the prediction accuracy and the real time performance of *ROP* prediction should be guaranteed. The prediction accuracy and computational complexity of *ROP* rely on the setting of motifs number k and the trained m_DBNs model based on the dataset. The k value can always be set following an in-depth analysis of the experimental results over different datasets and the requirement of specific application. A larger dataset is a positive factor affecting the prediction accuracy of *ROP*. During the execution life cycles of the component systems, more and more QoS parameters can be collected. The historical system execution parameters with more useful instances will then become available. To integrate these new instances and obtain a more accurate m_DBNs model, *ROP* needs to retrain the model. This can be time consuming. We regard this as another limitation of *ROP*. In the future, incremental learning technologies need to be investigated.

We plan to extend our approach via real applications. More specifically, we propose to focus on

1) designing an SoS oriented online fault removal framework based on the prediction result, and
2) applying this method to various types of service-oriented systems.

We will also develop a tool to achieve proactive service replacement based on *ROP* for various types of service-oriented systems. These extensions should be helpful in reconfirming the benefits of our approach and demonstrate its applicability in service-oriented SoS and other modern software systems.

ACKNOWLEDGMENTS

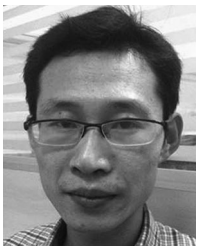
This work was partially supported by NSFC Projects (Nos. 61672152, 61232007, 61532013), Collaborative Innovation Centers of Novel Software Technology and Industrialization and Wireless Communications Technology, and Australian Research Council's *Linkage Projects* funding scheme (No. LP120200305). H. Wang is the corresponding author.

REFERENCES

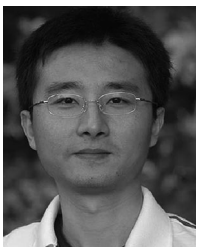
[1] A. Sousa-Poza, S. Kovacic, and C. Keating, "System of systems engineering: An emerging multidiscipline," *Int. J. Syst. Eng.*, vol. 1, no. 1, pp. 1–17, 2008.
[2] R. L. Ackoff, "Towards a system of systems concepts," *Manage. Sci.*, vol. 17, no. 11, pp. 661–671, 1971.
[3] M. Jamshidi, *System of Systems Engineering: Innovations for the Twenty-First Century*. Hoboken, NJ, USA: Wiley, 2011.
[4] FAA, "Next Generation Air Transportation System (NextGen)," 2014. [Online]. Available: http://www.faa.gov/nextgen/
[5] K. J. Rothenhaus, J. B. Michael, and M.-T. Shing, "Architectural patterns and auto-fusion process for automated multisensor fusion in SOA system-of-systems," *IEEE Syst. J.*, vol. 3, no. 3, pp. 304–316, Sep. 2009.
[6] M. B. Blake and M. N. Huhns, "Web-scale workflow: Integrating distributed services," *IEEE Internet Comput.*, vol. 12, no. 1, pp. 55–59, Jan./Feb. 2008.
[7] T. S. Cook, D. Drusinksy, and M.-T. Shing, "Specification, validation and run-time monitoring of SOA based system-of-systems temporal behaviors," in *Proc. IEEE Int. Conf. Syst. Syst. Eng.*, 2007, pp. 1–6.
[8] F. Salfner, M. Lenk, and M. Malek, "A survey of online failure prediction methods," *ACM Comput. Surveys*, vol. 42, no. 3, pp. 10:1–10:42, 2010.
[9] *IEEE Standard Glossary of Software Engineering Terminology*, ANSI/IEEE Std. 729–1991, 1991.
[10] Z. Zheng and M. R. Lyu, "Personalized reliability prediction of web services," *ACM Trans. Softw. Eng. Methodology*, vol. 22, no. 2, 2013, Art. no. 12.
[11] Q. Guan, Z. Zhang, and S. Fu, "Ensemble of Bayesian predictors and decision trees for proactive failure management in cloud computing systems," *J. Commun.*, vol. 7, no. 1, pp. 52–61, 2012.
[12] Z. Zheng and M. R. Lyu, "Collaborative reliability prediction of service-oriented systems," in *Proc. 32nd ACM/IEEE Int. Conf. Softw. Eng.*, 2010, pp. 35–44.
[13] H. E. Mansour and T. Dillon, "Dependability and rollback recovery for composite web services," *IEEE Trans. Serv. Comput.*, vol. 4, no. 4, pp. 328–339, Oct.–Dec. 2011.
[14] Z. Zheng and M. R. Lyu, "Optimal fault tolerance strategy selection for web services," *Int. J. Web Serv. Res.*, vol. 7, no. 4, pp. 21–40, 2010.
[15] L. Wang, H. Wang, Q. Yu, H. Sun, and A. Bouguettaya, "Online reliability time series prediction for service-oriented system of systems," in *Proc. 11th Int. Conf. Serv. Oriented Comput.*, 2013, pp. 421–428.
[16] R. Calinescu, L. Grunske, M. Z. Kwiatkowska, R. Mirandola, and G. Tamburrelli, "Dynamic QoS management and optimization in service-based systems," *IEEE Trans. Softw. Eng.*, vol. 37, no. 3, pp. 387–409, May/Jun. 2011.
[17] V. Cardellini, E. Casalicchio, V. Grassi, S. Iannucci, F. L. Presti, and R. Mirandola, "MOSES: A framework for QoS driven runtime adaptation of service-oriented systems," *IEEE Trans. Softw. Eng.*, vol. 38, no. 5, pp. 1138–1159, Sep./Oct. 2012.
[18] G. Fan, H. Yu, L. Chen, and D. Liu, "Petri net based techniques for constructing reliable service composition," *J. Syst. Softw.*, vol. 86, no. 4, pp. 1089–1106, 2013.
[19] Q. Yu, "Decision tree learning from incomplete QoS to bootstrap service recommendation," in *Proc. 19th IEEE Int. Conf. Web Serv.*, 2012, pp. 194–201.
[20] Q. Yu, Z. Zheng, and H. Wang, "Trace norm regularized matrix factorization for service recommendation," in *Proc. 20th IEEE Int. Conf. Web Serv.*, 2013, pp. 31–41.
[21] Q. Yu, "QoS-aware service selection via collaborative QoS evaluation," *World Wide Web*, vol. 17, no. 1, pp. 33–57, 2012.

[22] Z. Zheng, H. Ma, M. Lyu, and I. King, "Collaborative web service QoS prediction via neighborhood integrated matrix factorization," *IEEE Trans. Serv. Comput.*, vol. 6, no. 3, pp. 289–299, Jul.–Sep. 2013.
[23] M. Silic, G. Delac, and S. Srbljic, "Prediction of atomic web services reliability based on k-means clustering," in *Proc. 9th Joint Meeting Found. Softw. Eng.*, 2013, pp. 70–80.
[24] J. Cardoso, A. Sheth, J. Miller, J. Arnold, and K. Kochut, "Quality of service for workflows and web service processes," *Web Semantics: Sci. Serv. Agents World Wide Web*, vol. 1, no. 3, pp. 281–308, 2004.
[25] A. Amin, L. Grunske, and A. Colman, "An automated approach to forecasting QoS attributes based on linear and non-linear time series modeling," in *Proc. 27th IEEE/ACM Int. Conf. Automated Softw. Eng.*, 2012, pp. 130–139.
[26] A. Amin, A. Colman, and L. Grunske, "An approach to forecasting QoS attributes of web services based on ARIMA and GARCH models," in *Proc. IEEE 19th Int. Conf. Web Serv.*, 2012, pp. 74–81.
[27] A. Amin, L. Grunske, and A. Colman, "An approach to software reliability prediction based on time series modeling," *J. Syst. Softw.*, vol. 86, no. 7, pp. 1923–1932, 2013.
[28] H. Elfawal-Mansour, A. Mansour, and T. Dillon, "Composite web QoS with workflow conditional pathways using bounded sets," *Serv. Oriented Comput. Appl.*, vol. 7, no. 2, pp. 101–116, 2011.
[29] A. Zisman, G. Spanoudakis, J. Dooley, and I. Siveroni, "Proactive and reactive runtime service discovery: A framework and its evaluation," *IEEE Trans. Softw. Eng.*, vol. 39, no. 7, pp. 954–974, Jul. 2013.
[30] P. K. McKinley, S. M. Sadjadi, E. P. Kasten, and B. H. Cheng, "Composing adaptive software," *IEEE Comput.*, vol. 37, no. 7, pp. 56–64, Jul. 2004.
[31] B. H. Cheng, et al., "Software engineering for self-adaptive systems: A research roadmap," in *Software Engineering for Self-Adaptive Systems*. Berlin, Germany: Springer, 2009, pp. 1–26.
[32] R. De Lemos, et al., "Software engineering for self-adaptive systems: A second research roadmap," in *Software Engineering for Self-Adaptive Systems II*. Berlin, Germany: Springer, 2013, pp. 1–32.
[33] A. Csenki, "Bayes predictive analysis of a fundamental software reliability model," *IEEE Trans. Rel.*, vol. 39, no. 2, pp. 177–183, Jun. 1990.
[34] J. D. Pfefferman and B. Cernuschi-Frías, "A nonparametric nonstationary procedure for failure prediction," *IEEE Trans. Rel.*, vol. 51, no. 4, pp. 434–442, Dec. 2002.
[35] Y. Liang, Y. Zhang, M. Jette, A. Sivasubramaniam, and R. Sahoo, "BlueGene/L failure analysis and prediction models," in *Proc. Int. Conf. Depend. Syst. Netw.*, 2006, pp. 425–434.
[36] K. Vaidyanathan and K. S. Trivedi, "A measurement-based model for estimation of resource exhaustion in operational software systems," in *Proc. 10th Int. Symp. Softw. Rel. Eng.*, 1999, pp. 84–93.
[37] A. Andrzejak and L. Silva, "Deterministic models of software aging and optimal rejuvenation schedules," in *Proc. 10th IFIP/IEEE Int. Symp. Integr. Netw. Manage.*, 2007, pp. 159–168.
[38] G. A. Hoffmann, K. S. Trivedi, and M. Malek, "A best practice guide to resource forecasting for computing systems," *IEEE Trans. Rel.*, vol. 56, no. 4, pp. 615–628, Dec. 2007.
[39] G. Hamerly and C. Elkan, "Bayesian approaches to failure prediction for disk drives," in *Proc. 18th Int. Conf. Mach. Learn.*, 2001, pp. 1–9.
[40] J. F. Murray, G. F. Hughes, and K. Kreutz-Delgado, "Machine learning methods for predicting failures in hard drives: A multiple-instance application," *J. Mach. Learning Res.*, vol. 6, no. 5, pp. 783–816, 2005.
[41] D. Turnbull and N. Alldrin, "Failure prediction in hardware systems," University of California, San Diego, CA, Tech. Rep., 2003, Available: http://www.cs.ucsd.edu/~dturnbul/Papers/ServerPrediction.pdf
[42] E. Kiciman and A. Fox, "Detecting application-level failures in component-based internet services," *IEEE Trans. Neural Netw.*, vol. 16, no. 5, pp. 1027–1041, Sep. 2005.
[43] A. Daidone, F. Di Giandomenico, S. Chiaradonna, and A. Bondavalli, "Hidden Markov models as a support for diagnosis: Formalization of the problem and synthesis of the solution," in *Proc. 25th IEEE Symp. Reliable Distrib. Syst.*, 2006, pp. 245–256.
[44] M. Y. Chen, E. Kiciman, E. Fratkin, A. Fox, and E. Brewer, "Pinpoint: Problem determination in large, dynamic internet services," in *Proc. Int. Conf. Depend. Syst. Netw.*, 2002, pp. 595–604.
[45] G. F. Hughes, J. F. Murray, K. Kreutz-Delgado, and C. Elkan, "Improved disk-drive failure warnings," *IEEE Trans. Rel.*, vol. 51, no. 3, pp. 350–357, Sep. 2002.
[46] A. Ward, P. Glynn, and K. Richardson, "Internet service performance failure detection," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 26, no. 3, pp. 38–43, 1998.
[47] Y.-Y. M. Chen, et al., "Path-based failure and evolution management," Ph.D. dissertation, Univ. California, Berkeley, CA, USA, 2004.
[48] J. Korbicz, *Fault Diagnosis: Models, Artificial Intelligence, Applications*. Berlin, Germany: Springer-Verlag, 2004.
[49] A. R. Ward and W. Whitt, "Predicting response times in processor-sharing queues," in *Analysis of Communication Networks: Call Centres, Traffic, and Performance*. Providence, RI, USA: Amer. Math. Society, 2000, pp. 1–29.
[50] F.-T. Cheng, S.-L. Wu, P.-Y. Tsai, Y.-T. Chung, and H.-C. Yang, "Application cluster service scheme for near-zero-downtime services," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2005, pp. 4062–4067.
[51] J. Crowell, M. Shereshevsky, and B. Cukic, "Using fractal analysis to model software aging," Lane Dept. CSEE, West Virginia Univ., Morgantown, WV, USA, pp. 1–12, 2002, Available: http://www3.fi.mdp.edu.ar/fc3/SisDin2009/temasAlumnos/Spennatto-IngSoft/Experimental_Results_from_Multiple_OS_Parameters.pdf
[52] M. Shereshevsky, J. Crowell, B. Cukic, V. Gandikota, and Y. Liu, "Software aging and multifractality of memory resources," in *Proc. Int. Conf. Depend. Syst. Netw.*, 2003, pp. 721–730.
[53] J. Lunze, *Automatisierungstechnik*. Berlin, Germany: Oldenbourg-Verlag, 2003.
[54] K. Hatonen, M. Klemettinen, H. Mannila, P. Ronkainen, and H. Toivonen, "TASA: Telecommunication alarm sequence analyzer or how to enjoy faults in your network," in *Proc. IEEE Netw. Operations Manage. Symp.*, 1996, pp. 520–529.
[55] R. Vilalta, C. V. Apte, J. L. Hellerstein, S. Ma, and S. M. Weiss, "Predictive algorithms in the management of computer systems," *IBM Syst. J.*, vol. 41, no. 3, pp. 461–474, 2002.
[56] D. Levy and R. Chillarege, "Early warning of failures through alarm analysis a case study in telecom voice mail systems," in *Proc. 14th Int. Symp. Softw. Rel. Eng.*, 2003, pp. 271–280.
[57] F. Salfner, et al., "Modeling event-driven time series with generalized hidden semi-Markov models," Dept. Comput. Sci., Humboldt Univ., Tech. Rep. 208, 2006.
[58] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison, *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge, U.K.: Cambridge Univ. Press, 1998.
[59] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. Cambridge, MA, USA: MIT Press, 2009.
[60] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. Cambridge, MA, USA: MIT Press, 2012.
[61] F. Brosch, H. Koziolek, B. Bühnová, and R. Reussner, "Architecture-based reliability prediction with the Palladio component model," *IEEE Trans. Softw. Eng.*, vol. 38, no. 6, pp. 1319–1339, Nov./Dec. 2012.
[62] M. Palviainen, A. Evesti, and E. Ovaska, "The reliability estimation, prediction and measuring of component-based software," *J. Syst. Softw.*, vol. 84, no. 6, pp. 1054–1070, 2011.
[63] M. R. Lyu, *Handbook of Software Reliability Engineering*. New York, NY, USA: McGraw-Hill, 1996.
[64] J. Rao and C. Xu, "Online capacity identification of multitier websites using hardware performance counters," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 3, pp. 426–438, Mar. 2011.
[65] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, "Network motifs: Simple building blocks of complex networks," *Sci. Signaling*, vol. 298, no. 5594, pp. 824–827, 2002.
[66] E. R. Shellman, C. F. Burant, and S. Schnell, "Network motifs provide signatures that characterize metabolism," *Mol. BioSystems*, vol. 9, no. 3, pp. 352–360, 2013.
[67] A. Mueen, E. J. Keogh, Q. Zhu, S. Cash, and M. B. Westover, "Exact discovery of time series motifs," in *Proc. SIAM Int. Conf. Data Mining*, 2009, pp. 473–484.
[68] T. Savor and R. E. Seviora, "An approach to automatic detection of software failures in real-time systems," in *Proc. 3rd IEEE Real-Time Technol. Appl. Symp.*, 1997, pp. 136–146.
[69] N. Ye and Q. Chen, "An anomaly detection technique based on a chi-square statistic for detecting intrusions into information systems," *Quality Rel. Eng. Int.*, vol. 17, no. 2, pp. 105–112, 2001.
[70] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, 3rd ed. Amsterdam, The Netherlands: Elsevier, 2011.

Hongbing Wang received the PhD degree in computer science from Nanjing University, China. He is a full professor in the School of Computer Science and Engineering, Southeast University, China. He served as an visiting scientist with CSIRO ICT Center, Australia, from Apr. 2009 to Mar. 2010. Prior to this, he has visited the Hongkong University and the Waterloo University during 2003-2008. His research interests include service computing, cloud computing, and software engineering. He published more than 50 refereed papers in international journals and conferences, e.g., the *Journal of Web Semantics,* the *Journal of Systems and Software,* the *IEEE Transactions on Parallel and Distributed Systems,* the *IEEE Transactions on Services Computing*, ICSOC, ICWS, SCC, etc. He is a member of the IEEE.



Lei Wang is currently working toward the PhD degree in the School of Computer Science and Engineering, Southeast University, China. He is also a lecturer with Nanjing Forestry University, China. His research interests include service computing, software reliability engineering, and data mining. His publications have appeared in international journals and popular conferences, e.g., the *IEEE Transactions on Parallel and Distributed Systems,* the *IEEE Transactions on Services Computing*, ICSOC, ICWS, SCC, etc.



Qi Yu received the PhD degree in computer science from Virginia Polytechnic Institute and State University (Virginia Tech). He is an associate professor in the College of Computing and Information Sciences, Rochester Institute of Technology. His current research interests lie in the areas of service computing, databases, and data mining. His publications have mainly appeared in well-known journals (e.g., the *VLDB Journal*, the *ACM Transactions on the Web*, the *World Wide Web Journal*, and the *IEEE Transactions on Services Computing*) and conference proceedings (e.g., ICSOC and ICWS). He is a member of the IEEE.

Zibin Zheng received the PhD degree from The Chinese University of Hong Kong, in 2011. He is an associate professor with Sun Yat-sen University, Guangzhou, China. He received the ACM SIGSOFT Distinguished Paper Award at ICSE'10, Best Student Paper Award at ICWS'10, and IBM PhD Fellowship Award. His research interests include services computing, software engineering, and data mining. He is a member of the IEEE.

Athman Bouguettaya received the PhD degree in computer science from the University of Colorado, Boulder, in 1992. He is the head of the School of Information Technologies, University of Sydney, Sydney, Australia. He was a science leader with the CSIRO ICT Centre, Canberra, Australia. His current research interests include the foundations of web service management systems and sensor cloud service. He is a fellow of the IEEE and a distinguished scientist of the ACM.

Michael R. Lyu received the BS degree in electrical engineering from National Taiwan University, Taipei, Taiwan, R.O.C, in 1981, the MS degree in computer engineering from the University of California, Santa Barbara, in 1985, and the PhD degree in computer science from the University of California, Los Angeles, in 1988. He is currently a professor in the Department of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong, China. His research interests include software reliability engineering, distributed systems, fault-tolerant computing, mobile networks, web technologies, multimedia information processing, and E-commerce systems. He was elected as an IEEE fellow (2004), AAAS fellow (2007), and ACM fellow (2015) for his contributions to software reliability engineering and software fault tolerance. He was also named Croucher Senior Research fellow in 2008 and IEEE Reliability Society engineer of the Year in 2010.