M-BSRM: Multivariate BayeSian Runtime QoS Monitoring Using Point Mutual Information

Pengcheng Zhang, Huiying Jin, Hai Dong, and Wei Song

Abstract—Quality of Service (QoS) is well acknowledged as a decisive means for ascertaining the performance of third-party Web services. QoS has high uncertainty in complex and dynamic network environments. QoS monitoring is considered as one of the most effective techniques to detect QoS violations at runtime. However, existing QoS monitoring approaches only consider single QoS attribute and do not provide a promising solution for comprehensively monitoring multivariate QoS attributes. To overcome this problem, a novel QoS monitoring approach, named M-BSRM (Multivariate BayeSian Runtime Monitoring), is proposed. First, M-BSRM adopts the point mutual information theory to initialize the weights of different environmental impact factors and solves the problem of uneven distribution between classes brought by traditional algorithms. Second, each single QoS attribute is integrated with user preference using the information fusion theory. Finally, a Bayesian classifier is used to comprehensively evaluate multivariate QoS attributes at runtime. The experimental results on both the real-world and simulated data sets show that M-BSRM is more effective, practical and efficient than the other approaches.

Index Terms—Quality of Service, Runtime monitoring, Point mutual information, Information fusion, Bayesian classifier.

1 Introduction

With the wide application of Service Oriented Architecture (SOA), Web services become very popular in many areas of our life, including business, industry, medicine, and entertainment, etc. [1], [2]. The number of Web services deployed on the Internet is rapidly increasing. This causes a phenomenon that different service providers may provide services with similar functions. How to select a service that satisfies a user's need among many similar services has become the focus of attention in recent years [3]. To this end, the concept of QoS (Quality of Service) has been introduced. QoS represents many non-functional attributes of services and mainly includes *response time*, *throughput*, *reliability*, *availability* and so on [4].

Service providers usually provide QoS values of their offered Web services. However, the QoS values might be affected by external factors such as deployment time, geographical locations of servers and clients [5]. The fluctuations of QoS values may lead to potential deployment failures [6]. Consequently, it is extremely important to effectively monitor QoS values and use them as basic references for Web service deployment in order to precisely understand the present state of QoS and avoid potential failures.

Each QoS attribute corresponds to an actual value (observed at runtime) and a required value (pre-defined in Service Level Agreements (SLAs)). The main objective of the QoS monitoring is to prevent the actual value from violating the required value [7]. Therefore, the problem how to effectively monitor QoS attributes is transformed into the problem how to effectively judge whether or not the monitored QoS values will meet the pre-defined QoS requirements using collected QoS data. In fact, the vast majority of QoS requirements can be described by probabilistic quality attributes [8]. For example, "the probability that a service is available per 24 hours should be greater than 90%" or "the probability that a service is offline per 24 hours should be less than 2%" [8]. In this way, we transform the problem of QoS monitoring into probabilistic analysis and calculation of collected QoS data, which is recently termed as probability monitors [9].

Some researchers directly use state-of-the-art probability monitoring approaches for service monitoring [9], [10], [11]. However, the main difference between QoS monitoring and traditional software monitoring is that QoS values are more sensitive and easily impacted by various environmental impact factors, such as locations of services and users. Ignoring these factors may lead to wrong or inaccurate monitoring results [7]. Consequently, several recently proposed QoS monitoring approaches [7], [12] take into account the environmental impact factors during the monitoring process, which makes them more practical.

However, there are still two main problems with these approaches. First, the state-of-the-art approaches can only monitor single QoS attribute values. Because of the complexity of the services and the fluctuations of the QoS values in the dynamic environment, the information of the service state reflected by a single QoS attribute contains some extent of uncertainty and misinformation, which inevitably leads to low accurate QoS monitoring decision-making judgment or even misjudgment. In fact, in dynamic Internet environments, service providers and service consumers always have multiple QoS requirements and have multivariate QoS attributes. These multivariate QoS attributes may have un-

- *P. Zhang and H. Jin are with the College of Computer and Information, Hohai University, Nanjing, 211100, China*
  E-mail: pchzhang@hhu.edu.cn; 367046895@qq.com
- *H. Dong is with the School of Science, Royal Melbourne Institute of Technology, Melbourne, VIC 3001, Australia*
  Email: hai.dong@rmit.edu.au
- *W. Song is with the School of Computer Science and Engineering, Nanjing University of Science and Technology, China*
  E-mail: wsong@njust.edu.cn

Manuscript received XXX XX, XXXX; revised XXX XX, XXXX.

certain relations among each other [13], [14]. For example, the relationship between *latency* and *throughput* is uncertain. Generally speaking, *throughput* refers to how much information can be delivered over a network within a specific period of time. *Latency* describes how quick information can be delivered over a network. Higher throughput does not necessarily lead to lower latency, and vice versa, according to Little's Law [1]. Existing monitoring approaches based on single QoS attributes cannot give comprehensive results on multivariate QoS attributes due to a lack of means for discovering the inter-QoS relations. Considering the uncertain relationship between *latency* and *throughput*, when a single QoS attribute based monitoring approach is applied on *latency* and *throughput* separately, the monitoring results might be *rejection* and *acceptance* respectively. However, the overall evaluation for the comprehensive QoS set (both *latency* and *throughput*) is still unclear.

Second, current QoS approaches use the TF-IDF (term frequency-inverse document frequency) algorithm to initialize the weights of various environmental impact factors that impose negative effects on QoS values. It may lead to the uneven distribution between different monitoring categories (*rejection* or *acceptance*) [7].

To address the two aforementioned problems, we propose a novel multivariate QoS monitoring approach termed as M-BSRM (Multivariable BayeSian Runtime Monitoring) using point mutual information (PMI) theory. Compared with the existing approaches, this approach makes the following novel contributions:

- To address the uneven distribution problem between different monitoring categories and improve the accuracy of initial weights, PMI is used to initialize the weights of environmental impact factors, which can provide more precise weights.
- To comprehensively evaluate multivariate QoS attributes, each single QoS attribute is integrated with user preference using *information fusion theory* [15]. Next, a *Bayesian classifier* is used to comprehensively monitor multivariate QoS attributes.
- Furthermore, since the weighs of environmental impact factors may be expired in dynamic environments, this may lead to imprecise monitoring results. Consequently, *sliding window mechanism* and *information gain theory* are combined to update the weights at runtime, which can provide more precise and timely monitoring results.
- Finally, we have performed a set of experiments using a simulated dataset with a pre-defined error probability and a real-world public data set. The experimental results show that M-BSRM outperforms the other approaches.

The paper is structured as follows: Section 2 reviews existing QoS monitoring approaches and analyses their limitations. Section 3 provides the basic concepts used in the paper. The detailed description of the M-BSRM approach is provided in Section 4. We validate M-BSRM using two datasets in Section 5. Section 6 concludes the paper and plans future work.

1. https://blog.flux7.com/blogs/benchmarks/littles-law

2 Related work

2.1 QoS Monitoring Approaches Beyond Probability Monitors

Zeng et al. [16] propose a monitoring system driven by a QoS observation model. The model defines enterprise-level metrics and related evaluation formulas, and integrates them into the SOA architecture. The proposed monitoring system detects and routes service operation events systematically. The system does not take into account user requirements and is not validated in large-scale service environments. Radovanovic et al. [17] deploy a monitoring system in a cloud environment through the TR-069 remote management protocol. They develop a cloud access interface which provides mobile applications with enough information to visualize QoS parameters. In general, the monitoring device is feasible in most cases, but the entire framework lacks a security mechanism. Michlmayr et al. [18] propose a QoS monitoring framework from both client and server sides, which can rely on handling events and inform interested users about the current QoS. Coppolino et al. [19] propose two EC-funded projects (SRT-15 and SocIoS), where the SRT-15 framework is used for monitoring and the SocIoS framework is used for service quality assessment. Raimond et al. [17] propose a non-intrusive and online monitoring method with respect to service level agreements (SLAs). This method finds the situations of SLA violations by inferring the type of message exchanged and the time stamp.

2.2 Probabilistic QoS Monitoring Approaches

According to the underlying theories, approaches in this field can be classified into the classes of simple probability calculation [20], Bayesian theory [7], [11] and classical hypothesis testing [9], [10], [21], [22].

Chan et al. [20] use the PCTL [23] language to define non-functional attributes of services. The ratio of the number of samples satisfying the probabilistic quality standard to the total number of monitored samples is compared with a pre-defined standard. If the standard is satisfied, the service is considered to be operating normally; otherwise the service is considered to be violated. This approach does not include statistical analysis and is easily affected by sample distributions, which may make the calculated probability inconsistent with the real probability.

Grunske et al. [9] propose a novel monitoring approach using hypothesis testing. It extends existing statistical model checking technologies and defines a probabilistic logic called CSL^{Mon} (Continuous Stochastic Logic) for monitoring. Based on the performance of SPRT [24] (Sequential Probabilistic Ratio Test) at significance levels, the evaluation demonstrates the effectiveness of the CSL^{Mon} formulae.

Zhang et al. [10] define a novel approach using Probabilistic Timed Property Sequence Chart (PTPSC), a probabilistic and timed extension of the existing specification Property Sequence Chart [25]. They design an SDT (Syntax-Directed Translator) to automatically generate probability monitors, which include two parts. The first part is the operation of analysis using TBA (Timed Buchi Automata) [26], and the second part is the analysis of monitoring results using SPRT. However, the approach is no longer accurate when the monitoring results fall in the undecided area.

Grunske et al. [22] improve the SPRT approach to enable continuous monitoring by back-off statistical analysis and reusing previous results, which reduces runtime overhead. However, when the actual probability is in the undecided area, the number of required samples are unlimited. It is then difficult for the approach to make a right decision.

To address the above problem, Zhu et al. [11] present a new approach using Bayesian theory. The approach checks the runtime information to determine whether the monitoring results support the null or the alternative hypothesis. To avoid Bayesian factors from falling into the undecided area, previous monitoring results are reused. The accuracy of the approach is sensitive to prior distributions since it is very difficult to select an appropriate prior distribution.

All the previous approaches have not considered environmental impact factors. Consequently, they are not suitable for real QoS monitoring problems. Zhang et al. [7] propose a weighted naive Bayesian monitoring approach termed as wBSRM. The method is the first work which considers the influence of environmental impact factors and makes QoS monitoring approaches more practical. The approach utilizes TF-IDF to quantify the initial weights of factors. However, due to the dynamic environments, the weights may be expired and useless in the further monitoring process.

As a subsequent project, Zhang et al. [12] propose an improved approach using the information gain theory, called IgS-wBSRM. The approach can update the weights in dynamic environments. Consequently, it can correct the expired weights problem in the previous monitoring approaches. The experimental results show that IgS-wBSRM outperforms the previous QoS monitoring approaches. However, the approach still cannot monitor multivariate QoS attributes.

In general, existing QoS monitoring approaches only monitor a single QoS attribute and do not support multivariate QoS attributes, which cannot meet the real QoS monitoring requirements. To address this problem, this paper proposes a novel multivariate QoS monitoring approach named M-BSRM.

3 PRELIMINARIES

In this section, Section 3.1 first gives the concepts of *impact factors* and *impact factor combinations*. Section 3.2 introduces Bayesian classifier. Next, the concepts of information fusion and information gain are depicted in Section 3.3 and Section 3.4, respectively. Finally, Section 3.5 presents the Pointwise Mutual Information theory.

3.1 Impact Factor and Impact Factor Combination

Compared with traditional software systems, Web service QoS values are sensitive to different kinds of environmental impact factors, such as locations of servers and users, usage periods, etc. Those factors are termed as *impact factors* of QoS attributes for simplicity. Furthermore, since QoS attributes are always affected by many factors, their combinations are termed as *impact factor combinations*. For example, locations of servers and users is an *impact factor combination*.

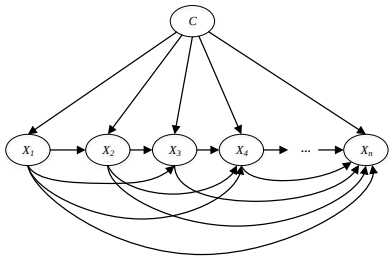


Fig. 1. The structure of the Bayesian classifier

3.2 Bayesian Classifier

Bayesian, as one of the important probability theories, was first proposed by British mathematician Bayes in the 18th century [27], [28]. For QoS monitoring, Bayesian theory is characterized by adding historical empirical data to existing predictive judgments, which combines prior probabilities and likelihood probabilities to represent all forms of uncertainty. Zhu et al. [11] propose a Bayesian-based probabilistic attribute monitoring method. Zhang et al. [7] present a weighted naive Bayesian probabilistic monitoring method considering the influence of environmental factors.

General Bayesian theorem is defined as the probability of the occurrence of event A in the case of the known occurrence of event B. In order to solve the probability of the occurrence of event B under the conditions of the occurrence of event A, it can be expressed in the following formulae:

$$p(A|B) = \frac{p(A)p(B|A)}{p(B)} \tag{1}$$

Bayesian classifier relies on Bayesian theory and combines with prior probability and class conditional probability density to form a more efficient pattern recognition model. The key idea of solving the classification problem is to use prior probabilities to express uncertainty in the set of categories. It uses the probability calculation method to estimate the distribution of sample data, and probability to express the degree of trust of the classification results.

Fig. 1 describes the main structure of a Bayesian classifier. The entire classifier has a large number of calculations during the calculation process. Furthermore, Naive Bayes has been widely recognized for its high computation efficiency and accuracy. The basic idea is that, for a specified item, the occurrence probability of each category in the occurrence of the classification item is solved. The probability result is sorted, and the classification item belongs to the item with the largest probability value in the category.

In general, the naive Bayes classification is defined as follows: Let $x = \{a_1, a_2, ..., a_m\}$ be the item to be classified and a_i be the feature attribute of the item x, where $i = 1, 2, ..., m$. Y is the category set, where $Y = \{y_1, y_2, ..., y_n\}$. Then we need to calculate $p(y_1|x)$, $p(y_2|x)$, ..., $p(y_n|x)$ one by one.

If $p(y_k|x) = max\{p(y_1|x), p(y_2|x), ..., p(y_n|x)\}$, $x \in y_k$, where $k \in \{1, 2, ..., n\}$. According to Bayes's formulae (1) we can get:

$$p(y_k|x) = \frac{p(x|y_k)p(y_k)}{p(x)} \tag{2}$$

Because Naive Bayes assumes that each feature attribute is conditionally independent, it is deduced as follows:

$$p(x|y_k) = \prod_{i=1}^{m} p(a_i|y_k) \tag{3}$$

When Equation 3 is substituted by Equation 2, we can get the following formulae

$$p(y_k|x) = \frac{p(y_k)\prod_{i=1}^{m} p(a_i|y_k)}{p(x)} \tag{4}$$

Since the denominator $p(x)$ is an invariant constant for all classes y_k, the category of the numerator's maximum is the classification of x. Consequently, Naive Bayes decision formulae can be induced in the following:

$$y(x) = argmax_{y_k \in Y} \left\{ p(y_k) \prod_{i=1}^{m} p(a_i|y_k) \right\} \tag{5}$$

3.3 Information Entropy and Information Fusion

To solve the problem of quantitative measurement of information, Shannon proposed the concept of "information entropy" [29] in 1948. In general, information entropy is used to describe the probability of random events and the certainty of the variables [30]. A random variable X whose domain is $\{x_1, x_2, ..., x_n\}$. The information entropy $H(X)$ of this variable is defined as:

$$H(x) = -\sum_{x \in X} p(x) log\, p(x) \tag{6}$$

In Equation 6, $p(x)$ is the probability density function for X. A quantitative measure of information is called entropy, which reflects the uniformity of any energy distribution in the world. If the energy distribution is more concentrated, the entropy value is smaller; on the contrary, if the energy distribution is more uniform, the entropy value is larger. The use of information entropy in QoS monitoring is expressed as: if the information entropy of the impact factor combination of the current sample is calculated before the monitoring decision being made, it represents the uncertainty measure of the classification result of the monitoring decision; if it is calculated after the dynamic sample being added, it represents the amount of information that can be obtained from the impact factor combination of the current dynamic sample.

Information fusion [15] was originally called data fusion. The simplest and most intuitive method of information fusion is the weighted average method. It weights and averages a set of redundant information. The result is used as a fusion value. Information fusion has a unique processing method with multi-dimensional QoS attribute information. The amount of QoS information in a single dimension obviously has limitations. According to the basic principle of information fusion, the information of multidimensional information that is fused by single dimensional information is clearly greater than that of any single dimensional information. Consequently, the information fusion of multidimensional QoS attributes is uniquely superior when it comes to solving QoS monitoring problems.

3.4 Information Gain

Information gain is used to denote the difference of the information entropy before and after the monitoring decision is made by the environmental impact factor of the sample [31]. The detail definition is described as:

$$IG(s) = H(C) - H(C|s) = -\sum_{c_j \in C} p(c_j * log(p(c_j))) + p(s) * \sum_{c_j \in C} p(c_j|s) * log(p(c_j|s)) + p(\overline{s}) * \sum_{c_j \in C} p(c_j|\overline{s}) * log(p(c_j|\overline{s}) \tag{7}$$

Among them, $C = \{c_0, c_1\}$, and C is the category set of monitoring results, which corresponds to two monitoring results c_0 and c_1. $IG(s)$ is the information gain of the added impact factor combination s. $H(C)$ is the information entropy of s before the addition of the impact factor combination s of the dynamic sample, which represents a measure of the uncertainty during the monitoring process. $H(C|s)$ represents the information entropy of s after the impact factor s of dynamic samples, which represents the amount of information that can be known from s. $p(s)$ represents the probability that the impact factor combination s belongs to the category C, and $p(\overline{s})$ represents the probability that the impact factor s appears in the sample set but does not belong to the category C.

The information gain of the impact factor combination s represents the amount of information that s can provide for monitoring decision classification. The larger the value, the more classification information the impact factor combination s carries. The more important it is in the classification process, the higher the weight it should be given.

From the calculation process of information gain, we know that when the impact factor combination's distribution is more concentrated in one class of the monitoring category and less in the others, the information gain value calculated by Equation 7 is larger. A non-uniform distribution of impact factors among clusters has a better classification effect than a uniform distribution of impact factors among clusters. In summary, for classification monitoring, the weight of the impact factor combination is directly proportional to the information gain value in the sample set. That is to say, the importance of the impact factor combination in the monitoring classification can be reflected by the value of the information gain value.

3.5 Pointwise Mutual Information

Mutual information [32] is a useful measure of information in information theory. It can be seen as the amount of information contained in a random variable about another random variable, or it is a decrease in the uncertainty of a random variable due to another known random variable. Point mutual information [33] is actually derived from the concept of mutual information in information theory. In general, mutual information is defined as follows:

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) log \frac{p(x,y)}{p(x)p(y)} \tag{8}$$

It measures the correlation between two random variables, that is, the amount of information contained in one random

variable about another random variable. It can be seen that mutual information is a weighted sum of point mutual information for all possible values of X and Y. Point mutual information is often used to measure the correlation between two things using the following equation. For QoS monitoring, point mutual information is used to measure the impact factor combination carried by samples and the correlation between the two classifications of monitoring.

$$I(x;y) = log\frac{p(x,y)}{p(x)p(y)} \tag{9}$$

4 The M-BSRM Approach

In this section, the problems of existing approaches are deeply analyzed in Section 4.1. In Section 4.2, we give the general overview of M-BSRM. The theoretical descriptions of M-BSRM are presented in Section 4.3. The detailed algorithms of M-BSRM are described in Section 4.4.

4.1 Problem Identification

There are two main problems in existing QoS monitoring approaches [7], [11], [12]. We identify them in the following.

First, as we already mentioned, all current monitoring approaches can only deal with single QoS attributes, which cannot satisfy the real QoS requirements. In fact, both service provider and service user may request various QoS attributes. Furthermore, there exist uncertain relationships among these different QoS attributes. For example, the *latency* value of a Web service is 0.5s, which satisfies the QoS requirements. However, the *throughput* value of the Web service may not satisfy the QoS requirements. Clearly, current QoS monitoring approaches cannot give a comprehensive evaluation of these multivariate QoS attributes Therefore, it is especially important for QoS monitoring approaches to consider multivariate QoS attributes.

Second, previous work uses the TF-IDF algorithm for QoS monitoring problem to measure combinational impact factors. It defines the weight of an impact factor combination as follows: if the impact factor combination has a higher frequency in current monitoring category and a lower frequency in the entire sample set, it is considered that it has a good monitoring category with a higher distinguishing ability. Then its ability to distinguish the current monitoring category is low. Let $w_{c_j}^{R_i}$ be the weight of an impact factor combination R_i on the monitoring category. Then the weight value can be calculated by Equation 10:

$$w_{c_j}^{R_i} = TF * IDF(R_i) = (n_{c_j}^{R_i}/n_{c_j}) * log(n/n_{R_i}) \tag{10}$$

where $w_{c_j}^{R_i}$ denotes the weight of the impact factor combination R_i, $n_{c_j}^{R_i}$ represents the number of samples that carry the impact factor combination R_i after being tested by QoS standards and belongs to the category c_j. n_{c_j} is the number of categories that the QoS samples belong to the category c_j, n is the number of all the samples, and n_{R_i} is the total number of the samples that carry the impact factor combination R_i.

TF-IDF considers a sample set as a whole in the monitoring classification process and does not consider the distribution of an impact factor combination among classes, which might lead to deviations in weight calculation.

TABLE 1
The frequency table of impact factor combination

Impact factor combination	Categories			
	c_0		c_1	
	$c0\_old$	$c0\_new$	$c1\_old$	$c1\_new$
$\langle Japan, Singapore\rangle$	8	4	0	0
$\langle Italy, Germany\rangle$	0	6	6	0

To illustrate the above problem, we give an example in the following. There are two monitoring categories: c_0 and c_1 and there are also two time windows in each category. We consider the impact factor combination $\langle Japan, Singapore\rangle$ and $\langle Italy, Germany\rangle$, where the former represents the geographical location of a client and the latter represents the geographical location of a server. Assuming that there are 24 samples in total, the number of samples carrying the two impact factors is 12 and the distribution among them is shown in Table 1. From the table, we can see that the impact factor combination $\langle Japan, Singapore\rangle$ is only distributed in the category c_0, while $\langle Italy, Germany\rangle$ is evenly distributed in both the categories. It is clear that the impact factor combination $\langle Japan, Singapore\rangle$ carries more classification information, which has better classification ability and should be given a higher weight. In contrast, the impact factor combination $\langle Italy, Germany\rangle$ is evenly distributed in the two classifications and should be given a lower weight. When TF-IDF is used to calculate the weights, the calculation results are described in Table 2.

TABLE 2
Weight table of impact factor combination obtained by TF-IDF algorithm

Impact factor combination	Categories			
	c_0		c_1	
	$c0\_old$	$c0\_new$	$c1\_old$	$c1\_new$
$\langle Japan, Singapore\rangle$	0.0894	0.0603	0	0
$\langle Italy, Germany\rangle$	0	0.0904	0.0885	0

The impact factor combination $\langle Italy, Germany\rangle$ in Table 2 has a higher weight, which surpasses the impact factor combination $\langle Japan, Singapore\rangle$ that has a good classification ability. This is obviously wrong. It also means that the TF-IDF algorithm does not consider the inter-class distribution, and will lead to the deviation of the weight calculation.

Taking the frequency of the impact factor combination in different classifications in Table 1 as an example, we can use the PMI algorithm to calculate the weights. The results are shown in Table 3. When the PMI algorithm initializes the weights, it adjusts the uneven distribution between classes, and corrects the calculation deviation caused by the uneven distribution of impact factors. Since the initial weight has a significant influence on the accuracy of the entire monitoring model, our approach using the PMI algorithm to calculate the initial weights is clearly more accurate and makes the monitoring model more sensitive.

TABLE 3
Weight table of impact factor combination obtained by PMI algorithm

Impact factor combination	Categories			
	c_0		c_1	
	$c0\_old$	$c0\_new$	$c1\_old$	$c1\_new$
$\langle Japan, Singapore \rangle$	0.3680	0.1249	0	0
$\langle Italy, Germany \rangle$	0	0	0	0

4.2 Overview of M-wBSRM

The framework of M-BSRM is shown in Fig. 2. The approach is mainly divided into two phases: *training phase* and *monitoring phase*.

Training phase. The historical QoS data samples are preprocessed to remove noise data, missing information and invalid samples. To avoid that the normalization process may cause the data to "distort" or lose important information, a single QoS attribute is checked against a predefined threshold before the normalization. Different normalization approaches are utilized for positive and negative QoS attributes. According to user preference for each QoS attribute, multivariate QoS attribute samples are integrated into a comprehensive QoS sample using information fusion theory. At the same time, impact factor combinations are extracted, and the frequency of satisfying the comprehensive QoS threshold is counted. The PMI algorithm is used to initialize the weights of the impact factor combinations. The prior probability and the class conditional density of the Bayesian classifier is obtained by the success or failure statistics of the training samples.

Monitoring stage. Like historical samples, the incoming real-time sample data stream is also denoised, discretized, and normalized. After each QoS attribute value is tested against a threshold separately, information fusion is used to fuse incoming QoS attribute samples. Then the impact factor combination is extracted, and the information gain value of the incoming sample is calculated through the sliding window mechanism, and the initial weight value is updated dynamically. The sliding window mechanism is used to update the related parameters in the Bayesian classifier and the classifier is invoked to obtain the monitoring results.

4.3 The Detailed Description of M-BSRM

Our approach mainly considers four QoS attributes: *response time*, *throughput*, *reliability*, and *availability*. For n sets of QoS samples of a Web service, a sample vector consists of n sets of QoS attribute values. Then, four QoS attributes can form an $n * 4$ matrix $Q = (Qi, j; 1 <= i <= n, 1 <= j <= 4)$. In the matrix Q, each row corresponds to sample data of different QoS attributes, and each column corresponds to sample data of the same QoS attribute. The matrix is represented by Equation 11.

$$Q = \begin{bmatrix} Q_{11} & Q_{12} & Q_{13} & Q_{14} \\ Q_{21} & Q_{22} & Q_{23} & Q_{24} \\ \dots & \dots & \dots & \dots \\ Q_{n1} & Q_{n2} & Q_{n3} & Q_{n4} \end{bmatrix} \tag{11}$$

The comprehensive QoS requirement can be defined as follows: "the probability that the comprehensive QoS value is greater than 0.5 is 50%". Then, the QoS attribute that is satisfied is denoted as c_0, and the QoS attribute that is not satisfied is denoted as c_1. Then, the detailed process of M-BSRM is described in the following steps:

Threshold test and data preprocessing. Normally, QoS attributes are divided into two categories: positive and negative constraints. The first category is a positive attribute constraint, which indicates that the higher the QoS attribute value, the better the QoS, such as *throughput*, *reliability*, and *availability*. The second type is a negative attribute constraint, which indicates that the higher the value, the worse the QoS, such as *response time* and *price*. Each single QoS attribute is checked against a threshold individually to determine whether they are within users' acceptable range. For QoS attributes with positive constraints, an acceptable minimum value is set. For QoS attributes with negative constraints, an acceptable maximum value is set. These restrictions preserve the necessary information for a single QoS attribute sample to weigh the overall QoS assessment. After all the single QoS attributes are checked against the corresponding thresholds, the normalization is performed.

For positive and negative QoS attributes, different normalization methods are used to map the QoS attribute values to the range [0,1]. Overall, the higher the normalized values, the better the QoS. Equation 12 and Equation 13 are used to normalize QoS attribute values with positive and negative constraints, respectively. $V_{i,j}$ represents the normalized value of the j-th QoS attribute sample of the i-th sample. $Q_{i,j}$ represents the true value of the j-th QoS attribute of the i-th sample. Q_j^{min} represents the minimum value of the j-th QoS attribute in the sample set. Q_j^{max} represents the maximum value of the j-th QoS attribute in the sample set.

$$V_{i,j} = \begin{cases} \frac{Q_{i,j} - Q_j^{min}}{Q_j^{max} - Q_j^{min}}, Q_j^{max} - Q_j^{min} ! = 0 \\ 1, Q_j^{max} - Q_j^{min} = 0 \end{cases} \tag{12}$$

$$V_{i,j} = \begin{cases} \frac{Q_j^{max} - Q_{i,j}}{Q_j^{max} - Q_j^{min}}, Q_j^{max} - Q_j^{min} ! = 0 \\ 1, Q_j^{max} - Q_j^{min} = 0 \end{cases} \tag{13}$$

Information fusion. According to user preference for each QoS attribute, weights are assigned to these QoS attributes. At the same time, each QoS attribute information is weighted and fused according to Equation 14. $ComprehensiveQoS_i$ is the comprehensive QoS value of each QoS attribute of the i-th sample after information fusion. W_j is the user preference for the j-th QoS attribute, and $\sum_{j=1}^{4} W_j = 1$.

$$ComprehensiveQoS_i = \sum_{j=1}^{4} (V_{i,j} * W_j) \tag{14}$$

Traditional information fusion technology does not perform threshold checking before normalization, which may lose the information that should be attributed to each QoS attribute and cause wrong decision. For example, a QoS sample vector of a Web service <response time (s), throughput (bps), reliability, availability> is <20, 34238, 0.8, 0.75>, It is observed that the response time is far from meeting the QoS requirements. If the higher *throughput* sample is incorporated, the comprehensive QoS value may satisfy the

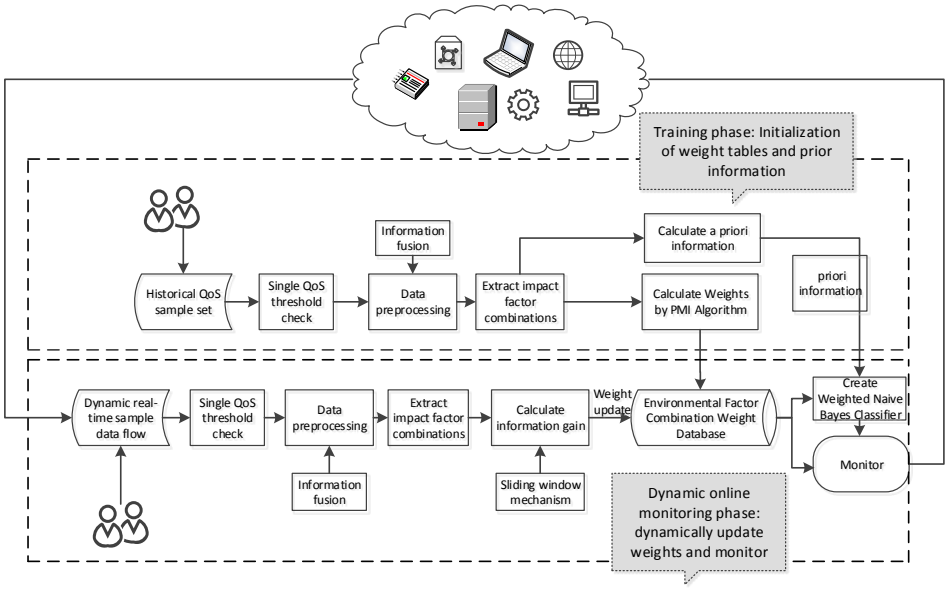


Fig. 2. Overview of M-wBSRM approach

QoS threshold and the monitor may decide that the sample meets the QoS requirements. However, in most cases, when *response time* fails to meet the requirements, the user will not be satisfied with the service even if the other QoS attributes satisfy the threshold.

Impact factor combination extraction. The approach extracts the impact factor of the sample, classifies the factors, and carries out the threshold test. The success or failure analysis of the samples is performed, and the number of samples which satisfies the QoS threshold is calculated. We use the map structure to store impact factor combinations and their corresponding statistical parameters. The parameters include the total number of samples carrying an impact factor combination, the number of times the sample carrying the impact factor combination belongs to the c_0 class, and the number of times the sample belongs to the c_1 class. These parameters will be used for the PMI algorithm to calculate the initial weights.

Constructing Naive Bayes Classifier. The probability that the sample set $X = \{x_1, x_2, x_3, ..., x_n\}$ belonging to the class c_j can be described using the Bayesian theory:

$$\begin{aligned} P(c_j|x_1, x_2, ..., x_n) &= \frac{p(c_j)p(x_1, x_2, ..., x_n|c_j)}{p(x_1, x_2, ..., x_n)} \\ &= \alpha p(c_j) \prod_{i=1}^{n} p(x_i|x_1, x_2, ..., x_{i-1}) \end{aligned} \tag{15}$$

Let $\alpha = \frac{1}{p(x_1, x_2, ..., x_n)}$, and α is a constant that is independent of the class value. Based on the theory of Maximum posterior (MAP) criterion, the Bayesian classifier can identify the category with the biggest posterior probability. Therefore, the naive Bayes classifier can be described by Equation 16.

$$C(X) = \arg\max_{c_j \in C} \{P(c_j) \prod_{i=1}^{n} p(x_i|c_j)\} \tag{16}$$

It is necessary to learn parameters (such as prior probability $p(c_j)$ and probability density $p(x_i|c_j)$) based on the training set. Since $p(c_j)$ and $p(x_i|c_j)$ are relatively small in practical applications, the requirements for calculation accuracy are relatively high. or the ease of calculation, the following discriminant is used:

$$C(X) = \arg\max_{c_j \in C} \{\log(p(c_j)) + \sum_{i=1}^{n} \log(p(x_i|c_j))\} \tag{17}$$

Naive Bayes assumes that each sample does not depend on other samples. However, this assumption is not true in actual monitoring environment. Thus, the weighted Bayesian classifier is described as:

$$C(X) = \arg\max_{c_j \in C} \{\log(1+p(c_j)) + \sum_{i=1}^{n} w_{c_j}^{R_i} * \log(1+p(x_i|c_j))\} \tag{18}$$

where $w_{c_j}^{R_i}$ is the weight of each sample. Considering that $p(x_i|c_j)$ is no greater than 1 and $log(p(x_i|c_j))$ is no greater than 0, the sample can be determined based on the important level of the classification. The log function is monotonously increasing with a finite interval, consequently $log(1 + p(x_i|c_j))$ is adopted to make the weights correct, and the final classification decision is not affected.

Based on the weighted Naive Bayesian decision function, we need to calculate $p(x_i|c_j)$ and $p(c_j)$. $p(c_j)$ denotes the prior probability and its calculation formula is:

$$p(c_j) = \frac{n_{c_j} + 1}{n} \tag{19}$$

Among them, the number of times that the dynamic sample set belongs to the c_j class is validated by the QoS standard, and n represents the number of all the samples.

$p(x_i|c_j)$ denotes the occurrence probability of the sample x_i in the class c_j. The formulae is discribed as follows:

$$p(x_i|c_j) = \frac{n_{c_j}^{x_i} + 1}{n_{c_j} + 2} \tag{20}$$

where $n_{c_j}^{x_i}$ is the number of the occurrences of x_i within c_j, and n_{c_j} is the number of samples in c_j. To prevent the product being zero in Equation 16, Laplacian smoothing [34] is performed on Equation 19 and Equation 20. Finally, all parameters are substituted into the weighted naive Bayes classifier (Equation 18) to obtain the final monitoring results.

Initializing impact factor weight table. Information theory uses mutual information to measure the correlation between two random events [35]. Assuming there are two random events X and Y, their mutual information is defined as:

$$I(X;Y) = \sum_{x \in X, y \in Y} p(x,y) log \frac{p(x,y)}{p(x)p(y)} \tag{21}$$

The weighted sum of point mutual information for all possible values of X and Y is mutual information. Point mutual information is a measure of the relevance between two things. Therefore, point mutual information is used to measure the impact factor combination carried by the sample and to monitor the correlation between the two classifications. The formulae for calculating the correlation between the impact factor combination R_i and the monitoring classification c_j is:

$$w_{c_j}^{R_i} = PMI(R_i, c_j) = log \frac{p(R_i, c_j)}{p(R_i)p(c_j)} = log \frac{n_{c_j}^{R_i}/n}{(n_{R_i}/n)(n_{c_j}/n)} \tag{22}$$

The calculated impact factor combination is stored in the weight database for c_0 and c_1 classes.

Dynamic updating. Similar to the initial samples, threshold test, normalization and information fusion are also performed on the incoming sample data stream. In a dynamic environment, the distribution of impact factors combination across monitoring categories will continue to change and the initial weights will expire. Both information gain and point mutual information are used to quantify the amount of information that belongs to the same order of magnitude. The information gain contained in the new sample is proportional to the classification ability. In Equation 23, the weight of an impact factor combination is updated dynamically using information gain. In the formulae, $IG(R_i)$ denotes the information gain of the newly added sample.

$$\begin{aligned} w_{c_j}^{R_i} new &= PMI(R_i, c_j) * IG(R_i) \\ &= log \frac{n_{c_j}^{R_i}/n}{(n_{R_i}/n)(n_{c_j}/n)} * IG(R_i) \end{aligned} \tag{23}$$

$$\begin{aligned} IG(R_i) = &- \sum_{c_j \in C} (\frac{n_{c_j}}{n}) * log(\frac{n_{c_j}}{n}) \\ &+ \frac{n_{R_i}}{n} * \sum_{c_j \in C} \frac{n_{c_j}^{R_i} + 1}{n_{R_i} + 1} * log(\frac{n_{c_j}^{R_i} + 1}{n_{R_i} + 1}) \\ &+ \frac{n - n_{R_i}}{n} * \sum_{c_j \in C} \frac{n_{c_j} - n_{c_j}^{R_i} + 1}{n - n_{R_i} + 1} * \\ &log(\frac{n_{c_j} - n_{c_j}^{R_i} + 1}{n - n_{R_i} + 1}) \end{aligned} \tag{24}$$

Relying on a sliding window mechanism, all the relevant parameters of the Bayesian classifier can be updated. Only the samples within the sliding window are used. When a new sample is collected, the earliest one will be discarded. In this way, a dynamic update mechanism is implemented.

Monitoring. The Bayesian classifier selects the classification which has the largest posterior probability as the classification to which the sample belongs, and finally invokes the Bayesian classifier decision (Equation 18) to obtain the monitoring results.

4.4 The Algorithm Description

Algorithm 1 performs comprehensive QoS tests on multivariate QoS attribute samples to obtain whether the comprehensive QoS sample belongs to a successful sample or a failed sample. $getmultiQoSDataList()$ reads a QoS sample vector and sequentially stores its attribute values into $multiQoSBean$ of $multiQoSDataList$. During the process of reading data, $getmaxAndMinValues()$ is used to calculate the maximum or minimum values of each QoS attribute to prepare for the subsequent normalization operation. $thresholdCheck$ method is used to test the threshold of a single QoS attribute. If the threshold test is not satisfied, the lag is set to false. $computeComQoSValue$ is used to merge multiple QoS attribute samples into one comprehensive QoS sample. The result is compared with the comprehensive QoS threshold, and probability statistics is performed to obtain the statistics of the samples meeting the comprehensive QoS attribute threshold. Among them, $record\_NQoS$ is a window queue for recording each comprehensive QoS vector. *YorN* is a window queue that records the success or failure of the comprehensive QoS standard. It prepares for discarding out-of-date samples and adjusting related parameters of the Bayesian classifier.

Algorithm 1 $comStandardDecision(comQoSsample)$

Require: Sample vector $X = (x_1, x_2, x_3, ..., x_n)$; QoS probability quality attribute standard: β; Comprehensive QoS attribute value threshold: com_Value
Ensure: Successful sample or failed sample.

```
getmultiQoSDataList()
getmaxAndMinValues()
for each multiQoSBean ∈ multiQoSDataList do
    record_NQoS.add(multiQoSBean),flag=true;
    if (!thresholdCheck(multiQoSBean)) then
        flag = false;
    end if
    if (flag == true) then
        currentComValue= computeComQoSValue();
    end if
    if (currentComValue <= comQoS_Value) then
        successQoS++;
    end if
    c=successQoS*1.0/n;
    if (c >= β) then
        n_{c_0}++,YorN.add(1);
        The current sample is a successful sample;
    else
        n_{c_1}++,YorN.add(0);
        The current sample is not a successful sample;
    end if
end for
```

Algorithm 2 is based on the PMI algorithm to calculate the initial weights of an impact factor combination for the two classifications. Comprehensive QoS standard tests are performed on the samples, and the test results of samples with different impact factor combinations are statistically classified to obtain the relevant parameters required by the PMI algorithm. We substitute these parameters into the formulae to count the weight of the impact factor combination for different monitoring categories, and store the weights in the weight table. Since the PMI algorithm considers the distribution of impact factor combination between different classes, the problem of uneven distribution among classes caused by the traditional TF-IDF algorithm is addressed.

Algorithm 2 $computeWeight()$

Require: Sample vector $X = (x_1, x_2, x_3, ..., x_n)$; QoS probability quality attribute standard: β;
Ensure: weightTable.

```
n = 0;
for each x_i ∈ X do
    n++;
    if (weightTable.contains(R_i)) then
        n_{R_i}++;
    else
        weightTable.add(R_i), n_{R_i} = 1;
    end if
    if (comStandardDecision(x_k^{R_i}) == successfulsampe) then
        n_{c_0}^{R_i}++, n_{c_0}++;
    else
        n_{c_1}^{R_i}++, n_{c_1}++;
    end if
    w_{c_0}^{R_i} = math.log((n_{c_0}^{R_i}+1)*1.0/(n_{c_0}+2)/n_{c_0}*1.0/n)
    w_{c_1}^{R_i} = math.log((n_{c_1}^{R_i}+1)*1.0/(n_{c_1}+2)/n_{c_1}*1.0/n)
    Correspond to the weights of impact factor combination R_i stored in the weight table
end for
return weightTable;
```

5 Experimental evaluation

To validate our approach, we compare M-BSRM with recently proposed approaches – wBSRM [7], iBSRM [11] and lgS-wBSRM [12] using two different types of data sets (a simulated dataset and a real-world dataset QWS [36]), respectively. Because real-world QWS dataset has no predefined failure rate, we perform the first group of experiments based on the simulated data set which has an error rate. The other experiments are performed over the QWS data set. The experiments are designed to investigate the effectiveness, practicability and efficiency of M-BSRM compared with the state-of-the-art approaches.

5.1 Experimental Setup

All experiments are performed based on the Eclipse platform using Java language and a computer with Intel Core i5-3337U 1.8GHz CPU, DDR4 12G RAM and 5400 rpm HD hard disk. The first set of datasets is a set of simulated datasets using given criteria. The second set of datasets is the Quality of Web Services (QWS) dataset, which contains 150 files, and each file contains a service user which calls QoS sample data for 100 services. Examples of the parameters in the QWS data sets are shown in Table 4.

TABLE 4
QoS parameter information of QWS data set

ClientIP	WSID	Response Time(ms)	Data Size	HTTP Code	HTTP Message
128.10.19.52	8953	20172	2626	-1	java.net.SocketTimeout Exception:connect timed out
128.10.19.52	9703	847	950	200	OK
128.10.19.52	1562	1115	1471	200	org.xml.sax.SAXException: Bad types (class [B ->c
128.10.19.52	10324	1118	709	200	OK
128.10.19.52	8717	853	728	200	OK
128.10.19.52	8742	20870	2626	-1	java.net.SocketTimeout Exception:connect timed out
128.10.19.52	16506	103	564	200	OK
35.9.27.26	8953	20176	2624	-1	java.net.SocketTimeout Exception:connect timed out

The impact factor combination considered by the experiments is the client's IP address (ClientIP) and the Web service's service ID (WSID). In Table 4, *Response time* is the Response Time (ms) in the table. *Throughput* can be calculated by DataSize and Response Time. The formula for calculating *throughput* is described as follows:

$$throughput = \frac{DataSize}{ResponseTime} * 1000bps \tag{25}$$

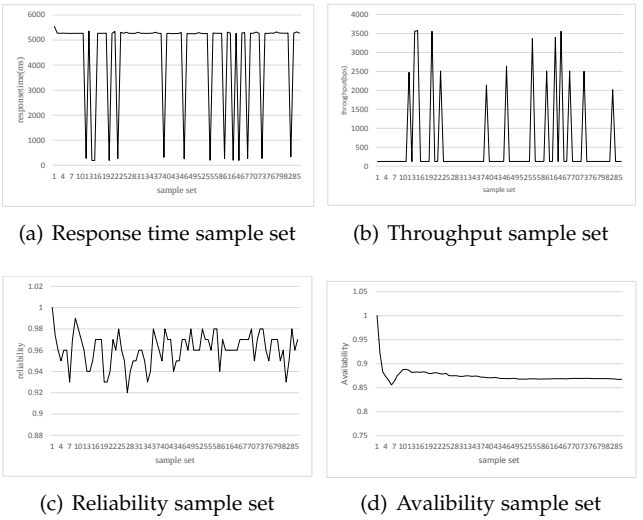
Reliability and *availability* can be obtained through the statistics of HTTP Code and HTTP Message data. When the HTTP code is 200 and the HTTP message is OK, the service is successfully executed. The quantification values are calculated based on the definitions of *Reliability* (Equation 26) and *availability* (Equation 27).

Reliability refers to the probability that a service is successfully executed, which reflects the degree to which the service request can be properly serviced. It is usually defined as follows:

$$Pr = \frac{M}{N} \tag{26}$$

Among them, M represents the total number of successful executions of the service, and N represents the total number of times the service is called within a certain period of time. Pr is used to denote the *reliability* of a service. Consequently, the higher the Pr value, the more likely the service is correctly called. The better the service, the more stable it is.

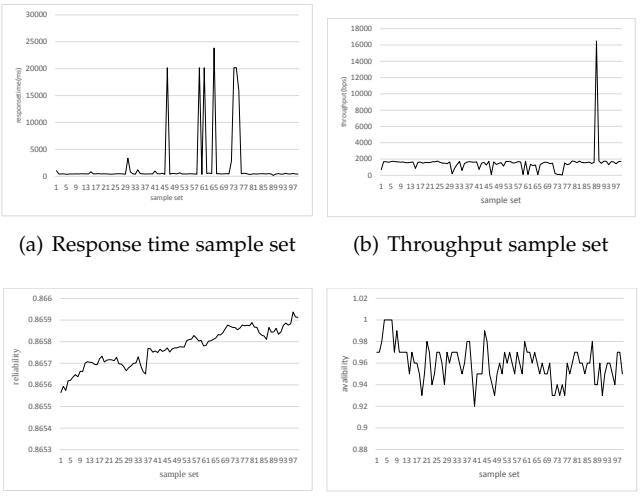
Availability is defined as the proportion of the number of successful Web service executions over a period of time. It is



(a) Response time sample set (b) Throughput sample set

(c) Reliability sample set (d) Avalibility sample set

Fig. 3. QoS data for $\langle 12.108.127.136, 13977 \rangle$



(a) Response time sample set (b) Throughput sample set

(c) Reliability sample set (d) Avalibility sample set

Fig. 4. QoS data for $\langle 128.83.122.179, 10324 \rangle$

usually expressed as a percentage of available time and can be calculated using the following formula:

$$Pa = \frac{t}{T} \tag{27}$$

where t is the time when the service can be used normally, and T is the total service running time. Pa is used to denote the *availability* of a service. General speaking, the higher the value of Pa, the more possible it is to obtain a service normally.

Fig. 3 shows the *response time*, *throughput*, *reliability*, and *availability* data for the impact factor combination (that is, the ClientIP and WSID) $\langle 12.108.127.136, 13977 \rangle$. Fig. 4 shows the *response time*, *throughput*, *reliability*, and *availability* data for the *impact factor combination* $\langle 128.83.122.179, 10324 \rangle$. Each QoS attribute data in Fig. 3 and Fig. 4 is kept within a certain range. From the figure, it can be observed that the combination of ClientIP and WSID can represent the tendency of the sample carrying the corresponding *impact factor combination*.

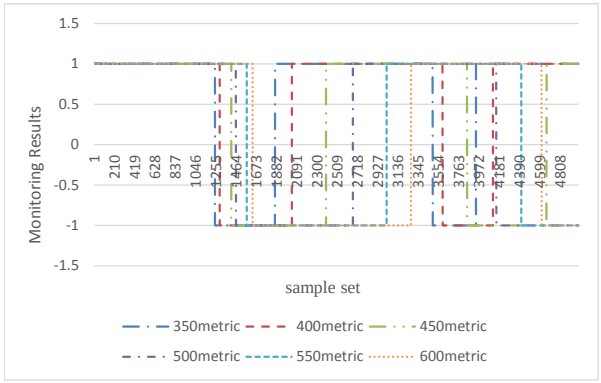


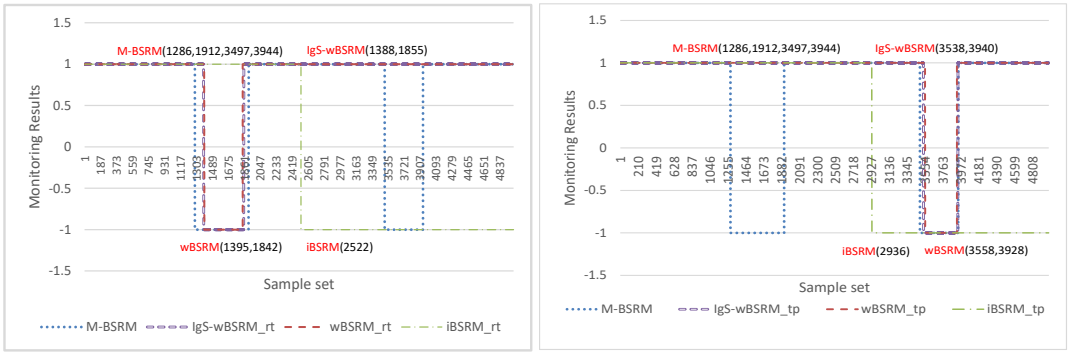
Fig. 5. Monitoring results under diffrent sizes of sliding windows

5.2 Experimental Results

5.2.1 *Effectiveness*

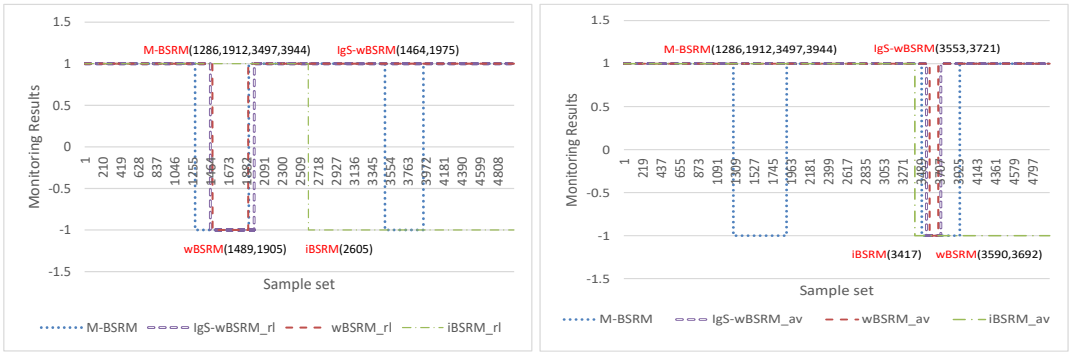
For effectiveness assessment, we verify whether M-BSRM is more sensitive and makes timely decision on injected failures compare with the other approaches. Since real world data set does not have a pre-defined error standard, the monitoring result in real world data set is unclear. Consequently, M-BSRM is first evaluated based on a simulated data set with controlled constraints using injected failures. The comprehensive QoS attribute is defined as "the probability of a comprehensive QoS value greater than 0.5 is not less than 50%". In our experiments, the user preference values for *response time*, *throughput*, *reliability*, and *availability* are all set to 0.25. During the information fusion process, user preferences of IgS-wBSRM, wBSRM, and iBSRM are set with default QoS attribute values except for the currently considered QoS attribute. For example, considering *response time*, the preference value of the *response time* is set to 1 when information is converged, and the other QoS attributes are set to default. The impact factor combinations involved in the experiment are $\langle 128.2.223.63, 1521 \rangle$ and $\langle 128.138.207.45, 20041 \rangle$. The initial weights of the impact factor combinations are same as the initial weights of the impact factor combinations obtained from the real dataset. The initial weights are obtained by training 1000 real data samples under the QWS data set. The percentage of error samples injected in *response time* and *reliability* is less than 50%, when the number of samples is 1200 to 1600, and the impact factor combination in the range of 1000 to 1800 is defined as $\langle 128.2.223.63, 1521 \rangle$. The weight corresponding to the class c_0 is 1.072981E-4, and the weight corresponding to the class c_1 is 1.754976E-4. The number of error samples injected in *throughput* and *availability* is less than 50% at 3300-3700 sample, and the impact factor combination in the 3200-4000 range is defined as $\langle 128.138.207.45, 20041 \rangle$. The weight corresponding to c_0 class is 1.241754E-4, and the weight corresponding to c_1 is 1.350249E-4.

To validate the effect of sliding window size on the monitoring results, the effect of M-BSRM is first analyzed using static sliding windows of different sizes. The experimental results are shown in Fig. 5. The ordinate indicates the classification of the monitoring results, where "1" represents acceptance and "-1" represents rejection. The abscissa indicates the number of monitored data samples. The lines



(a) Comparison of monitoring results between M-BSRM and other approaches (*Response Time*)

(b) Comparison of monitoring results between M-BSRM and other approaches (*Throughput*)



(c) Comparison of monitoring results between M-BSRM and other approaches (*Reliability*)

(d) Comparison of monitoring results between M-BSRM and other approaches (*Availability*)

Fig. 6. Comparison monitoring results under the simulated data set

parallel to the ordinate represent changes of the monitoring decisions. From the figure, it can be seen that in general when the window size is smaller, the approach is more sensitive; when the window size is larger, the classication result is less likely changed and the approach needs more time to make a judgment. From our experiments, the range of 400-450 is considered to be the optimal sliding window size. Both good and bad data can be accurately detected in this range, and there are no long delays resulting in wrong decisions. Consequently, we take 400 as the window size and compare it with IgS-wBSRM, wBSRM, and iBSRM which also incorporate the sliding window mechanism.

The comparison monitoring results are shown in Fig. 6. General speaking, from the figure we can see that the monitoring results of M-BSRM are consistent with the data of the simulated QoS data set since it merges multi-dimensional QoS information. During the entire monitoring process, iBSRM always violates the rules of the pre-defined data set and is mostly ineffective.

By taking a close look at Fig. 6a, M-BSRM detects service error prior to IgS-wBSRM and wBSRM at sample number 1286. On the one hand, since M-BSRM integrates multi-dimensional QoS information, it can avoid the erroneous sample statistics of each single QoS attribute, and checks service failure firstly. Because of the frequency variation of the impact factor combination in the two categories, the class-to-class deviation of the monitoring classification is generated. M-BSRM using information gain theory can dynamically adjust at runtime. Between samples 1200 and 1600 where the erroneous samples are completed, M-BSRM does not produce a change decision of the two categories, and the decision change is a little behind IgS-wBSRM and wBSRM. This is due to the fact that there is a certain sample of fault-tolerance during the middle region, which has not reached the decision-making change criterion for the posterior probability ratio. The error samples of multivariate QoS attributes in the 400 samples need a certain number of data samples for fault tolerance, and the difference among the samples whose QoS attributes do not meet the probabilistic quality attribute standard is not empty. Thus, M-BSRM delays the decision change of monitoring classification. Between samples 3497 and 3944, M-BSRM detects service failure and is consistent with the results in the simulated dataset. Since IgS-wBSRM and wBSRM can only monitor *response time*, they cannot detect service errors under error samples containing other QoS attributes.

In general, from Fig. 6, we can conclude that the multi-dimensional QoS information that is fused by the single-dimensional QoS information is greater than the information of any single-dimensional QoS information. M-BSRM can use fewer samples to detect service violations and service failure caused by other QoS attributes (that is, QoS attributes that are not considered by the single QoS attribute), and more effectively monitor QoS results.

To verify the effectiveness of the PMI algorithm, the monitoring performance respectively with the TF-IDF and PMI weight initialization are compared. The experimental results are shown in Fig. 7. It can be seen that the overall

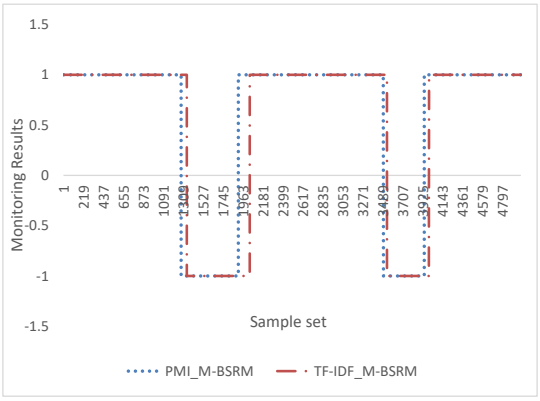


Fig. 7. Comparison of monitoring results with PMI and TF-IDF based weight initialization

monitoring decisions of the two algorithms are consistent and the PMI based monitoring method detects the service failure more quickly. This is because the calculation deviation caused by the uneven distribution of environmental factors among classes is corrected when using the PMI algorithm, in comparison to the TF-IDF algorithm that gradually reduces the deviation with weight updating.

5.2.2 Practicability

This experiment is to evaluate the practicability of M-BSRM towards real-world Web services. More specifically, we validate whether M-BSRM is able to perform the monitoring function over the real world QWS data set and achieve reasonable results in comparison to the other approaches. For the QWS dataset, a total of 6000 samples are used, and the first 1000 are taken as training samples, and the following 5000 samples are used for monitoring. The user preference for each QoS is set as follows: *response time* and *throughput* are both 0.4, and *reliability* and *availability* are both 0.1. The comprehensive QoS attribute is described as "the probability that the comprehensive QoS value is greater than 0.5 is not less than 50%". In the experiment, the other QoS attributes in IgS-wBSRM, wBSRM, and iBSRM use default values except when single QoS attributes are considered by the models. Since user preference values on *response time* and *throughput* are large, this experiment is only based on *response time* and *throughput*.

Fig. 8a describes the comparatively monitoring results of M-BSRM and the other approaches over the QWS data set, where M-BSRM considers multivariate QoS attributes while IgS-wBSRM, wBSRM, and iBSRM only consider *response time*. The monitoring results for the data samples 1300-1600 are shown in Fig. 8b. Fig. 8c describes the comparatively monitoring results of M-BSRM and the other approaches while IgS-wBSRM, wBSRM, and iBSRM only consider *throughput*. The monitoring results for the data samples 3340-4000 are shown in Fig. 8d. As shown in Fig. 8a and Fig. 8c, from the macro perspective, wBSRM, M-BSRM, and IgS-wBSRM can generate consistent results. Compared with the other approaches, M-BSRM is more comprehensive. It can detect faulty samples of different QoS attributes, while IgS-wBSRM and wBSRM can only detect faulty samples of the currently considered QoS attributes.

M-BSRM is more stable than wBSRM, and iBSRM has the wrong judgment sometimes. wBSRM makes frequent changes in decision making among certain data points. M-BSRM can detect the service failure before IgS-wBSRM and wBSRM since M-BSRM integrates information of multivariate QoS attributes. In data segment 465-613, M-BSRM can monitor the service failure, while IgS-wBSRM, as well as wBSRM, cannot detect the service failure. The reason is that M-BSRM incorporates decision metrics from the other QoS attribute error samples besides *response time*. At the same time, we observe the monitoring results of the same data samples in Fig. 8c. In sample 613, the service failure is judged and there is a certain delay. It can be determined that this is an erroneous sample with *throughput*. In Fig. 8a, from data samples 2377 to 2465, service failure is detected due to the comprehensiveness of M-BSRM. Fig. 8b and Fig. 8d clearly show that the noise phenomenon is more serious for wBSRM. The reason is that the ratio of the posterior probability of wBSRM has been dissociated from about 1 and thus the decision-making has been changed. This kind of monitoring results is obviously false.

In general, compared with IgS-wBSRM, wBSRM, and iBSRM, M-BSRM can monitor QoS more accurately and practically. It incorporates multi-dimensional QoS information and can use fewer samples to detect service failures. There may be some delays in the decision making in some cases. This is because the wrong samples of multivariate QoS attributes require more correct samples for fault tolerance.

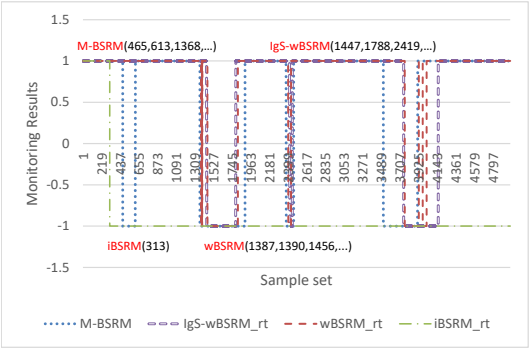
5.2.3 Efficiency

For the efficiency assessment, we investigate the time consumed for different approaches. In general, the monitoring time includes two aspects. The first is the *training time* for initializing the weight. The second is the *monitoring time* for different algorithms at runtime. For the comparative approaches, iBSRM has no training phase, and the training algorithms for wBSRM and IgS-wBSRM are the same. Consequently, we only list the training time of M-BSRM and IgS-wBSRM under different QoS standards in Table 5. From the table, we can see that the training time of the M-BSRM and IgS-wBSRM is relatively close.

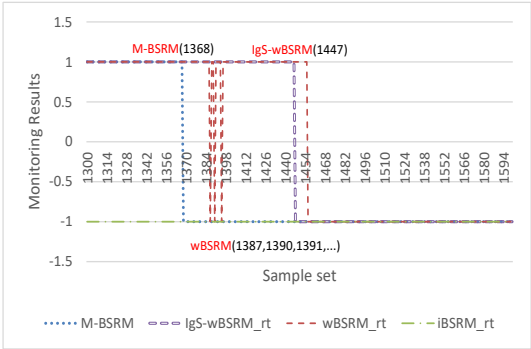
TABLE 5
Comparison of time efficiency in the training phase

time (ms)	QoS Standards							
	0.47	0.48	0.49	0.50	0.51	0.52	0.53	0.54
M-BSRM	22.22	29.83	20.30	24.81	20.55	28.05	21.03	28.98
IgS-wBSRM	26.59	27.23	26.72	22.55	28.24	23.78	26.46	20.72

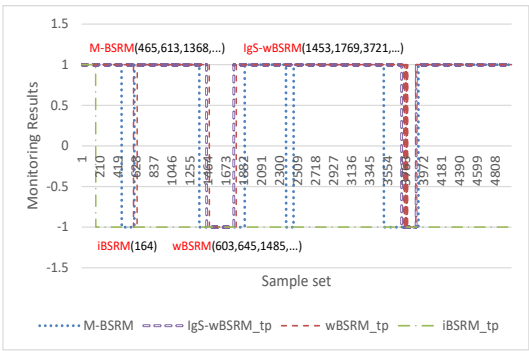
Fig. 9 compares the average monitoring time of different QoS standards for the four approaches over the QWS data set. The abscissa represents the different QoS standards, and the ordinate represents the average time consumed for different approaches. From Fig. 9, the time consumed for iBSRM is shorter than that of wBSRM, IgS-wBSRM, and M-BSRM. This is because iBSRM does not consider different kinds of impact factors on monitoring results. Consequently, there is no retrieval time spent in the weight library. However, the monitoring accuracy of the other approaches is



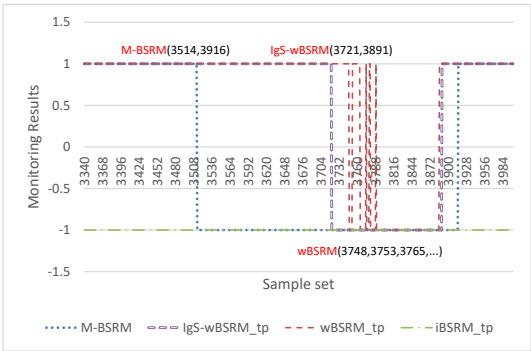
(a) Comparison of M-BSRM and other methods (response time only) monitoring results



(b) The monitoring result of the sample data segment 1300-1600



(c) Comparison of M-BSRM and other methods (throughput only) monitoring results



(d) The monitoring result of data segment 3340-4000

Fig. 8. The Monitor results under QWS data sets

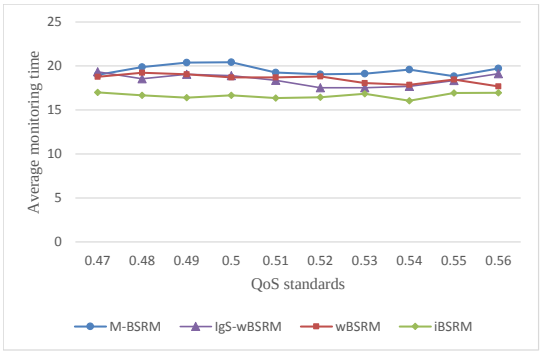


Fig. 9. Average monitoring time

obviously better than iBSRM. The monitoring time of M-BSRM is a little higher than that of IgS-wBSRM and wBSRM. The reason is that it requires the fusion of multivariate QoS attributes. Furthermore, since M-BSRM incorporates the sliding window mechanism, it can reduce the number of data statistics. Consequently, the total time complexity only increases a little. Since M-BSRM can consider multi-dimensional QoS attributes, the overall efficiency of M-BSRM outperforms IgS-wBSRM and wBSRM.

6 Conclusions

To solve the problem that the existing QoS monitoring approaches do not consider multivariate QoS attributes, a multivariate QoS monitoring approach named M-BSRM is proposed. The experimental results prove that M-BSRM is more effective, practical and efficient than other approaches.

Future work considers the following aspects. First, for different monitoring environments, multiple sets of experiments with different window sizes are required to determine the optimal sliding window range. We will further explore how to determine the optimal window range. Second, as QoS attribute data might be contextually related, the Bayesian classifier model might be optimized by considering the contextual relevance during the monitoring process.

Acknowledgments

The work is supported by the National Natural Science Foundation of China under Grant No. 61572171 and No. 61761136003, the Natural Science Foundation of Jiangsu Province under Grant No. BK20191297 and No. BK20171427, and the Fundamental Research Funds for the Central Universities under Grant No. 2019B15414. Special thanks to students *Yan Jiang* and *Zhipeng He* from Hohai University who help on designing and performing the experiments.

References

[1] M. Urkia Kortabarria, Web service performance on heterogeneous systems : A performance comparison between J2EE and .NET on heterogeneous systems, School of Humanities & Informatics.

[2] J. K. Han, H. S. Lee, G. J. Yong, SOA-based web service application and analysis - forcused to japan electronic government 3 (1) (2017) 25–28.

[3] H. Billhardt, R. Hermoso, S. Ossowski, R. Centeno, Trust-based service provider selection in open environments (2007) 1375–1380.
[4] J. El Hadad, M. Manouvrier, M. Rukoz, TQoS: Transactional and QoS-aware selection algorithm for automatic web service composition, IEEE Transactions on Services Computing 3 (1) (2010) 73–85.
[5] N. B. Mabrouk, S. Beauche, E. Kuznetsova, N. Georgantas, V. Issarny, QoS-aware service composition in dynamic service oriented environments, in: ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing, Springer, 2009, pp. 123–142.
[6] Z. Zheng, Y. Zhang, M. R. Lyu, Distributed QoS Evaluation for Real-World Web Services, in: IEEE International Conference on Web Services, 2010, pp. 83–90.
[7] P. Zhang, Y. Zhuang, H. Leung, W. Song, Y. Zhou, A Novel QoS Monitoring Approach Sensitive to Environmental Factors, in: IEEE International Conference on Web Services, 2015, pp. 145–152.
[8] L. Grunske, Specification patterns for probabilistic quality properties, in: ACM/IEEE International Conference on Software Engineering, 2008, pp. 31–40.
[9] L. Grunske, P. Zhang, Monitoring probabilistic properties, in: Joint Meeting of the European Software Engineering Conference and the ACM Sigsoft International Symposium on Foundations of Software Engineering, 2009, Amsterdam, the Netherlands, August, 2009, pp. 183–192.
[10] P. Zhang, W. Li, D. Wan, L. Grunske, Monitoring of Probabilistic Timed Property Sequence Charts, Software-Practice & Experience 41 (7) (2011) 841–866.
[11] Y. Zhu, M. Xu, P. Zhang, W. Li, Bayesian Probabilistic Monitor: A New and Efficient Probabilistic Monitoring Approach Based on Bayesian Statistics, in: International Conference on Quality Software, 2013, pp. 45–54.
[12] P. Zhang, H. Jin, Z. He, H. Leung, W. Song, Y. Jiang, Igs-wbsrm: A time-aware web service qos monitoring approach in dynamic environments, Information and Software Technology 96 (4) (2018) 14–26.
[13] P. Zhang, L. Wang, W. Li, H. Leung, W. Song, A web service qos forecasting approach based on multivariate time series, in: 2017 IEEE International Conference on Web Services (ICWS), IEEE, 2017, pp. 146–153.
[14] P. Zhang, H. Jin, H. Dong, W. Song, L. Wang, LA-LMRBF: Online and long-term web service QoS forecasting, IEEE Transactions on Services Computing, 2019, DOI: 10.1109/TSC.2019.2901848.
[15] M. Haghighat, M. Abdel-Mottaleb, W. Alhalabi, Discriminant correlation analysis: Real-time feature level fusion for multimodal biometric recognition, IEEE Transactions on Information Forensics and Security 11 (9) (2016) 1984–1996.
[16] L. Zeng, H. Lei, H. Chang, Monitoring the QoS for Web Services, in: International Conference on Service-Oriented Computing, 2007, pp. 132–144.
[17] H. Rachidi, A. Karmouch, A framework for self-configuring devices using TR-069, in: International Conference on Multimedia Computing and Systems, 2011, pp. 1–6.
[18] A. Michlmayr, F. Rosenberg, P. Leitner, S. Dustdar, Comprehensive qos monitoring of web services and event-based sla violation detection, in: Proceedings of the 4th international workshop on middleware for service oriented computing, ACM, 2009, pp. 1–6.
[19] L. Coppolino, S. D Antonio, L. Romano, F. Aisopos, K. Tserpes, Effective QoS Monitoring in Large Scale Social Networks 511 (2014) 249–259.
[20] K. Chan, I. Poernomo, H. Schmidt, J. Jayaputera, A model-oriented framework for runtime monitoring of nonfunctional properties, in: International Conference on Quality of Software Architectures and Software Quality, and Proceedings of the Second International Conference on Software Quality, 2005, pp. 38–52.
[21] I. Lee, O. Sokolsky, J. Regehr, et al., Statistical runtime checking of probabilistic properties, in: International Workshop on Runtime Verification, Springer, 2007, pp. 164–175.
[22] L. Grunske, An effective sequential statistical test for probabilistic monitoring, Information & Software Technology 53 (3) (2011) 190–199.
[23] H. Hansson, B. Jonsson, A logic for reasoning about time and reliability, Formal Aspects of Computing 6 (5) (1994) 512–535.
[24] A. Wald, Sequential tests of statistical hypotheses, Annals of Mathematical Statistics 16 (2) (1945) 117–186.
[25] P. Zhang, B. Li, L. Grunske, Timed property sequence chart., Journal of Systems & Software 83 (3) (2010) 371–390.
[26] R. Alur, D. L. Dill, A theory of timed automata, Theoretical Computer Science 125 (2) (1994) 183–235.
[27] A. L. Rukhin, Bayes and Empirical Bayes Methods for Data Analysis, Chapman & Hall/CRC, 1998.
[28] X. Wang, M. Zang, G. Xiao, Epigenetic change detection and pattern recognition via bayesian hierarchical hidden markov models, Statistics in medicine 32 (13) (2013) 2292–2307.
[29] J. A. Nez, P. M. Cincotta, F. C. Wachlin, Information entropy, Celestial Mechanics & Dynamical Astronomy 64 (1-2) (1996) 43–53.
[30] G. Y. Wang, H. Yu, D. C. Yang, Decision table reduction based on conditional information entropy, Chinese Journal of Computers 25 (7) (2002) 759–766.
[31] J. T. Kent, Information gain and a general measure of correlation, Biometrika 70 (1) (1983) 163–173.
[32] H. Peng, F. Long, C. Ding, Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy, IEEE Transactions on pattern analysis and machine intelligence 27 (8) (2005) 1226–1238.
[33] G. Bouma, Normalized pointwise mutual information in collocation extraction, Proceedings of GSCL (2009) 31–40.
[34] D. A. Field, Laplacian smoothing and delaunay triangulations, International Journal for Numerical Methods in Biomedical Engineering 4 (6) (1998) 709–712.
[35] H. Akaike, Information theory and an extension of the maximum likelihood principle, Springer New York, 1992.
[36] Z. Zheng, Collaborative reliability prediction of service-oriented systems, in: ACM/IEEE International Conference on Software Engineering, 2010, pp. 35–44.

Pengcheng Zhang received the Ph.D. degree in computer science from Southeast University in 2010. He is currently an associate professor with the College of Computer and Information, Hohai University, Nanjing, China, and was a visiting scholar at San Jose State University, USA. His research interests include software engineering, services computing and data science. He has published in premiere or famous computer science journals, such as the IEEE Transactions on Big Data, the IEEE Transactions on Emerging Topics in Computing, the IEEE Transactions on Services Computing, the Information and Software Technology, the Journal of System and Software, and the Software: Practice and Experience. He was the co-chair of IEEE AI Testing 2019 conference. He served as technical program committee member on various international conferences. He is a member of the IEEE.

Huiying Jin is a PHD candidate with the College of Computer and Information, Hohai University, Nanjing, China. She received her bachelor degree in Software Engineering from Yangzhou University, Yangzhou, China, 2017. Her current research interests include services computing and data mining.

Hai Dong is a lecturer with School of Science in RMIT University, Melbourne, Australia. He received a PhD from Curtin University, Australia and a Bachelor degree from Northeastern University, China. His research interests include Services Computing, Artificial Intelligence, Cyber Security, Data Analytics, Cloud Computing, and Internet of Things. He has published a monograph and over 80 research articles in international journals and conferences. He is a member of the ACM and the IEEE.

Wei Song received the Ph.D. degree from Nanjing University, China, in 2010. He is a full professor with the School of Computer Science and Engineering, Nanjing University of Science and Technology, China. He was a visiting scholar at Technische Universität München, Germany. His research interests include software engineering and methodology, program analysis and testing, services and cloud computing, and process analysis and mining. He was invited to the Schloss Dagstuhl Seminar "Integrating Process-Oriented and Event-Based Systems" held in August, 2016. He has published in premiere computer science journals such as *IEEE Transactions on Cloud Computing*, *IEEE Transactions on Dependable and Secure Computing*, *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Services Computing*, and *IEEE Transactions on Software Engineering*, as well as premiere international conferences such as ESEC/FSE, ASE. He is a member of the IEEE.