Reputation Measurement and Malicious Feedback Rating Prevention in Web Service Recommendation Systems

Shangguang Wang, *Member, IEEE*, Zibin Zheng, *Member, IEEE*, Zhengping Wu, *Member, IEEE*, Fangchun Yang, *Member, IEEE*, Michael R. Lyu, *Fellow, IEEE*

Abstract—Web service recommendation systems can help service users to locate the right service from the large number of available Web services. Avoiding recommending dishonest or unsatisfactory services is a fundamental research problem in the design of Web service recommendation systems. Reputation of Web services is a widely-employed metric that determines whether the service should be recommended to a user. The service reputation score is usually calculated using feedback ratings provided by users. Although the reputation measurement of Web service has been studied in the recent literature, existing malicious and subjective user feedback ratings often lead to a bias that degrades the performance of the service recommendation system. In this paper, we propose a novel reputation measurement approach for Web service recommendations. We first detect malicious feedback ratings by adopting the Cumulative Sum Control Chart, and then we reduce the effect of subjective user feedback preferences employing the Pearson Correlation Coefficient. Moreover, in order to defend malicious feedback ratings, we propose a malicious feedback rating prevention scheme employing Bloom filtering to enhance the recommendation performance. Extensive experiments are conducted by employing a real feedback rating dataset with 1.5 million Web service invocation records. The experimental results show that our proposed measurement approach can reduce the deviation of the reputation measurement and enhance the success ratio of the Web service recommendation.

Index Terms—Web service recommendation, feedback rating, reputation, Cumulative Sum Control Chart, Pearson Correlation Coefficient.

1 INTRODUCTION

Web service technologies create an environment where users and applications can search and compose services in an automatic and seamless manner. In the service-oriented environment where everybody is allowed to offer services, it is natural that there will be numerous offers of services providing equivalent or similar functionality [1]. Moreover, Web services that span diverse organizations and computing platforms can be composed to create new, value-added service-oriented applications efficiently. However, some Web services may act maliciously. Hence, a key requirement is to provide an effective mechanism in recommending trustworthy services for users.

- *S. Wang and F.Yang are with State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Box 187#, No.10, Xi Tu Cheng Road, Beijing, China, 100876. E-mail: {sgwang; fcyang}@bupt.edu.cn*
- *Z. Zheng and Michael R. Lyu are with Department of Computer Science & Engineering, The Chinese University of Hong Kong. E-mail: {zbzheng; lyu}@cse.cuhk.edu.hk*
- *Z. Wu is with Department of Computer Science and Engineering, University of Bridgeport. E-mail: zhengpiw@bridgeport.edu*

Manuscript received 0000; revised 0000; accepted 0000; published online 0000.
For information on obtaining reprints of this article, please send e-mail to: tsc@computer.org, and reference IEEECS Log Number TSC-0000-00-0000. Digital Object Identifier no. 10.1109/TSC.0000.0.

Web Service recommendation systems can be employed to recommend the optimal Web service for satisfying user's requirements [2]. Service recommendation is helpful for users when two or more Web services have the same functionality but different Quality-of-Service (QoS) performance. QoS is defined as a set of non-functional properties, including reputation, response time, reliability, etc. Web service recommendation can provide the user with necessary information to help decide which Web service should be selected [3].

Most QoS-aware Web service recommendation schemes are based on the qualities promised by service providers. However, service providers may fail partially or fully in delivering the promised quality at runtime [4]. It is not an easy task since some service providers may not fulfill their promised service quality. The reputation of Web service needs to be considered when making a service selection. Web service reputation is regarded as a metric of its future behavior. It is a collective measurement of the opinions of a community of users regarding their actual experience with the Web service [3]. It is computed as an aggregation of users' feedback ratings over a specific period of time (a sample interval) and reflects the reliability, trustworthiness, and credibility of the Web service and its provider.

With the Web service reputation taken into consideration, the probability of recommending the optimal service and the success ratio of the composite services can be increased. However, as it is not realistic to assure that the user feedback ratings are fairly accurate and non-malicious [5], several studies have recognized the importance of reputation measurements of Web services. The proposed solutions [6-10] employ different techniques to measure Web service reputations based on user feedback ratings. Although previous work has explored the efficiency and robustness of various measurement approaches, most of them [6-10] suffer from the weaknesses described as follows.

First, it is difficult to ensure the purity of user feedback ratings because of the existence of malicious users. Malicious users could provide malicious feedback ratings to affect the measurement results for commercial benefit. In open service-oriented environments, there are no widely-employed user verification mechanisms. Participating users are usually represented by a pseudonym. In such environment, a special threat comes from Sybil attacks [11]. This attack allows a single malicious user to be represented by an arbitrary number of forged users. Hence, malicious users can initiate a flood of malicious feedback ratings to subvert the reputation system of Web services.

Second, previous approaches fail to ensure the accuracy of feedback ratings. There is a large variety of users on the Internet. Users have different feedback rating styles [12]. Different users often give different feedback ratings to the same service. For a reputation mechanism to be fair and objective, it is essential to measure reputation on the basis of fair and objective feedback ratings.

Finally, most previous research focused on various feedback rating aggregation schemes of reputation measurement, and little work investigated preventing malicious feedback ratings. If the Web service recommendation system cannot prevent malicious feedback ratings, any effective reputation measurement approach will become invalid since these malicious feedback ratings suppress benign feedback ratings. Hence, an effective malicious feedback rating prevention scheme is very essential for the reputation measurement of Web services.

In our previous work [13], we briefly analyze the importance of a reputation measurement in service computing, which lacks of deep research on reputation measurement and malicious feedback rating prevention. To address these weaknesses, this paper extends our previous work by proposing a reputation measurement approach to reduce the deviation of the reputation measurement of Web services and to improve the success ratio of the service recommendation. Moreover, to prevent malicious users from suppressing benign feedback ratings, this paper presents a malicious feedback rating prevention scheme. This paper makes the contributions: 1) we adopt the Cumulative Sum Control Chart to identify malicious feedback ratings to lessen the influence of malicious feedback ratings on the trusted reputation measurement; 2) we devise feedback similarity computation to shield the different preferences in feedback ratings of users using the Pearson Correlation Coefficient; 3) we propose a malicious feedback rating prevention scheme to prevents malicious users from suppressing benign feedback ratings using a standard Bloom filter; 4) we validate our proposed malicious feedback rating prevention scheme through theoretical analysis, and also evaluate our proposed measurement approach experimentally on a real feedback rating dataset involving 1.5 million real-world Web service invocation records.

The remainder of this paper is organized as follows. Section 2 introduces related work. Section 3 describes the proposed reputation measurement approach. A malicious feedback rating prevention scheme is proposed in Section 4. Section 5 gives the theoretical analysis about the proposed measurement approach. Section 6 conducts experiments to evaluate the proposed measurement approach and Section 7 concludes the paper.

2 Related Work

To provide accurate reputation measurement for Web service recommendation, some notable reputation measurement schemes have been proposed.

Conner et al. [7] proposed a reputation-based trust management framework that supports the synthesis of trust-related feedback ratings from multiple services that are hosted within an infrastructure. The core of the framework is a trust management service (TMS). TMS allows each service to use its own trust metrics, to meet its local trust requirements, and to support multiple reputation scoring functions. This framework has a significant advantage in that it supports multiple reputation measurement approaches, which are suitable to multiple Web service environments. TMS takes the client, the service, the normalized transaction feedback rating, and the set of optional attributes to create a service invocation history record. However, for malicious feedback ratings, malicious users often collude with other users. Then TMS cannot find malicious feedback ratings. Moreover, TMS calculates the trustworthiness of a given peer as the average feedback weighted by the scores of the feedback users. Unfortunately, a feedback user with high trustworthiness is not consistently reliable and it also provide malicious feedback ratings for the illegal acquisition of economic benefits. Hence, TMS cannot get accurate the reputation when good feedback users become bad or bad users become good.

Limam and Boutaba [10] proposed a feedback computation model, derived from the expectancy disconfirmation theory from market science, was used to generate a feedback from service utility and cost, and then a reputation derivation model had also been proposed to aggregate feedbacks into a reputation value that better reflects the behavior of the service at selection time. However, the model cannot shield users' different feedback preferences, which makes the reputation value biased, and lessens the accuracy. Moreover, it is very difficult to predict the feedback ratings in real Web service environments, especially, existing malicious feedback behaviors. Hence, the model cannot obtain the deserved reputation value. Z. Wang and J. Cao [14] proposed a two-layer method for evaluating and selecting QoS guaranteed resources from a number of potential Grid resource candidates. In the bottom layer, the informed users contribute their experiences and make fuzzy-based judgments about a resource individually. In the top layer, the approach selects judgments from all representatives and makes a comprehensive decision. The two-layer method is stable and accurate in different grid environments. R. Zhou and K. Hwang [15] proposed a P2P reputation system called PowerTrust. The PowerTrust dynamically selects a small number of power nodes and then by using a look ahead random walk strategy and leveraging the power nodes, the PowerTrust improves the global reputation accuracy and aggregation speed. What's more, the PowerTrust is robust and scalable in peer joining, peer leaving and malicious peers, which can significantly achieve high query success rate in P2P file-sharing applications. However, most P2P systems deployed on the Internet are unstructured. Unfortunately, the schemes [15,16] can not support unstructured P2P system.

S. D. Kamvar et al. [17] presented an effective method to minimize the impact of malicious peers on the performance of a P2P system. The method computes a global trust value for a peer by calculating the left principal eigenvector of a matrix of normalized local trust values, thus taking into consideration the entire system's history with each single peer being able to decrease the number of inauthentic files in the P2P system. J. Caverlee et al. [18] presented the SocialTrust framework that supports tamper-resilient trust establishment in the presence of large-scale manipulation by malicious users, clique formation, and dishonest feedback. By distinguishing relationship quality from trust, incorporating a personalized feedback mechanism for adapting as the community evolves and tracking user behavior, the SocialTrust can significantly support the robust trust establishment in online social networks. In contrast to existing schemes which suffer from low performance because of malicious feedback ratings and different user preferences, our approach can mitigate these malicious or subjective feedback ratings by filtering and adjusting these data. Our approach can find malicious feedback ratings since it does not rely on the trustworthiness of each feedback user. Moreover, it can also reduce the influences of different users' feedback preferences on the accuracy of Web service reputation measurement but existing schemes cannot support the essential case.

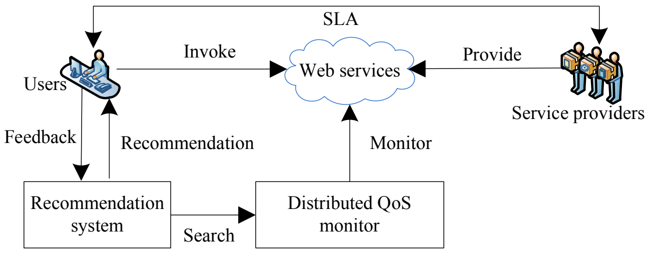


Fig. 1. Framework of feedback rating monitoring.

3 The Reputation Measure

The reputation represents a collective perception of the users in the community about a Web service, that is, the reputation of a given service is a collective feedback rating of the users that have interacted with or used the service in the past.

Feedback rating is the perception of each user about invoked services. It could be a single value representing an overall perception or a vector representing a value for each QoS attribute of a Web service, such as a response time, reliability, and availability.

Fig. 1 shows what occurs when a user sends a service request to the recommendation system. With a Service Level Agreement (SLA) between a user and a service provider, the user selects a Web service that satisfies his QoS requirements and then invokes the service. After the service is consumed, the user reports a feedback rating for the service regarding the performance of the Web service. Finally, the recommendation system collects the feedback rating and other feedback ratings from other users with a Data Collector, calculates the reputation (scores), updates these scores in a QoS repository, and provides the scores when recommending services to the users.

In this study, for the *j*-th invoked service $s_j (j = 1, 2, ...)$, a user provides a feedback rating that indicates the level of satisfaction with the service after each interaction with the service. A feedback rating is simply an integer that ranges from 1 to R (e.g., R=10), where R means extreme satisfaction and 1 means extreme dissatisfaction. Then users maintain n feedback ratings which represent their perception of s_j's performance. We take $q(s_j)$ to represent the reputation score of s_j over a global time. Then $q(s_i)$ can be calculated with the following:

$$q(s_j) = \frac{1}{n}\sum_{i=1}^{n} r_i \tag{1}$$

where r_i represents the *i*-th feedback rating, $n(n = 1, 2, ...)$ represents the number of feedback ratings.

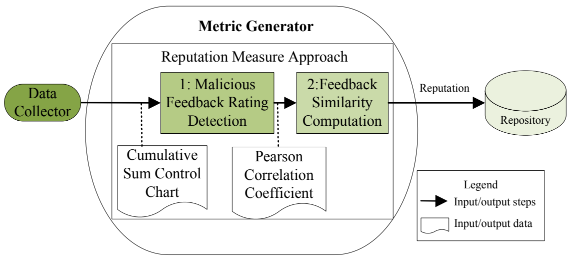

Metric Generator
Reputation Measure Approach
Data Collector
1: Malicious Feedback Rating Detection
2:Feedback Similarity Computation
Reputation
Repository
Cumulative Sum Control Chart
Pearson Correlation Coefficient
Legend
Input/output steps
Input/output data

Fig. 2. Procedures of the reputation measurement approach.

However, because the reputation influences the recommendation of an interaction partner, some dishonest service providers misuse the system. These service providers might have a direct interest in improving the chances of a certain candidate to become selected or to diminish the chances of others. Moreover, the feedback rating can be individual because it is based on users' personal expectations and opinions. The different users that invoke the same service may provide varied feedback ratings. Therefore, the main challenge is addressing services that attempt to provide misleading feedback ratings, either unfair or subjective feedback ratings. Hence, a recommendation system needs appropriate mechanisms for filtering and weighting services with a reputation metric. However, in evidence-based reputation measure approaches, the trust an entity has in another entity is usually linked to a pseudonym that influences the accuracy of the reputation measurement. Moreover, they may fail to recognize the feedback ratings with users' preferences. They do not cater to the accuracy of a reputation measurement, which makes the reputation of a Web service deviate from its actual value in a composition system or an e-commerce application. Hence, to solve the problem, we propose a reputation measurement approach that is based on a feedback rating evaluation for the Web service recommendation.

As shown in Fig. 2, our proposed measurement approach mainly contains two phases, i.e., a malicious feedback rating detection and a feedback rating adjustment. The first phase involves detecting malicious feedback ratings collected by a Data Collector using the Cumulative Sum Control Chart (called CUSUM). The second phase involves computing the feedback similarity of different users using the Pearson Correlation Coefficient to adjust the feedback ratings. Finally, the Repository stores the reputation measured scores and provides the scores when requested by the recommendation system. Details of these phases are presented in Section 3.1 and Section 3.2, respectively.

3.1 Phase 1: Malicious Rating Detection

3.1.1 *Data sampling and the CUSUM*

A special threat to the reputation measurement of Web services comes from malicious feedback ratings such as Sybil attacks [19-20]. Hence, malicious feedback ratings must be considered in reputation measurements of Web services.

Under normal situations, each user selects a recommended Web service, invokes it with an expected QoS, and ends with a feedback rating. When malicious users attack the reputation system, there are more negative feedback ratings than the usual situation (an example of the malicious feedback ratings is shown in the appendix). Therefore, under abnormal situations, there would be more malicious feedback ratings than benign feedback ratings in a sampling interval. In practical applications, the reputation system of Web services can become invalid with mass malicious feedback ratings. Consequently, the reputation system is unable to reply to user recommendation requirements effectively. Hence, our aim is to recognize attacks by detecting an imbalance in the feedback rating flow for an anomalous shift in the positive or negative direction.

In this study, in order to more accurately find the anomalous shift than the above example, we apply the Cumulative Sum Method (CUSUM) to detect and handle malicious feedback ratings. The CUSUM as a sequential analysis technique is typically used for monitoring change detection based on hypothesis testing. It is developed for independent and identically distributed random variables. For example, for a process $\{y_i\}\,(i = 1, 2, ...)$, there are two hypotheses, θ_0 and θ_1, with probability density functions $p\theta_0(y_i)$ and $p\theta_1(y_i)$. The first hypothesis corresponds to the statistical distribution prior to a change and the second hypothesis corresponds to the distribution after a change. The CUSUM for signaling a change is based on the log-likelihood ratio C_n, which is given by the following:

$$C_n = \sum_{i=1}^{n} c_i \tag{2}$$

with $c_i = \ln \frac{p\theta_1(y_i)}{p\theta_0(y_i)}$.

The typical behavior of the log-likelihood ratio contains a negative drift before a change and a positive drift after the change (we give an example in Fig. 3). Hence, the relevant information for detecting a change is from the difference between the log-likelihood ratio, $C_n(n = 1, 2, ...)$, and its current minimum value (i.e.,

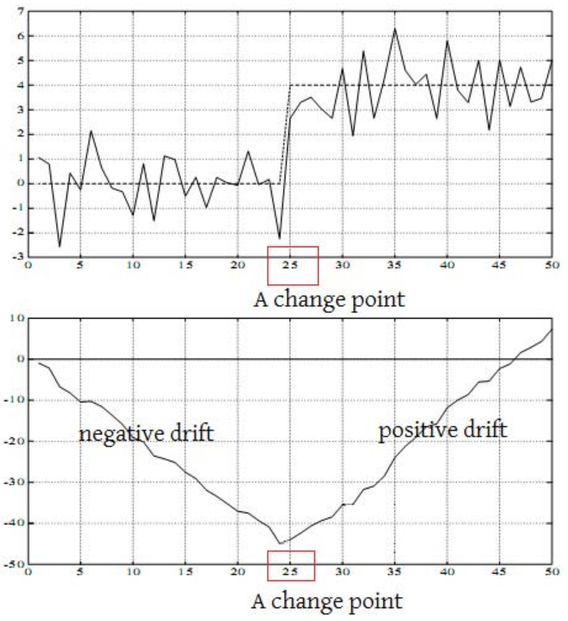


Fig. 3. An example: the typical behavior of the log-likelihood ratio. The horizontal axis in these figures is the number of sampling interval. The vertical axis in the above figure represents the detection object and The vertical axis in the below figure represents the detection results.

a threshold value), $h(h > 0)$. If $C_n \geq h$, then a positive shift occurs in the n-th sample, i.e., there is an abnormal detection point. It is well known that CUSUM is well-suited for checking the abnormal shift and has been widely used for detecting the small and moderate mean shift [21-22]. We can take the malicious feedback ratings as the abnormal shift according to Eq.3; hence, CUSUM can be used for detecting the malicious feedback ratings. As shown in Table 1, in order to understand our detection mechanism by using CUSUM, we make a table listing parameters and meanings for the reader's reference. The detailed detection mechanism is described in the following section.

TABLE 1
CUSUM parameters

Parameter	Meaning
n	The number of sample intervals.
y_i	The number of detection objects (such as feedback ratings) in the i-th sample interval.
θ_0	The parameter of the statistical distribution prior to a change.
θ_1	The parameter of the statistical distribution after a change.
$p\theta_0(y_i)$	The probability density function with the parameter θ_0.
$p\theta_1(y_i)$	The probability density function with the parameter θ_1.
C_n	The log-likelihood ratio.
h	The threshold value.

3.1.2 *Detection mechanism*

This section focuses on the application of CUSUM to detect and handle the malicious feedback ratings, including positive malicious feedback ratings (i.e., unfairly high feedback ratings) and negative malicious feedback ratings (i.e., unfairly negative feedback ratings). Because a negative malicious feedback rating (negative drifts) detection is similar to a positive malicious feedback rating (positive drifts) detection [22], in this study, we only consider positive malicious feedback rating detection.

For each feedback rating, CUSUM monitors a set of $n(n = 1, 2, ...)$ feedback rating sample intervals $\{y_1, \cdots, y_n\}$. The variable $y_j(y_j = \sum_{i=1}^{m} r_i)(1 \leq j \leq n)(m = 1, 2, ...)$ is the sum of all of the feedback ratings (called feedback rating traffic in this paper) in the j-th sample interval (a specific period of time) where m is the number of feedback rating in the sample interval. According to the literature [20,23], assume that the change feedback rating traffic $\{y_j\}$ is an independent Gaussian distribution with a known variance σ^2 (the assumption of a Gaussian distribution about $\{y_j\}$ can be utilized better by removing no stationary behavior [24]). The assumption remains the same after the change, and μ_0 and μ_1 are the mean feedback rating traffic before and after the change. Then CUSUM can be described as follows:

$$f_n = \left[f_{n-1} + \frac{\mu_1 - \mu_0}{\sigma^2}(y_n - \frac{\mu_1 + \mu_0}{2}) \right]^+, \quad (3)$$

where if $f > 0$, $f^+ = f$; otherwise, $f^+ = 0$.

In addition, to reduce the complex and time-consuming calculations, we consider a simple approach to apply CUSUM to $\tilde{x}_n$, with the following:

$$\tilde{x}_n = x_n - \bar{\mu}_{n-1}(n = 1, 2, ...), \quad (4)$$

where x_n is the sum of all of the feedback ratings in the n-th sample interval, and $\bar{\mu}_n$ is an estimate of the mean rate at the n-th sample interval.

We can obtain $\bar{\mu}_n$ of Eq.4 with an exponential weighted moving average, as follows:

$$\bar{\mu}_n = \lambda\bar{\mu}_{n-1} + (1 - \lambda)x_n, \quad (5)$$

where $\lambda \in [0, 1]$ (its setting depends on the application preference) is the exponential weighted moving average (EWMA) factor, i.e, the weight given to the most recent rational subgroup mean.

The mean feedback rating traffic of $\tilde{x}_n$ prior to a change is zero; hence, the mean in Eq.4 is $\mu_0 = 0$. A remaining issue that must be addressed is the value of μ_1, which is the mean rate after the change. This value cannot be known beforehand and its setting depends on the application preference. Hence, we approximate it with $\alpha\bar{\mu}_n$, where α is an amplitude percentage parameter, which corresponds to the increasing mean rate after a change.

Then the CUSUM from Eq.3 can be written as follows:

$$f_n = \left[f_{n-1} + \frac{\alpha\bar{\mu}_{n-1}}{\sigma^2}(x_n - \bar{\mu}_{n-1}\frac{\alpha\bar{\mu}_{n-1}}{2})\right]^+ . \quad (6)$$

If $f_n \geq h$ ($h > 0$ is the predesigned CUSUM threshold parameter, i.e., the average attack strength), then a malicious attack has been detected and the feedback ratings are dropped.

3.2 Phase 2: Rating adjustment

Although malicious feedback ratings can be detected using CUSUM, feedback ratings are often subject to the different preferences of the user with the same service, which fails to ensure the accuracy of the feedback ratings. It is well known that there is a large variety of users on the Internet. These users, who have different preferences, report feedback ratings that are often subject to their preferences. Some users may be conservative, whereas some others may be aggressive or neutral. Hence, it is imperative to shield the influence of conservative, aggressive, or neutral feedback ratings for the same service.

In our study, feedback similarity computation is proposed to shield the influence of different preferences of users and to adjust their feedback ratings with the Pearson Correlation Coefficient (PCC) [25].

We assume that there are m users and n Web services, and the relationship between users and Web services is denoted with an $m \times n$ matrix [2]. Then each entry $r_{a,i}$ in the matrix denotes the feedback rating of the Web service i rated by the user a where $r_{a,i}$ is from Phase 1 and is a normal feedback rating.

The PCC uses the following equation to compute the similarity between user a and user u based on their commonly-rated Web services:

$$Sim(a,u) = \frac{\sum\limits_{i \in I_a \cap I_u} (r_{a,i} - \overline{r_a})(r_{u,i} - \overline{r_u})}{\sqrt{\sum\limits_{i \in I_a \cap I_u} (r_{a,i} - \overline{r_a})^2}\sqrt{\sum\limits_{i \in I_a \cap I_u} (r_{u,i} - \overline{r_u})^2}}, \quad (7)$$

where $Sim(a,u) \in [-1,1]$ represents the similarity of two users (a larger value indicates a higher similarity), $I_a \cap I_u$ is a set of commonly rated Web services by both users a and u, $r_{a,i}$ and $r_{u,i}$ are the two feedback ratings of Web service i rated by user a and u, respectively ($r_{a,i}$ and $r_{u,i}$ are from Phase 1), and $\overline{r_a}$ represents the average feedback rating of all of the Web services that are rated by user a.

After calculating and ranking the PCC similarity values between the current user and the other users, a set of similar users $S(a)$ can be identified, as follows:

$$S(a) = \{u | Sim(a,u) \geq Sim_k, Sim(a,u) > 0, a \neq u\}, \quad (8)$$

where Sim_k is the k-th largest PCC value with the current user u where k presents the number of the similar users (i.e., they have larger PCC values than others), and $Sim(a,u) > 0$ is to exclude the dissimilar users (dissimilar users, e.g., those with negative PCC value, will influence the reputation measurement accuracy).

After obtaining the set of similar users, according to a set of community Web services $SS^k = \{s_1^k, \cdots, s_l^k\}$, which contains l services used by the K users, we can calculate the feedback similarity between user a and user u as the following:

$$FS^k(a,u) = \begin{cases} 1 - \sqrt{\frac{\sum\limits_{u \in S(a)} (r_{a,i}^k - r_{u,i}^k)^2}{|SS^k|}}, & if\ |SS^k| \neq 0 \\ 0 & ,\ if\ |SS^k| = 0 \end{cases}, \quad (9)$$

where $FS^k(a,u) \in [0,1]$ represents the similarity of two users (a larger value indicates a higher similarity) and $|SS^k|$ is the number of services in SS^k.

Having calculated the feedback similarity, we can use $FS^k(a,u)$ to adjust the feedback ratings of user a according to the feedback ratings of other similar users with the following:

$$\hat{r}_{a,i} = \sum_{u \in S(a)} \frac{FS^k(a,u)}{\sum\limits_{u \in S(a)} FS^k(a,u)} \times r_{u,i}, \quad (10)$$

where $\hat{r}_{a,i}$ is the adjusted feedback rating of the i-th rated service from user a, $r_{u,i}$ is the feedback ratings of Web service i rated by user u.

Having executed the two phases mentioned above, to gain the accurate reputation measurement, we transform Eq.1 into Eq.11 by the following equation:

$$q(s_j) = \frac{1}{m}\sum_{a=1}^{m} \hat{r}_{a,i}, \quad (11)$$

where s_j represents the j-th Web service and $q(s_j)$ represents the measured reputation of the service s_j.

4 Malicious Rating Prevention

In this section, in order to prevent malicious feedback ratings from reaching the QoS repository of service brokers, we propose a malicious feedback rating prevention scheme. Its aim is to cooperate with the proposed reputation measurement approach to enhance the performance of the recommendation system. The idea is to identify the IP addresses with the offending feedback ratings and filter them out. In order to achieve this, we employ a standard Bloom filter to prevent the anomalous feedback ratings.

The Bloom filter was formulated by Burton H. Bloom in the 1970s [26]. It is first "programmed" with each message in the set, and then queried to determine the membership of a particular message, i.e., whether an element is a member of a set. It is a data structure used for representing a set of messages succinctly, and is widely used for different purposes of Internet applications. For the convenience of the reader, we give an overview of the Bloom filter in the following section.

4.1 Overview of the Bloom Filter

We begin by presenting the mathematics behind a standard Bloom filter. A standard Bloom filter for representing a set $S = \{x_1, x_2, ..., x_n\}$ of n elements, is described by an array of m bits, initially all set to 0. A standard Bloom filter uses k different hash functions $h_1, h_2, ..., h_k$, each of which maps or hashes some set element to one of the m array positions with a uniform random distribution over the range $1...m$. For each element $x \in S$, the bits $h_j(x)$ are set to 1 for $1 \leq j \leq k$. A location can be set to 1 multiple times, but only the first change has an effect. To check if an item y is in S, we check whether all $h_j(y)$ are set to 1. If not, then clearly y is not a member of S. If all $h_j(y)$ bits are set to 1, y is in S. If all $h_j(y)$ bits are found to be 1 and y is not a member of S, then it is a false positive (The false positive is sufficiently small, and almost can be ignored according to practical application [26]).

4.2 Prevention Scheme

The key of the prevention is to identify the IP addresses that are associated with malicious feedback ratings, and then inform the service broker to block malicious users from rating these Web services. Hence, our proposed prevention scheme contains two stages, i.e., activating stage and blocking stage.

In the activating stage, the first step to implement a Bloom filter is initializing the following parameters: the upper bound on false match probability of the Bloom filter, the filter size m of the Bloom filter, and the number of hash functions k of the Bloom filter. The second step is to identify a malicious feedback rating IP address set $S = mrip_1, mrip_2, ..., mrip_n$ with n items. We will first show how a Bloom filter is represented through a series of item insertion operations. Algorithm 1 includes the details regarding the process of the activating prevention operation. It is clear that when malicious feedback ratings are detected in the i-th sample interval by using the CUSUM algorithm (Section 3.1.2), the set S collects IP addresses of feedback ratings in the sample interval. Because attackers often provide malicious feedback ratings in a short time, we assume that S can collect all malicious IPs. The final step is to use k independent hash functions $h_1, h_2, ..., h_k$ to map each item of S to the bit vector $1, ..., m$ uniformly. When inserting $mrip$, Algorithm 1 sets the bits at all these positions to 1. Hence, it is convenient to represent S as a Bloom filter by invoking Algorithm 1 repeatedly. After achieving the Bloom filter, the blocking stage starts to run from the $(i+1)$-th sample interval to the n-th sampler interval when $f_i \geq h$ in the i-th sample interval. It can block malicious feedback ratings by checking IP addresses based on the Bloom filter instead of S. The detailed blocking process is illustrated in Algorithm 2, which uses an item ip as input. If all the hash[j] bits are set to 1 for $1 \leq j \leq k$ in the Bloom filter, then the

Algorithm 1 Activating Prevention

```
Input: $mrip$, malicious feedback rating IP address elements
Output: An activated Bloom filter
if $f_n \geq h$ and $S$ is not null then
    Initiate $k$ and $m$;
    Obtain the set $S$;
    for $i$=1 to $n$ do
        for $j$=1 to $k$ do
            Vector(hash[j](mrip[i]))$\leftarrow$ 1;
        end for
    end for
end if
```

item ip is a member of S, i.e., it is with a malicious feedback rating. Otherwise, ip is not a member of S, i.e., it is with a benign feedback rating. In the blocking stage, we define the blocking ratio (BR) as the ratio of the number of the IPs with malicious feedback ratings and all IPs in the same sample interval, i.e., $BR = \theta/n$ where the better the algorithm is, the larger the blocking ratio is.

After we identify the malicious IPs, the remote service broker (RSB) will be responsible for finding out the malicious clients who rated those Web services. Finally, the authentication and authorization module (AAM) of the RSB will block these malicious users. This stage is relatively straightforward and is not the focus of this paper. By our proposed prevention scheme, once an attacker has been detected, we can drop the feedback ratings that are associated with the attacker or the victim by discriminating the IP addresses. With the help of the RSBs, our reputation system can shield against the malicious feedback ratings from the reputation measurement of each Web service.

Algorithm 2 Blocking malicious feedback ratings

```
Input: $ip$, IP address elements
Output: $BR$, the blocking ratio
$\theta = 0$;
for $i$=1 to $n$ do
    for $j$=1 to $k$ do
        if Vector(hash[j](ip[i]))=0 then
            $\theta++$;
            RSB.get(ip[i]);
            AAM.block(ip[i]);
        end if
    end for
end for
$BR \leftarrow \theta/n$;
Return $BR$;
```

5 Theoretical Analysis and Limitation

In this section, we give a theoretical analysis of the proposed reputation measurement approach and malicious feedback ratings prevention scheme, and discuss the limitation of our approach. First, we show the efficiency of the proposed reputation measurement approach. Then, we study the proposed prevention scheme on the false positive probability and success probability. Finally, the limitation is discussed.

5.1 Efficiency of measurement approach

As described above in Section 3, we use the CUSUM and PCC to solve the reputation measurement problem. For the proposed measurement approach, to actually measure the reputation of each Web service, the feedback ratings associated with the Web service will be computed by detecting malicious feedback ratings and adjusting the subjective feedback ratings.

In the context of the malicious feedback rating detection, for each Web service, the CUSUM monitors a set of n feedback rating sample intervals. Each sample interval is assigned a score $z(y_i)$, i.e., $z(y_i) = \frac{\mu_1-\mu_0}{\sigma^2}(y_i - \frac{\mu_1+\mu_0}{2})$. When a sample y_i is available, we update the CUSUM f_i as follows:

$$f_i = max(f_{i-1} + z(y_i), 0). \tag{12}$$

If $f_i \geq h$, then take action where $h > 0$ is the pre-specified CUSUM threshold. Note that if a sample i follows malicious feedback ratings, then the expected score $E(z(y_i))$ should be positive so that f_i will eventually rise above threshold h. Moreover, $E(z(y_i))$ should be negative when the samples follow benign feedback ratings. We justify the choice of $z(y_i)$ in Section 3.1.

The CUSUM is adequate for identifying any abrupt change of benign feedback rating traffic to malicious feedback rating traffic. To understand this, note that if these feedback ratings are benign, $z(y_i)$ is negative (in the expected sense) and the corresponding f_i will stay around the zero value, regardless of how long the benign feedback rating traffic has been observed. However, when the benign feedback rating traffic turns to malicious feedback rating traffic, f_i increases and eventually surpasses the threshold h. Hence, the CUSUM prevents a malicious user from suppressing f_i with a long history of benign feedback rating traffic. This ensures that the CUSUM detects malicious feedback ratings in a timely manner. Of course, it is not a good approach when the number of feedback ratings is very little or none, e.g., for newly deployed Web services.

In the context of the feedback rating adjustment, the PCC is used to shield the influence of different preferences of users and adjust their subjective feedback ratings. The PCC adopts Eq.14 to obtain a set of similar users. Then the feedback similarity between two users can be computed by Eq.16. Based on the feedback similarity values, the feedback ratings with different preferences could be adjusted. Finally, the accurate reputation score can be computed. Hence, based on the above analysis, our proposed approach can measure the reputation of each Web service accurately and efficiently even when malicious and subjective feedback ratings fill the Web service environment.

5.2 False positive probability of prevention scheme

From [26], we can prove that the proposed prevention scheme can block malicious feedback ratings with a very small false positive probability (all the k bits are found to be 1 but the IP with a malicious feedback rating is not a member of S) by setting $k = \ln 2m/n$. Hence, the false positive probability of the prevention scheme is very small. For example, when $k = 10$, the minimal $f \approx 0.1\%$. Hence, our proposed prevention scheme can block malicious feedback rating with a very small false positive probability, which indirectly improves the accuracy of reputation measurement for each Web service.

Moreover, the Bloom Filter of our approach stores each IP address with hash function. When malicious attack is found in the *i*-the sample interval, we only discard the IP addresses of the sample interval. Because the number of feedback ratings within a sample intervals is very few, if the discarded feedback rating is good, the influence is also very limited for reputation measurement.

5.3 Success probability of prevention scheme

From [26], a malicious feedback rating over a Web service can be blocked with high probability by querying whether its IP is a member of S, assuming that the RSB could efficiently run. Hence, the proposed prevention scheme can block malicious feedback ratings with high success probability.

Moreover, it can support different development environments of reputation systems with special false positive probability constraints. The Bloom filter guarantees no false negative, and in an ideal case, the success probability could reach 100%. But the proposed prevention scheme can not block malicious feedback rating with 100% probability because of these existing factors such as dynamic IP addresses, the low intensity of malicious feedback ratings and so on. Hence, the validation demonstrated that our proposed prevention scheme can block the malicious feedback ratings with very high probability.

5.4 Limitations of our proposed approach

- The detection scheme of our approach may fail when the intensity of malicious feedback ratings is very low. The higher the intensity of malicious feedback ratings is, the better the detection performance of our proposed approach is.
- The adjustment scheme of our approach is not suitable for a new service or the service used rarely, since the number of feedback ratings and

users are very low. Of course, if there is not a adequate service community with massive services invoked, it cannot also work.

- Note that we propose a malicious rating prevention, but the dynamic IP addresses and distributed rating attack using difference IP addresses cannot be distinguished and blocked in this paper.
- When the performance of a service sudden changes from good/bad to bad/good, if users give very bad/good feedback ratings, this approach will have a false positive (note that if the performance changes from fair to bad/good, this approach is till effective).
- There is a tradeoff between measurement accuracy and computation load. There is a heavy overload because of the complex computation in rating adjustment phase.

6 Performance Evaluation

This section uses experiments to evaluate the guarantees of our proposed approach. We use a real world Web service QoS dataset and a feedback rating dataset in the experiment. We also choose to use simulation to generate feedback ratings because it enables us to study large-scale malicious and subjective feedback ratings of the reputation measurements of Web services in service recommendations.

6.1 Experiment Setup

For the experiments on the deviation, we use an actual feedback rating dataset[1]. The dataset consists of data from a real online dating service (Libimseti) [28]. Overall the dataset contains 194,439 users, who provided 11,767,448 feedback ratings. Ratings are on a 1-10 scale, where ”10” is the best (integer feedback ratings only). Only users who provided at least 20 feedback ratings are included.

It is worth noting that because of the current limited availability of feedback rating data, many existing reputation systems [16,29-30] used simulation data for performance evaluation. In the simulation data, The simulated malicious and subjective feedback can reflect the real situations by setting the magnitude (e.g.,1,2,...,10) of subjective feedback ratings and the density (e.g.,10%,20%,...,100%) of malicious feedback ratings [4, 5, 7, 10, 16]. Hence, in our experiments, we also employ simulation to generate malicious and biased feedback ratings to evaluate the proposed approach, as follows.

Malicious and biased feedback ratings are generated synthetically, which allows us to control the characteristics of the feedback ratings. Hence, to investigate the performance of the reputation measurement for different feedback ratings, we simulated 500 services and 500 users. These users reported their feedback

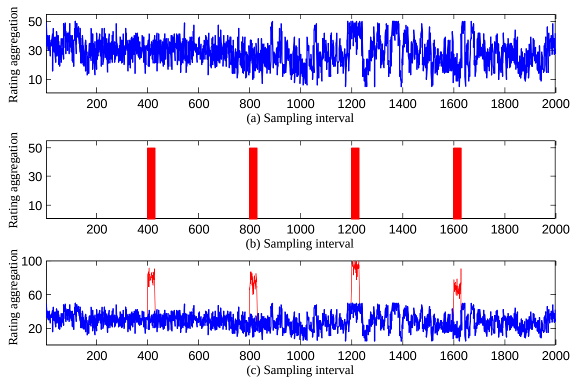


Fig. 4. An example: the synthesis of original feedback ratings and malicious Ratings.

ratings with two types, i.e., biased feedback ratings and malicious feedback ratings. Every feedback rating is also limited to an integer feedback rating from 1 to 10. The malicious feedback ratings contain malicious positive feedback ratings and malicious negative feedback ratings. In order to facilitate experimental comparison with other approaches in a same experimental environment, as shown in Fig. 4, we choose a part of the feedback rating traffic in which a sampling interval (a specific period of time) contains 5 feedback ratings and where feedback rating aggregation (y-axis) denotes the sum of 5 feedback ratings. In Fig. 4(a), the background feedback rating traffic is shown. We believe that there are only a few malicious feedback ratings in the Libimseti dataset because it has no business benefit or benefit conflicts in the network dating site[2]. In Fig. 4(b), only (positive) malicious feedback ratings that are from the simulated malicious users are shown. As shown in Fig. 4(c), the original feedback ratings with malicious feedback ratings are generated synthetically, which allow us to investigate the performance of our approach.

Unless otherwise noted, the parameters for the CUSUM algorithm are set to ($\lambda = 0.5$, $\alpha = 0.7$, and $h = 0.7$). In comparisons, all of the test cases and the runtime environment are the same. Each experimental result is collected as an average after each approach is run 10 times.

We conduct our experimental results from a PC with an Intel Core2 2.0GHz processor and 2.0GB of RAM. The machine is running Windows XP SP3, Matlab 7.6 and Java 1.4.8. We compare our approach with the reputation measurement approaches in [7] and [10], with respect to the deviation of the reputation measurement and the reliability of the composition service. The approach in [7] takes the client, the service, the normalized transaction feedback rating, and the set of optional attributes to create a service invocation history record that is used to measure the reputation. Based on a combination of a perception

1. http://www.occamslab.com/petricek/data/

2. http://libimseti.cz/

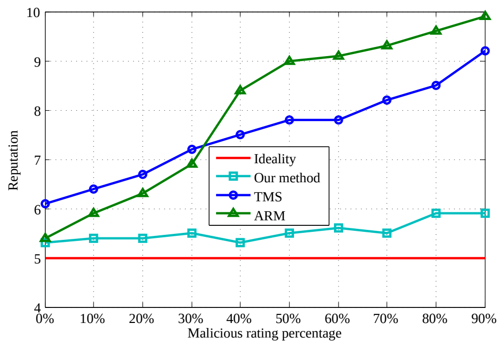


Fig. 5. Positive malicious feedback rating percentages.

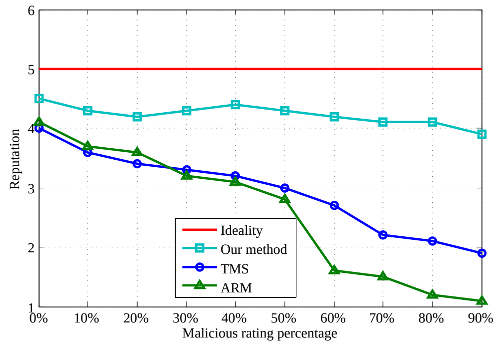


Fig. 6. Negative malicious feedback rating percentages.

function and a disconfirmation function, the approach in [10] designed a feedback rating computation model, and then adopted the simple exponential smoothing approach to compute reputation scores. For illustration purposes, OA represents our approach, TMS represents the approach in [7] and ARM represents the approach in [10].

6.2 Experiment on Deviation

In this experiment, we compare our approach with TMS and ARM with respect to the deviation of reputation measurement under a malicious feedback rating condition and normal feedback ratings conditions.

Definition 1. Deviation: We define the deviation of the reputation measurement for each individual service as the difference between the ideal reputation (all of the feedback ratings are objective, fair, and benign) and the actual reputation (feedback ratings are subjective or malicious).

6.2.1 Malicious feedback ratings

In this experiment, we vary malicious feedback ratings from 0% to 90%, with a step of 10% with 100 random independent services. The 100 services are counted on an abstract service for a more objective measurement. As shown in Figs. 5-6, the measurement results $(\sum_{i=1}^{100} r_i \Big/ 100)$ are collected on average.

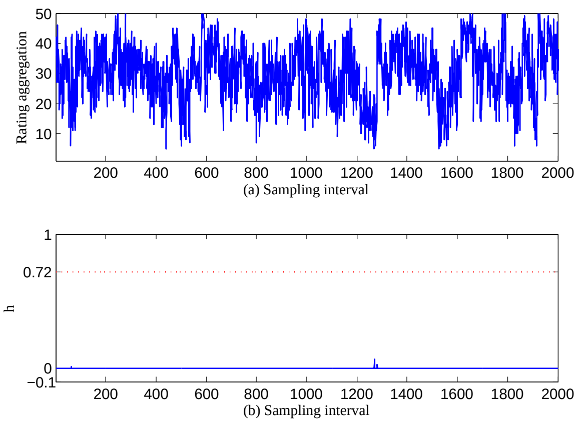


Fig. 7. Benign feedback rating detection. The parameter h represents the threshold value.

In Figs. 5-6, the Ideality line presents the ideal reputation. Although, in reality, it is impossible to obtain an ideal reputation for each service. With the Ideality line, we can effectively evaluate the performance of the three approaches. In other words, the better the approach is, the smaller the deviation is.

From Fig. 5, we can see that the deviation of our approach is 5.53 on average, but the others are 7.54 (TMS) and 7.98 (ARM), respectively. When the positive malicious feedback rating percentage increases, the deviations of TMS and ARM become larger. These relationships exaggerate the actual reputation value of the service and deceive or mislead users. Fortunately, our approach is not sensitive to the positive malicious feedback ratings. With an increasing number of positive malicious feedback ratings, it still has good performance.

From Fig. 6, we can see that the deviation of our approach is 4.23 on average, while the deviations of TMS and ARM are 2.54 and 2.59, respectively. Specifically, when the negative malicious feedback rating percentage is more than 50%, the measured reputations of the TMS and ARM will sharply decrease. Clearly, the measured reputation scores by TMS and ARM are inaccurate, which masks the actual reputation of the service and makes the re-evaluated service fail to compete with existing services for market share. In contrast, our approach still works well despite the existing negative malicious feedback ratings.

In summary, from Figs. 5-6, with different numbers of malicious feedback ratings, the deviation of our approach is much smaller than those of the other approaches.

6.2.2 benign feedback ratings

In the experiment, we apply our approach to the original feedback ratings without adding any malicious attack. The CUSUM algorithm is used to analyze the actual feedback ratings of benign users. Fig. 7 shows a part of the original feedback ratings traces. Fig.7

(a) shows the original feedback ratings, in which a sampling interval contains 5 feedback ratings, and feedback rating aggregation (y-axis) denotes the sum of 5 feedback ratings. Fig. 7(b) shows the results, where all h values are mostly zeros and always much smaller than the threshold. Hence, no false alarms are reported, which demonstrates that our proposed approach does not have any effect on the accuracy of the reputation system under benign conditions.

6.3 Experiment on Optimality

In practical application, an important aim of a reputation measurement is to help service recommendation systems find the optimal services under reputation attribute constraints. However, because of the existence of malicious and biased feedback ratings, the reputation score of a Web service often cannot reflect a service provider's real performance, which prevents users from customizing the best services according to their QoS requirements. Hence, in this section, we compare the optimality of the composition service to further evaluate our approach (The composition service that is one of the primary research issues of Service Computing is a service aggregating smaller and fine-grained services [6,8]).

For experiments on optimality, we use an actual QoS dataset named WS-DREAM[3] from [2]. The WS-DREAM dataset contains approximately 1.5 million Web service invocation records of 150 users in 24 countries. Values of three QoS attributes (i.e., *Response Time, Response Data Size, and Failure Probability*) are collected by these 150 users on 10,258 Web services.

Based on the measured reputation scores of the three approaches, the QoS attributes can be extended into four attributes (Response Time, Response Data Size, Failure Probability, and Reputation). Hence, we can find the best services under reputation attribute constraints with Mixed Integer Programming [31].

In our study, the overall utility [31] is the aggregation of the three QoS attributes (Response Time, Response Data Size, and Failure Probability) of the composition service under reputation constraints. To facilitate a comparison, we select "RUX" to represent the overall utility under reputation constrains, where the reputation scores are measured using our approach. Similarly, "RUY" and "RUZ" represent the overall utility, for which the reputation scores are measured using TMS and ARM, respectively. We use "OUT" to represent the optimal result of the composition service; in other words, the overall utility is only the aggregation of the three QoS attributes (Response Time, Response Data Size, and Failure Probability).

Definition 2. Optimality: We define the optimality of the composition service as the ratio of the overall utility and the optimality result with the following:

$$optimality = 100\% \times RUi/OUT, i = X, Y, Z, \quad (13)$$

3. http://www.wsdream.net

where the better the approach, the larger the optimality.

The optimality results of the three approaches are shown in Table 2. The number of QoS attributes is set to 4, and the number of QoS constraints is set to 1. The number of service candidates per service class is from 10 to 50, with steps of 10, and the number of service classes is set to 5. We vary the number of similar users (K) from 2 to 10, with steps of 2.

From Table 2, we can see that with the different number of similar users, the optimality of OA is the largest. Its optimality is 91.4% on average, while those of TMS and ARM are only 72.4% and 72.1%, respectively. Compared with TMS, most results of OA are larger than 90%, while all of the results of TMS are smaller than 90%. Compared with ARM, the results of OA are more significant. Hence, the performance of OA is the best among the three approaches. Thus, with OA, the service selection algorithm can obtain the optimal services. As a result, our approach can significantly improve the performance of the service selection for the service composition system in open service environments.

TABLE 2
The optimality of the composition service. The parameter K in the table represents the number of similar users.

K	Method	The number of service candidates				
		10	20	30	40	50
2	OA	**88.2%**	**89.5%**	**89.3%**	**84.2%**	**85.9%**
	TMS	79.2%	77.5%	72.6%	69.8%	69.9%
	ARM	72.5%	70.5%	70.4%	72.7%	77.4%
4	OA	**90.9%**	**91.4%**	**89.7%**	**90.8%**	**91.0%**
	TMS	68.8%	66.9%	68.3%	65.6%	63.5%
	ARM	74.7%	74.8%	73.4%	71.9%	72.4%
6	OA	**91.6%**	**91.0%**	**92.8%**	**89.4%**	**89.9%**
	TMS	77.2%	78.4%	76.3%	71.5%	61.8%
	ARM	80.2%	78.5%	66.7%	72.3%	79.1%
8	OA	**93.4%**	**94.5%**	**91.2%**	**93.0%**	**95.4%**
	TMS	72.9%	72.1%	72.6%	71.5%	65.8%
	ARM	79.3%	79.8%	74.5%	66.7%	69.8%
10	OA	**94.5%**	**91.5%**	**96.5%**	**95.0%**	**93.7%**
	TMS	72.2%	73.8%	61.3%	59.4%	65.9%
	ARM	66.8%	71.4%	64.9%	60.3%	61.7%

6.4 Experiment on success ratio

In a service recommendation system, another important goal is to recommend reliable services for users. However, because of the failure of the reputation measurement schemes, the selected service often deviates from the user's expectations, which may lead to service composition failure in practical applications. Thus, the aim of this experiment is to compare the success ratio of our proposed approach with other approaches, with respect to the number of end-to-end QoS constraints. For this purpose, we fixed the number of service candidates per service class to 100 services, and we varied the number of QoS constraints (*NQC*) from 1 to 3, i.e., *NQC*=1,2,3. Furthermore, we

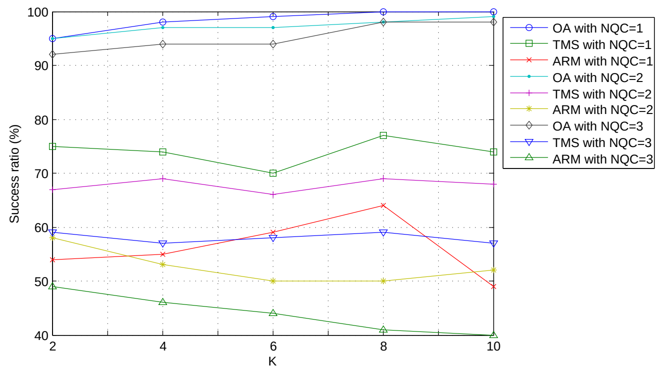


Fig. 8. Comparison of success ratios. The parameter K in the figure represents the number of similar users. The parameter NQC in the figure represents the number of QoS constraints.

also give the definition of the success ratio where the higher the success ratio of one approach is, the better its performance.

Definition 3. Success Ratio (SR) is how often the ratio of users' QoS constraints (C_i) to the monitored aggregated QoS values (U_i) is greater than or equal to 1 for n composition services, i.e., $SR = \frac{srn}{n} \times 100\%$, $srn = \sum_{i=1}^{n} \begin{cases} 1, \bigcap_{i=1}^{l} \frac{C_i}{U_i} \geq 1 \\ 0, otherwise \end{cases}$ where l denotes the number of end-to-end QoS constraints that are negative QoS attributes.

Fig. 8 shows the comparison of the success ratios among the approaches, where the parameter n is set as $n = 100$. With different NQC, the success ratio of our approach is much higher than those of the other two approaches. The overall success ratio of our approach is 96.9% on average, while those of the other two approaches are 66.6% (TMS) and 50.9% (ARM), respectively. These experimental results indicate that our approach effectively reduces the influence of malicious and unfair feedback ratings on the success ratio of composition services.

6.5 Studies on the Parameters

In this section, we study the effect of the parameters of our proposed approach on the optimality and success ratio results. As shown in Fig. 9, the parameters contain the f_n parameter α, the EWMA parameter λ, the CUSUM threshold h, and the number of similar users K. In our experiments, the number of QoS constraints is 1, and the number of service candidates per service class is 30.

6.5.1 Effect of the f_n parameter α

Fig. 9(a) shows the effect of the α of the f_n for our reputation approach. To show its impact clearly, we vary the value of α from 0.1 to 1 with a step value of 0.1. We set $\lambda = 0.7$, $h = 0.7$, and $K = 10$ in the experiment. The figure shows the following: (1) the optimality is significantly reduced when the value of α is increased from 0.7 to 1. This observation indicates that the optimality will be reduced when the most probable percentage of the mean rate after the occurrence of mass malicious feedback ratings has occurred increases; (2) the success ratio is not substantially influenced by the value of the α; and (3) the best performance of the approach is for values of α in the interval [0.4, 0.7].

6.5.2 Effect of the EWMA parameter λ

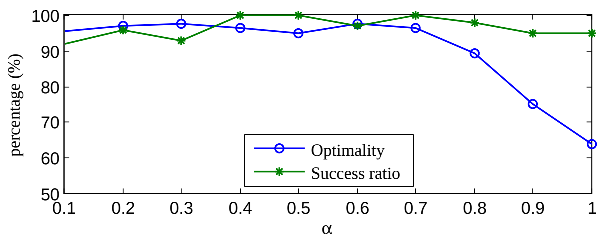
Fig. 9(b) shows the effect of the EWMA parameter λ for our reputation measurement approach. To show its impact clearly, we vary the value of λ from 0.1 to 1 with a step value of 0.1. We set $\alpha = 0.5$, $h = 0.7$, and $K = 10$ in the experiment. Similar to before, the figure is obtained by taking the average of 10 runs. The figure shows the following: (1) the optimality is increased when the value of λ is increased from 0.1 to 0.5. However, it is significantly reduced when the value of λ is increased from 0.6 to 1; (2) the success ratio is gradually increased at the early stage and is steady at the late stage. This observation indicates that the success ratio will be steady when the current feedback ratings play a larger role than the historic feedback ratings; (3) the success ratio is not substantially influenced by the value of λ in the interval [0.4, 1]; and (4) the best performance of the approach is for values of λ in the interval [0.4, 0.7].

6.5.3 Effect of the CUSUM threshold h

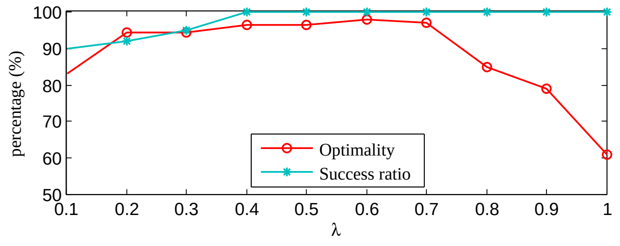
Fig. 9(c) shows the effect of the CUSUM threshold h of our reputation approach where we vary the value of h from 0.1 to 1 with a step value of 0.1. We set $\alpha = 0.5$, $\lambda = 0.7$, and $K = 10$ in the experiment. The figure is obtained by taking the average of 10 runs. Fig. 9(c) shows the following: (1) the optimality is increased when the value of h is increased from 0.1 to 0.6. However, it is significantly reduced when the value of h is increased from 0.7 to 1; (2) the success ratio is steady at the early stage and is reduced when the value of h changes from 0.9 to 1. This observation indicates that the success ratio will be reduced because, with the increasing threshold value, the approach cannot filter malicious feedback ratings effectively; and (3) the best performance of the approach is for values of h in the interval [0.4, 0.8].

6.5.4 Effect of the PCC parameter K

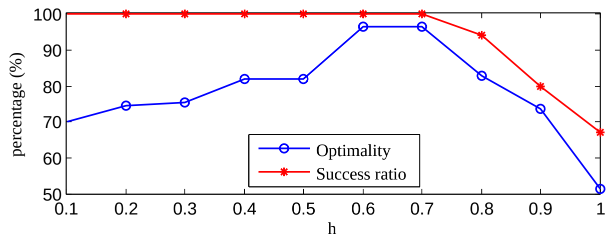
Fig. 9(d) shows the effect of the PCC parameter K in which the measurement experiments are taken, which vary the value of K from 2 to 10 with a step value of 2. We set $\alpha = 0.5$, $\lambda = 0.7$, and $h = 0.7$ in the experiment. Fig. 9(d) shows that the optimality and the success ratio are increased when the value of K is increased from 2 to 10. Although the amplitude is not very large,



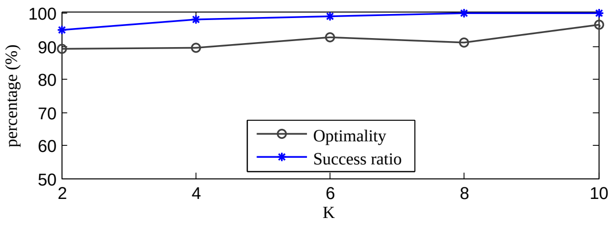
(a) Effect of the f_n parameter α



(b) Effect of the EWMA parameter λ



(c) Effect of the CUSUM threshold h



(d) Effect of the PCC parameter K

Fig. 9. Effect of the parameters. The parameter α represents the increasing mean rate of the change feedback rating traffic after a malicious attack. The parameter λ is a constant that determines the depth of memory of the EWMA, i.e., it determines the rate at which "older" data enter into the calculation of the EWMA. The parameter h is the alarm threshold for detecting malicious attacks. The parameter K represents the number of similar users that uses the same Web service.

this observation indicates that the higher the value of K is, the better the performance of the approach is, i.e., the more objective the reputation score is.

7 Conclusion

The proposed reputation measurement approach utilizes malicious feedback rating detection and feedback similarity computation to measure the reputation of Web services. The efficiency of our proposed approach is evaluated and validated by the theoretical analysis and extensive experiments. The experimental results show that our proposed approach can accomplish a trustworthy reputation measurement of Web services and greatly improve the service recommendation process. The proposed prevention scheme can identify the IP addresses with the offending feedback ratings and block them using a standard Bloom filter. The theoretical analysis indicates the efficiency of the proposed prevention scheme in blocking malicious feedback ratings within the Web service recommendation system.

Our on-going research includes investigating the parameters of sampling interval according to the number of feedback ratings, the number of sampling, duration and storage space, and constructing a common malicious feedback rating prevention scheme for Web service recommendation systems.

Acknowledgments

The work is supported by the NSFC(61202435); the Beijing Natural Science Foundation (4132048); and the Fundamental Research Funds for the Central Universities.

References

[1] X. Chen, X. Liu, Z. Huang, and H. Sun. RegionKNN: A scalable hybrid collaborative filtering algorithm for personalized web service recommendation, *In Proceedings of the 8th IEEE International Conference on Web Services (ICWS'10)*, pages 9-16, 2010.

[2] Z. Zheng and M. R. Lyu. Collaborative reliability prediction of service-oriented systems. *In Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering (ICSE'10)*, pages 35-44, 2010.

[3] E. M. Maximilien and M. P. Singh. Conceptual model of web service reputation. *SIGMOD Record*: 31(4): 36-41, 2002.

[4] Z. Malik and A. Bouguettaya. Evaluating rater credibility for reputation assessment of web services. *In Proceedings of the 8th International Conference on Web Information Systems Engineering (WISE'07)*, pages 38-49, 2007.

[5] Z. Xu, P. Martin, W. Powley, and F. Zulkernine. Reputation-enhanced QoS-based web services discovery. *In Proceedings of the IEEE International Conference on Web Services (ICWS'07)*, pages 249-256, 2007.

[6] D. Ardagna and B. Pernici. Adaptive service composition in flexible processes. *IEEE Transactions on Software Engineering*, 33(6): 369-384, 2007.

[7] W. Conner, A. Iyengar, T. Mikalsen, I. Rouvellou, and K. Nahrstedt. A trust management framework for service-oriented environments. *In Proceedings of the 18th international conference on World Wide Web (WWW'09)*, pages 891-900, 2009.

[8] S. Nepal, Z. Malik, and A. Bouguettaya. Reputation Propagation in Composite Services. *In Proceedings of the IEEE International Conference on Web Services (ICWS'09)*, pages 295-302, 2009.

[9] R. Jurca, B. Faltings, and W. Binder. Reliable QoS monitoring based on client feedback. *In Proceedings of the 16th international conference on World Wide Web (WWW'07)*, pages 1003-1012, 2007.

[10] N. Limam and R. Boutaba. Assessing Software Service Quality and Trustworthiness at Selection Time. *IEEE Transactions on Software Engineering*, 36(4): 559-574, 2010.

[11] J. R. Douceur. The Sybil Attack. *In Proceedings of the First International Workshop on Peer-to-Peer Systems (IPTPS'01)*, pages 251-260, 2002.

[12] F. Li, F. Yang, K. Shuang, and S. Su. A Policy-Driven Distributed Framework for Monitoring Quality of Web Services.

In Proceedings of the IEEE International Conference on Web Services (ICWS'08), pages 708-715, 2008.

[13] S. Wang, Z. Zheng, Q. Sun, H. Zou, and F. Yang. Evaluating feedback ratings for measuring reputation of web services. *In Proceedings of the IEEE International Conference on Services Computing (SCC'11)*, pages 192-199, 2011.

[14] Z. Wang and J. Cao. Committee-based Evaluation and Selection of Grid Resources for QoS Improvement. *In Proceedings of the 10th IEEE/ACM International Conference on Grid Computing (Grid'09)*, pages 138-144, 2009.

[15] R. Zhou, K. Hwang. PowerTrust: A Robust and Scalable Reputation System for Trusted Peer-to-Peer Computing. *IEEE Transactions on Parallel and Distributed Systems*, 2007, 18(4): 460-473.

[16] X. Li and L. Ling. PeerTrust: supporting reputation-based trust for peer-to-peer electronic communities. *IEEE Transactions on Knowledge and Data Engineering*, 16(7): 843-857, 2004.

[17] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The Eigentrust algorithm for reputation management in P2P networks. *In Proceedings of the 12th international conference on World Wide Web (WWW'03)*, pages 640-651, 2003.

[18] J. Caverlee, L. Liu, and S. Webb. Socialtrust: tamper-resilient trust establishment in online communities. *In Proceedings of the 8th ACM/IEEE-CS joint conference on Digital libraries (JCDL'08)*, pages 04-14, 2008.

[19] S. Ries and E. Aitenbichler. Limiting Sybil Attacks on Bayesian Trust Models in Open SOA Environments. *In Proceedings of the 2009 Symposia and Workshops on Ubiquitous, Autonomic and Trusted Computing (UIC-ATC'09)*, pages 178-183, 2009.

[20] C. Dellarocas. Immunizing online reputation reporting systems against unfair ratings and discriminatory behavior. *In Proceedings of the 2th ACM conference on Electronic Commerce (EC'00)*, pages 150-157, 2000.

[21] V. A. Siris and F. Papagalou. Application of anomaly detection algorithms for detecting SYN flooding attacks. *Computer Communications*, 29(9): 1433-1442, 2006.

[22] R. Radharamanan, A. Galelli, D. T. Alex, and L. L. Perez. Sensitivity analysis on the CUSUM method. *International Journal of Production Economics*, 33(1): 89-95, 1994.

[23] Z. Yanzhen, G. Liang, L. Ge, X. Bing, and M. Hong. Rectifying prejudicial feedback ratings in reputation based trust management. *In Proceedings of the IEEE International Conference on Services Computing (SCC'07)*, pages 530-535, 2007.

[24] J. L. Hellerstein, F. Zhang, and P. Shahabuddin. A statistical approach to predictive detection. *Computer Networks*, 35(1): 77-95, 2001.

[25] U. Shardanand and P. Maes. Social information filtering: algorithms for automating 'word of mouth'. *In Proceedings of the Conference on Human Factors in Computing Systems (CHI'95)*, pages 210-217, 1995.

[26] Bloom BH. Space/Time Trade-offs In Hash Coding with Allowable Errors.*Communications of the ACM*,13(7):422-426, 1970.

[27] Dharmapurikar S, Krishnamurthy P, Taylor DE. Longest prefix matching using bloom filters. *IEEE/ACM Transactions on Networking*, 14(2):397-409, 2006.

[28] L. Brozovsky and V. Petricek. Recommender System for Online Dating Service. *In Proceedings of the 6th Conference Zalosti (Zalosti'07)*, pages 1-12, 2007.

[29] I. Maarouf, U. Baroudi, and A. R. Naseer. Efficient monitoring approach for reputation system-based trust-aware routing in wireless sensor networks. *IET Communications*, 3(5): 846-858, 2009.

[30] Z. Malik and A. Bouguettaya. Reputation Bootstrapping for Trust Establishment among Web Services. *IEEE Internet Computing*, 13(1): 40-47, 2009.

[31] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang. QoS-aware middleware for Web services composition. *IEEE Transactions on Software Engineering*, 30(5): 311-327, 2004.

Shangguang Wang is an associate professor at Beijing University of Posts and Telecommunications. He received his Ph.D. degree at Beijing University of Posts and Telecommunications in 2011. He has served as reviewers for numerous journals, including IEEE Transactions on Parallel and Distributed System, IEEE Transactions on Service Computing, etc. He served as PC member of IEEE SCC, BigData, etc. His research interests include service computing and cloud services.

Zibin Zheng is an associate research fellow at Shenzhen Research Institute, The Chinese University of Hong Kong. He received Outstanding Ph.D. Thesis Award of The Chinese University of Hong Kong at 2012, ACM SIGSOFT Distinguished Paper Award at ICSE2010, Best Student Paper Award at ICWS2010, and IBM Ph.D. Fellowship Award at 2010. He served as PC member of IEEE CLOUD, ICWS, SCC, ICSOC, SOSE, etc. His research interests include service computing and cloud computing.

Zhengping Wu is an assistant professor of Computer Science and Engineering at the University of Bridgeport. He received his Ph.D. in Computer Science from the University of Virginia in 2008. His research interests include services computing, network security, and distributed systems.

Fangchun Yang received his PhD degree in communication and electronic system from the Beijing University of Posts and Telecommunication in 1990. He is currently a professor at the Beijing University of Posts and Telecommunication, China. His current research interests include network intelligence and services computing. He is a fellow of the IET.

Michael R. Lyu is currently a Professor in the Department of Computer Science and Engineering, The Chinese University of Hong Kong. His research interests include software reliability engineering, distributed systems, service computing, information retrieval, social networks, and machine learning. He has published over 400 refereed journal and conference papers in these areas. Dr. Lyu is an IEEE Fellow and an AAAS Fellow for his contributions to software reliability engineering and software fault tolerance.