Research Article
Privacy-Preserving and Scalable Service Recommendation Based on SimHash in a Distributed Cloud Environment

Yanwei Xu, Lianyong Qi, Wanchun Dou, and Jiguo Yu

1School of Information Science and Engineering, Chinese Academy of Education Big Data, Qufu Normal University, Rizhao 276826, China
2State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China

Correspondence should be addressed to Lianyong Qi; lianyongqi@gmail.com

Received 4 August 2017; Revised 9 November 2017; Accepted 19 November 2017; Published 18 December 2017

Academic Editor: Chuan Zhou

Copyright © 2017 Yanwei Xu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

With the increasing volume of web services in the cloud environment, Collaborative Filtering-(CF-) based service recommendation has become one of the most effective techniques to alleviate the heavy burden on the service selection decisions of a target user. However, the service recommendation bases, that is, historical service usage data, are often distributed in different cloud platforms. Two challenges are present in such a cross-cloud service recommendation scenario. First, a cloud platform is often not willing to share its data to other cloud platforms due to privacy concerns, which decreases the feasibility of cross-cloud service recommendation severely. Second, the historical service usage data recorded in each cloud platform may update over time, which reduces the recommendation scalability significantly. In view of these two challenges, a novel privacy-preserving and scalable service recommendation approach based on SimHash, named SerRecSimHash, is proposed in this paper. Finally, through a set of experiments deployed on a real distributed service quality dataset WS-DREAM, we validate the feasibility of our proposal in terms of recommendation accuracy and efficiency while guaranteeing privacy-preservation.

1. Introduction

With the ever-increasing volume and variety of web services in various web-based communities, it becomes a challenging task to find the web services that a target user is really interested in [1–3]. In this situation, various service recommendation techniques are introduced to alleviate the heavy burden on the service selection decisions of target users, for example, the well-adopted user-based Collaborative Filtering (i.e., UCF). According to traditional UCF, the similar friends of a target user are often employed to make recommendations to the target user [4]. Therefore, similar friend discovery is the key step to the subsequent service recommendation.

Generally, the bases for similar friend discovery, that is, historical service usage data (e.g., service quality observed by users) are centralized; in this situation, it is easy to determine the similar friends of a target user. However, in the age of IoT (Internet of Things), the quality data of various services are often monitored and collected by geographically distributed sensors and stored in different cloud platforms [5]. In this situation, the historical service usage data are not centralized, but distributed. Such a distributed service recommendation scenario calls for data sharing and collaboration between different cloud platforms. However, as work [6] indicates, this kind of cross-platform data sharing may bring additional privacy leakage risk, which decreases the feasibility of cross-cloud service recommendation severely. Besides, for the involved multiple cloud platforms, their volume of service quality data may become increasingly huge with updates over time, which leads to a frequent recalculation of user similarity and hence reduces the recommendation scalability significantly.

In view of these two challenges, a novel privacy-preserving and scalable service recommendation approach based on SimHash, named SerRecSimHash, is put forward in this paper. Our SerRecSimHash can achieve a good recommendation performance in terms of accuracy, efficiency, and privacy-preservation.
Generally, the contributions of this paper are threefold:

(1) To the best of our knowledge, existing research works seldom consider the service recommendation in a distributed cloud environment, as well as the resulting privacy-preservation problems. In this paper, we formalize this privacy-preserving service recommendation problem and clarify its research significance.

(2) We put forward a novel service recommendation approach based on offline SimHash technique [7], named SerRecSimHash, to protect the private information of most users in different cloud platforms, and meanwhile improve the service recommendation efficiency and scalability.

(3) We conduct a set of experiments based on a real distributed service quality dataset WS-DREAM to validate the feasibility of our proposed SerRecSimHash approach. Experiment results show that SerRecSimHash achieves a good performance in terms of recommendation accuracy and scalability while guaranteeing privacy-preservation.

The rest of the paper is organized as follows. Related work is presented in Section 2. Research motivation is demonstrated in Section 3. In Section 4, we introduce the details of our proposed service recommendation approach SerRecSimHash. In Section 5, a set of experiments are conducted based on WS-DREAM dataset, to validate the feasibility and advantages of our proposal. And finally, in Section 6, we summarize the paper and point out the future research directions.

2. Related Work

Collaborative Filtering (i.e., CF) has become one of the most effective techniques in various recommender systems. User-based CF and item-based CF are brought forth for high-quality service recommendation in [4] and [8], respectively. In order to combine their advantages, a hybrid CF recommendation approach is introduced in [9]. Experiment results show that the hybrid approach improves the recommendation performance. As the quality of a web service often depends on the service execution context (e.g., time, location), time-aware CF and location-aware CF are proposed in [10] and [11], respectively, to improve the accuracy of recommended results. However, the above approaches cannot handle the recommendation problems where historical service usage data are very sparse. In view of this drawback, a belief propagation-based approach is proposed in [12], to find the potential friends of the target user.

However, the above approaches all assume that the service recommendation bases, that is, historical service usage data, are centralized, without considering the distributed service recommendation scenarios as well as the resulting privacy leakage risk. In view of this drawback, the authors in [13] suggest that a user should release only a small portion of his/her observed service quality data to the public so that the remaining majority of user-service quality data are secure. However, the released small portion of data can still reveal part of a user’s private information. In order to protect user privacy completely, the data obfuscation technique is adopted in [14] to hide the real service quality data by adding an obfuscated data item. However, as the service quality data used to make service recommendations have been obfuscated, the recommendation accuracy is decreased accordingly; besides, additional time cost is brought by the adopted data obfuscation operation. Similarly, a segment-based data hiding approach is introduced in [15], where each piece of user-service quality data is divided into several data segments, and then the data segments are employed to calculate user similarity approximately and make further service recommendation. However, there are still two shortcomings in this approach. First, the data segmentation process often takes much time, which decreases the recommendation efficiency heavily. Second, it fails to protect some important privacy information appropriately, for example, the information of the service intersection commonly invoked by two users. Locality-sensitive hashing technique is recruited in [16] to protect and realize the privacy-preservation purpose; however, only partial private information of users can be protected very well.

In view of the drawbacks of existing approaches, a novel privacy-preserving and scalable service recommendation approach based on SimHash, that is, SerRecSimHash, is proposed in this paper, to cope with the service recommendation problems in the distributed cloud environment. Next, an example is presented in Section 3 to further demonstrate the research motivation of our paper.

3. Research Motivation

An intuitive example is presented in Figure 1 to motivate our paper. Here, \( u_{\text{target}} \) denotes a target user to whom Amazon platform intends to recommend services; \( u_1 \) and \( u_2 \) are two users whose observed service quality data are recorded in Microsoft and IBM platforms, respectively; \( \{w_{s_1}, \ldots, w_{s_n}\} \) are the candidate services for recommendation. Specifically, if a
user has never invoked a service, the corresponding service quality data is null.

Next, according to traditional UCF, the first step is to calculate user similarity \( \text{sim}(u, v) \) as to determine the similar friends of \( u \). However, the above user similarity calculation process involves the cross-platform collaborations and hence faces the following two challenges:

1. Generally, Microsoft and IBM are not willing to share their recorded service quality data to Amazon due to privacy concerns, which decreases the feasibility of cross-cloud user similarity calculation and subsequent service recommendation severely.
2. In Amazon, Microsoft, and IBM, the volume of service quality data may become increasingly huge with updates over time; in this situation, the collaboration efficiency and scalability are often reduced significantly and hence cannot satisfy the quick recommendation requirements from target users.

In view of these two challenges, a privacy-preserving and scalable service recommendation approach, that is, SerRecSimHash, is proposed in this paper, which will be introduced in detail in the next section.

4. A SimHash-Based Service Recommendation Approach

In this section, a privacy-preserving and scalable approach, that is, SerRecSimHash, is proposed to handle the distributed service recommendation problems. The main idea behind SerRecSimHash is: the users who have invoked the most common services can be regarded as “probably similar” friends [17]; therefore, we first utilize SimHash to look for a small number of “probably similar” friends of the target user, in a privacy-preserving and scalable way; afterwards, we determine the target user’s “really similar” friends from the “probably similar” ones; finally, we make service recommendations to the target user based on the preferences of his/her “really similar” friends.

Concretely, SerRecSimHash consists of the four steps in Box 1. Here, \( u_{\text{target}} \) denotes a target user, \( U \) is the user set in multiple involved cloud platforms, \( (w_{s1}, \ldots, w_{s_n}) \) is the candidate service set, and \( H(u) \) denotes the hash value of user \( u \) on SimHash.

Step 1 (building user indexes offline based on SimHash). For each user \( u \in U \), calculate his/her hash value \( H(u) \) offline based on SimHash. Then \( H(u) \) is regarded as the index for \( u \).

Step 2 (finding “probably similar” friends of the target user). According to the same hash function adopted in Step 1, calculate user index for \( u_{\text{target}} \), that is, \( H(u_{\text{target}}) \). If the Hamming Distance between \( H(u_{\text{target}}) \) and \( H(u) \) is smaller than 3, then \( u \) is considered as a “probably similar” friend of \( u_{\text{target}} \).

Step 3 (finding “really similar” friends of the target user). For a “probably similar” friend \( u_i \) obtained in Step 2, calculate his/her similarity with \( u_{\text{target}} \); if the similarity is larger than a threshold \( P \), then \( u_i \) is a “really similar” friend of \( u_{\text{target}} \).

Step 4 (service recommendation). According to \( u_{\text{target}} \& \) “really similar” friends derived in Step 3, predict the quality of services never invoked by \( u_{\text{target}} \) and recommend the quality-optimal services to \( u_{\text{target}} \).

Box 1: Four steps of service recommendation approach SerRecSimHash.
reduce the time cost. Besides, through SimHash, each user \( u_i \) is encapsulated into a less-sensitive user index \( H(u_i) \), without revealing his/her sensitive information (e.g., whether he/she has invited a service or not, a service’s running quality observed by him/her) to other platforms. Therefore, user privacy is protected.

**Step 2** (finding “probably similar” friends of the target user). According to same hash functions adopted in Step 1, we calculate the index for the target user, that is, \( H(u_{\text{target}}) \). Next, we calculate the Hamming Distance between \( H(u_{\text{target}}) \) and \( H(u_i) \) (\( 1 \leq i \leq m \)), denoted by \( D(H(u_{\text{target}}), H(u_i)) \). Concretely, suppose \( H(u_{\text{target}}) \) and \( H(u_i) \) are denoted by \( r \)-dimensional vectors \((v_{\text{target}-1}, \ldots, v_{\text{target}-r})\) and \((v_{i-1}, \ldots, v_{i-r})\), respectively. Then \( D(H(u_{\text{target}}), H(u_i)) \) can be calculated by (2), where \( a_k \) is a Boolean value calculated by (3). Here, symbol “\( \oplus \)” denotes the XOR operation.

\[
D(H(u_{\text{target}}), H(u_i)) = \sum a_k \quad (1 \leq k \leq r) \tag{2}
\]

\[
a_k = v_{\text{target}-k} \oplus v_{i-k} = \begin{cases} 
1 & \text{if } v_{\text{target}-k} \neq v_{i-k} \\
0 & \text{if } v_{\text{target}-k} = v_{i-k} \end{cases} \tag{3}
\]

According to SimHash [6], if \( D(H(u_{\text{target}}), H(u_i)) \) < 3 holds, then we can conclude that the services invoked by \( u_{\text{target}} \) and \( u_i \) are approximately the same. In other words, \( u_i \) can be regarded as a “probably similar” friend of \( u_{\text{target}} \) and then put into set Prob_Sim_Friend(\( u_{\text{target}} \)). Moreover, the size of \( \text{Prob}_\text{Sim}_\text{Friend}(u_{\text{target}}) \), that is, \( |\text{Prob}_\text{Sim}_\text{Friend}(u_{\text{target}})| \), is often small (\( \ll m \)) due to the nature of SimHash.

**Step 3** (finding “really similar” friends of the target user). The users in set \( \text{Prob}_\text{Sim}_\text{Friend}(u_{\text{target}}) \) (obtained in Step 2) are only “probably similar” friends of the target user, not necessarily “really similar” friends. Considering this point, in this step, we further determine the “really similar” friends of the target user from set \( \text{Prob}_\text{Sim}_\text{Friend}(u_{\text{target}}) \). Concretely, for any \( u_i \in \text{Prob}_\text{Sim}_\text{Friend}(u_{\text{target}}) \), we calculate his/her similarity with \( u_{\text{target}} \), that is, \( \text{Sim}(u_{\text{target}}, u_i) \), based on Pearson Correlation Coefficient (PCC) [18] in (4) (as \( |\text{Prob}_\text{Sim}_\text{Friend}(u_{\text{target}})| \) is often small, only a small number of users take part in the user similarity calculation process in (4); as a consequence, we can protect the private service quality data observed by the remaining majority of users).

In (4), symbol \( I \) denotes the service intersection invoked by \( u_{\text{target}} \) and \( u_i \); \( q \) is a quality dimension of web services, for example, response time; \( q_{\text{target}-j} \) and \( q_{i-j} \) represent service \( ws_j \)’s quality values over dimension \( q \) observed by \( u_{\text{target}} \) and \( u_i \), respectively; \( \overline{q}_{\text{target}} \) and \( \overline{q}_i \) denote \( u_{\text{target}} \)'s and \( u_i \)'s average quality values over dimension \( q \) of all the services invoked by \( u_{\text{target}} \) and \( u_i \), respectively. Specifically, if the service intersection \( I = \text{Null} \), \( \text{Sim}(u_{\text{target}}, u_i) = 0 \) holds. Moreover, if condition in (5) holds, \( u_i \) can be regarded as a “really similar” friend of \( u_{\text{target}} \) and put into set Real_Sim_Friend(\( u_{\text{target}} \)). Here, symbol \( P \) is a predefined similarity threshold (0.5 ≤ \( P \leq 1 \)).

\[
\text{Sim}(u_{\text{target}}, u_i) = \frac{\sum_{ws_j \in I} (q_{\text{target}-j} - \overline{q}_{\text{target}}) \times (q_{i-j} - \overline{q}_i)}{\sqrt{\sum_{ws_j \in I} (q_{\text{target}-j} - \overline{q}_{\text{target}})^2} \times \sqrt{\sum_{ws_j \in I} (q_{i-j} - \overline{q}_i)^2}} \tag{4}
\]

\[
\text{Sim}(u_{\text{target}}, u_i) \geq P \tag{5}
\]

**Step 4** (service recommendation). For all the users \( u_i \) in set \( \text{Real}_\text{Sim}_\text{Friend}(u_{\text{target}}) \) (obtained in Step 3), we rank them by \( \text{Sim}(u_{\text{target}}, u_i) \) (see (4)) in descending order and return the Top 3 (at most) similar friends (denoted by set \( U_{\text{top-3}} \)) of the target user. Afterwards, for each service never invoked by the target user, denoted by \( ws_j \), we predict its quality over dimension \( q \) observed by \( u_{\text{target}} \), that is, \( q_{\text{target}-j} \), by (6), where \( u_i \in U_{\text{top-3}} \) and \( q_{i-j} \) represents service \( ws_j \)'s quality value over dimension \( q \) observed by \( u_i \). Finally, we select the service with the optimal predicted quality and recommend it to the target user, so as to finish the whole service recommendation process.

\[
q_{\text{target}-j} = \frac{\sum_{u_i \in U_{\text{top-3}}} \text{Sim}(u_{\text{target}}, u_i) \times q_{i-j}}{\sum_{u_i \in U_{\text{top-3}}} \text{Sim}(u_{\text{target}}, u_i)} \tag{6}
\]
5. Experiments

5.1. Experiment Configurations. In this section, a set of experiments are deployed on WS-DREAM dataset [19] to validate the feasibility of our proposed recommendation approach SerRecSimHash. WS-DREAM is a real-world service quality (e.g., throughput) set obtained from 339 users on 5825 web services from different countries. To simulate the recommendation scenario that we focus on in this paper (i.e., recommendation in a distributed cloud environment), each country is regarded as a cloud platform.

We compare our approach with a benchmark approach UPCC [20] and another two up-to-date privacy-preserving recommendation approaches, that is, P-UIPCC [14] and PPICF [15]. Many works, for example, [21–23], consider the time cost and the MAE as the evaluation criteria; likewise, we also adopt these two criteria in this paper (in our SerRecSimHash approach, most user privacy information, e.g., whether a user has invoked a service or not and service quality observed by a user, can be protected by the intrinsic nature of SimHash; therefore, we will not evaluate the capability of privacy-preservation of our proposal here).

1. Time cost: the consumed time for recommending a web service to the target user, which can be used to measure the recommendation efficiency and scalability.

2. MAE: the difference between the predicted quality and real quality of recommended services (the smaller the better), which can be used to measure the recommendation accuracy.

The density of user-service quality matrix is set at 3% and the experiments are conducted on a Lenovo laptop with 2.40 GHz processor and 12.0 GB RAM. The laptop is running under Windows 10 and JAVA 8. Each experiment is repeated 10 times and the average experiment results are reported.

5.2. Experiment Results and Analyses. Concretely, the following four profiles are tested and compared, respectively. Here, \( m \) and \( n \) denote the number of users and number of web services, respectively; user similarity threshold \( P = 0.5 \) holds.

Profile 1: Recommendation Efficiency Comparison. In this profile, we test the time cost of our proposal with respect to \( m \) and \( n \) and compare it with the remaining three approaches. The experiment parameters are set as follows: \( m \) is varied from 50 to 300; \( n \) is varied from 1000 to 5000. The concrete experiment results are shown in Figure 3 (\( n = 5000 \) holds in Figure 3(a) and \( m = 300 \) holds in Figure 3(b)).

As shown in Figure 3(a), the time costs of UPCC, P-UIPCC, and PPICF approaches all increase approximately linearly with the growth of \( m \); this is because more time is needed to calculate user similarities when the number of users, that is, \( m \), becomes larger, while our proposed SerRecSimHash approach outperforms those three ones in terms of time cost, as most jobs (e.g., user indexes building) can be finished offline before a service recommendation request arrives. Furthermore, after the hashing process, only a few “probably similar” friends of the target user are obtained; as a consequence, little time is taken to find the “really similar” friends of the target user from the small number of “probably similar” friends. Due to the above two reasons, the recommendation efficiency and scalability of our proposed SerRecSimHash approach are improved significantly. Similar comparison results can be observed from Figure 3(b), whose reasons are the same as those in Figure 3(a) and will not be discussed repeatedly.

Profile 2: Recommendation Accuracy Comparison. Accuracy is a key criterion to evaluate the quality of a recommender system. Therefore, in this profile, we test the MAE (the smaller the better) of our proposal and compare it with the remaining three approaches. The experiment parameters are set as follows: \( m \) is varied from 50 to 300; \( n \) is varied from 1000 to 5000. The experiment results are presented in Figure 4 (\( n = 5000 \) holds in Figure 4(a) and \( m = 150 \) holds in Figure 4(b)).

As Figure 4 shows, the recommendation accuracy values of P-UIPCC and PPICF approaches are often low (i.e., MAE values are high), as many approximate operations are
recruited in these two approaches to protect the user privacy, for example, data obfuscation technique adopted in P/UIPCC approach and data segmentation-merging technique recruited in PPICF approach. These techniques on one hand can protect the privacy information of users effectively and on the other hand decrease the accuracy of recommended results, while our proposed SerRecSimHash approach achieves the approximate service recommendation accuracy as the benchmark approach UPCC, as the SimHash technique adopted in SerRecSimHash can guarantee finding the “really similar” friends of a target user with high probability and thereby can achieve a high recommendation accuracy.

Profile 3: Number of “Probably Similar” Friends of the Target User in SerRecSimHash with respect to \( m \) and \( n \). In our SerRecSimHash approach, a small number of “probably similar” friends (the number is \(|\text{Prob}_\text{Sim}_\text{Friend}(u_{\text{target}})|\)) of a target user are obtained. In this profile, we test the relationship between \(|\text{Prob}_\text{Sim}_\text{Friend}(u_{\text{target}})|\) and \( m \) and \( n \). Experiment parameters are set as follows: \( m \) is varied from 50 to 300; \( n \) is varied from 1000 to 5000. The concrete experiment results are presented in Figure 5.

As Figure 5(a) shows, the value of \(|\text{Prob}_\text{Sim}_\text{Friend}(u_{\text{target}})|\) increases approximately linearly with the growth of \( m \); this is because it is more probable to find a “probable friend” of the target user when the candidate user space becomes larger. As Figure 5(b) shows, the value of \(|\text{Prob}_\text{Sim}_\text{Friend}(u_{\text{target}})|\) increases relatively slowly when \( n \) rises, whose reasons are twofold. First, more valuable recommendation information is available when the number of services, that is, \( n \), increases; as a consequence, more “probably similar” friends of the target user can be found by our proposed SerRecSimHash approach. Second, due to the intrinsic nature of SimHash technique adopted in our SerRecSimHash approach, the number of services, that is, \( n \), does not influence the finding process of “probably similar”
friends directly in our proposal and, hence, the influence of parameter \( n \) stressed on \( |\text{Prob}_\text{Sim\_Friend}(u_{\text{target}})| \) is not so obvious as that in Figure 5(a).

Profile 4: Recommendation Failure Rate of \( \text{SerRecSimHash} \) with respect to \( m \) and \( n \). The SimHash technique adopted in this paper is essentially a kind of probability-based similar neighbor finding approach [24]. Therefore, our proposed \( \text{SerRecSimHash} \) approach may fail to return any recommended neighbors finding approach [24]. Therefore, our proposed approach decreases with the growth of \( m \); this is because it is more probable to find the "probably similar" friends of a target user when the candidate space of users becomes larger. Moreover, the failure rate approaches 0 when \( m \) is large enough, for example, when \( m = 200, 250, \) or 300. Figure 6(b) shows the relationship between failure rate of \( \text{SerRecSimHash} \) and the number of services, that is, \( n \). As indicated in Figure 6(b), the failure rate approximately drops with the growth of \( n \); this is because when the number of services increases, the probability that two users have invoked the common services grows accordingly, and hence it is more probable to find the "probably similar" friends of a target user. Furthermore, as can be seen from Figure 6(b), the failure rate of \( \text{SerRecSimHash} \) approach approaches 0 when \( n \) is large enough, for example, when \( n = 5000 \).

![Figure 6: Recommendation failure rate of \( \text{SerRecSimHash} \) with respect to \( m \) and \( n \).](image)

5.3. Shortcoming Analyses. In terms of the experiment results, we can conclude that \( \text{SerRecSimHash} \) approach achieves a good tradeoff among the recommendation accuracy, efficiency, and failure rate while guaranteeing privacy-preservation. However, other evaluation criteria are not discussed in depth, such as the well-known consistency criterion (e.g., the inferred friend consistency) suggested in work [25]. Besides, as [26] indicates, weight plays an important role in the final evaluation results; however, we do not consider the weight of found friends in this paper for simplicity.

6. Conclusions and Future Work

In the distributed cloud environment, a cloud platform is often not willing to share its recorded user-service invocation data with other cloud platforms due to privacy concerns, which decreases the feasibility of cross-cloud collaborative service recommendation severely. Besides, the user-service invocation data recorded by each cloud platform may update over time, which reduces the recommendation scalability significantly. In view of these two challenges, a novel privacy-preserving and scalable service recommendation approach based on SimHash, that is, \( \text{SerRecSimHash} \), is put forward in this paper. To validate the feasibility of our proposal, we conduct a set of experiments based on a real distributed service quality dataset WS-DREAM. Experiment results show that \( \text{SerRecSimHash} \) outperforms the other up-to-date approaches in terms of recommendation accuracy and efficiency while guaranteeing privacy-preservation.

As work [27] indicates, SimHash is essentially a probability-based search technique and, hence, failure is inevitable in certain situations. Considering this point, in the future, we will continue to refine our proposal so as to further decrease the recommendation failure rate and boost the recommendation robustness. Besides, due to the inherent shortcoming of various hash-based privacy-preservation techniques suggested in [28], it is hard to evaluate the privacy-preservation performance of our proposal. In the future, we hope to find well-adopted technical criteria.
to evaluate the effectiveness of our proposal in terms of privacy-preservation. Moreover, work [29] proposes to utilize the semantic information to improve the retrieval performance; likewise, we hope to refine our work by adding more semantic information in the future.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This paper is partially supported by the Natural Science Foundation of China (no. 61402258, no. 61672276, no. 61373027, and no. 61672321), Key Research and Development Project of Jiangsu Province (no. BE2015154, no. BE2016120), and Open Project of State Key Laboratory for Novel Software Technology (no. KFKT2016B22).

References


Submit your manuscripts at
https://www.hindawi.com