Knowledge-Based Systems 147 (2018) 25–35

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Service recommendation based on quotient space granularity analysis and covering algorithm on Spark

Yi-wen Zhang[a], Yuan-yuan Zhou[a,*], Fu-tian Wang[a], Zheng Sun[b], Qiang He[c,*]

[a] Key Laboratory of Intelligent Computing & Signal Processing, Ministry of Education, Anhui University, Hefei, Anhui, China
[b] School of Science, Arcadia University, Glenside, PA, USA
[c] School of Software and Electrical Engineering, Swinburne University of Technology, Melbourne, Australia

ARTICLE INFO

Article history:
Received 22 July 2017
Revised 5 February 2018
Accepted 6 February 2018
Available online 10 February 2018

Keywords:
Covering algorithm
Quotient space
Recommender systems
Association matrices

ABSTRACT

The rapid growth of Web services has made it a challenge for users to find appropriate Web services because it is very difficult for traditional Web service recommendation approaches to process the large amount of service-relevant data. To address this issue, this paper proposes CA-QGS (Covering Algorithm based on Quotient space Granularity analysis on Spark), a scalable approach for accurate Web service recommendation in large-scale scenarios. CA-QGS first clusters users and Web services based on users' past quality experiences on co-invoked Web services. It then performs granularity analysis on the clustering results to identify users and Web services that are similar to the target user and Web service, and employs the collaborate filtering technique to predict the quality of the target Web service for the target user. This way, appropriate Web services can finally be recommended to the target user. To increase the efficiency of CA-QGS, we parallelize CA-QGS on Spark. Extensive experiments show that CA-QGS outperforms existing approaches in both recommendation accuracy and efficiency.

1. Introduction

Big data analytics is being employed in many areas to improve the quality and value of a variety of services [1]. In the big data environment, more and more services are deployed in the cloud to provide different functionalities [2]. According to Programmable Web, an online Web service repository, the number of published Web services has increased by four times since 2009. The statistics published webservices.seekda.com, a Web service search engine, also indicate an exponential growth in the number of published Web services in the past several years [38]. The rapid increase in the number of Web services, as well as users, has generated a large amount data on users, Web services and users' experiences on Web services. It is essential for researchers to explore and analyze this big data to extract useful information for predicting the quality of Web services and recommending Web services to users. In recent years, the collaborative filtering (CF) technique, such as item-based [21,23–25] and user-based [17,18,22] methods, have been widely employed in the implementation of recommender systems for Web services. Its fundamental theory is to analyze the historical quality values of Web services experienced by similar users to predict the quality of a given Web service for a target user.

Although traditional CF techniques have been successfully applied in many e-commerce recommender systems, they cannot be directly employed in the big data environment for web service recommendation due to three major issues:

1. Data sparsity [3–5]: Inspecting every Web service for its quality value experienced by every user is impractical because it is extremely resource and time consuming. This inevitably leads to high data sparsity in the user-service matrix for Web service recommendation.
2. Scalability [6–9]: Traditional CF techniques are inefficient in processing an extremely user-service matrix for Web service recommendation.
3. Trust [5,31]: There are potentially ratings on quality values provided by untrustworthy users, which decrease the prediction accuracy for Web service recommendation. To address the above issues, researchers have proposed a number of improved CF-based methods.

A native and widely adopted solution to the above issues is to employ clustering techniques to partition users and Web services into clusters so that similar users and Web services can be identified. Then the similar users and Web services are taken by CF techniques to perform quality prediction for Web services. One of the most popular clustering methods is the k-means [10] algorithm.

* Corresponding authors.
E-mail addresses: zhangyiwen@ahu.edu.cn, zywahu@qq.com (Y.-w. Zhang), yyzhouahu@qq.com (Y.-y. Zhou), wft@ahu.edu.cn (F.-t. Wang), zsun@arcadia.edu (Z. Sun), qhe@swin.edu.au (Q. He).

https://doi.org/10.1016/j.knosys.2018.02.014
0950-7051/

The k-means algorithm is an unsupervised method that partitions given data points into a manually chosen number (k) of clusters with one initial center randomly selected for each cluster. However, there are two major inherent limitations to Web service recommendation approaches based on k-means. First, it is difficult to determine a proper k to ensure that the corresponding clustering results can properly indicate the similarity between users as well as Web services. Second, the clustering results heavily rely on the random selection of the initial centers. Thus, k-means based quality prediction approaches cannot ensure the accuracy and stability of the prediction results.

To address the above issues, this paper proposes CA-QGS (Covering Algorithm based on Quotient space Granularity analysis on Spark), a novel approach for Web service recommendation. Given a set of users and Web services, CA-QGS first employs a covering-based clustering algorithm to partition users into clusters based on the similarity between their historical quality experiences on each of the co-invoked Web services through granularity analysis. The granularity analysis visualizes the initial clustering results which help users understand the meanings of within-class and among-class from different perspectives. Following a similar procedure, CA-QGS also partitions Web services into clusters based on each user's quality experiences on Web services. Then, from the final clustering results, CA-QGS chooses a number of users and Web services that are most similar to the target user u and the target Web service s to predict the quality of Web service s for user u. Finally, CA-QGS recommends Web services with the best quality to user u. In order to increase the efficiency of CA-QGS in processing big data, we parallelize the partitioning operation of CA-QGS on Spark.

The major contributions of this paper are:

1. We employ an improved covering-based clustering algorithm to partition users and Web services into clusters respectively based on users' historical quality experiences through granularity analysis to increase the prediction accuracy. This clustering approach does not require the number of clusters to be prespecified or the initial centers to be manually selected.
2. We present a novel approach for rapid identification of users and Web services that are most similar to the target user and the target Web service by implementing descending operations on each row element in the user-service matrix. This novel mechanism can address the issue of user trust.
3. We propose a parallel covering algorithm on the Spark platform. Spark is a distributed compute system that allows an efficient, general-purpose programming language to be used interactively to process large-scale datasets on *resilient distributed datasets* (RDDs). The parallelized covering algorithm significantly improves the efficiency and effectiveness of Web service recommendation in the big data environment.
4. We conduct extensive experiments to comprehensively compare CA-QGS with existing approaches, i.e., traditional collaborative filtering based approaches and k-means-based approaches in prediction accuracy based on a real-world dataset named WS-Dream. The experimental results demonstrate that CA-QGS outperforms those approaches significantly.

This paper is organized as follows. Section 2 overviews the preliminary knowledge. Section 3 discusses the technical details of CA-QGS. Section 4 introduces CA-QAS's procedure for Web service recommendation. Section 5 presents the experimental results and analysis. Section 6 reviews the related work and Section 7 concludes this paper.

2. Preliminary knowledge

In this section, we first formally define four concepts: user association matrix M_u, service association matrix M_s, the set of similar users $A(u)$ and the set of similar services $A(s)$. Then, we give corresponding examples.

Definition 1. (*User association matrix* M_u). M_u is an $m \times m$ matrix used to record the number of times that m users are partitioned into the same clusters.

An element in M_u denoted by $u_{i,j}$, $1 \le i,j \le m$, is the total number of times that user i and user j are partitioned into the same clusters during the n executions of the clustering algorithm, where n is the number of Web services. Element $u_{i,j}$ is updated when user i and user j are partitioned into the same cluster by setting $u_{i,j} = u_{i,j} + 1$ and $u_{j,i} = u_{j,i} + 1$ simultaneously. M_u is illustrated below:

$$M_u = \begin{bmatrix} u11 & u12 & \dots & u1m \\ u21 & u22 & \dots & u2m \\ \dots & \dots & \dots & \dots \\ um1 & um2 & \dots & umm \end{bmatrix} \quad (1)$$

The user similarity matrix is a symmetric matrix, where $M^u = M_u^T$, $u_{i,j} = u_{j,i}$, $1 \le i,j \le m$ and $u_{i,j} = u_{j,i} = 0$ if $i = j$.

Definition 2 (. *Service association matrix* M_s). M_s is an $n \times n$ matrix used to record the number of times that n Web services are partitioned into the same clusters.

An element in M_s denoted by $s_{i,j}$,$1 \le i,j \le n$, is the total number of times that service i and service j are partitioned into the same clusters during the m executions of the clustering algorithm, where m is the number of users. Element $s_{i,j}$ is updated when service i and service j are partitioned into the same cluster by setting $s_{i,j} = s_{i,j} + 1$ and $s_{j,i} = s_{j,i} + 1$ simultaneously. M_s is illustrated below:

$$M_s = \begin{bmatrix} S11 & S12 & \dots & S1m \\ S21 & S22 & \dots & S2m \\ \dots & \dots & \dots & \dots \\ Sn1 & Sn2 & \dots & Snn \end{bmatrix} \quad (2)$$

The service similarity matrix is a symmetric matrix, where $M^s = M_s^T$, $s_{i,j} = s_{j,i}$, $1 \le i, j \le n$ and $s_{i,j} = s_{j,i} = 0$ if $i = j$.

Definition 3. (*Similar users A(u)*). Given a user u, its similar users, denoted by $A(u)$, are defined as the users that have similar quality experiences on a same set of co-invoked Web services.

$A(u)$ is expressed as $A(u_i) = \{u_a | u_{i,a} \ge u_{i,k},\ u_{i,a} > 0,\ 1 \le a \le m,\ u_a \in U,\ u_{i,k} \in M_u\}$, where $U = \{u_1, u_2, \dots, u_m\}$ represents a set of users, $u_{i,k}$ denotes the kth element obtained by the descending operation of the ith row elements in the Mu and the same applies to $u_{i,a}$.

Definition 4. (*Similar Web services A(s)*). Given a Web service s, its similar Web services, denoted by $A(s)$, are defined as the Web services that have delivered similar quality experiences to a same group of users.

$A(s)$ is expressed as $A(s_i) = \{s_b | s_{i,b} > s_{i,k},\ s_{i,b} > 0,\ 1 \le b \le n,\ s_b \in S,\ s_{i,k} \in M_s\}$, where S represents a set of Web services and $S = \{s_1, s_2, \dots, s_n\}$, $s_{i,k}$ denotes the kth element obtained by the descending operation of the ith row elements in the Ms and the same applies to $s_{i,b}$.

Given a user u, a Web service s, a set of users $U = \{u_1, u_2, \dots, u_m\}$ and a set of Web services $S = \{s_1, s_2, \dots, s_n\}$, the procedure for predicting the quality of Web service s consists of two major steps: (1) to identify $A(u)$ from U, and $A(s)$ from S; (2) to predict the quality of Web service s for user u based on $A(u)$ and $A(s)$.

Fig. 1 presents an example with 9 users, $u_1, \dots, u_9$, and 9 Web services, $s_1, \dots, s_9$. The matrix contains the response times of the Web services experienced by the users. Such a matrix is referred to as a *user-service matrix*, denoted by $M = [q_{u,s}]_{m \times n}$, where m is the

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9
u_1		380		?		490	280	480	390
u_2					280	330	470	390	
u_3		380		410	400	460			290
u_4	300		310	420	410				
u_5	430	460	270				440		
u_6		?		320		460	410	440	310
u_7		480		400	250	310			390
u_8	440		430	350	270				
u_9	370	410	380					460	

Fig. 1. A user-service matrix of response time (milliseconds).

number of users and n is the number of Web services. Each element $q_{u,s}$ in the matrix, $1 \le u \le m$, $1 \le s \le n$, is the one-dimensional quality value of Web service s experienced by user u in its invocation(s) of s in the past. Here $q_{u,s}$ can be the quality value experienced in one invocation or the average quality value in multiple invocations. An empty cell $q_{u,s}$ indicates that Web service s has not been invoked by user u and hence its quality is unknown to user u. The set of Web services that user u has invoked before is denoted by $S(u)$ and the set of users that have invoked a Web service s is denoted by $U(s)$. Take the user-service matrix in Fig. 1 for example, there are $S(u_6)=\{s_4, s_6, s_7, s_8, s_9\}$ and $U(s_4)=\{u_3, u_4, u_6, u_7, u_8\}$.

Now, suppose we want to predict $q_{1,4}$, i.e., the quality of Web service s_4 for user u_1. In Fig. 1, we can see that $S(u_6) \cap S(u_1)=\{s_6, s_7, s_8, s_9\}$. This indicates that user u_6 and user u_1 have both invoked Web services s_6, s_7, s_8, s_9. If u_6 and u_1's past quality experiences on s_6, s_7, s_8, s_9 are similar, it is likely that they will also have similar quality experiences on s_4. Now, $q_{6,4}$ is available, i.e., user u_6's quality experience on service s_4. Thus, taking into account the similarity between $\{q_{6,6}, q_{6,7}, q_{6,8}, q_{6,9}\}$ and $\{q_{1,6}, q_{1,7}, q_{1,8}, q_{1,9}\}$, together with $q_{6,4}$, $q_{1,4}$ can be predicted. This works in both ways. In turn, $q_{6,2}$ can also be predicted in a similar way. If we look at user u_8 and user u_1, there is $S(u_8) \cap S(u_1)=\emptyset$, meaning that u_8 and u_1 have not invoked any common Web services. According to Definition 3, u_8 and u_1 are not similar. Thus, u_8 is not useful and must not be taken into account for the prediction of $q_{1,4}$. The inclusion of as many users similar to user u_1 as possible will increase the accuracy of the prediction of $q_{1,4}$. Thus, the key is to identify users similar to u_1, e.g., u_6, and in the meantime, to exclude users dissimilar to u_1, e.g., u_8. This also applies to the identification of similar Web services.

3. CA-QGS mechanisms

This section first introduces the improved covering algorithms, including the definition of quotient space and the granularity computing. Then, the implementation of the covering algorithm on Spark is discussed.

3.1. Covering algorithm based on quotient space granularity analysis

The covering algorithm is proposed by Zhang and Zhang according to the geometric meaning of neural networks [11]. It employs a neuron model called M-P to obtain a rule based on field covering and does not require the number of clusters and initial centroids to be pre-specified. Quotient space is one of the three major theories of granular computing [12]. Granular computing is a simulation of human's global analysis capability. It is widely recognized

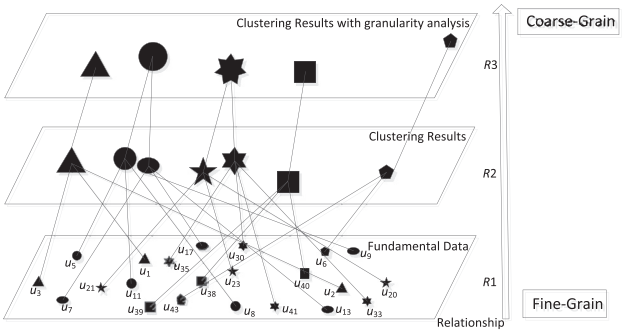

Fig. 2. Covering clustering based on granularity analysis.

that it can analyze and observe the same problem from very different granularity. People can not only solve the problem in a different granularity world, but also jump from one granularity world to another quickly. This capacity to process different granularity worlds is human's most powerful capability in solving problems. The granularity analysis can be analyzed and discussed from the perspective of different granularity of quotient space. Therefore, we propose an improved covering algorithm which considers the initial clusters effect of granularity and merges two clusters into one cluster when the similarity between them is greater than a granularity threshold θ.

As demonstrated in Fig. 2, we use the user-based covering clustering based on granularity analysis as an example. R in Fig. 2 refers to a set of all-equivalence relationships on the data, and the different point elements refer to users' quality experiences on the Web service. If the $R=R1$ ($R1$ is a fine-grain), the whole elements are self-contained. Then, the covering algorithm is employed to cluster these elements into seven classes, indicated by bold graphics on the clustering results layer. A new granularity $R2$ is generated through the clustering operation and $R3 < R2 < R1$. Any two clusters are merged into one cluster if their similarity is greater than θ. In Fig. 2, similar circles and ellipses are merged into one cluster. Similar five-pointed stars and hexagonal stars are merged into one cluster. And finally, there are five clusters with a new granularity $R3$. Through granularity analysis, a suitable granularity $R3$ is obtained. Similar elements are partitioned into the same clusters and dissimilar elements with exceptional values are far from other elements on the suitable granularity.

Similar users and Web services are partitioned into the same clusters and dissimilar users and Web services with usual quality values are far from the other users and Web services on the suitable granularity. Thus, those "usual" users and Web services are usually not partitioned into the same cluster with the other users and Web services. They will be excluded from the prediction by the approach for similar user selection discussed in Section 4.3. This way, untrustworthy users cannot jeopardize the accuracy of the prediction results.

3.2. Implementation of covering algorithm on Spark

Spark is the de facto distributed compute platform for large data processing, and is particularly suitable for iterative calculation. A main component of Spark is the *resilient distributed dataset* (RDD), which represents a read-only collection of objects partitioned across multiple machines that can be rebuilt if a partition is lost. Users can explicitly cache an RDD in memory across multiple machines and reuse it in multiple parallel operations. It is the main reason why Spark is able to process big data efficiently. Due to the performance of memory computing, data locality and transport optimization of Spark, it is particularly suitable for performing recur-

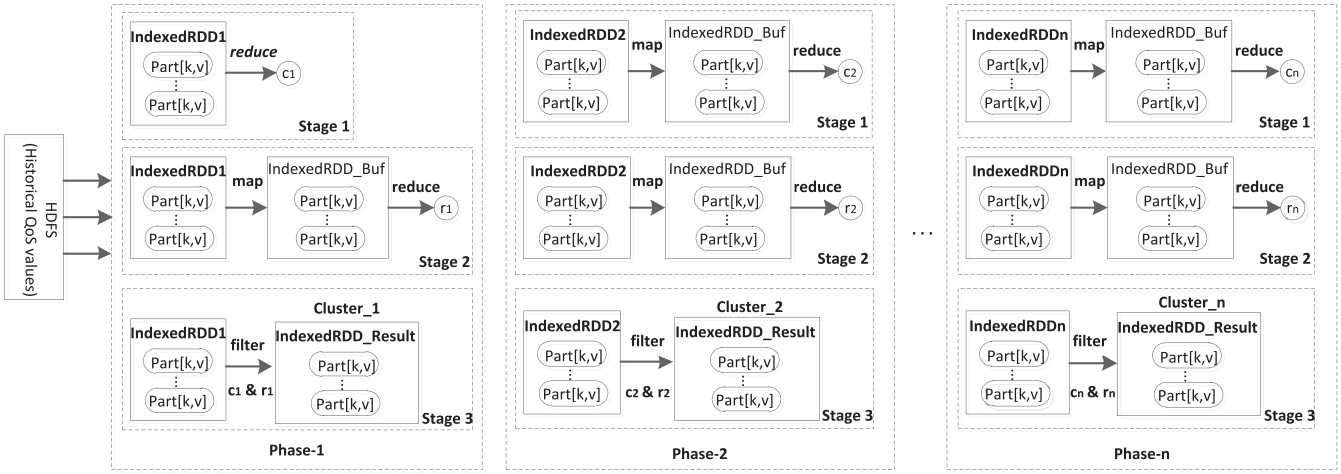

Fig. 3. Parallel covering process.

sive operations on big data [13]. Thus, CA-QGS implements the iterative covering algorithm on Spark to maximize its efficiency.

Applications implement on Spark read data from distributed files and transform the data that need to be processed into *IndexedRDD* [14]. In this research, the data to be processed is users' quality experiences on Web services, which represented by key-value pairs. *IndexedRDD* improves the efficiency of the selecting, updating and deleting operations through building indexes according to the key-value pairs stored in the memory across multiple machines. This can shorten the running time of CA-QGS and improve its throughput.

The covering algorithm implemented on Spark is illustrated in Fig. 3. As demonstrated, the distributed files are read from HDFS and transformed into *IndexedRDD*. The parallel covering process consists of many phases. Each phase comprises three stages which obtain the cluster's center, radius and the cluster respectively. Phase 1 describes the process for obtaining the first cluster. Stage 1 in phase 1 obtains the first cluster c_1 through the *reduce* operation. This operation obtains the data point that is nearest to the centroid of the whole data in parallel. Stage 2 in phase 1 obtains the radius r_1 of the cluster through the *map* and *reduce* operations. Specifically, *IndexedRDD_Buf* is obtained, an intermediate variable, through the *map* operation. The *map* operation calculates the distance between cluster c_1 and each uncovered data point and forms *IndexedRDD_Buf* . Then, radius r_1 is obtained through the *reduce* operation. This operation produces radius r_1 by calculating the *average* of the *IndexedRDD_Buf* in parallel. Stage 3 in phase 1 obtains *cluster_1* through the *filter* operation. In the meantime, the *filter* operation filters the data points where the distances between center c_1 and each uncovered data point are shorter than the radius r_1. The radius and center are acquired in stages 1 and 2. The stages in next phases are similar to those in phase 1. These phases are repeated until no more data point can be identified, because it indicates that all data points are clustered by these clusters. That is the end of the covering process.

After the covering process, CA-QGS performs granularity analysis and obtain the final clustering results. CA-QGS uses the improved algorithm executed on $U(s)$ for every Web Service s in S, through a total of n executors. Algorithm 1 presents the pseudocode for the above covering algorithm.

3.3. Computational complexity analysis

This section discusses the computational complexity of the covering algorithm. In line 5, the computational complexity is $O(m)$

Algorithm 1 CA-QGS clusters users $U(s)$ that have invoked Web service s.

Input: $U(s)$ and s
Output: Results of parallel covering with granularity analysis—a set of clusters C_1, C_2, ...
Begin
1: center $c = $ null
2: **Set** $C_u = U(s)$
3: **while** C_u != null **do:**
4: **if** $c ==$ null
5: center $c \leftarrow$ parallel_get_center(C_u)
6: **else**
7: center c $\leftarrow$ parallel_get_FutherSample(c, C_u)
8: **end if**
9: radius $r \leftarrow$ parallel_get_radius(c, C_u)
10: Covering C_b = parallel_get_covering(c, r, C_u)
11: $C_u = C_u$.subtract(C_b)
12: **end while**
13: max_similarty $\leftarrow$ get_max_similarty(C_b[])
14: **if** max_ similarity $\geq \theta$ **then**
15: $C_b \leftarrow$ merge($C_b[i]$, $C_b[i+1]$)
16: update(C_b[])
17: max_similarty $\leftarrow$ get_max_similarty(C_b[])
18: **end if**
19: return C[]()
End

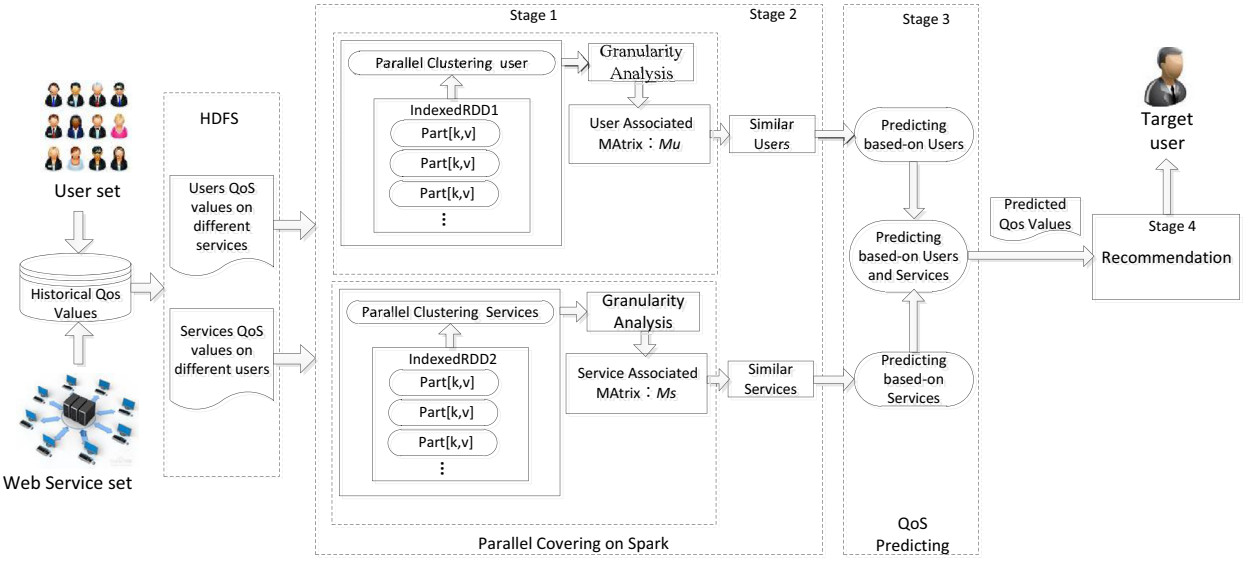

Fig. 4. Recommended process based on CA-QGS.

because there are a maximum of m data points in D. Similarly, the computational complexity of lines 7, 9, 10 and 11 are also $O(m)$. Lines 4–11 need to be repeated until all data points in D are covered. In line 9, the radius of a cluster is the average distance between the center of the cluster and all the data points are not covered by any clusters. On average, each newly created cluster covers half of the uncovered data points. Accordingly, the computational complexity is $O(\log m)$. In line 13, the computational complexity is $O(p)$ because there are p clusters after the initial covering process. Similarly, the computational complexity of lines 15–17 is $O(p)$ since there are a maximum of p clusters. Lines 15–17 need to be repeated until the max similarity exceeds θ. The number of clusters is much smaller than m. Thus, the computational complexity of Algorithm 1 is $O(m)+O(\log m)+O(p)=O(m)$. For n Web services, Algorithm 1 needs to be executed for n times. Thus, the computational complexity of clustering m users for all n Web services is $O(mn)$.

With an algorithm similar to Algorithm 1, CA-QGS partitions the Web services in S into clusters based on their quality values experienced by each of their common users. The clustering algorithm is executed for a total of m times, one for each of the m users. Similar to Algorithm 1, the computational complexity of the algorithm for clustering Web services is $O(mn)$. Overall, the computational complexity of the entire clustering process is $O(mn)+O(mn)=O(mn)$.

4. Web service recommendation

This section first introduces CA-QGS's process for Web service recommendation, and then discusses the four stages in detail.

4.1. Web service recommendation process

Considering that target users want to find desirable Web services through the recommender systems, the main task of Web service recommendation is to predict the QoS values of Web services which are unknown to the target user. Then, the Web services with optimal QoS values are recommended to the target user.

As presented in Fig. 4, given a user u, a Web service s, CA-QGS first predicts $q_{u,s}$, i.e., the quality of Web service s for user u, then recommends the Web services with the best $q_{u,s}$ value to user u. The recommendation procedure consists of four stages. Stages 1 and 2 perform the parallel covering on Spark. Stage 3 predicts the QoS. Stage 4 selects the Web services with the best QoS values. The four stages are discussed in detail below:

1. Based on users' quality experiences on each Web service s in S, denoted by $U(s)$, CA-QGS employs the covering algorithm to partition similar users and Web services into clusters. It then performs granularity analysis based on the initial clustering results and obtains the final clustering results. Two association matrices, Mu and Ms, are built based on the final clustering results.
2. Based on Mu and Ms, CA-QGS selects a set of users similar to user u, denoted by $A(u)$, from $U(s)$ and a set of Web services similar to Web service s, denoted by $A(s)$, from $S(u)$.
3. CA-QGS calculates $q_{u,\,s}(u)$, the quality of Web service s for user u based on $A(u)$. In the meantime, it calculates $q_{u,\,s}(s)$ based on $A(s)$. Then, it predicts $q_{u,s}$ by combining $q_{u,\,s}(u)$ and $q_{u,\,s}(s)$.
4. CA-QGS recommends the Web services with the optimal QoS values to the target user.

4.2. Stage 1: parallel covering algorithm based on granularity analysis

To identify $A(u)$, CA-QGS first partitions the users $u \epsilon U$ in U into clusters based on their historical QoS experiences on each individual Web service in S. For each web service $s \epsilon S$, CA-QGS maps the p-dimensional quality of the users in U that have invoked s into a p-dimensional space in the past, where each data point represents a use's quality experience on s. CA-QGS transforms all users' quality experiences into *IndexedRDD* and conducts the parallel covering algorithm on each service $s \epsilon S$ based on *IndexedRDD*, then returns initial clustering results. The relationship between clusters is adjusted based on the initial clustering results by performing granularity analysis with the global optimal similarity threshold θ. After that, CA-QGS obtains the final covering clustering results. As illustrated in Fig. 5, CA-QGS partitions the data points into clusters. The users in the same cluster have similar or even the same experiences on specific Web services. There are four services in Fig. 5. We employ the User Rating attribute to describe the covering process in a concise and clear way. We employ the covering-based clustering algorithm to partition users into clusters based on their experiences on Web service s_1. This way, we obtain four clusters, i.e., $\{u_1, u_6, u_8\}$, $\{u_9\}$, $\{u_2, u_4\}$ and $\{u_3, u_5, u_7\}$. The granularity analy-

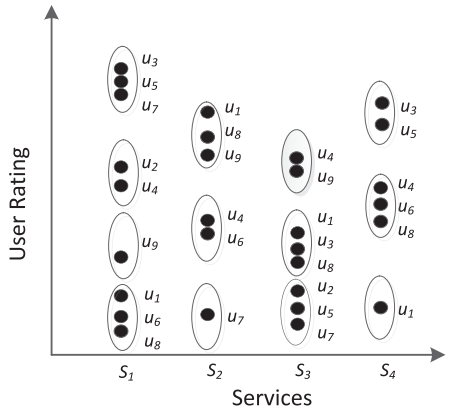

Fig. 5. Covering based on user.

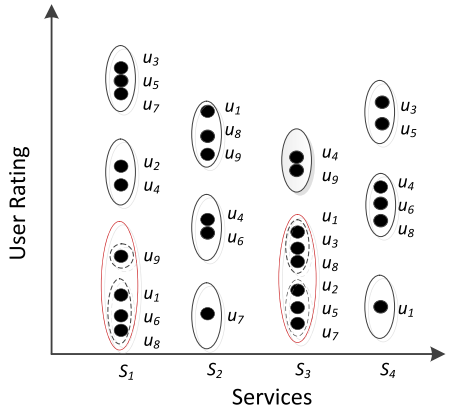

Fig. 6. User covering based on granularity analysis.

sis merges two clusters into one cluster if the similarity between them is greater than θ, as presented in Fig. 6.

Based on the clustering results, the user association matrix *Mu* is updated based on the clustering results from each clustering operation, following one rule: if user i and user j are partitioned into a same cluster under the service s, then $u_{ij} = u_{ij} + 1$ and $u_{ji} = u_{ji} + 1$.

With a clustering algorithm similar to Algorithm 1, CA-QGS also partitions the Web services in S into clusters based on their quality values experienced by each of their common users. This clustering algorithm is executed for a total of m times, one for each of the m users. The final clustering results are recorded in the service association matrix *Ms*.

4.3. Stage 2: the selection of similar neighbors

The next step is to select users similar to user u and Web services similar to Web service s. The selection must prioritize the users and Web services that are the most similar to user u and Web service s respectively. The similarity between user i and user j can be determined by two factors: (1) their common quality experiences, i.e., how similar their quality experiences are on each of the Web services they have both invoked in the past; (2) their common Web services, i.e., how many Web services they have both invoked in the past. Algorithm 1 has taken into account the first factor. Given a Web service s that has been invoked by both user i and user j, if their quality experiences on s are similar, they will be partitioned into the same cluster by the execution of Algorithm 1 for Web service s. The Web services that have not been invoked by either user i or user j will not even be considered by Algorithm 1. Now, we discuss how consider the second factor.

If the number of Web services co-invoked by two users is large, their quality experiences on those Web services can significantly contribute to the computation of their similarity. This is a fundamental assumption of collaborative filtering [37]. It also complies with the principle of statistical significance—high statistical significance indicates that the result is not attributed to chance. For example, if two users have both invoked a large number of Web services and have experienced highly similar, or even the same response times, it is highly likely that they will have similar, or even the same experiences on the response times of other Web services. A possible (but not necessarily the only) reason is that they might be located within the same local area network and rely on the same network infrastructure when accessing the same Web service. On the contrary, if two users have only co-invoked a very small number of Web services and have had similar quality experiences, it does not necessarily mean that they will have similar quality experiences on other Web services. For example, two users have only co-invoked one Web service in the past and have experienced similar or the same response times. It is too soon to determine that they will also experience similar or the same response times when invoking other Web services. Statistically, the same experiences they have on that only co-invoked Web service could be attributed to chance.

CA-QGS prioritizes users that have co-invoked a larger number of Web services with user i when selecting users similar to user i. A high value of $u_{i,j}$ in M_u indicates that users i and j have co-invoked a large number of Web services and have had similar quality experiences on those Web services. Formula (3) presents the user association matrix obtained from processing the user-service matrix in Fig. 1. Take user u_9 for example. User u_8 should be given the highest priority during the selection of users similar to u_9. User u_5 should be given the second highest priority. Users u_2, u_4 and u_6 should be given priorities lower than u_8 and u_5 but higher than u_1, u_3 and u_7. There are many ways to allocate different weights to different users during the selection of similar users according to their priorities. For example, a set of weights, w_1, w_2, ..., w_9, can be allocated to users u_1, u_2, ..., u_8 respectively to indicate their priorities, where $w_8 > w_5 > w_2$, w_4, $w_6 > w_1$, w_3, w_7 and $2\sum_{i=1}^{8} w_i = 1$. CA-QGS selects the top k_u $(1 \le k_u \le m - 1)$ users that have been partitioned into the same cluster for the most times. Accordingly, CA-QGS performs descending operations on each row elements in the user association matrix M_u and selects the top k_u users form $S(u)$:

$$M_u = \begin{array}{c} \begin{matrix} u_1 & u_2 & u_3 & u_4 & u_5 & u_6 & u_7 & u_8 & u_9 \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 3 & 1 & 1 & 2 & 0 & 0 & 1 \\ 0 & 3 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 0 & 1 & 0 & 2 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 2 & 2 \\ 1 & 2 & 0 & 0 & 1 & 0 & 3 & 0 & 1 \\ 0 & 0 & 0 & 2 & 0 & 3 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 2 & 0 & 1 & 0 & 3 \\ 0 & 1 & 0 & 1 & 2 & 1 & 0 & 3 & 0 \end{bmatrix} \end{array} \begin{matrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \\ u_9 \end{matrix} \quad (3)$$

Given a Web service s, CA-QGS employs a similar to select top k_s $(1 \le k_s \le n - 1)$ Web services similar to Web service s.

The optimal k value can be obtained efficiently through experimenting with different k values, which does not require time-consuming re-clustering of users and Web services. Parameter k (k_u or k_s) is a domain-specific parameter. Different applications and data sets usually have their own characteristics, and hence inherit different optimal k values. On the one hand, a too small k value cannot ensure that all useful information on the users is used to perform quality predictions. On the other hand, a too large k value may lead to the inclusion of dissimilar users into the quality prediction, and consequently decreases the prediction accuracy. There-

fore, the k value should be set domain-specifically based on experiences and/or experiments. In Section 5.4, we study the impact of k on the prediction accuracy with a real-world data set.

4.4. Stage 3: prediction

Based on the quality experiences obtained from the selected k_s users similar to user u, i.e., $A(u)$, CA-QGS employs Formula (4) as a user-based method to predict the quality of Web service s for user u with a weighted average of the quality experienced by the k_u users:

$$q_{u,s}(u) = \frac{\sum_{t=1}^{k_u} (r_{a_u(t)} \times n_{a_u(t)})}{N_u} \tag{4}$$

where k_u is the number of users similar to the target user u, $a_u(t)$ represents the tth user in the set of users similar to user u, $r_{a_u(t)}$ denotes the user $a_u(t)$ that invoked Web service s, $n_{a_u(t)}$ indicates the number of times that user u and $a_u(t)$ are partitioned into a same cluster, and N_u represents the total number of times that user u and its similar users are partitioned into a same cluster.

Similarly, CA-QGS employs formula (5) as an item-based method to predict the quality of Web service s for user u with a weighted average of the quality of the selected k_s services similar to Web service s, i.e., $A(s)$:

$$q_{u,s}(s) = \frac{\sum_{t=1}^{k_s} (r_{a_s(t)} \times n_{a_s(t)})}{N_s} \tag{5}$$

where k_s is the number of Web services similar to the target service s, $a_s(t)$ represents the tth service in the set of Web service similar to s, $r_{a_s(t)}$ denotes service $a_s(t)$ that was invoked by user u, $n_{a_s(t)}$ indicates the number of times that service s and $a_s(t)$ are clustered into a same cluster, and N_s represents the total number of times that service s and its similar services are clustered into a same cluster.

Formulas (4) and (5) are used to predict the quality of a Web service based on only similar users and similar Web services respectively. However, it is possible that there are only a small number of similar users or similar Web services due to the sparsity of the user-service matrix. Quality prediction by formula (4) or (5) is prone to be inaccurate due to the lack of adequate data in the user-service matrix. Thus, it is necessary to combine the prediction results obtained by the user-based and item-based methods. Thus, CA-QGS combines formulas (4) and (5) with the weighting factor λ, which is experimentally studied in Section 5.4, to achieve more accurate prediction results:

$$q_{u,s} = \lambda q_{u,s}(u) + (1-\lambda) q_{u,s}(s) \tag{6}$$

where $\lambda \epsilon [0,1]$ is a weighting parameter that indicates the priority given for the results obtained from Formula (4). Parameter λ allows the prediction to adapt to different environments. When $\lambda = 1$, it is equivalent to prediction solely based on users. On the contrary, $\lambda = 0$ is equivalent to prediction solely based on services. If user u has not invoked any services and service i has not been invoked by any users, the average quality of all services is used as the predicted quality. Formula (7) summarizes the three different scenarios:

$$q_{us} = \begin{cases} \lambda q_{us}(u) + (1-\lambda) q_{us}(s) & A(u) \neq \emptyset \text{ and } A(s) \neq \emptyset \\ q_{us}(u) & A(u) \neq \emptyset \text{ and } A(s) = \emptyset \\ q_{us}(s) & A(u) = \emptyset \text{ and } A(s) \neq \emptyset \\ null & A(u) = \emptyset \text{ and } A(s) = \emptyset \end{cases} \tag{7}$$

4.5. Stage 4: Web service recommendation

After predicting the quality values of Web services with formula (7), the Web services with the optimal QoS values are selected and recommended to user u.

5. Experimental evaluation

In this section, we experimentally compare CA-QGS with six state-of-the-art approaches in their effectiveness (measured by predication accuracy) and efficiency (measured by computational overhead). In the experiments, we change the values of different parameters, including the matrix density, λ, k, the similarity threshold θ and the number of clusters.

The Spark cluster used to implement CA-QGS is built on a cluster with nine connected nodes. Each node runs Windows 7 64 bit and Spark 1.4 on Intel Core i7-4970 3.6 GHZ, 16G RAM. The total number of CPU cores is 72 and the total memory capacity is 450G.

5.1. Experimental setup

The experiments are conducted on WS-Dream [15–17], a real-world dataset that contains two-dimensional quality values of 5,825 real-world Web services, including their round-trip times (RTT) and throughput (TP). The data were collected from 339 distributed users' invocations on those Web services. In total, there are 1,974,675 records. This dataset has been used by many researchers [10,32]. In this section, we present the results obtained from experiments on the RTT data. The ones from experiments on the TP data are similar, thus are omitted.

In the experiments, we randomly select a certain percentage of Web services from the dataset as a training set. The remaining Web services are used as a test set. The percentage of training set varies from 5% to 30% in step of 5%. We also change the matrix density from 0.1 to 0.3 in steps of 0.05, k from 1 to 10, λ from 0.1 to 1, and the similarity threshold θ from 0.4. This way, we comprehensively evaluate the ability of CA-QGS to handle datasets with different characteristics in various application scenarios. We also evaluate the efficiency of CA-QGS by comparing the time consumption with a state-of-the-art parallel prediction approach based on k-means. In those experiments, we change the total number of CPU cores from 10 to 50 in step of 10.

5.2. Evaluation metrics

To evaluate the prediction accuracy, we employ two metrics, namely mean absolute error (MAE) and root mean square error (RMSE), in the experiments.

MAE is defined as:

$$\text{MAE} = \frac{\sum_{u,s} |r(u,s) - p(u,s)|}{N} \tag{8}$$

where $r(u, s)$ denotes the predicted QoS values between service user u and Web service s, $p(u, s)$ denotes the measured QoS value, and N is the number of Web services in the prediction results.

Root mean square error (RMSE) presents the standard deviation of the prediction error, which is expressed as follow:

$$\text{RMSE} = \sqrt{\frac{\sum_{u,s} (r(u,s) - p(u,s))^2}{N}} \tag{9}$$

Lower MAE and RMSE values indicate higher prediction accuracy.

5.3. Performance comparison

CA-QGS is compared with the following state-of-the-art prediction approaches:

- UPCC (User-based collaborative filtering method using Pearson Correlation Coefficient): This method employs values contributed by similar users to predict missing QoS values [17].

Table 1
Accuracy comparison (Top-k= 2, λ= 0.4).

Quality dimension	Method	MAE						RMSE					
		Matrix density											
		0.05	0.10	0.15	0.20	0.25	0.3	0.05	0.10	0.15	0.20	0.25	0.3
RT	UPCC	0.6274	0.5346	0.486	0.4665	0.4523	0.4365	1.4877	1.3845	1.312	1.2944	1.2669	1.2265
	IPCC	0.8685	0.8581	0.8256	0.7116	0.671	0.6129	1.6641	1.6915	1.5274	1.4674	1.4652	1.3936
	UIPCC	0.7121	0.682	0.6356	0.5626	0.5346	0.4985	1.4784	1.4464	1.3276	1.2826	1.2642	1.21
	PKU	0.5365	0.4845	0.4539	0.433	0.418	0.4018	1.5863	1.51	1.4444	1.4064	1.3912	1.3388
	PKI	0.4079	0.3757	0.3675	0.3598	0.3544	0.3431	1.3791	1.3296	1.308	1.3082	1.3071	1.2535
	PKUI	0.4202	0.3864	0.3718	0.363	0.3521	0.3401	1.3036	1.251	1.2173	1.2139	1.1998	1.1547
	UCA-QGS	0.4851	0.4347	0.4074	0.389	0.3786	0.3708	1.471	1.3807	1.3232	1.2874	1.2772	1.251
	ICA-QGS	0.4173	0.3899	0.3763	0.369	0.3612	0.3491	1.2934	1.2389	1.2011	1.1966	1.1871	1.1444
	CA-QGS	**0.4054**	**0.3715**	**0.354**	**0.3437**	**0.3355**	**0.3268**	**1.2401**	**1.1808**	**1.1365**	**1.1298**	**1.1163**	**1.0828**

- IPCC (Item-based collaborative filtering method using Pearson Correlation Coefficient): This method employs values contributed by similar services to predict missing QoS values [23].
- UIPCC: This method combines UPCC and IPCC with a specific parameter lambda to predict missing values based on both similar users and similar services [27].
- PKU: The parallel K-means method is used to cluster users and the similar users are obtained. Then, the missing QoS values of unused Web services are predicted based on similar users for recommendation.

PKU shares the same methodology as our user-based prediction approach UCA-QGS to cluster users for the identification of similar users. To ensure the comprehensiveness of the comparison, we have also implemented the following two prediction approaches:

- PKI: The parallel K-means method is used to cluster Web services to identify similar Web services. Then, the missing QoS values of unused Web services are predicted based on similar users for recommendation.
- PKUI: Combining PKU and PKI, this approach predicts the missing quality values of unused Web services based on both similar users and similar Web services.

In this series of experiments, the matrix density varies from 5% to 30% in steps of 5%. The value of k is set to 2, and $\lambda = 0.4$. The accuracies achieved by different approaches are presented in Table 1.

The experimental results show that CA-QGS achieves the highest prediction accuracy across all scenarios. The increase in the matrix density increases the prediction accuracy obtained by CA-QGS.

5.4. Impact of parameters

5.4.1. Impact of Top-k

Parameter k determines the number of similar users or services selected for prediction. In order to investigate the impact of k on prediction accuracy, we set the *matrix density* as 20%, λ as 0.4 and similarity threshold θ as 0.6, and change the value of k from 1 to 10 in steps of 1. The experimental results are illustrated in Fig. 7.

According to the experimental results, the increase in the value of k does not necessarily lead to a higher prediction accuracy. Overall, as the value of k increases, the prediction accuracy increases at the beginning before reaching its optimal value 2, and then starts to decrease because k continues to increase. This indicates that k must not be too small or too big. A k value that is too small does not include adequate information on similar users and Web services in the prediction. This is demonstrated in Fig. 7 by the decreases in MAE and RMSE in both series of experiments as the value of k starts to increase from 1. On the other hand, a k value that is too large may involves users and/or Web services that are not similar to target user and target Web service. Such users and Web services may decrease the prediction accuracy significantly.

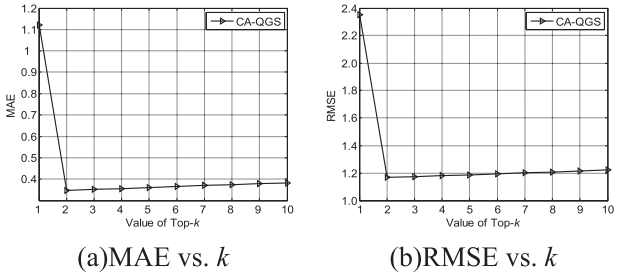

Fig. 7. Impact of k on CA-QGS.

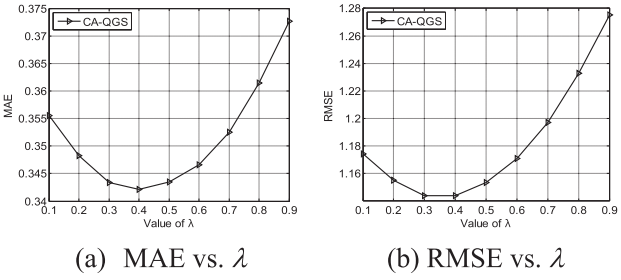

Fig. 8. Impact of λ.

5.4.2. Impact of λ

Parameter λ determines how much CA-QGS replies on similar users and similar services respectively, when predicting the quality of a Web service for a user. As discussed in Section 4.4, when $\lambda = 0$, CA-QGS is equivalent to ICA-QGS as only similar Web services are considered. Similarly, when $\lambda = 1.0$, CA-QGS is equivalent to UCA-QGS. Thus, we vary the λ value from 0.1 to 0.9 in steps of 0.1 to evaluate its impact on CA-QGS. In this series of experiments, we set the matrix density to 0.2, similarity threshold θ to 0.6 and k to 2. The experimental results are illustrated in Fig. 8.

Fig. 8 demonstrates that λ impacts the prediction accuracy of CA-QGS significantly. Overall, MAE and RMSE in Fig. 8 increase with the increase of λ before reaching their optimal values. Then, MAE and RMSE decrease because the λ value continues to increase. Similar to our findings of k, λ must not be too small or too big. A suitable λ value provides the best prediction accuracy because it enables a proper combination of the information on similar users and similar Web services.

5.4.3. Impact of θ

Finding the global optimal similarity threshold θ is vital to improve the prediction accuracy of CA-QGS. We conduct extensive ex-

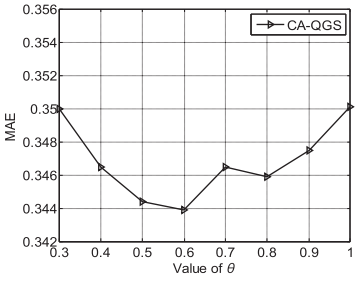

Fig. 9. Impact of θ.

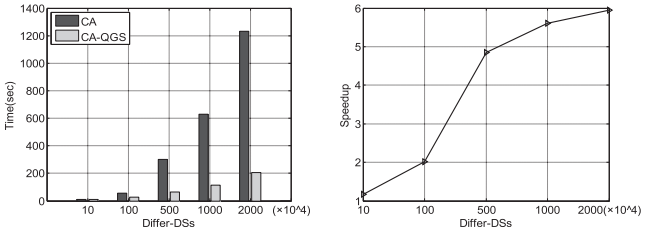

Fig. 11. Efficiency of CA-QGS and CA.

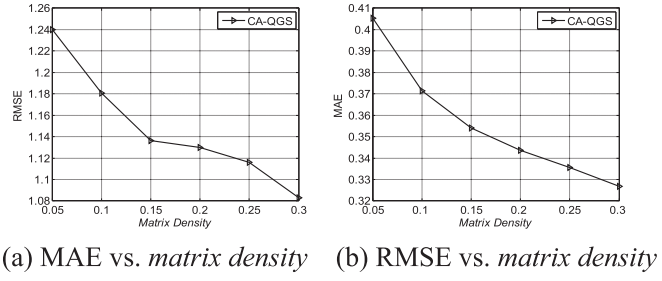

Fig. 10. Impact of *matrix density*.

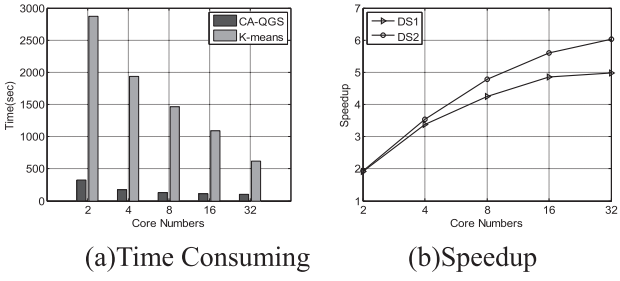

Fig. 12. Efficiency of CA-QGS and K-means.

periments to study the impact of θ. In this series of experiments, we set the matrix density to 0.2, λ to 0.4 and k to 2. In the meantime, we vary the value of θ from 0.3 to 1 in steps of 0.1. Then, we obtain user association matrices and service association matrices with different values of θ. Finally, we employ those matrices to perform prediction and obtain the optimal value of θ. The experimental results are illustrated in Fig. 9.

If θ is set to a small number, the data may be only clustered into a few clusters. In contrast, if the threshold is set to a large number, the data may be clustered into many clusters without considering the correlation between the clusters after the initial clustering. Fig. 9 shows the clustering results with different θ values. It demonstrates that an appropriate θ allows an accurate prediction. In our experiments, we obtain the most accurate prediction results when $\theta = 0.6$. Once θ changes, the clustering results will change. This leads to changes in the association matrices, which impact the prediction results. Therefore, the change in θ has a great influence on the prediction results.

5.4.4. Impact of matrix density

Section 5.2 compares the prediction accuracy achieved by different approaches under various matrix density settings. In this section, we investigate the impact of the matrix density on CA-QGS. The experimental results shown in Fig. 10 demonstrate that the matrix density has a significant impact on the prediction accuracy obtained by CA-QGS.

The experimental results show that both MAE and RMSE decrease because the matrix density increases in all cases. A data set with a higher matrix density contains more information on users and Web services that can be employed to make predictions. Thus, it allows CA-QGS to achieve higher prediction accuracy.

5.5. Efficiency evaluation

Besides the effectiveness measured by the prediction accuracy, the efficiency of a recommendation approach also plays an important role in its practicality. The time consumption of a recommendation approach with high effectiveness might still not be practical in very large-scale scenarios. In recent years, cloud computing has significantly promoted the development of big data [26]. With the prevalence of cloud computing and service computing, more and more services have been co-invoked by a large number of users, resulting in massive invoking records. We also conduct a series of experiments to compare the efficiency of CA-QGS with K-means in processing a large number of service invocation records. We synthetically generate five large datasets based on the WS-Dream dataset, which include 100,000, 1,000,000, 5,000,000, 10,000,000, and 20,000,000 service invocation records to simulate very large-scale scenarios. These datasets are generated by extracting the corresponding number of users' service invocation records on each Web service and the all service invocation records are regarded as those generated by many users on a Web service.

To evaluate the performance of the parallel covering-based clustering algorithm introduced in Section 4.2, we compare it with the original covering algorithm (CA) on the five abovementioned large-scale datasets. The results are shown in Fig. 11 where Differ-DSs refer to five large data sets. Fig. 11(a) shows the time consumption taken by CA-QGS and CA respectively, while Fig. 11(b) shows the speedup of CA-QGS against CA.

CA-QGS takes 10 s to process the dataset with 100,000 service invocation records and CA takes 12 s. To process the largest datasets with 20,000,000 service invocation records, CA-QGS takes only 192 s, 18 times more (192 s versus 10 s), while CA takes 1230 s, 102 (1230 s versus 12 s). The results indicate that, as the dataset becomes larger, the time taken by CA increases more much more rapidly than CA-QGS. As we can see in Fig. 11(b), with the increase in the size of the dataset, the speedup of CA-QGS also increases. This demonstrates that CA-QGS is suitable for large-scale scenarios.

We also implement extensive experiments to study the impact of the total number of CPU cores on CA-QGS and the prediction approach based on k-means. In Fig. 12, Speedup refers to the ratio of the time spent by a single processor system and a multi-processor system. DS1 and DS2 refer to the two datasets, one with 5,000,000 service invocation records and the other with 10,000,000, used in this series of experiments.

Fig. 12(a) demonstrates the time consumption taken by CA-QGS and k-means to process the dataset with 10,000,000 service invocation records. With the increase in the number of CPU cores, the time taken by CA-QGS and k-means gradually decrease. How-

ever, CA-QGS consumes much less time than k-means which indicates that the efficiency of our method is significantly better than the prediction approach based on k-means. The speedups of CA-QGS are shown in Fig. 12(b). We can observe that, with the same number of CPU cores, a bigger dataset results in a more significant speedup. When the number of CPU cores increases, the speedup also increases but has less significant. This is caused by the increase in the communication between CPU cores, which contributes to the total clustering time. Thus, in scenarios with an even larger dataset or where extremely fast predictions are needed, CA-QGS is a more suitable choice due to its outstanding advantage in efficiency over the traditional covering algorithm and the prediction approach based on k-means.

6. Related works

Web service recommendation has become a very active research area in the past few years. CF is the dominant technique in current recommender systems [4,5,19,20] for Web service recommendation. In recent years, many researchers have employed clustering techniques to improve the drawbacks of CF in predicting the quality of Web services. This section will first review the related work on CF-based approaches for Web service recommendation in general, then the approaches based on clustering.

6.1. CF-based approaches

CF approaches are widely adopted in recommender systems [4,19,20] for Web services. Many approaches based on CF have been proposed in recent years for Web service recommendation [9,18,21–25]. To name a few, the approach proposed by Shao et al. [17] first calculates the similarity measured by the Pearson Correlation Coefficient (PCC) between users based on their past experiences on the quality of Web services. It then predicts the quality values of the target Web services based on users' experiences that are similar to the target user. This method is implemented in our experiments as UPCC. The method named IPCC, which is also implemented in our experiments, is a prediction approach based on only similar Web services. This approach was first invented and used by Amazon [23] and has been extensively investigated and widely employed [24]. This approach identifies similar Web services using PCC and predicts the missing quality values of unused Web services based on identified similar Web services.

Making predictions only based on similar users like UPCC suffer from poor prediction accuracy when there are not adequate similar users and the same for IPCC when there are not adequate similar Web services. To address this issue, Zheng et al. [27] combine the user-based and item-based CF techniques and introduce weights to balance the impact of user-based and item-based CF approaches. Their approach improves the prediction accuracy significantly. Researchers have also employed other modified CF-based approaches to perform Web service recommendation. Liu et al. [28] present a new user similarity model to improve the recommendation accuracy while only a few service invocation records are available for the calculation of user similarity. The model considers not only the local context information of service invocation records, but also the global preferences for user behaviors. Luo et al. [29] design an incremental CF-based approach enhanced with the Regularized Matrix Factorization (RMF). They investigate the training process of MF-based recommendation approaches, and incrementally update the trained parameters according to new data. As a result, they can obtain an efficient strategy used in Recommender. Liu et al. [30] propose a combination of location-aware and personalized CF method. They employ not only the quality values but also the location of Web services to perform service recommendation. The location of a user is estimated based on its IP address. Similar users are identified based on both their quality experiences on Web services and their locations. A location-aware and personalized Web service recommendation approach is designed. Wang et al. [31] propose a reputation measurement method to prevent biased and malicious users' quality ratings of Web services from being used in Web service recommendation. This approach detects user preferences and malicious users' quality ratings of Web services by adopting the Cumulative Sum Control Chart. It then reduces the influence of subjective users' quality ratings using PCC. This way, the prediction accuracy can be improved.

6.2. Clustering-based approaches

Data sparsity, poor scalability the computational overhead are the three major issues that CF techniques have yet to properly address. Many researchers have attempted to cluster users and Web services to identify similar users and Web services before Web service recommendation.

Kumara et al. [32] propose a clustering-based service recommendation approach that employs the bottom-up hierarchical clustering technique to cluster users and Web services based on their quality experiences and semantic similarity. They also consider the association between services and recommend services using the generated clusters and services with better QoS values selected by a filtering process. Chen et al. [33] cluster users into different regions according to their locations and quality experiences with the hierarchical clustering technique. Web services are clustered based only on the similarity between their QoS values. The quality of Web services are then performed by finding target user's or target service's similar neighbors. However, their approach does not consider the information on similar users. Zhang et al. [34] propose an approach that employs a fuzzy clustering approach to cluster users. The approach takes the advantages of both fuzzy clustering and PCC. The major limitation of their approach is the significant impact of initially selected centroids on the clustering results. Due to its simplicity and popularity, the k-means algorithm has also attracted many researchers' attention as a means for clustering users and Web services. Yu et al. [3] propose an approach named CluCF that employs the k-means algorithm to cluster users and Web services with the focus on lowering the complexity of updating clusters upon new users and new Web services. Similar to [3], Wu et al. [10] also employ the k-means algorithm to cluster users, but with a different aim to identify untrustworthy users. Su et al. [35] propose a trust-aware approach TAP that combines k-means for Web service recommendation. They use k-means to cluster users and Web services and then calculate the reputation of users and Web services based on the clustering information and their beta reputation. Next, they identify trustworthy similar users and similar Web services. Finally, they perform predictions for target users by combining similar users and similar Web services. Although those recommendation approaches perform well in off-line testing on small datasets, they are too computationally expensive on large datasets. Therefore, Wang et al. [36] propose a new collaborative filtering recommendation algorithm based on k-means. This approach first clusters the longitude and latitude of users and then calculates the similarity of users within each cluster. However, the critical limitation to prediction approaches based on k-means remains unsolved - the clustering results heavily rely on the pre-specified number of clusters and the initially selected centroids.

In this paper, we propose CA-QGS, a novel approach for predicting the quality of Web services and recommending services to target users. CA-QGS efficiently addresses the limitations and issues of existing approaches with a novel covering-based clustering technique. CA-QGS does not require the number of clusters or the centroids of clusters to be pre-specified. Implemented on Spark, CA-QGS can efficiently mitigate the issue of data scalability. The exper-

imental results demonstrate that CA-QGS significantly outperforms state-of-the-art prediction approaches for Web service recommendations in both effectiveness and efficiency.

7. Conclusion and further study

In this paper, we propose CA-QGS, a novel covering-based approach for Web service recommendation with quotient space granularity analysis. CA-QGS first employs a covering algorithm to partition users into clusters based on the similarity between their historical quality experiences on each of the co-invoked Web services with a granularity analysis mechanism. Based on the initial clustering results, the granularity analysis mechanism helps users visually understand the physical meaning of within-class and among-class from different perspectives. Following a similar procedure, CA-QGS also partitions Web services into clusters based on each user's quality experiences on them. Then, from the final clustering results, CA-QGS chooses a number of users and Web services that are most similar to the target user and the target Web service as the basis for quality prediction. Based on the selected similar users and Web services, CA-QGS predicts the quality of Web service s for u. Finally, CA-QGS recommends the Web services with optimal QoS values to the target user. CA-QGS can be implemented on Spark to increase its efficiency in processing large datasets. The results of the experiments on the real-world WS-Dream dataset demonstrate that CA-QGS significantly outperforms state-of-the-art prediction approaches for Web services. The results also demonstrate that CA-QGS is more efficient in both clustering and prediction.

In our future work, we will enhance CA-QGS to solve the cold-start problem and consider other QoS values, such as the location information of the Web services, to facilitate more accurate Web service recommendation.

Acknowledgments

This work was supported by the National Key Technology R&D Program (no. 2015BAK24B01), the General Research for Humanities and Social Sciences Project of Chinese Ministry of Education (no. 15YJAZH112) and the Key Project of Nature Science Research for Universities of Anhui Province of China (no. KJ2016A038).

References

[1] J. Yin, Y. Tang, W. Lo, et al., From big data to great services, in: IEEE International Congress on Big Data, IEEE, 2016, pp. 165–172.
[2] R. Hu, W. Dou, J. Liu, Clubcf: a clustering-based collaborative filtering approach for big data application, IEEE Trans. Emerg. Top. Comput. 2 (3) (2014) 302–313.
[3] C. Yu, L. Huang, CluCF: a clustering CF algorithm to address data sparsity problem, Serv. Oriented Comput. Appl. 11 (1) (2017) 33–45.
[4] M. Al-Hassan, H. Lu, J. Lu, A semantic enhanced hybrid recommendation approach: a case study of e-Government tourism service recommendation system, Decis. Support Syst. 72 (2015) 97–109.
[5] Q. Shambour, J. Lu, An effective recommender system by unifying user and item trust information for B2B applications, J. Comput. Syst. Sci. 81 (7) (2015) 1110–1126.
[6] L. Song, C. Tekin, M. van der Schaar, Online learning in large-scale contextual recommender systems, IEEE Trans. Serv. Comput. 9 (3) (2016) 433–445.
[7] S. Deng, L. Huang, G. Xu, et al., On deep learning for trust-aware recommendations in social networks, IEEE Trans. Neural Netw. Learn. Syst. 28 (5) (2016) 1164–1177.
[8] S. Wang, Z. Zheng, Z. Wu, M.R. Lyu, F. Yang, Reputation measurement and malicious feedback rating prevention in web service recommendation systems, IEEE Trans. Serv. Comput. 8 (5) (2015) 755–767.
[9] K. Huang, Y. Liu, S. Nepal, Y. Fan, S. Chen, W. Tan, A novel equitable trustworthy mechanism for service recommendation in the evolving service ecosystem, in: Proceedings of the International Conference on Service-Oriented Computing (ICSOC), 2014, pp. 510–517.
[10] C. Wu, W. Qiu, Z. Zheng, et al., QoS prediction of web services based on two-phase K-means clustering, in: 2015 IEEE International Conference on Web Services (ICWS), IEEE, 2015, pp. 161–168.
[11] L. Zhang, B. Zhang, A geometrical representation of McCulloch-Pitts neural model and its applications, IEEE Trans. Neural Netw. 10 (4) (1999) 925–929.
[12] L. Zhang, B. Zhang, The quotient space theory of problem solving, Fundam. Inf. 59 (2–3) (2004) 287–298.
[13] A.G. Shoro, T.R. Soomro, Big data analysis: apache spark perspective, Global J. Comput. Sci. Technol. 15 (1) (2015) 9–14.
[14] S. Peng, J. Sankaranarayanan, H. Samet, SPDO: high-throughput road distance computations on spark using distance oracles, in: 2016 IEEE 32nd International Conference on Data Engineering (ICDE), IEEE, 2016, pp. 1239–1250.
[15] Z. Zheng, M.R. Lyu, Personalized reliability prediction of web services, ACM Trans. Softw. Eng. Methodol. 22 (2) (2013) 1–25.
[16] Z. Zheng, X. Wu, Y. Zhang, et al., QoS ranking prediction for cloud services, IEEE Trans. Parallel Distrib. Syst. 24 (6) (2013) 1213–1222.
[17] L. Shao, J. Zhang, Y. Wei, et al., Personalized QoS prediction for web services via collaborative filtering, in: IEEE International Conference on Web Services (ICWS), 2007, pp. 439–446.
[18] T. Könik, R. Mukherjee, J. Katukuri, Subjective similarity: personalizing alternative item recommendations, in: Proceedings of the 24th International World Wide Web Conference (WWW), 2015, pp. 1275–1279.
[19] J. Lu, D. Wu, M. Mao, W. Wang, G. Zhang, Recommender system application developments: a survey, Decis. Support Syst. 74 (2015) 12–32.
[20] W. Wang, G. Zhang, J. Lu, Collaborative filtering with entropy-driven user similarity in recommender systems, Int. J. Intell. Syst. 30 (8) (2015) 854–870.
[21] J. Yao, W. Tan, S. Nepal, S. Chen, J. Zhang, D. De Roure, C. Goble, Reputationnet: reputation-based service recommendation for e-science, IEEE Trans. Serv. Comput. 8 (3) (2015) 439–452.
[22] Z. Zheng, H. Ma, M.R. Lyu, et al., Wsrec: a collaborative filtering based web service recommender system, in: IEEE International Conference on Web Services (ICWS), IEEE, 2009, pp. 437–444.
[23] G. Linden, B. Smith, J. York, Amazon.com recommendations: item-to-item collaborative filtering, IEEE Internet Comput. 7 (1) (2003) 76–80.
[24] B. Sarwar, G. Karypis, J. Konstan, et al., Item-based collaborative filtering recommendation algorithms, in: Proceedings of the 10th International Conference on World Wide Web, ACM, 2001, pp. 285–295.
[25] M. Deshpande, G. Karypis, Item-based top-n recommendation algorithms, ACM Trans. Inf. Syst. 22 (1) (2004) 143–177.
[26] Y.N. Xia, M.C. Zhou, X. Luo, et al., A stochastic approach to analysis of energy-aware DVS-enabled cloud datacenters, IEEE Trans. Syst. Man Cybern. Syst. 45 (1) (2015) 73–83.
[27] Z. Zheng, H. Ma, M.R. Lyu, I. King, QoS-aware web service recommendation by collaborative filtering, IEEE Trans. Serv. Comput. 4 (2) (2011) 140–152.
[28] H. Liu, Z. Hu, A. Mian, et al., A new user similarity model to improve the accuracy of collaborative filtering, Knowl. Based Syst. 56 (2014) 156–166.
[29] X. Luo, Y. Xia, Q. Zhu, Incremental collaborative filtering recommender based on regularized matrix factorization, Knowl. Based Syst. 27 (2012) 271–280.
[30] J. Liu, M. Tang, Z. Zheng, et al., Location-aware and personalized collaborative filtering for web service recommendation, IEEE Trans. Serv. Comput. 9 (5) (2016) 686–699.
[31] S. Wang, Z. Zheng, Z. Wu, M.R. Lyu, F. Yang, Reputation measurement and malicious feedback rating prevention in web service recommendation Systems, IEEE Trans. Serv. Comput. 8 (5) (2015) 755–767.
[32] B.T.G.S. Kumara, I. Paik, T. Siriweera, et al., Cluster-based web service recommendation, in: 2016 IEEE International Conference on Services Computing (SCC), IEEE, 2016, pp. 348–355.
[33] X. Chen, Z. Zheng, Q. Yu, M.R. Lyu, Web service recommendation via exploiting location and QoS information, IEEE Trans. Parallel Distrib. Syst. 25 (7) (2014) 1913–1924.
[34] M. Zhang, X. Liu, R. Zhang, et al., A web service recommendation approach based on QoS prediction using fuzzy clustering, in: 2012 IEEE Ninth International Conference on Services Computing (SCC), IEEE, 2012, pp. 138–145.
[35] K. Su, B. Xiao, B. Liu, et al., TAP: a personalized trust-aware QoS prediction approach for web service recommendation, Knowl. Based Syst. 115 (2017) 55–65.
[36] Z. Wang, N. Yu, J. Wang, User attributes clustering-based collaborative filtering recommendation algorithm and its parallelization on spark, in: Asian Simulation Conference, Springer, Singapore, 2016, pp. 442–451.
[37] K. Goldberg, T. Roeder, D. Gupta, et al., Eigentaste: a constant time collaborative filtering algorithm, Inf. Retriev. 4 (2) (2001) 133–151.
[38] Q. He, J. Yan, H. Jin, et al., Quality-aware service selection for service-based systems based on iterative multi-attribute combinatorial auction, IEEE Trans. Softw. Eng. 40 (2) (2014) 192–215.