Advance Access publication on 6 January 2020
doi: 10.1093/comjnl/bxz090

Effective User Preference Clustering in Web Service Applications

Yan Wang[1], Jian-tao Zhou[1,*], Xinyuan Li[1] and Xiaoyu Song[2]

[1]*Inner Mongolia Engineering Lab of Cloud Computing and College of Computer Science, Inner Mongolia University, No. 235 Daxue West Street, Saihan district, Hohhot 010021, China*
[2]*Department of Electrical and Computer Engineering, Portland State University, 1900 SW Fourth Ave., Suite 160 Portland, OR 97207, USA*
**Corresponding author: ndcswy@sina.com*

The research on personalized recommendation of Web services plays an important role in the field of Web services technology applications. Fortunately, not all users have completely different service preferences. Due to the same application scenarios and personal interests, some users have the same preferences for certain types of Web services. This paper explores the problem of user clustering in the service environment, grouping users according to their service preferences. It helps service providers to identify and characterize the preferences of similar users and provide them with customized services. We propose two combination-based clustering algorithms which make full use of the advantages of the *K*-means algorithm and the affinity propagation algorithm. In addition, a three-stage clustering process is elaborated to improve the accuracy of user clustering. To reduce the time complexity of the algorithms, we create a parallel execution model of the algorithms implemented by a higher-order MapReduce sequence linking technology. Extensive experiments on simulated datasets and real datasets are performed on the comparisons between the proposed algorithms and the other combination-based clustering algorithms. The experimental results substantiate that the proposed algorithms can effectively distinguish user group with different preferences.

Keywords: clustering algorithm; combination; user preference; Affinity propagation; K-means

Received 14 November 2018; Revised 8 June 2019; Accepted 12 July 2019
Handling editor: Fionn Murtagh

1. INTRODUCTION

With the arrival and rapid development of information science and technology, web services become more convenient, fast and efficient than before. In order to meet the needs of service requesters, a large number of web services with different types are provided by service providers on the Internet platform [1]. Especially when there are many web services with a similar function but different performance in the service environment, some problems have emerged. On the one hand, users cannot choose a satisfactory service based solely on some performance parameters which are given by service providers [2]. On the other hand, service providers cannot efficiently plan and manage their web services [3]. They do not know how to recommend different services for different users with different preferences. For example, homogeneous web services provided by different service providers differ in response time, cost, availability, successful execution and throughput. Different users have different QoS requirements for this type of web services. Some users are concerned about the response time of the service, while others are more concerned about the cost of the service. If the users can be divided into different groups based on their preferences, an individual will be transformed into a node in the form of a group. So a service provider can recommend services, publish messages or post ads to a group according to the common preferences in the group [4]. And a user can select an unknown service based on the recommendations from the users in the same group with him or her [5]. Based on this idea, we will study the clustering methods for grouping users based on their preferences in the paper. Due to the limitation of using a single clustering algorithm in the actual application [6], the presented algorithms combine the *K*-means algorithm with the affinity propagation (AP) algorithm to solve the problem of grouping users based on preference similarity without any prior knowledge.

2. BACKGROUND AND RELATED WORKS

Clustering is a process of grouping a set of objects into classes of similar objects, which has been extensively studied in many areas [7]. At present, there are many clustering methods to divide data nodes into groups [6], such as neural network-based clustering method [8–9], spectral clustering method [10], hierarchical clustering method [11–12], density-based clustering method [13] and partitioning clustering method [14]. Different clustering algorithms have different advantages and disadvantages, which are suitable for different application scenarios. For instance, as a new neural network-based clustering technique, the FACT (fuzzy adaptive clustering technique) is able to consider several similarity criteria to solve the problem of real-world sized complexity, which is trained to cluster machines and parts for cellular manufacturing [8]. The spectral clustering algorithm utilizes the top eigenvectors of a matrix derived from the distance between points to find good clusters, which are widely used in computer vision and VLSI (very-large-scale integration) design [15].

The three other clustering methods, which are hierarchical clustering method, density-based method and partitioning clustering method, are more common, which partition the data nodes according to the similarity or intensity between them [16]. The hierarchical clustering algorithm iteratively merges the most similar pairs until to meet some preset ultimate condition. The density-based algorithm divides a set of nodes into subsets of similar densities [17]. The partitioning clustering algorithm uses graph partition to cluster the nodes, applying iterative optimization to adjust the fitness between nodes so as to get the optimal results, such as K-means [14], AP [18] and fuzzy C-means (FCM) [19].

But the vast majority of single clustering algorithms are sensitive to the initial condition. Because many empirical datasets have the structural imperfections, the hierarchical clustering method is prone to classification errors when the empirical data departs from the ideal conditions of compact isolated clusters [20]. The performance of density-based clustering algorithm depends on two specified parameters: the maximum radius of a neighborhood and the minimum number of the data points contained in such neighborhood [13]. The partitioning clustering algorithm always needs to set the cluster number in advance.

Through the above analysis, the advantages and disadvantages of single clustering algorithm are obvious. In order to take advantage of single clustering algorithm and overcome its limitation on the prior condition, some combination-based clustering methods have been presented. Zhao *et al*. proposed a new technology for a large scale of data clustering based on the combination of Canopy algorithm and K-means algorithm [21]. The main idea of the algorithm is to use the Canopy algorithm to roughly divide the datasets into several intersecting submanifolds and then use the K-means algorithm for a more accurate classification on each submanifold. It can reduce the time complexity of the algorithm greatly and does not need to preset the cluster number. However, a new problem appeared. The two distance threshold parameters T1 and T2 of the actual dataset need to be set in the Canopy algorithm. But it comes from the experience or can be obtained by cross validation, so it is difficult to determine them directly.

In the literature [22–24], the definition of density is introduced into the K-means algorithm. This kind of algorithm processes the isolated points based on distance and statistical method and then adopts a variable radius to select the initial cluster centers, which can optimize the running condition of the K-means algorithm. In [25], the authors integrated the probabilistic neural networks with K-medoids clustering to detect phishing websites. But these methods still need to determine whether the value of K has been accurately set or not, and it has a great impact on the clustering results.

Considering that the AP algorithm does not need to input the estimated cluster number beforehand, some clustering algorithms have been designed to discover the optimal partition result through the combination of AP algorithm and other algorithms. In [26], a novel parameter-free clustering algorithm, named as APSCAN, was presented, which combined the DBCSAN algorithm with the AP algorithm. Du *et al*. divides the clustering process into two phases: coarsening clustering and exact clustering [27]. In the first phase, the AP algorithm is used for coarsening clustering. In the second phase, the self-tuning spectral clustering (SSC) is used to the previous result to perform the exact clustering. However, most clustering algorithms are applicable to a certain field. The above algorithms are mainly used in the field of image recognition.

The goal of this paper is to cluster the users based on their preferences in the service environment. To reduce the sensitivity of the parameters and improve the effectiveness and efficiency of the clustering method, we design two algorithms, called AAK (AP + AP + K-means) and AKK (AP + K-means + K-means), which both combine the K-means algorithm with the AP algorithm to group users based on user preferences. In addition, the MapReduce programming model is used to improve the computational efficiency of the clustering algorithms.

The remainder of this paper is structured as follows. Section 3 formulates the user clustering problem in the service environment. Section 4 elaborates our algorithm, as well as our considerations and insights. Section 5 demonstrates and interprets our evaluations. We conclude this paper in Section 6.

3. PROBLEM DEFINITION AND FORMULATION

In the web service environment with a large number of similar services, user preferences are particularly important for understanding users' needs and characteristics. The most direct data that can reflect a user's preferences is the user's scores on the used services. Meanwhile, the scores are easy to collect in the actual service environment. So mining the implicit knowledge

about users' preferences from the large amount of scoring data has become a subject which has great significance in practical applications. It helps the service providers better understand their customers and provides more suitable services to these customers.

But how to mine a user's complex preferences from some simple scoring data? Clustering the users that have similar scores for the same services is one of the most widely used methods to solve this problem at present. If the users are clustered with the same or similar preferences, we can further analyze the common features of similar users or build credible user recommendation based on a group with similar preferences. Consider an example with two users, A and B, who belong to the same group. If A thinks a service is good, B will like it with great probability.

The user clustering analysis in the web services environment is a process for compartmentalizing. In the process, all users are grouped into some collections of users. There is a high similarity between the intra-cluster users, while the inter-cluster users are quite different. The problem of cluster analysis based on user preferences can be described by the following elements:

$$\Gamma = \{\mathrm{U}, \mathrm{E}, \mathrm{R}, \mathrm{A}, \mathrm{C}\}.$$

The meanings of various symbols in the problem of Γ are given below. U represents a collection of users in the web service environment, which includes the entities that need to be clustered. E represents a collection of all the user scores for the used services in the service environment. R represents some clustering criteria, which are related to user preferences. *A* represents a clustering algorithm, which is used to cluster the elements in U based on R. C represents a clustering results, which is a partition of U in terms of preferences. The formalized definitions of some concepts involved in Γare given.

3.1. Definition 1 (User Set U)

U is a non-empty set, $U = \{u_1, u_2, \ldots, u_n\}$, where n is for the number of users.

3.2. Definition 2 (Service Set S)

S is a non-empty set, $S = \{s_1, s_2, \ldots, s_m\}$, where m is for the number of services.

3.3. Definition 3 (User Score Set E)

Each element e in E can be represented as a triple $< u_x, s_i, g_{xi} >$, $E = \{e | e =< u_x, s_i, g_{xi} >, u_x \in U, s_i \in S, g_{xi} \in R\}$, g is real, representing the score of user u_x for service s_i.

Clustering criterion is the main basis for the clustering analysis. In [28–29], we have focused on the clustering criteria of users in the service environment. The design of criterion is considered from the two directions: merging similar patterns and separating dissimilar patterns. Similarity merging is the most common form of clustering. In *E*, the fact that two users choose a few of the same services suggests that they have a certain similarity in service preferences. The more they choose the same services, the more similar their preferences are. We use it as a merge rule for users.

3.4. Definition 4 (Similarity Rule SR)

Let $N(u_x) = \{s_i | \exists (u_x, s_i, g_{xi}) \in E\}$ be a set of scored records of u_x. For $\forall u_x, u_y \in U$, $u_x \neq u_y$, the similarity between u_x and u_y is a function of $N(u_x)$ and $N(u_y)$, formally [28]:

$$\mathrm{Sim}\left(u_x, u_y\right) = \begin{cases} \frac{|N(u_x) \cap N(u_y)|}{\min\{N(u_x), N(u_y)\}} & N(u_x) \neq 0 \wedge N\left(u_y\right) \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where $|N(u_x)|$ is for the number of services in $N(u_x)$;$|N(u_x) \cap N(u_y)|$ is for the number of services in the intersection of $N(u_x)$ and $N(u_y)$.

Sometimes, users may give different scores for a similar service, which reflects the differences in their preferences. We use it as a separate rule for users.

3.5. Definition 5 (Dissimilarity Rule DR)

For $\forall u_x, u_y \in U$, $u_x \neq u_y$, the dissimilarity between u_x and u_y is a function of $N(u_x)$ and $N(u_y)$, formally [28]:

$$\mathrm{Dis}(u_x, u_y) = \begin{cases} \frac{\sum_{s_i \in N(u_x) \cap N\left(u_y\right)} \mathrm{Abs}\left(g_{xi} - g_{yi}\right)}{|N(u_x) \cap N\left(u_y\right)| * \alpha} & |N(u_x) \cap N(u_y)| \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where $\mathrm{Abs}(g_{xi} - g_{yi})$ *is for the absolute difference of the scores for s_i between u_x and u_y, and* α is *for the normalizing factor (α is set to the maximum of the score difference between users).*

Whether u_x and u_y should be grouped together or not is determined by $\mathrm{Sim}(u_x, u_y)$ and $\mathrm{Dis}(u_x, u_y)$. When $\mathrm{Sim}(u_x, u_y) = 0$ and $\mathrm{Dis}(u_x, u_y) = 0$, u_x and u_y are completely different in terms of service preferences. When $\mathrm{Sim}(u_x, u_y) \neq 0$ but $\mathrm{Dis}(u_x, u_y) = 0$, the clustering criteria lies completely with $\mathrm{Sim}(u_x, u_y)$. Next, we give a comprehensive clustering criterion to cluster the users in the service environment.

3.6. Definition 6 (Preference Criterion R)

For $\forall u_x, u_y \in U$, $u_x \neq u_y$, the clustering criterion between u_x and u_y is a function of $\mathrm{Sim}(u_x, u_y)$ *and* $\mathrm{Dis}(u_x, u_y)$, *formally* [28]:

$$R\left(u_x, u_y\right) = \frac{\mathrm{Sim}\left(u_x, u_y\right)}{e^{\mathrm{Dis}(u_x, u_y)}} \tag{3}$$

The clustering result is to divide all the users in *U* into several non-empty subsets. Each user in *U* belongs to at least one

subnet, and only one subnet. It can be formally described as follows.

3.7. DEFINITION 7 (CLUSTERING RESULT C)

The clustering Result C is a partition of U, formally:

$$C = \{c_1, c_2, \ldots, c_k\},$$

subject to
(i) $c_i \subseteq U$, $c_i \neq \varnothing$ (i=1,2, ...,k);
(ii) $c_i \cap c_j = \varnothing$ ($i \neq j$);
(iii) $\bigcup_{i=1}^{k} c_i = U$.

The different algorithms used in the clustering analysis often result in different results. The termination condition or clustering size of the algorithm is difficult for the user clustering problem. Below, we begin to discuss the clustering analysis and algorithm design in Γ.

4. DESIGN OF CLUSTERING ALGORITHMS

A combination-based clustering algorithm is a kind of advanced clustering algorithms, but this kind of algorithm is only valid in some particular field or has some obvious disadvantages after combination. In order to improve the practicability of algorithm for problemΓ, we firstly do a detailed analysis on the K-means algorithm and the AP algorithm when applying them to cluster the users in the service environments.

4.1. Analysis on related algorithms

K-means is one of the simplest unsupervised learning algorithms that classify a given dataset through a certain number of cluster fixed a priori conditions [14]. When the number of clusters can be accurately estimated, the K-means algorithm has high classification accuracy. The steps of clustering the users through this algorithm are as follows: (i) input the number of clusters k; (ii) select k users as the initial clustering centers; (iii) divide each user into a cluster in which he or she has the highest similarity with the cluster center; (iv) recalculate the clustering centers; (v) repeat (iii) and (iv) until the users within a cluster are no longer changed; and (vi) output the clustering result.

If the value of k is suitable, the K-means algorithm can efficiently divide the users whose preference similarity is high into a group and the users whose preference similarity is low into different groups. However, in the actual service environment, it is difficult to determine the clustering number in advance. Therefore, grouping users only by using the K-means algorithm is impractical.

The AP algorithm is a clustering algorithm based on 'information transfer' between data points. Rather than requiring that the number of clusters be prespecified, the algorithm takes a real number $s(k, k)$ for each data point k as input so that data points with larger values of $s(k, k)$ are more likely to be chosen as exemplars, and then generate the optimal cluster representative based on iterative information between exemplar neighbors [18]. Each user is regarded as a potential candidate cluster center. Two message parameters are passed in this algorithm: responsibility and availability [18]. The $R(i, k)$ represents the numerical message sent from u_i to u_k, which describes whether u_k is appropriate as the clustering center of u_i, while $A(i,k)$ represents the numerical message sent from u_k to u_i, which describes whether u_i selects u_k as its clustering center. The greater the value of $R(i, k)$ and $A(i, k)$ are, the greater likely u_k is used as the clustering center of u_i. The AP algorithm updates the responsibility and availability of each user according to the similarity matrix between users in the iterative process until it no longer produces the new clustering centers. Finally, each user is assigned to a corresponding group.

The biggest problem of the AP algorithm is that all eligible users can be selected as the clustering centers, which make it generate too many groups. So it is unable to obtain the ideal result of user groups.

Through analysis of the above two algorithms, there is no way to overcome the disadvantages caused by the algorithms when using a single clustering algorithm. The key of the combination-based clustering algorithm is to overcome the shortcomings of the single clustering algorithm so as to improve the accuracy of clustering. A large number of experimental results show that if we can take the appropriate clustering number as an input to the K-means algorithm, it will be able to effectively partition the users in the service environment based on R. But the biggest disadvantage of the K-means algorithm is that it cannot automatically determine the appropriate clustering number. As an exploratory analysis method, the AP algorithm can conduct an exploratory cluster analysis without any prior knowledge. This advantage of the AP algorithm can compensate for the disadvantages of the K-means algorithm. We design two clustering algorithms through combining the K-means with the AP algorithm in this paper, called AAK and AKK. The basic idea of the algorithms is to firstly determine the most suitable clustering number and then cluster users through a single algorithm.

4.2. Design of AAK and AKK

The input of the AAK algorithm and the AKK algorithm are both U and E. In the process of algorithm implementation, user clustering is divided into three stages. Each stage uses a single clustering algorithm separately. The implementation flow of the AAK and AKK algorithms is shown in Fig. 1.

AAK algorithms: in the first stage, the AP algorithm is used to obtain the maximum cluster number $K_{\max}$. It means that all users can be divided into $K_{\max}$ groups at most. In the second stage, the number K_{best}, which is the optimal cluster number, is

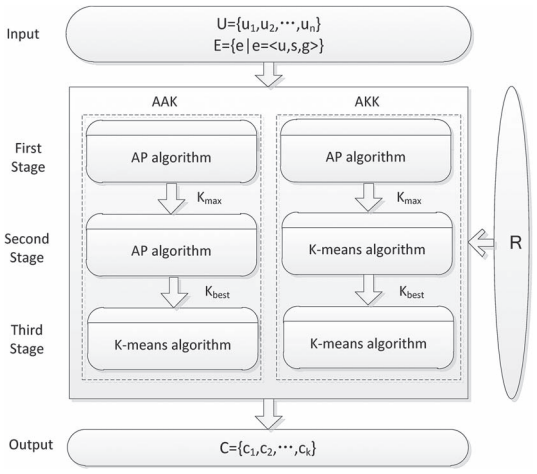

FIGURE 1. Flow chart of AAK algorithm.

searched in the range [2, $K_{\max}$] by using the AP algorithm. In the third stage, the K-means algorithm is implemented, which takes K_{best} as the initial cluster number, to get the ideal results of grouping users.

AKK algorithm: the first and third stages of this algorithm are the same as those of the AAK algorithm, and the difference is in the second stage. In this stage, the value of K_{best} is searched in the range [2, $K_{\max}$] by using the K-means algorithm, not the AP algorithm.

In the above process, the BWP (between–within proportion) index [30], a clustering validity index, is used to evaluate the quality of user group structure. In [30], BWP is used to determine the optimal number of clusters in the K-means clustering algorithm. It evaluates the results based on the degree of separation between groups and the degree of compactness inside a group, which is independent of special algorithms. When the cluster number equals k, the average value of clustering validity is calculated by Equation (4) [31].

$$\text{AvgBWP}(k) = \frac{1}{n}\sum_{j=1}^{k}\sum_{i=1}^{n_j}\text{BWP}\,(j,i) \tag{4}$$

Of these, n is the number of users and k is the cluster number; n_j is the number of users in group j. The BWP value of u_i in group j is denoted by BWP(j,i). The formula is

$$\text{BWP}\,(j,i) = \frac{bs\,(j,i) - ws\,(j,i)}{bs\,(j,i) + ws\,(j,i)} = \frac{bsws\,(j,i)}{baws\,(j,i)} \tag{5}$$

where $bs(j,i)$ is the minimum average similarity between u_i in group j, and the users in other groups, called the degree of separation between groups, $ws(j,i)$, is the average similarity in group j that u_i belongs to, called the degree of compactness inside a group.

In the second stage of AAK, BWP needs to be calculated in every clustering iteration. After all iterations ends, all the values of BWP will be compared. The greater the absolute value of BWP is, the better the effect of clustering results is. Therefore, K_{best}, which makes avgBWP(k) reach its maximum value, is the optimal clustering number. Take K_{best} as an input of the K-means algorithm in the third stage so as to achieve the accurate result.

In the second stage of AKK, the K-means algorithm is used to group the users when the initial cluster number is set from 2 to $K_{\max}$. Each clustering result is evaluated through the BWP index. When a result makes the BWP reach its maximum value, the corresponding clustering number is optimal.

The description of two algorithms is as follows.

Algorithm: AAK and AKK

Input: U, E
Output: C = {c1, c2, . . . , ck}
Procedure Preparation (U, E).
{
1: S ← *Sim*(*U*, *E*);//according to Equation (1).
2: D ← *Dis*(*U*, *E*);//according to Equation (2).
3: P ← *Preference*(S, D);//according to Equation (3).
4: return(P);
}

Procedure AP1 (P, lam, maxiter, con_iter).
//'maxiter' Maximum number of iterations.
//'con_iter' If the exemplars stay fixed for con_iter iterations,
AP1 terminates early.
//'lam' Damping factor.
{
5: θ ← *median*(*P*);//Set preference to median similarity.
6: X ← *Setmatrix*(θ, *P*);
7: [R1 A1] = Initialization (X);
8: while (curm<= maxiter)&& (curc<=con_iter).
{
9: R_1 ← *Update*(A_1, *X*);//update responsibilities.
10: A_1 ← *Update*(R_1, *X*);//update availabilities.
11: for i = 1 to n.
{
12: if R(i, i)' + A(i, i)' > 0.
13: *Cen* ← *SetCenter*(*i*);//Generate the cluster centers
}//end for.
14: curm = curm+1;
15: if *Unchange*(*Cen*).
16: curc = curc+1;
}//end while.
17: $K_{\max}$ = *count*(*Cen*);
18: return($K_{\max}$,X);
}

Procedure AP2 (X, lam).

```
    //'lam' Damping factor.
    {
19:     [R2 A2] = Initialization (X);
20:     for curt = 2 to K_max.
    {
21:     R_2 ← Update(A_2, X);//update responsibilities.
22:     A_2 ← Update(R_2, X);//update availabilities.
23:     for i = 1 to n.
    {
24:     cn = 0;
25:     if R(i, i)' + A(i, i)' > 0.
26:     Cen ← SetCenter(i);//Generate the clustering centers
    }//end for.
27:     C = cluster(Cen, curt);//Generate the clustering result.
28:     BWP(curt) = evaluation(C);//evaluate the clustering
result.
    }//end for.
29:     K_best = Max(BWP);//Compare the BWP
30:     return(K_best);
    }

    Procedure K-means (P, K_best, kmaxiter, kcon_iter).
    //'kmaxiter 'Maximum number of iterations.
    //'kcon_iter' If the all the users stay fixed for con_iter
iterations, K-means terminates early.
    {
31:     C ← InitialCluster(K_best);
32:     while (curm<= kmaxiter)&& (curc<=kcon_iter).
    {
33:     For x = 1 to n
    {
34:     For i = 1 to K_best
    {
35:     Avgpre(u_x, c_i) = Σ_{u_j ∈ c_i} P(u_j, u_x)/|c_i|;
36:     Add(Max(Avgpre(u_x, c_i), u_x, c_i);
    }//end for.
    }//end for
37:

                C_best = Generate () ;

38:     curm = curm+1;
39:     if Unchange(Cen).
40:     curc = curc+1;
    }//end while.
41:     return(C_best);
    }

    Procedure K-means2(P,K_max,kmaxiter, kcon_iter).
    //'kmaxiter 'Maximum number of iterations.
    //'kcon_iter' If the all the users stay fixed for con_iter.
    //iterations, K-means1 terminates early.
    {
42:     for i = 2 to K_max.
    {
43:

            C ← K-means (P, i, kmaxiter, kon_iter) ;

44:     BWP(i) ← evaluation(C);
    }
45:     K_best = Max(BWP);
    }

    Procedure AAK(U, E).
    {
46:     P=Preparation (U, E);
47:     [K_max,X] = AP1 (P, lam, maxiter, con_iter);
48:     K_best= AP2(X, lam);
49:     C = K-means3 (P,K_best,kmaxiter, kcon_iter);
    }

    Procedure AKK(U, E).
    {
50:     P=Preparation (U, E);
51:     [K_max,X] = AP1 (P, lam, maxiter, con_iter);
52:     K_best= K-means(P,K_max,kmaxiter, kcon_iter);
53:     C = K-means(P, K_best, kmaxiter, kcon_iter);
    }
```

It can be seen from the description of the algorithms that neither the AAK algorithm nor the AKK algorithm needs to preinput the clustering number and preset the clustering centers. *Preparation* is responsible for preprocessing the data extracted from the service environment and calculating the preference similarity between users based on R. The main task of *AP1* is to select all the users that might become clustering centers, assigning the number of clustering centers to $K_{\max}$. *AP2* or *K-means (line 52)* is used to find the most appropriate clustering number within a smaller range from 2 to $K_{\max}$. *K-means (line 53)* divided the users into K_{best} groups based on R. After three stages of clustering analysis, the AAK algorithm and the AKK algorithm can improve the accuracy of clustering result without prior knowledge.

Although the tasks of each phase of the two algorithms are the same, their time complexity is different. The time complexity of the first and third stages of the algorithms is the same as that of the AP algorithm and the K-means algorithm, respectively. The difference between the two algorithms is the time complexity of the second stage. In the second stage of the AAK algorithm, all the cluster representatives in the result of *AP1* are sorted according to their probabilities of being selected as clustering centers. In each iteration of *AP2,* only the first t users, which changes from 2 to $K_{\max}$, are selected as clustering centers. So its time complexity is the same as that of the AP algorithm. Unlike the AAK algorithm, the AKK algorithm determines the optimal clustering number by the K-means algorithm. In this process, the K-means algorithm is executed about $K_{\max}$ times, as shown in lines 42–44. So its time

complexity is $K_{\max}$ *O (K-means). We cannot compare the time complexity of the AAK and AKK algorithms without knowing the distribution of data. In different scenarios, both algorithms have their advantages and disadvantages.

4.3. MapReduce model of the algorithms

Due to the higher complexity of the combinatorial clustering algorithm, the serial computation using a single processor will not be able to meet the actual needs of computational ability or internal storage with the expanding of user scale in the service environment. MapReduce, which is a concurrent programming model, can extend data processing applications to multiple compute nodes [32]. Therefore, a parallel programming implementation of the combination-based clustering algorithm on the distributed system infrastructure platform (that is, Hadoop) is finished, and the related experiments are conducted to verify the execution effect in this paper.

At first, the whole data processing task needs to be divided into several relatively independent subtasks when the logic of the data processing flow is complex. Then, each subtask is accomplished by a separate MapReduce operation. Hadoop can provide a variety of link technology to make multiple MapReduce jobs, for example, sequential link technology and complex dependency links technology [33]. In this paper, the process of the combination-based clustering algorithm is divided into two relatively independent MapReduce tasks by means of sequential link technology. The second stage of algorithm is completed by the first MapReduce task, and the third stage is completed by the other MapReduce task. The description of the parallel MapReduce implementation model is shown in Fig. 2. The algorithm is completed by two order-linked MapReduce tasks; the output of the first MapReduce is the input of the second MapReduce. MapReduce1 is used to find out an optimal number of clusters from the AP algorithm or K-means algorithm; MapReduce2 is used to cluster the users based on their preferences using the K-means algorithm borrowing the number of k from the previous step.

MapReduce1: Its main task is to realize the parallelization in the second stage of the algorithms.

For the AAK algorithm, upload similarity matrix P and a percentage of k in the range of [2, $K_{\max}$] to the Map 1 nodes, then execute them in parallel, and finally obtain an optimal clustering number. Mapper1 is responsible for calculating and finding the maximum of BWP in the range of $K_{\max}$, which generates some key-value pairs <Key, Value> and passes them to Shuffle1 as the intermediate result. In the key-value pairs, Key is the value of k, Value is the value of AvgBWP(k). In the process of Shuffle1, the key-value pairs are grouped and sorted as the input of Reduce1. Reduce1 is responsible for comparing all the key-value pairs and finding the maximum of BWP and the corresponding k. The final output of Reduce1 is the value of K_{best} and the corresponding BWP.

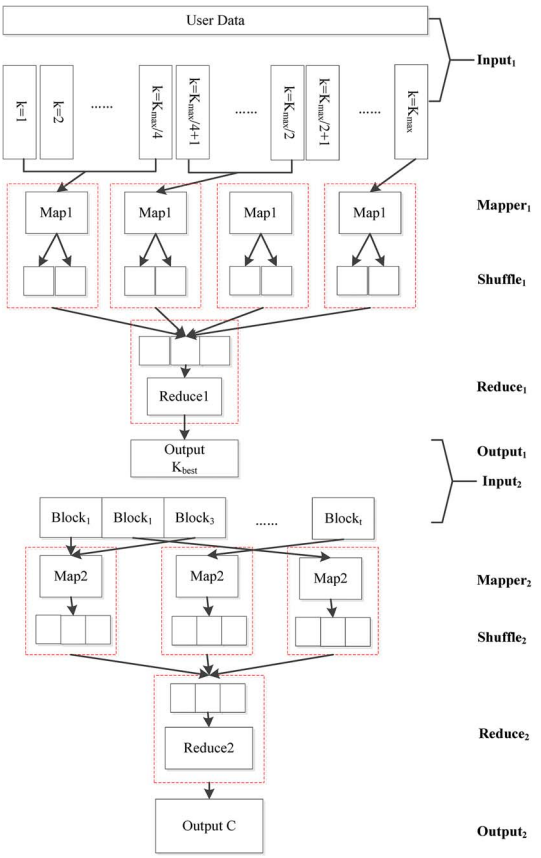

FIGURE 2. Parallel processing of AAK algorithm.

For the AKK algorithm, upload matrix P to HDFS and then execute them in parallel, and finally obtain an optimal clustering number. Mapper1 is responsible for calculating which group each user should be divided into. After clustering, output the key-value pairs <Key, Value>, where Key is for group ID and Value is for user ID. Shuffle1 is responsible for sorting the results of Map 1, and the results are passed to Reduce1. Reduce1 is responsible for merging the users belonging to the same group, evaluating the validity of the clustering results and calculating the value of AvgBWP(k). After comparing all the key-value pairs, output the maximum of BWP and the corresponding k.

MapReduce2: When completing the above MapReduce1, MapReduce2 starts with the output of MapReduce1. The MapReduce2 process of the AKK algorithm is the same as that of the AKK algorithm. Upload the user data blocks and the files about initial clustering centers to each Map 2 node, and then execute them in parallel. Mapper2 is responsible for assigning each user of the blocks to a group having the highest preference similarity with him or her. It generates some key-value pairs <Key, Value> and passes them to Reduce2 as the intermediate result. In the key-value pairs, Key is a user ID and

Value is a group ID. Reducer2 is responsible for merging all the users that belong to a group and comparing the clustering result with that in previous iteration. If the clustering result is unchanged after *kcon_iter* consecutive iterations, the task is completed; otherwise, it still needs further iterations. After MapReduce2 is completed, the grouping result can be output from HDFS.

5 EXPERIMENTS AND ANALYSIS

In this section, we design some experiments to test the effect and performance of the AKK algorithm (Available at https://github.com/ndcswy/Cluster/tree/ndcswy-patch-1) and AAK algorithm (Available at https://github.com/ndcswy/Cluster/tree/ndcswy-patch-2). The first kind of experiments will cluster users according to their preferences on simulated dataset and real dataset, respectively. And then we will prove the effectiveness of the algorithm through comparing with other clustering algorithms. The second kind of experiments will design the parallel model of the two algorithms on the Hadoop platform. And then we will compare the performance of the improved algorithms with the original algorithms under different clustering scales. In order to verify the accuracy of the two algorithms, we compared them with the Canopy-*K*-means algorithm [21] and the density-based *K*-means algorithm [22]. The *D*-function comparing method is used as the evaluation standard of the experimental results [10]. The value of *D*-function is the average difference between sets, ranging in [0, 1]. If the grouping result is exactly the same with the preset groups, $D = 0$, instead, $D = 1$. The two types of datasets are used in the experiments: simulated datasets and real datasets.

5.1 Experiments on simulated datasets

The experiments are divided into three groups. In the first group, the users' preferences are obvious. So it is easy to divide the users into groups because the users in different virtual groups have obvious difference. Assume there are some users in the service environment, which belong to four groups, as shown in Fig. 3. The users in a group are highly similar, but the users in different group are obviously dissimilar.

The experimental results under a different user scale are shown in Table 1. In Table 1, n represents the number of users, K_{default} represents the number of initial virtual groups, K_{best} represents the number of groups in the results and D is calculated by D function, which represents the similarity between initial virtual group structure and the experimental result. From Table 1, the effects of the AAK algorithm and AKK algorithm are both better than two other clustering algorithms. Under a different user scale, their results are consistent with the preset.

In the second group, some users' preferences are fuzzy, as shown in Fig. 4. In this figure, a user is similar with the users from two or more different groups simultaneously, such as User 3 and User 6. The experimental results under different user scale are shown in Table 2. In the experimental results, the fuzzy users are divided randomly into a group which has higher similarity with them.

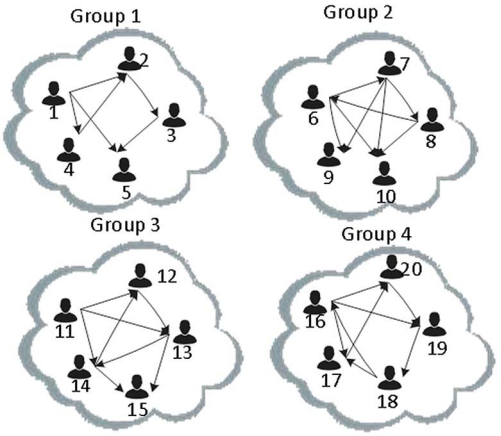

FIGURE 3. Structure map of simulated Dataset 1.

TABLE 1. The results of simulated Dataset 1.

		K_{default}	K_{best}	D
Canopy-*K*-means	$n = 20$	4	3	0.06
	$n = 200$	5	8	0.14
	$n = 2000$	10	25	0.18
	$n = 20000$	50	94	0.39
Density-based	$n = 20$	4	6	0.12
K-means	$n = 200$	5	12	0.18
	$n = 2000$	10	36	0.21
	$n = 20000$	50	129	0.27
AAK	$n = 20$	4	4	0.00
	$n = 200$	5	5	0.00
	$n = 2000$	10	10	0.00
	$n = 20000$	50	50	0.00
AKK	$n = 20$	4	4	0.00
	$n = 200$	5	5	0.00
	$n = 2000$	10	10	0.00
	$n = 20000$	50	50	0.00

Therefore, K_{best} may be different with K_{default} in the results. But compared with the two other algorithms, our algorithms still have a high accuracy.

In the third group, a user's service choice is randomly generated, as shown in Fig. 5. The experimental results are shown in Table 3. Because the users' preferences are not obvious, the users are divided into many groups, and the value of D is significantly higher than that in the results of Dataset 1 and Dataset 2. But the AAK algorithm and the AKK algorithm still have an ability to divide some users that have selected more same services into a group. The other two algorithms perform less well in this aspect.

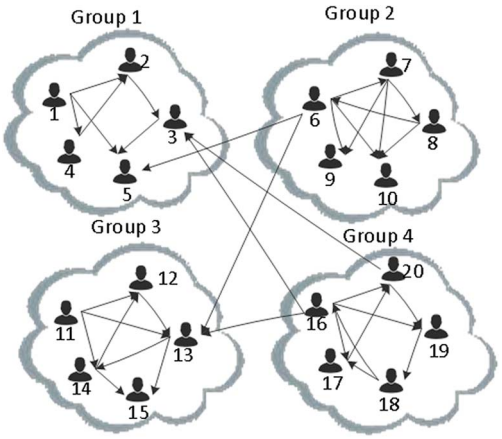

FIGURE 4. Structure map of simulated Dataset 2.

TABLE 2. The results of simulated Dataset 2.

		$K_{default}$	K_{best}	D
Canopy-K-means	$n = 20$	4	3	0.09
	$n = 200$	5	7	0.24
	$n = 2000$	10	25	0.21
	$n = 20000$	50	80	0.49
Density-based	$n = 20$	4	7	0.14
K-means	$n = 200$	5	19	0.26
	$n = 2000$	10	31	0.29
	$n = 20000$	50	166	0.34
AAK	$n = 20$	4	4	0.02
	$n = 200$	5	7	0.06
	$n = 2000$	10	16	0.11
	$n = 20000$	50	60	0.14
AKK	$n = 20$	4	4	0.02
	$n = 200$	5	8	0.05
	$n = 2000$	10	11	0.09
	$n = 20000$	50	53	0.11

5.2 Experiments on real datasets

We adopted the MovieLens dataset as the real dataset, containing 18 genres in 1682 movies. At first, the similarity matrix between users is calculated based on the users' scores. However, some of the genres cannot form a statistical scale because few users focus on them. So, all the users are divided into 14 groups on the basis of the genres. The results on real datasets are shown in Table 4.

In Table 4, n represents the number of users who come from different genres, $K_{default}$ represents the number of actual genres and K_{best} represents the number of groups in the results. In the real dataset, because of the high randomness of the users' choices, the group feature of the users turns much fuzzier. Therefore, the value of D becomes larger in the clustering results than that in the simulated dataset.

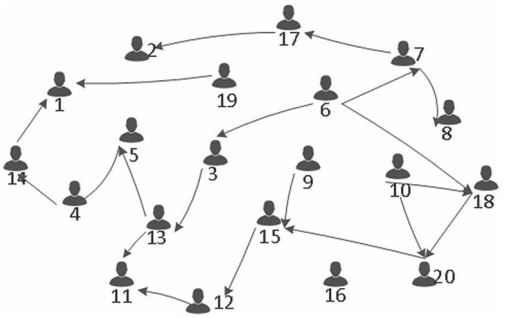

FIGURE 5. Structure map of simulated Dataset 3.

TABLE 3. The results of simulated Dataset 3.

		K_{best}	D
Canopy-K-means	$n = 20$	15	0.79
	n = 200	172	0.66
	$n = 2000$	1025	0.81
	$n = 20000$	4880	0.91
Density-based	$n = 20$	12	0.65
K-means	$n = 200$	119	0.86
	$n = 2000$	1026	0.89
	$n = 20000$	8665	0.94
AAK	$n = 20$	7	0.12
	$n = 200$	62	0.46
	$n = 2000$	698	0.75
	$n = 20000$	3652	0.78
AKK	$n = 20$	6	0.1
	$n = 200$	45	0.35
	$n = 2000$	406	0.52
	$n = 20000$	2223	0.65

TABLE 4. The results of real Dataset.

		$K_{default}$	K_{best}	D
Canopy-K-means	$n = 189$	14	12	0.23
	$n = 754$	14	16	0.29
	$n = 943$	14	31	0.40
Density-based	$n = 189$	14	29	0.09
K-means	$n = 754$	14	32	0.47
	$n = 943$	14	65	0.76
AAK	$n = 189$	14	11	0.12
	$n = 754$	14	16	0.24
	$n = 943$	14	23	0.27
AKK	$n = 189$	14	11	0.12
	$n = 754$	14	14	0.20
	$n = 943$	14	16	0.17

The experimental results showed that the AAK algorithm and AKK algorithm have significantly higher accuracy and practicability than the other two algorithms. The maximum

TABLE 5. The comparison of running time.

	$n = 10000$	$n = 20000$	$n = 50000$
AAK (second)	356430	1042576	3481263
MapReduce-AAK (second)	171935	260531	632957
AKK (second)	374260	1040658	3014523
MapReduce-AKK (second)	172602	302306	570895

D value of the AAK algorithm and AKK algorithm is 0.27 and 0.20, respectively, which is far below that of Canopy-K-means algorithm and the density-based K-means algorithm.

5.3 Experiments of MapReduce model of the algorithms

Both the AAK algorithm and the AKK algorithm are composed of three serial stages, so the computational complexity of each algorithm depends on the stage with the highest complexity. For the AAK algorithm, the main flow of the AP algorithm is executed in the first two stages, so its computational complexity is the same as that of the AP algorithm. In the last stage, the main flow of the K-means algorithm is executed, so its computational complexity is the same as that of the K-means algorithm. Therefore, the computational complexity of the AAK algorithm is $O(N^*N^*\log N+N^*N^*\log N+K^*N)$, where N is the number of users, K is the clustering number of users, $O(N^*N^*\log N)$ is the computational complexity of the AP algorithm and $O(K^*N)$ is the computational complexity of the K-means algorithm. The AKK algorithm is similar to the AAK algorithm. According to the above analysis, the computational complexity of the combined clustering algorithm is similar to that of the AP algorithm. In order to improve the clustering efficiency, a clustering model based on MapReduce is proposed in Section 4.3. In this experiment, we implemented the model on the Hadoop platform.

In order to further validate the efficiency of MapReduce implementation of the two algorithms, the running time of serial implementation and MapReduce implementation of the algorithms when processing large-scale users is shown in Table 5. The dataset used for Table 5 is identical to that used for Table 1. The numbers of parallel processes in MapReduce1 and MapReuce2 are decided by $K_{\max}$ and K_{best}, respectively. It can be seen from Table 5 that the MapReduce method has greatly shortened the operation time of the combination-based clustering algorithms. When $n = 10\,000$, the implementations of the MapReduce model-based algorithms are about 50% less than their serial implementations. As the number of users increases, the algorithms based on the MapReduce model save more time than their serial implementations. When $n = 50\,000$, the running time of the algorithms based on the MapReduce model are about 20% of their serial implementations.

6 CONCLUSION

In order to overcome the shortcomings of using the K-means algorithm or the AP algorithm alone to group users in the service environment, two fast and efficient combination-based algorithms are presented in this paper. The algorithms can not only discover the group structure of the users in the service environment without any prior knowledge but also shorten the running time with its MapReduce implementation. Through a series of experiments, the clustering algorithms which combine the AP algorithm with the K-means algorithms can rapidly and accurately divide the users into the appropriate groups based on preference criterion and have certain robustness. Compared with the Canopy-K-means algorithm and the density-based K-means, our algorithms are more applicable to the clustering problem of the users in the service environment and have a broad prospect in applications.

FUNDING

Natural Science Foundation of China (61662054, 61262082); Research Program of Science and Technology with the Universities of Inner Mongolia Autonomous Region (NJZY008); Inner Mongolia Science and Technology Innovation Team of Cloud Computing and Software Engineering; Inner Mongolia Application Technology Research and Development Funding Project (201702168); CERNET Innovation Project (NGII20160511); Inner Mongolia Engineering Lab of Cloud Computing and Service Software; Inner Mongolia Engineering Lab of Big Data Analysis Technology.

REFERENCES

[1] Ding, S., Xia, C., Cai, Q. and Zhou, K. (2014) QoS-aware resource matching and recommendation for cloud computing systems. *Appl. Math. Comput.*, 247, 941–950.
[2] Zhao, L., Ren, Y., Li, M. and Sakurai, K. (2012) Flexible service selection with user-specific QoS support in service-oriented architecture. *J. Netw. Comput. Appl.*, 35, 962–973.
[3] Liu, T., Lu, T. and Wang, W. (2012) SDMS-O: A service deployment management system for optimization in clouds while guaranteeing users' QoS requirements. *Future Gener. Comp. Sy.*, 28, 1100–1109.
[4] Chang, D. and Wang, T. (2012) Consumer preferences for service recovery options after delivery delay when shop-ping online. *Soc. Behav. Personal. Int. J.*, 40, 1033–1043.
[5] Kim, M., Song, W., Song, S. and Kim, E. (2012) Efficient collaborative recommendation with users clustered for IPTV Services. In *Proceedings of ICHIT 2012, Daejeon, Korea, 23–25 August*, pp. ∼409–∼416. Springer-Verlag, Berlin.
[6] Podgorski, K. (2014) Advances in machine learning and data mining for astronomy. *Int. Stat. Rev.*, 82, 153–154.
[7] Chen, J., Tan, J. and Liao, H. (2012) Meaningful string extraction based on clustering for improving webpage classification. *China Commun.*, 9, 68–77.

[8] Chang, C. and Chen, S. (2015) A comparative analysis on artificial neural network-based two-stage clustering. *Cogent Eng.*, 2, 995785.
[9] Liu, A. and Zhu, Q. (2011) Automatic modulation classification based on the combination of clustering and neural network. *J. China Univ. Posts Telecom.*, 18, 13–19.
[10] Zhang, P., Li, M. and Wu, J. (2006) The analysis and dissimilarity comparison of community structure. *PhysicaA*, 367, 577–585.
[11] Maqbool, O. and Babri, H. (2007) Hierarchical clustering for software architecture recovery. *IEEE T. Software Eng.*, 33, 759–780.
[12] Lai, C. (2005) A novel clustering approach using hierarchical genetic algorithms. *Intell. Autom. Soft. Co.*, 11, 143–153.
[13] Lv, Y., Ma, T., Tang, M., Cao, J., Tian, Y., Al-Dhelaan, A. and AI-Rodhaan, M. (2016) An efficient and scalable density-based clustering algorithm for datasets with complex structures. *Neurocomputing*, 171, 9–22.
[14] Kanungo, T., Mount, D. and Netanyahu, N. (2000) An efficient K-means clustering algorithm: Analysis and implementation. *IEEE T. Pattern Anal.*, 24, 881–892.
[15] Corporation, H. (2011) A new fuzzy clustering algorithm based on clonal selection for land cover classification. *Math. Probl. Eng.*, 4, 253–166.
[16] Velmurugan, T. and Santhanam, T. (2011) A survey of partition based clustering algorithms in data mining: An experimental approach. *Info. Tech. J.*, 10, 478–484.
[17] Huang, S. (2013) A graph partition-based clustering algorithm for mixed attributes data. *Comput. Appl. Softw.*, 30, 11–17.
[18] Frey, B. and Dueck, D. (2007) Clustering by passing messages between data points. *Science*, 315, 972–976.
[19] Zhou, M., Xu, Y. and Ma, L. (2010) Radio-map establishment based on fuzzy clustering for WLAN hybrid KNN/ANN indoor positioning. *China Commun.*, 7, 64–80.
[20] Mangiameli, P., Chen, S. and West, D. (1996) A comparison of SOM neural network and hierarchical clustering methods. *Eur. J. Oper. Res.*, 93, 402–417.
[21] Zhao, Q. (2014) Efficient algorithm of canopy-Kmeans based on Hadoop platform. *Electron. Sci. Tech.*, 27, 29–31.
[22] Duraiswamy, K. (2010) A novel density based improved K-means clustering algorithm – Dbkmeans. *Int. J. Comput. Sci. Eng.*, 2, 213–218.
[23] Wang, L., Bo, L. and Jiao, L. (2006) A modified K-means clustering with a density-sensitive distance metric. *Lect. Notes Comput. Sci.*, 4062, 544–551.
[24] Zheng, C., Miao, D. and Wang, R. (2009) Improved rough K-means clustering algorithm with weight based on density. *Comput. Sci.*, 36, 219–222.
[25] EI-Alfy and EI-Sayed, M. (2018) Detection of phishing websites based on probabilistic neural networks and K-medoids clustering. *Comput. J.*, 60, 1745–1759.
[26] Chen, X., Liu, W. and Qiu, H. (2009) APSCAN: A parameter free algorithm for clustering. *Pattern Recogn. Lett.*, 32, 973–986.
[27] Du, H., Wang, Y. and Dong, X. (2015) Texture image segmentation using affinity propagation and spectral clustering. *Int. J. of Pattern Recogn. Artif. Intell.*, 29, 1555009.
[28] Wang, Y., Zhou, J. and Tan, H. (2015) CC-PSM: A preference-aware selection model for cloud service based on consumer community. *Math. Probl. Eng.*, 8, 1–13.
[29] Wang, Y., Zhou, J., Li, X. and Song, X. (2017) Grouping users using a combination-based clustering algorithm in the service environment. In *Proceedings of IEEE ICWS 2017, Hawaii, USA, 25–30, June*, pp. 721–727. IEEE Computer Society, Washington, DC, USA.
[30] Zhou, S.X.Z. and Tang, X. (2010) Method for determining optimal number of clusters in K-means clustering algorithm. *J. Comput. Appl.*, 30, 1995–1998.
[31] He, N., Liu, M. and Zhao, F. (2015) A Chinese dishes recommendation algorithm based on personal taste. In *Proceedings of CYBCONF 2015, Gdynia, Poland, 24–25, June*, pp. 277–280. IEEE Computer Society, Washington, DC, USA.
[32] Yang, X. (2011) Reliable estimation of execution time of MapReduce program. *China Commun.*, 8, 11–18.
[33] Fernández, A., Río, S. and López, V. (2014) Big data with cloud computing: An insight on the computing environment, MapReduce and programming frameworks. *Wires. Data. Min. Knowl.*, 4, 380–409.