Applying Evolutionary-based User Characteristic Clustering and Matrix Factorization to Collaborative Filtering for Recommender Systems

R. J. Kuo*, Zhen Wu

Department of Industrial Management, National Taiwan University of Science and Technology, Taiwan
rjkuo@mail.ntust.edu.tw, 382316799@qq.com

Abstract

In recent years, with the rise of numerous Internet service industries, recommender systems have been widely used as never before. Users can easily obtain the information, products or services they need from the Internet, and businesses can also increase additional revenue through the recommender system. However, in today's recommender system, the data scale is very large, and the sparsity of the scoring data seriously affects the quality of the recommendation. Thus, this study intends to propose a recommendation algorithm based on evolutionary algorithm, which combines user characteristic clustering and matrix factorization. In addition, the exponential ranking selection technology is employed for evolutionary algorithm. The experiment result shows that the proposed algorithm can obtain better result in terms of four indicators, mean square error, precision, recall, and *F* score for two benchmark datasets.

Keywords: Recommender systems, Collaborative filtering, Evolutionary algorithm, User characteristic clustering, Matrix factorization

1 Introduction

With the development of the Internet, more and more users can get the information, products or services they need easily. In addition, users will also leave a lot of comments, ratings, browsing history and other explicit or implicit records on the Internet. Recommender systems uses different sources of information to provide users with predictions and suggestions for items [1]. The goal of the recommender system is to make use of these evaluation records to explore user preferences as much as possible and recommend products to users that they may like. This is not only a product marketing tool, but also can increase the user's dependence to the site.

Data sparseness and poor scalability have always been the key factors affecting the quality of the recommender system. It means though there are many products can be selected by users, yet just a few products are reviewed. In order to solve this problem and accurately find similar users for company, clustering analysis is introduced to optimize the collaborative filtering algorithm. Clustering analysis is a kind of data mining techniques which is able to group the similar users tighter. Basically, the customers belonging to the same cluster may share the similar characteristics. It does not decompose the rating matrix into small-rank matrices, but reduce the search space by clustering similar users or items together [2].

Therefore, this study mainly focuses on the improvement of matrix factorization collaborative filtering for recommender system. On the basis of model proposed by Navgaran *et al*. [3], their evolutionary-based matrix factorization (EAMF) method is improved and combined with user clustering characteristic method. The proposed method, which eliminates the need to consider the entire rating matrix, can improve the quality of recommendation and speed up the process. In addition, most of evolutionary algorithm has a weakness which is to get stuck to the local minimum. Thus, this study improves the original evolutionary algorithm by using different selection mechanisms. Thereafter, two benchmark datasets are applied to evaluate the performance of the proposed algorithm in terms of mean square error (MSE), precision, recall, and *F* score.

2 Literature Review

This section will provide some reviews of recommender system and evolutionary computation algorithm.

2.1 Recommender Systems

With the development of information technology and the Internet, mankind has moved from an era of lack of information to an era of information overload, which can be solved by using recommender system. After years of accumulation, recommender systems have been well applied in many fields, such as e-government, e-business, e-learning, e-commerce, etc [4].

Basically, the mainstream recommendation algorithms are categorized into three groups: content-based, collaborative filtering and hybrid recommendation algorithms. The basic idea of content-based recommendation is to recommend items that are similar to the user's interest [5]. Collaborative filtering (CF), a state-of-the-art method to build RS, predicts a user's rating or preference on a candidate item based on other similar users and items. CF algorithm realizes recommendation by calculating the similarity between the users and/or the items [6]. The algorithms in this category are divided into two sub-categories of memory-based and model-based collaborative filtering recommenders. In memory-based CF, the method that emphasizes on the similarity between users is called user-

*Corresponding Author: R. J. Kuo; E-mail: rjkuo@mail.ntust.edu.tw
DOI: 10.53106/160792642022072304005

based CF [7]. Model-based CF algorithms use different techniques on the training data in order to find patterns in the data and learn a model for predicting new ratings [8]. Each recommender method has its pros and cons. In order to make the recommendation algorithm perform better in practical applications, a hybrid recommendation technology was proposed. This technology combines the best characteristics of two or more recommendation technologies into a hybrid recommendation technology [9].

In recent years, the development of machine learning and deep learning has provided methodological guidance for recommender systems. Karatzoglou *et al.* [10] introduced an application of deep learning in recommender systems, and presented content recommendation and collaborative filtering recommendation methods. In today's recommender system, the scale of data is very large and the sparsity of rating data seriously affects the quality of rating. This problem cannot be solved in nature. However, some methods can be adopted to alleviate this problem to a certain extent [11]. For example, for the vacant values of original data, some ratings can be pre-filled, so that the accuracy of the similarity measurement can be improved, thereby improving the accuracy of the algorithm [12]. The cold start is a common problem in recommender systems. This problem is related to recommendations for novel users or new items. In case of new users, the system does not have information about their preferences in order to make recommendations [13]. To deal with new user problem, it is often necessary to use the user's personal statistical information to improve the recommendation results [14]. However, due to personal privacy, users are generally unwilling to provide detailed personal information publicly. Therefore, the recommender system algorithm that protects the privacy of users has also become a research hotspot. Yin *et al.* [15] proposed an efficient privacy-preserving collaborative filtering algorithm, which is based on differential privacy protection and time factor. Wu *et al.* [16] proposed to generate a group of fake preference profiles, so as to cover up the user sensitive subjects, and thus protect user personal privacy in personalized recommendation.

2.1.1 Collaborative Filtering

As mentioned in Section 2.1, collaborative filtering can be divided into memory-based and model-based approaches. The memory-based approach uses the similarity between users or items to find their relevant neighbors. Then, it uses these neighbors to recommend or predict items (users). Therefore, this method is also called neighbor-based collaborative filtering. The most commonly used similarities are Pearson's correlation coefficient, Cosine, Jaccard coefficient and so on. Some studies also tried to improve the traditional similarity calculation method by suggesting different similarity. For instance, Jin *et al.* [17] proposed a method to adjust nonlinear equations by using singularity factor, and considered the scoring habits of users. On the other hand, the model-based method learns a parameterized model and fits it to the user-item rating matrix. Then, it uses the model to provide recommended tasks. Common model-based techniques include clustering techniques, matrix factorization, and regression-based and many other methods [18]. The memory-based method can provide a good recommendation explanation, and it can use the historical behavior of similar users or other users trusted by the user to explain the recommendation results. However, the model-based method cannot provide such an explanation. Although the hidden factor calculated by it does represent a type of interest or attribute, it is difficult to describe it in natural language and generate an explanation to show it to the user. On the other hand, for high-dimensional and sparse data, memory-based methods perform poorly. In the Netflix Prize Contest [19], the model based on matrix factorization is widely considered to have good scalability, dimensionality reduction characteristics and high accuracy. Several matrix factorization techniques using auxiliary information such as time or social trust have been proposed, and it has been found that data sparsity can be effectively reduced [18].

2.1.2 Matrix Factorization

The matrix factorization also known as latent factor model, is currently one of the most widely used collaborative filtering algorithms. The core idea of matrix factorization is to link user interests and items through latent factors. The following will introduce a basic matrix factorization method combined with the update procedure proposed by Navgaran *et al.* [3]. Assume R is a user-item rating matrix. The rating matrix is factorized into two low-order matrices, which represent the hidden features of the user and the hidden features of the item. P is a user-feature matrix, and Q is an item-feature matrix. R can be represented as:

$$\begin{aligned} \boldsymbol{R} &= \begin{bmatrix} r_{11} \cdots r_{1j} \\ \cdots \ \cdots\cdots \\ r_{i1} \ \cdots r_{ij} \end{bmatrix} \\ &\approx \hat{R} = P \times Q^{\mathrm{T}} = \begin{bmatrix} p_{11} \cdots p_{1k} \\ \cdots \ \cdots\cdots \\ p_{i1} \cdots p_{ik} \end{bmatrix} \times \begin{bmatrix} q_{11} \cdots q_{1j} \\ \cdots \ \cdots\cdots \\ q_{k1} \cdots q_{kj} \end{bmatrix}, \end{aligned} \tag{1}$$

In addition, i is the number of users, j is the number of items, and k is the number of hidden features. Each value r_{ij} in R is a number between 1 and 5 which reflects the degree of preferences of user i to item j. $r_{ij} = 5$ and $r_{ij} = 1$ are the highest and lowest preferences, respectively. By taking the dot product of the vectors user i and item j, we can get a prediction of a rating of an item item_j by user_i as follows:

$$\hat{r}_{ij} = p_i^T q_j = \sum_{k=1}^{K} p_{ik} q_{kj}, \tag{2}$$

The predicted rating matrix $\hat{R}$ obtained by matrix factorization may have errors in the known rating items from the original rating matrix R. The goal is to find the best factorization method to minimize the total error of the predicted rating matrix after factorization. Such a method is called gradient descent, aiming at finding a local minimum of the difference. The difference between the actual value and the estimated value for user_i on item_j is called the error e_{ij}. In order to update the prediction result, the loss function needs to be minimized iteratively. The formula of the mean absolute error is as follows:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{m}\sum_{j=1}^{m} e_{ij} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{m}\sum_{j=1}^{m}\left|r_{ij} - \hat{r}_{ij}\right|, \tag{3}$$

In order to prevent the occurrence of overfitting ($\hat{r}_{ij}$ is greater than the maximum rate value, 5), the update procedure adding regularization is utilized. Navgaran *et al*. proposed an update procedure as follows:

$$p'_{ik} = p_{ik} + \alpha \frac{\partial}{\partial q_{kj}} e_{ij}^2 = p_{ik} + \alpha(2e_{ij}q_{kj} - \beta p_{ik}) \quad \text{and}$$
$$q'_{kj} = q_{kj} + \alpha \frac{\partial}{\partial q_{kj}} e_{ij}^2 = q_{kj} + \alpha(2e_{ij}p_{ik} - \beta q_{kj}), \quad \textbf{(4)}$$

where $p_{ik} \in P$, $q_{kj} \in Q$ and α is the learning rate. The step size determines the length of each step in the negative direction of the gradient during the gradient descent iteration. And β is used to control the magnitudes of the user-feature and item-feature vectors such that P and Q would give a good approximation of R without having to contain large numbers.

2.2 Clustering Analysis for Users

Clustering can be used as a separate process to find the internal distribution structure of data, or as a pre-process for other learning tasks such as classification. For example, in some commercial applications, it is necessary to distinguish the types of new users, but the diversity of user characteristics makes it difficult for businesses to define user types. At this time, clustering algorithm can be used to cluster user data, define each cluster as a class according to the results of the clustering algorithm, and then train the classification model based on these classes [20]. Clustering algorithm is consisted of four main steps: (1) preprocessing data, (2) defining distance function, (3) clustering or grouping, and (4) evaluating clustering results. Since clustering is unsupervised learning without labels and the number of clusters is unknown, it is necessary to further interpret clustering results in combination with business knowledge. For example, data of the same classification should be further segmented from the dimension of user portrait and features extracted from the statistical dimension. In the following, *K*-means algorithm and user characteristic similarity will be presented.

2.2.1 *K*-means Algorithm

K-means algorithm is a clustering algorithm based on partition, which takes distance as the standard for measuring similarity between data objects [21]. In other words, the smaller the distance between data objects is, the higher their similarity is, and the more likely they are in the same class cluster.

2.2.2 User Characteristic Similarity

Each user will have its own characteristics, such as gender, age, occupation, place of residence, salary, nationality, etc. According to the research results of Liji *et al*. [2] on the movielens dataset, people of different ages have specific preferences for genres of movies. Therefore, user characteristics can be used as a basis for grouping users, and after the user characteristic values are vectorized, similar users can be better clustered and the prediction result can be improved.

Ba *et al.* [22] proposed a new method that combines clustering algorithm with SVD algorithm and a novel way to calculate the similarity of user characteristic. This similarity was applied to a clustering algorithm to cluster users. The user characteristics are divided into three dimensions, namely gender, age, and occupation. It is believed that users with the same characteristics will have similar consumption habits and hobbies, and these three attributes play a decisive role. The specific calculation formula of user feature similarity is defined as:

$$\mathrm{muc(u)} = \alpha \mathrm{S(u)} + \beta \mathrm{A(u)} + \gamma \mathrm{O(u)}, \quad \textbf{(5)}$$

where $\alpha + \beta + \gamma = 1$, and α, β, and γ present the correlation coefficient of each attribute. muc(u) presents the comprehensive characteristic value of user u. In addition, S(u) presents user u's characteristic value of gender:

$$\mathrm{S(u)} = \begin{cases} 0, \text{ user u is female} \\ 1, \text{ user u is male} \end{cases}, \quad \textbf{(6)}$$

A(u) presents user u's characteristic value of age:

$$\mathrm{A(u)} = \begin{cases} 0, & \text{the age of u less than 15} \\ \dfrac{\mathrm{Au} - 15}{40}, & \text{the age of user u between 15 and 55} \\ 1, & \text{the age of user u more than 55} \end{cases} \quad \textbf{(7)}$$

O(u) presents user u's characteristic value of occupation:

$$\mathrm{O(u)} = \begin{cases} 0, & \text{user u in leisure class} \\ 0.5, & \text{user u in culture class} \\ 1, & \text{user u in management class} \end{cases}, \quad \textbf{(8)}$$

2.3 Evolutionary Computation Algorithm

Evolutionary computation is not a specific algorithm, but an "algorithm cluster." The inspiration of evolutionary computation draws on the selection mechanism of biological evolution process in nature and the transmission law of genetic information. This process is simulated iteratively through a calculation program, the problem to be solved is regarded as a natural environment, and the optimal solution is sought through natural evolution among a population of possible solutions. The main branches of evolutionary computing are: genetic algorithm [23], genetic programming [24], evolutionary strategies [25], and evolutionary programming [26].

Different evolutionary computations have big differences in the way of evolution. There are many changes in the steps of selection, crossover, mutation, and population control.

Evolutionary computation is a robust method that can adapt to different problems in different environments, and in most cases, it can obtain satisfactory effective solutions. Therefore, evolutionary algorithms have a wide range of applications. The use of evolutionary computation to learn user profiles and recommendation is an emerging trend in the recommender systems research [27].

2.3.1 Selection Techniques

There are some methods to select parents for crossover in the population, and choosing an appropriate selection strategy will have a great impact on the overall performance of genetic algorithm. If the diversity of a selection algorithm decreases, it will lead to premature convergence of the population to the local optimum rather than the desired global optimum, which is called "precocity." If the selection strategy is too divergent, it is difficult for the algorithm to converge to the best. Thus, between these two points we need to strike a balance so that the genetic algorithm converges to the global best in an efficient way [28].

Roulette wheel selection strategy, proposed by Holland [29], is one of the most basic selection strategies. The strategy of tournament selection is pretty straightforward, where we take n individuals from the entire population and let them compete with each other and pick the best one. This method is characterized by efficiency of execution and ease of implementation [30].

Linear ranking selection and exponential ranking selection are both sorting-based selection strategies. The difference between them is the calculation of the selection formula [31]. In linear ranking selection, individuals in a population are first ranked according to fitness value, and then all individuals are assigned an order number. The best individual gets rank N, and the probability of being selected is $P_{\max}$, the worst individual gets rank 1, and the probability of being selected is $P_{\min}$. The probabilities of the other individuals among them can then be obtained by the following formula:

$$P_i = P_{min} + (P_{max} - P_{min})\frac{\mathrm{i}-1}{N-1}; i \in \{1, \cdots, N\}, \quad \textbf{(9)}$$

where P_{i} is the selection probability of ith individual.

Similar to linear ranking selection strategy, exponential ranking selection uses an exponential expression when determining the selection probability of each individual as follows:

$$p_i = \frac{c^{N-i}}{\sum_{j=1}^{N} c^{N-j}}; i \in \{1, \cdots, N\}, \quad \textbf{(10)}$$

where c is the base number and $0 < \mathrm{c} < 1$.

2.3.2 Evolutionary-based Matrix Factorization

Snasel *et al.* [32] established a model for binary matrix decomposition using genetic algorithm. It starts by constructing and initializing factors. The objective function is the hamming distance between the reconstructed matrix and the original matrix. Crossover will aim to modify weights factor, and mutation changes the matrix of factor-item.

Navgaran *et al.* [3] proposed an evolutionary based matrix factorization method. They used EAs to find the values of the two matrices, user-factor and factor-item. In their method, each chromosome will be represented as concatenation of user-factor and factor-item matrices. The fitness function is the total sum of all the errors. Then they used genetic algorithm to reduce the total error between generations.

Kilani *et al.* [33] improved the model by Navgaran *et al.* The method of Navgaran *et al.* decomposes the entire rating matrix into two sub-matrices, and then connects them as a chromosome. Kilani *et al.* mixes the idea of neighborhood-based model and only considers users and items related to the target user. This speeds up the recommendations process and results in a better recommendation quality. The model of this paper is based on the ideas of these two models.

3 Methodology

This section introduces the details of proposed method which is an evolutionary-based collaborative filtering system combining user characteristic clustering and matrix factorization, named matrix factorization with user clustering based on evolutionary algorithm (UC-EAMF). The research framework of this study is introduced in section 3.1, while section 3.2 details the proposed method. The performance evaluation of the proposed method is presented in section 3.3.

3.1 Research Framework

Figure 1 is the research framework of current study. The first step is to input data, including user rating data and user characteristic data. Then, in the second part, the proposed method is developed and implemented in order to obtain the result, which is recommendation. Finally, the performance of the proposed method is evaluated. This study will compare the proposed model with Kilani *et al.*'s model [33] and EAMF [3].

3.2 Proposed Method

When EAMF makes recommendations for a target user, it uses the entire matrix. Taking Movielens 100K as an example, it contains 100,000 ratings from 943 users of 1,682 movies. The size of the matrix is 943X1682, which is a very large matrix and consumes a lot of calculation time. Therefore, it is necessary to find a way to make EAMF more efficient. The model proposed by Kilani *et al.* is designed to improve EAMF, which needs to consider the entire rating matrix resulting in worse efficiency. The proposed method in this study is inspired by this point. Through a certain method to find groups related to the target user and eliminate irrelevant users' information, the recommendation results can be more accurate and the implementation more efficient. This study combines a method of clustering using user characteristic similarity with the evolutionary-based matrix factorization model proposed by Navgaran *et al.* [3] to form a new model. After users are clustered into groups, only user rating data of the same group is used, while EAMF considers the entire rating matrix.

This study uses the user characteristic similarity measurement introduced in section 2.2.2 to calculate the comprehensive characteristic value of users' characteristic attributes. The users' characteristic attributes are gender, age, and occupation in three dimensions. Then, according to the obtained comprehensive characteristic value, K-means clustering algorithm is used to group users.

The steps of user clustering are presented as follows:

Step 1: Set up user similarity parameters α, β, and γ.

Step 2: Calculate user characteristic similarity according to formula (5-8).

Step 3: Set up the number of clusters.

Step 4: Set up initial cluster centers.

Step 5: Calculate the distance from the individual to the centers of the group using the Euclidean distance, formula (11):

$$dist(X,Y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}, \quad (11)$$

where x_i is the ith eigenvalue of the individual and y_i is the ith eigenvalue of the group center y of the individual x.

Step 6: Assign all the individuals to the nearest cluster.

Step 7: Update cluster centers.

Step 8: Stop if the termination condition is satisfied. Otherwise, go back to Step 5.

Step 9: Use the silhouette coefficient to find the best K according to silhouette coefficient as follows:

$$s = \frac{b-a}{\max(a,b)}, \quad (12)$$

where a is the average distance between a sample and other samples in its cluster, and b is the average distance between a sample and other cluster samples.

Figure 2 is an example of user clustering. Through user clustering, 9 users are divided into three different groups.

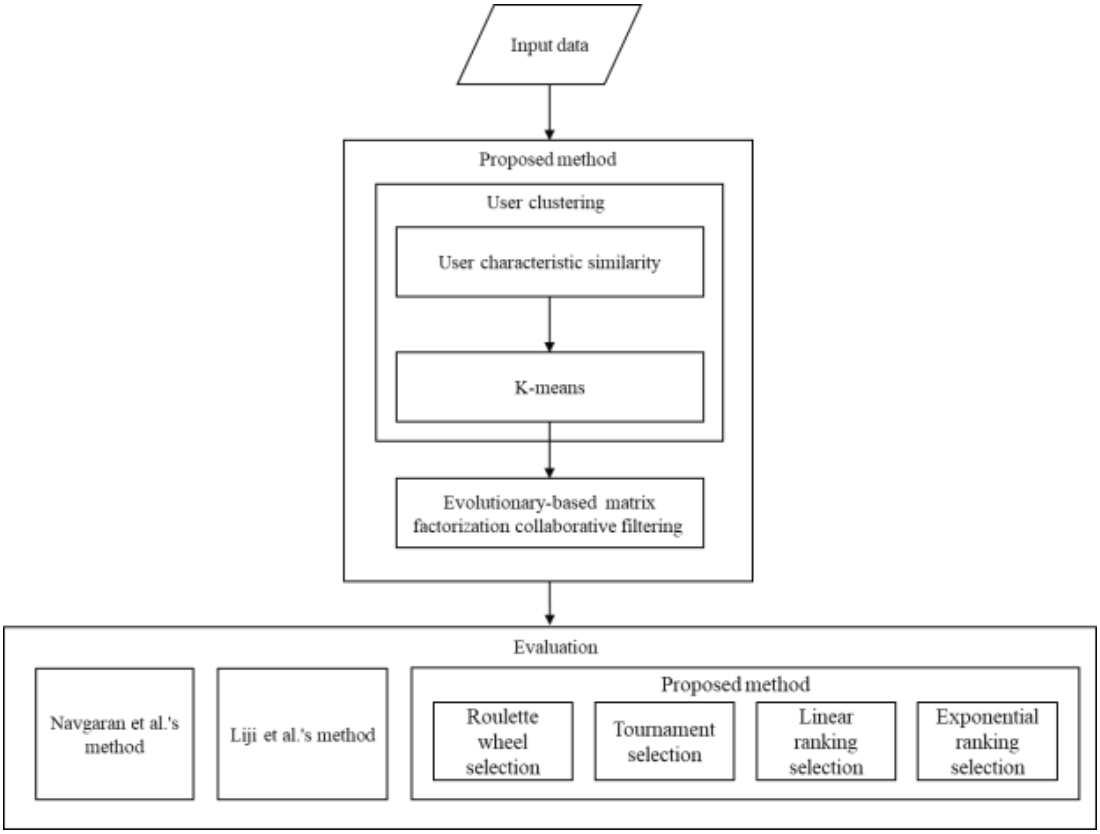

Figure 1. Research framework

	gender	age	occupation
U_1	F	21	student
U_2	F	19	student
U_3	M	18	student
U_4	F	24	marketing
U_5	F	19	artist
U_6	F	33	educator
U_7	F	18	writer
U_8	M	31	engineer
U_9	M	19	student

User clustering

	gender	age	occupation	class
U_1	F	21	student	1
U_2	F	19	student	1
U_4	F	24	marketing	1
U_3	M	18	student	2
U_5	F	19	artist	2
U_7	F	18	writer	2
U_9	M	19	student	2
U_6	F	33	educator	3
U_8	M	31	engineer	3

Figure 2. Example of user clustering

After the user clustering, then the evolutionary-based matrix factorization will be conducted. When the target user has been determined, the method will find the category to which the target user belongs, and extract the relevant user rating data to generate a new rating matrix. The chromosome in EA is a three-dimensional array (population size, factor, user+item), which represents the combination of user-factor and factor-item after the new matrix decomposition. A

complete rating matrix can be obtained after matrix multiplication of these two parts. Randomly generate chromosomes according to the set the number of hidden factors and the size of the new rating matrix. Figure 3 is an example of how to get the chromosome of the proposed method. The main steps are: (1) input data, (2) cluster and extract the rating data of the target user's category, and (3) get a new rating matrix and generate chromosome based on the new rating matrix.

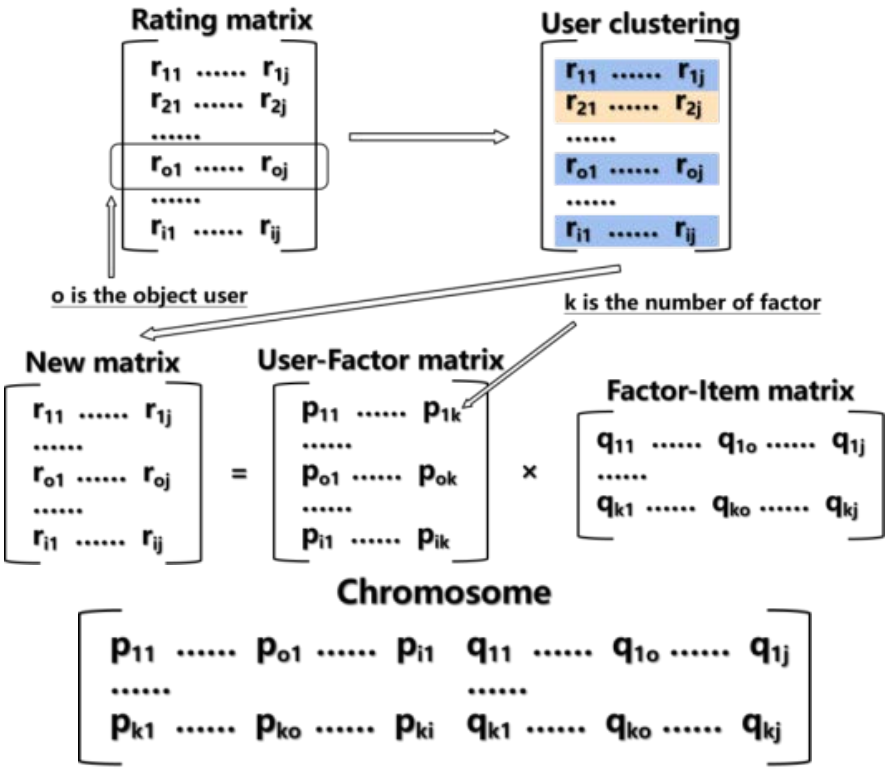

Figure 3. Example of the proposed method

In order to evaluate the fitness of each chromosome, it is necessary to calculate the error between the rated item in the original matrix and the corresponding rating in the chromosome, which is the loss function. This study employs the mean absolute error (MAE) as the fitness function. The purpose of the EA is to minimize this function to get the recommended result. The fitness function is as follows:

$$fitness = \frac{1}{n}\sum_{i=0}^{n}\frac{1}{m}\sum_{j=0}^{m}\left|r_{ij} - \hat{r}_{ij}\right|, \quad \textbf{(13)}$$

where r_{ij} is the rating of i^{th} user on j^{th} item, n is the number of users and m is the number of items.

After the evaluation, selection strategy is used to select parents. This study uses exponential ranking selection. Then crossover and mutation operators are performed. The crossover method used in this study is to randomly select two points in the chromosome and have the parents exchange this segment. As for mutation, randomly select a rated point in chromosome (assuming r_{ij}), and update the hidden layer data corresponding to this point (i.e., column i and column $i+j$ of chromosome) according to the formula given by Navgaran *et al*. as follows:

$$p'_{ik} = p_{ik} + \alpha(2e_{ij}q_{kj} - \beta p_{ik}) \quad \text{and} \quad \textbf{(14)}$$

$$q'_{kj} = q_{kj} + \alpha(2e_{ij}p_{ik} - \beta q_{kj}), \quad \textbf{(15)}$$

where $p_{ik} \in P$, $q_{kj} \in Q$, and α is the learning rate. β is used to control the magnitudes of the user-feature and item-feature vectors.

After reaching the stopping condition, the best individual in the population is selected to recommend the target user. The concept is to recommend the highest few items in the target user's predicted score to the user.

The main steps of the second part of the proposed method are as follows:

Step 1: Obtain user clustering results.

Step 2: Set up the target user.

Step 3: Find the corresponding cluster according to the target user and generate a new rating matrix.

Step 4: Set up the number of hidden factors, *k*.

Step 5: Set up the population size and maximum number of iterations of the GA.

Step 6: Initialize the chromosomes.

Step 7: Evaluate the chromosomes.

Step 8: Use selection strategy to choose parents.

Step 9: Implement crossover.

Step 10: Implement mutation.

Step 11: Update chromosomes.

Step 12: Stop if the termination condition is satisfied. Otherwise, go back to Step 7.

The flowchart of the proposed method is illustrated in Figure 4 and the pseudocode of the proposed algorithm is listed as follows:

```
User Cluster Algorithm
Set up the number of cluster center K
Initial the cluster center
While (stop condition):
    For i in range(number of users):
        If user[i].gender=male:
                S(i)=1
        Else: S(i)=0
        If user[i].age≤18:
                A(i)=0
        Elif user[i].age≤55:
                A(i)=(user[i].age-15)/40
        Else user[i].age≥55: A(i)=1
        If user[i].occupation belongs to leisure class:
                O(i)=0
        Elif user[i].occupation belongs to culture class:
                O(i)=0.5
        Else user[i].occupation belongs to management class:
                O(i)=1
    End
    Muc(i)=αS(i)+βA(i)+γO(i)
    For j in range(0,K):
        Compare muc(j) with center(j) to find the closest class to join in
    Refresh the cluster centers
    End
```

```
Evoluationary-based Matrix Factorization Algorithm
Set up the target user I
Generate the rating matrix of the user I's class
Set up EA parameter Popsize, MaxGeneration
Set up MF parameter k, alpha, beta
Initialize Chromosome
Generation Number=0
While Generation Number≤MaxGeneration:
        childN=0
        while childN≤Popsize:
                Selection(parent1,parent2)
                Crossover(parent1,parent2)
                Mutation(parent1,parent2)
                Fitness(child1,child2)
                Update(Chromosome)
                childN+=1
        End
        GenerationNumber+=1
End
```

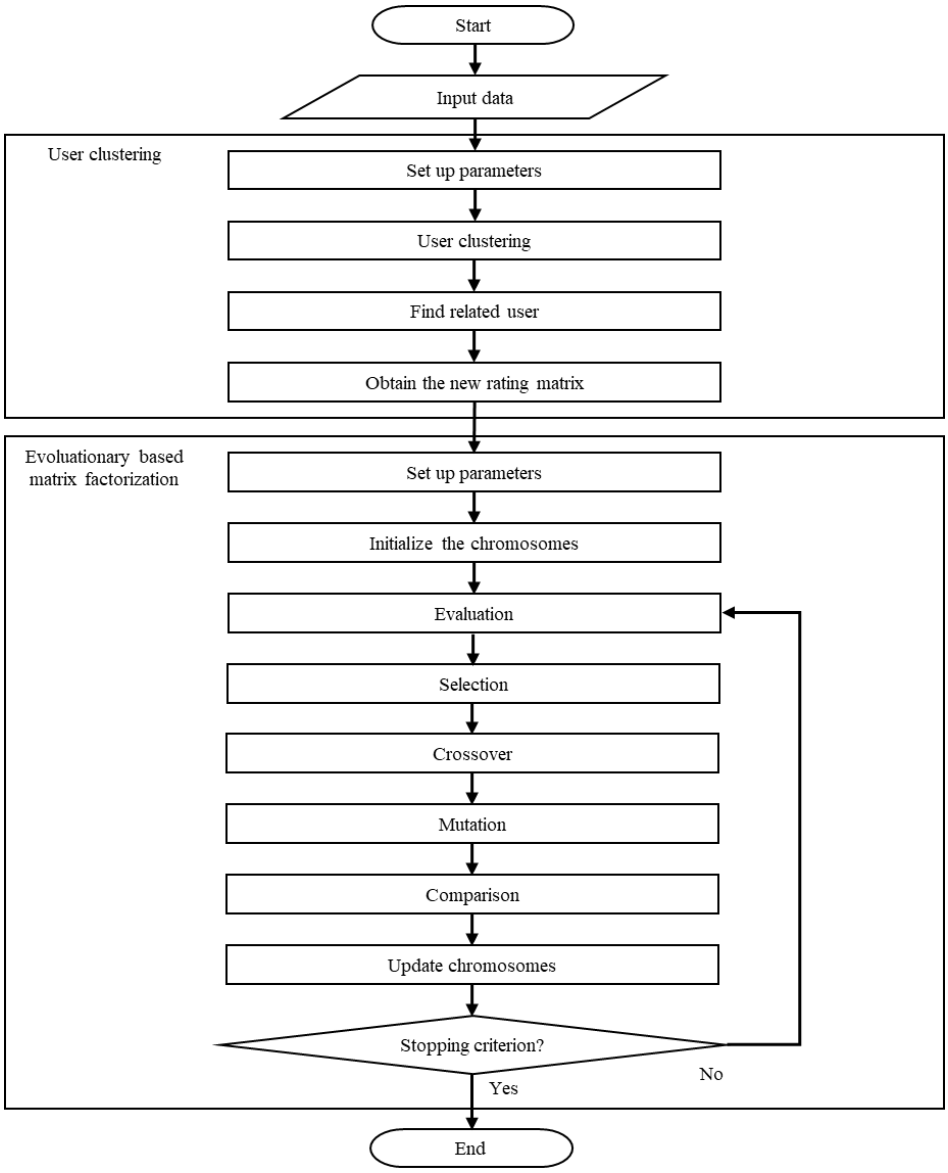

Figure 4. Flowchart of the proposed method

4 Experimental Results

This section presents the experimental results of the proposed UC-EAMF for evaluating and analyzing their performances. In this study, the proposed algorithm was coded by Python programming language and run on a PC with an Intel Core i7-7700 processor and 32GB RAM. The detailed experimental results are described as follow.

4.1 Datasets

There are two datasets used in this research including Movielens 100K and Movielens 1M. They are from social computing research at the University of Minnesota. Movielens 100K contains 100,000 ratings from 943 users of 1,682 movies, and the Movielens 1M contains 1,000,000 ratings from 6040 users of 3952 movies. Rating of each movie is on a scale of 1–5. Randomly select 80% of the dataset as the training set, and the remaining 20% as the test set.

4.2 Performance Measurement

This study will use MAE, precision, recall and *F* to evaluate the recommendation effect of the proposed model and apply the running time to evaluate the efficiency of the model, and compare the results with those of model proposed by Liji *et al.* and EAMF. In addition, this study will also compare the effects of four different selection strategies on the results. They are roulette wheel selection, tournament selection, linear ranking selection and exponential ranking selection. And the influence of the number of different user groups, the different combinations of α and β, and the number of hidden factors on the experimental results will also be explored.

In this study, Silhouette coefficient (SC) is employed to determine the suitable number of clusters, *N*, as follows:

$$\mathrm{S(i)} = \frac{b(i)-a(i)}{\max\{a(i),b(i)\}}, \tag{16}$$

where *i* represents user *i*, *a(i)* is the average of the dissimilar degree from the *i* vector to other points in the same cluster, which is the dissimilarity in the cluster. And *b(i)* is the minimum value of the average dissimilarity from the *i* vector to other clusters, which is the dissimilarity between clusters. The mean value of *S(i)* of all samples is the SC of the clustering result. The value of the SC is between [-1,1]. The larger the value, the closer the similar samples are. The farther the different samples are, the better the clustering effect.

Mean absolute error (MAE) is used to measure the accuracy of the recommendation, which is the most commonly used metric. It will also be applied to compare different algorithms. The lower the MAE, the better the recommendations performance is. Its formula has been mentioned above, as shown in Formula (3).

This study also uses top-*N* recommendation to evaluate the performance of the algorithm. So that it can be compared with other models. Recommendation precision and recall are widely used to measure the quality of recommendations. The formula for precision is defined as follows:

$$Precision = \frac{\sum u \in U |R(u) \cap T(u)|}{\sum u \in U |R(u)|}, \quad \textbf{(17)}$$

The formula for recall is defined as follows:

$$Recall = \frac{\sum u \in U |R(u) \cap T(u)|}{\sum u \in U |T(u)|}, \quad \textbf{(18)}$$

R(u) is the recommended result set of the proposed algorithm and *T*(u) is a set of user *u*'s favorite items in test set. Take movielens data set as an example, the highest score is 5. *T(u)* is a set of users scored 5 points and R(u) is the item with the highest score in the recommended result set of user u. R(u) and T (u) have the same set size.

As the number of top-*N* increases, the recall will increase. However, at the same time, precision will decrease. Therefore, *F* score is used to obtain an appropriate weighted combination of precision and recall metrics. The mathematical expression is as follows:

$$F = \frac{2 \times Precision \times Recall}{Precision + Recall}, \quad \textbf{(19)}$$

Basically, the higher the Precision, Recall and F score are, the better the recommendation performance is.

According to the different target users, the algorithm will use the user rating data of the corresponding class after the users are divided into some classes. In order to have an overall evaluation, all groups are measured, and the overall evaluation is calculated according to the proportion of users in each class to the total users.

4.3 Parameter Setting

Usually, parameters should be pre-determined before implementing the experiment since the performance will be influenced seriously by them. In the proposed UC-EAMF, *K*, *k*, *Alpha, Beta*, population size, crossover rate, mutation rate and the number of iterations should be pre-determined.

As shown in Figure 5 and Figure 6, for Movielens 100K, the clustering result is the best when *K* is equal to 20, while for Movielens 1M, the clustering result is the best when *K* is equal to 17. The *k* represents the number of latent features, and different *k* will affect the recommendation result of the algorithm. Too large *k* makes the computational time become very long, since if the number of latent features increases, the chromosome size will also increase. The experiment uses the best *K* that has been obtained and the preset parameters to find the optimal *k*. The optimal *k* of Movielens 100K can be found in Figure 7 and Figure 8. For *F* score, take top-10 as an example. As can be seen from Figure 7 and Figure 8, with the increase of *k*, *F* score is increasing and MAE is decreasing. When *k* reaches 4, *F* score has the highest value. If we continue to increase the *k*, *F* score begins to decline sharply, and MAE is kept within a certain range and begins to rise sharply after *k* is equal to 7. Thus, the optimal *k* is set as 4. From Figure 9 and Figure 10, when *k* is equal to 6, it is optimal in Movielens 1M.

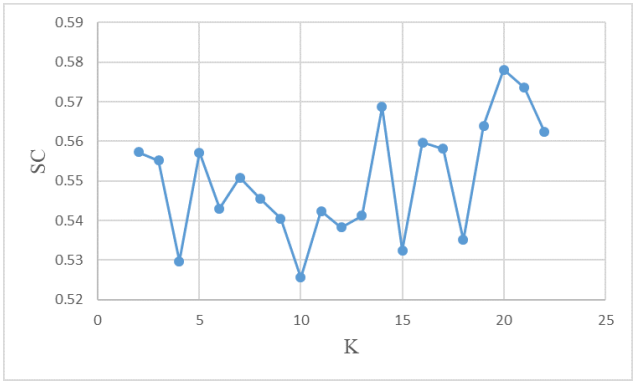

Figure 5. The SC of different *K* values for Movielens 100K

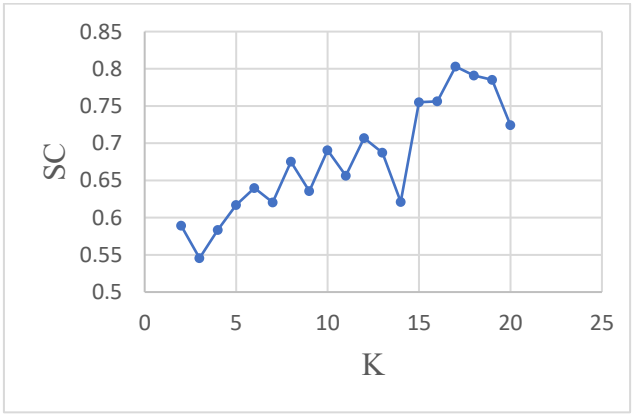

Figure 6. The SC of different *K* values for Movielens 1M

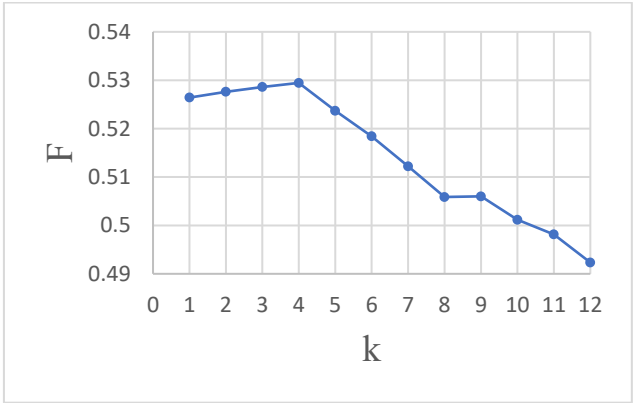

Figure 7. The *F* score of different *k* values for Movielens 100K

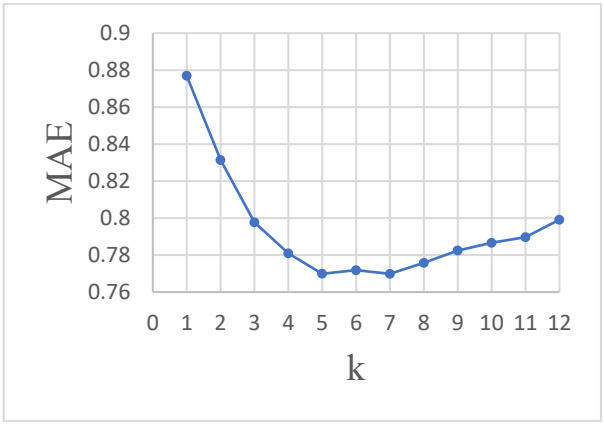

Figure 8. The MAE of different *k* values for Movielens 100K

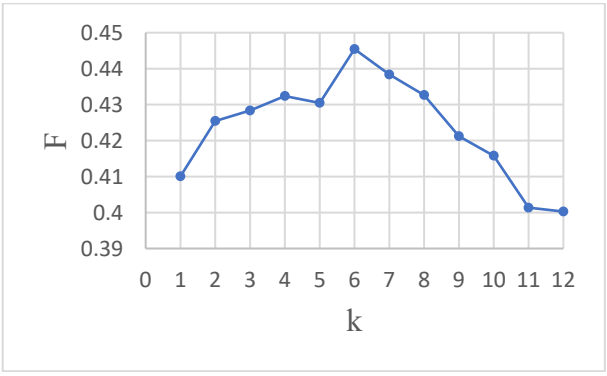

Figure 9. The *F* score of different *k* in Movielens 1M

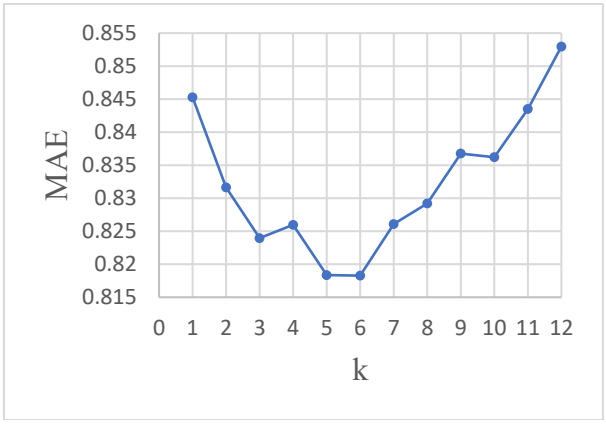

Figure 10. The MAE of different *k* in Movielens 1M

α is the learning rate, while β is used to control the magnitudes of the user-feature and item-feature vectors. They may affect each other. Thus, this study tested different combinations of α and β values to find the best α and β. The test results are shown in Table 1. It can be found that when α=0.03 and β=0.02, the results of *F* score and MAE are the best. As can be seen from Figures 11 and 12, MAE and *F* score converge when the number of iterations is equal to 1000. Thus, the number of iterations is set as 1000. Since the computational time is extremely long for Movielens 1M, these parameters have only been tested in Movielens 100K, and continue to use the same iteration number, α and β.

The parameters of evolutionary algorithm, after several trial experiments, the population size is set as 50, crossover rate is set as 0.8, and the mutation rate is set as 0.3.

Table 1. The MAE and *F* score of different combinations of α and β for 100K Movielens

alpha \ beta		0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.1
0.01	F	0.4965	0.4923	0.4938	0.4954	0.4972	0.4930	0.4916	0.4940	0.4952	0.4917
	MAE	0.9511	0.9611	0.9562	0.9619	0.9562	0.9623	0.9585	0.9667	0.9640	0.9671
0.02	F	0.5242	0.5295	0.5264	0.5254	0.5309	0.5281	0.5255	0.5251	0.5256	0.5256
	MAE	0.7821	0.7809	0.7821	0.7818	0.7818	0.7819	0.7809	0.7852	0.7848	0.7845
0.03	F	0.5391	0.5439	0.5362	0.5376	0.5367	0.5395	0.5400	0.5395	0.5368	0.5376
	MAE	0.7339	0.7217	0.7361	0.7354	0.7363	0.7355	0.7345	0.7374	0.7354	0.7351
0.04	F	0.5396	0.5438	0.5438	0.5385	0.5407	0.5424	0.5383	0.5420	0.5410	0.5426
	MAE	0.7231	0.7348	0.7194	0.7208	0.7228	0.7217	0.7221	0.7233	0.7222	0.7229
0.05	F	0.5423	0.5408	0.5425	0.5419	0.5417	0.5429	0.5431	0.5393	0.5380	0.5375
	MAE	0.7234	0.7224	0.7231	0.7237	0.7223	0.7198	0.7218	0.7212	0.7223	0.7210
0.06	F	0.5369	0.5390	0.5383	0.5421	0.5409	0.5393	0.5399	0.5378	0.5389	0.5395
	MAE	0.7313	0.7322	0.7295	0.7308	0.7297	0.7292	0.7295	0.7289	0.7293	0.7264
0.07	F	0.5354	0.5319	0.5367	0.5315	0.5335	0.5356	0.5360	0.5358	0.5344	0.5360
	MAE	0.7436	0.7431	0.7408	0.7439	0.7405	0.7388	0.7401	0.7392	0.7381	0.7406
0.08	F	0.5331	0.5301	0.5334	0.5353	0.5323	0.5317	0.5334	0.5318	0.5335	0.5324
	MAE	0.7569	0.7579	0.7562	0.7554	0.7541	0.7539	0.7544	0.7507	0.7489	0.7486
0.09	F	0.5257	0.5284	0.5290	0.5286	0.5292	0.5323	0.5294	0.5284	0.5301	0.5304
	MAE	0.7749	0.7715	0.7701	0.7724	0.7708	0.7645	0.7641	0.7641	0.7622	0.7633
0.1	F	0.5249	0.5240	0.5290	0.5220	0.5233	0.5245	0.5273	0.5287	0.5258	0.5221
	MAE	0.7899	0.7861	0.7865	0.7844	0.7865	0.7817	0.7837	0.7818	0.7793	0.7806

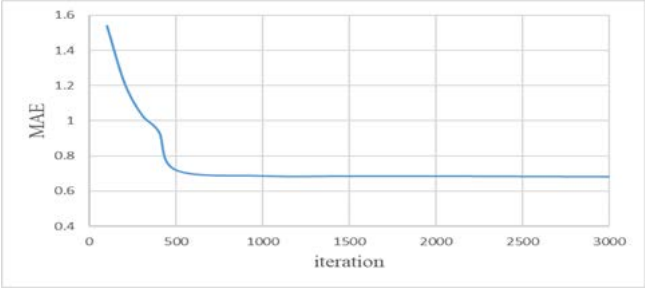

Figure 11. The convergence of MAE for Movielens 100K

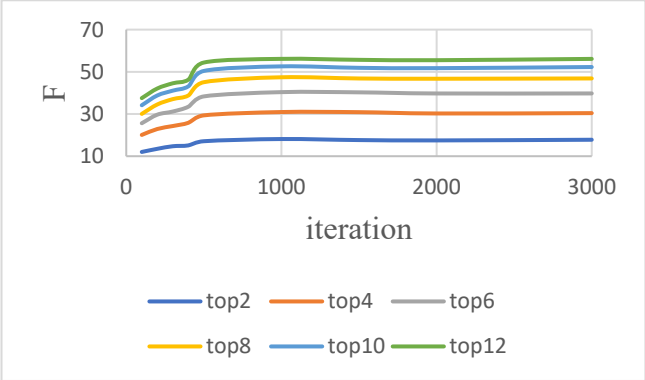

Figure 12. The convergence of top-N F score for Movielens 100K

4.4 Experimental Results

The experiment testified UC-EAMF using different selection methods, including roulette wheel selection, tournament selection, linear ranking selection and exponential ranking selection, and each method was executed 30 times for Movielens 100K and Movielens 1M. The purpose is to prove that using exponential ranking selection can have a better result than the other selection methods. From Table 2 and Table 3, it reveals that for Movielens 100K and Movielens 1M, according to the precision, recall, *F* score and MAE, the exponential ranking selection method has better performance than other selection methods. In EAMF, roulette wheel selection is used. During the test, it will be found that when MAE is used as fitness, in the same generation, all fitness values will be very close. Thus, when parents are randomly selected, the probabilities for all chromosomes are very close. However, if tournament selection is used to randomly select individuals for generating their parents after the competition, it is easier to find better parents than roulette wheel selection, but the process of selecting individual is completely random, and there is no guarantee that better individuals are more likely to be selected as parents. Ranking selection ranks individuals using fitness, and assigns the probability of each individual according to the ranking. This can ensure randomness and provide a higher probability of being selected for better individuals. Thus, ranking selection has a better performance. In this experiment, the probability distribution formula of exponential ranking selection is more suitable, so it can result in better results.

Table 2. The experiment results of UC-EAMF with different selection methods for Movielens 100K

Number of top-N	Metrics (%)	Roulette Wheel Selection	Tournament Selection	Linear Ranking Selection	Exponential Ranking Selection
Top 2	P	63.84	65.85	67.92	*68.03
	R	9.82	10.13	10.45	*10.47
	F	17.02	17.56	18.11	*18.14
Top 4	P	62.49	64.29	65.43	*65.75
	R	19.23	19.78	20.13	*20.23
	F	29.41	30.25	30.79	*30.94
Top 6	P	60.62	62.69	63.66	*63.93
	R	27.98	28.93	29.38	*29.50
	F	38.29	39.59	40.21	*40.38
Top 8	P	58.35	60.84	61.64	*62.23
	R	35.91	37.44	37.93	*38.30
	F	44.46	46.36	46.96	*47.42
Top 10	P	56.85	58.61	60.13	*60.47
	R	43.73	45.09	46.25	*46.51
	F	49.44	50.97	52.28	*52.58
Top 12	P	55.39	56.96	58.09	*58.47
	R	51.13	52.58	53.62	*53.98
	F	53.17	54.69	55.76	*56.14
	MAE	0.741	0.714	0.687	*0.684

Table 3. The experiment results of UC-EAMF with different selection methods for Movielens 1M

Number of top-N	Metrics (%)	Roulette Wheel Selection	Tournament Selection	Linear Ranking Selection	Exponential Ranking Selection
Top 2	P	58.55	59.56	60.62	*61.30
	R	9.01	9.16	9.33	*9.43
	F	15.61	15.88	16.16	*16.35
Top 4	P	56.66	57.46	59.53	*60.25
	R	17.43	17.68	18.32	*18.54
	F	26.66	27.04	28.01	*28.35
Top 6	P	55.15	56.06	58.09	*58.92
	R	25.45	25.87	26.81	*27.19
	F	34.83	35.41	36.69	*37.21
Top 8	P	53.85	54.70	56.90	*57.52
	R	33.14	33.66	35.02	*35.39
	F	41.03	41.68	43.35	*43.82
Top 10	P	52.55	53.62	55.87	*56.39
	R	40.42	41.25	42.98	*43.38
	F	45.70	46.63	48.58	*49.04
Top 12	P	51.32	52.46	54.71	*55.29
	R	47.37	48.42	50.50	*51.04
	F	49.27	50.36	52.52	*53.08
	MAE	0.993	0.907	0.773	*0.759

In addition, this study also compared the performance of UC-EAMF with EAMF using Movielens 100K and Movielens 1M. In addition, the comparison with the method that also uses user characteristic data in the Movielens 100K dataset is also examined. The proposed algorithm eliminates irrelevant users, eliminates the influence of irrelevant materials on the recommendation results, and makes the results more accurate. The computational results are listed in Table 4 and Table 5. It can be found that the proposed method UC-EAMF has a greater improvement compared to EAMF in the performance of precision, recall, *F* score and MAE. The rating sparsity of Movielens 1M is higher than that of Movielens 100K, so the recommended performance of the model is reduced for Movielens 1M. Therefore, it is important to eliminate data irrelevant to target users, since this can have a great improvement compared with EAMF.

Precision and recall will have different emphasis according to different needs. However, precision and recall are generally contradictory, so the F score is used to comprehensively consider the recommended performance. Compared with the method proposed by Liji *et al*., although the method proposed in this study has a gap with it in precision. However, from the perspective of the comprehensive index *F* score, the performance is better than the method of Liji *et al*.

Figure 13 illustrates the convergence curves both for EAMF and the proposed UC-EAMF. We can see that UC-EAMF outperforms EAMF. The proposed model is able to converge faster and provide better result.

Table 4. The experiment results of UC-EAMF, EAMF, and Liji et al.'s method for Movielens 100K

Number of top-N	Metrics (%)	Proposed method	Navgaran *et al.*'s method	Liji *et al.*'s method
Top 2	P	68.03	47.24	***93.21**
	R	*10.47	7.27	**8.52**
	F	*18.14	12.60	**15.55**
Top 4	P	65.75	47.06	***91.78**
	R	*20.23	14.48	**16.20**
	F	*30.94	22.14	**27.36**
Top 6	P	63.93	45.48	***91.11**
	R	*29.50	20.99	**23.03**
	F	*40.38	28.72	**36.55**
Top 8	P	62.23	45.17	***90.40**
	R	*38.30	27.80	**29.08**
	F	*47.42	34.42	**43.69**
Top 10	P	60.47	44.02	***90.10**
	R	*46.51	33.86	**34.50**
	F	*52.58	38.28	**49.61**
Top 12	P	58.47	43.38	***89.55**
	R	*53.98	40.04	**39.16**
	F	*56.14	41.65	**54.13**
	MAE	*0.684	1.035	0.746

Table 5. The experiment results of UC-EAMF and EAMF for Movielens 1M

Number of top-N	Metrics (%)	Proposed method	Navgaran *et al.*'s method
Top 2	P	*61.30	56.80
	R	*9.43	8.74
	F	*16.35	15.15
Top 4	P	*60.25	53.63
	R	*18.54	16.50
	F	*28.35	25.24
Top 6	P	*58.92	51.56
	R	*27.19	23.80
	F	*37.21	32.57
Top 8	P	*57.52	50.18
	R	*35.39	30.88
	F	*43.82	38.23
Top 10	P	*56.39	48.84
	R	*43.38	37.57
	F	*49.04	42.47
Top 12	P	*55.29	47.76
	R	*51.04	44.09
	F	*53.08	45.85
	MAE	*0.759	1.140

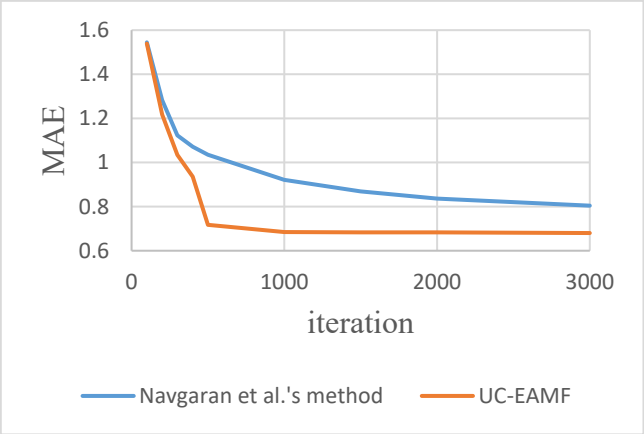

Figure 13. The MAE convergence curves of EAMF and UC-EAMF for Movielens 100K

4.5 Statistical Hypothesis

In order to make sure that the proposed method is statistically better than the other existing methods, one-tailed *t* test is used to testify whether UC-EAMF is significantly better than EAMF or not. $u_{UC-EAMF\_MAE}$ represents the overall mean value of MAE of proposed algorithm UC-EAMF in this paper, u_{EAMF_MAE} represents the overall mean value of MAE of the existing method EAMF. $u_{UC-EAMF\_topN}$ represents the overall mean value of topN F score of proposed algorithm UC-EAMF in this paper, $u_{UC-EAMF\_topN}$ represents the overall mean value of topN F score of the existing method EAMF.

Test for MAE:

H_0: $u_{UC-EAMF\_MAE} \geq u_{EAMF\_MAE}$

H_1: $u_{UC-EAMF\_MAE} < u_{EAMF\_MAE}$

Test for top 2:

H_0: $u_{UC-EAMF\_top2} \leq u_{EAMF\_top2}$

H_1: $u_{UC-EAMF\_top2} > u_{EAMF\_top2}$

Test for top 4:

H_0: $u_{UC-EAMF\_top4} \leq u_{EAMF\_top4}$

H_1: $u_{UC-EAMF\_top4} > u_{EAMF\_top4}$

Test for top 6:

H_0: $u_{UC-EAMF\_top6} \leq u_{EAMF\_top6}$

H_1: $u_{UC-EAMF\_top6} > u_{EAMF\_top6}$

Test for top 8:

H_0: $u_{UC-EAMF\_top8} \leq u_{EAMF\_top8}$

H_1: $u_{UC-EAMF\_top8} > u_{EAMF\_top8}$

Test for top 10:

H_0: $u_{UC-EAMF\_top10} \leq u_{EAMF\_top10}$

H_1: $u_{UC-EAMF\_top10} > u_{EAMF\_top10}$

Test for top 12:

H_0: $u_{UC-EAMF\_top12} \leq u_{EAMF\_top12}$

H_1: $u_{UC-EAMF\_top12} > u_{EAMF\_top12}$

Since EAMF needs longer computational time for Movielens 1M, there is not enough data to do statistical hypothesis testing. Therefore, in here, we only testify the experimental results using Movielens 100K. Null hypothesis $\mathbf{H_0}$ will be rejected when *P*-value is less than or equal to the significance level ($\alpha = 0.05$). Table 6 illustrates the *P*-values of statistical hypothesis for Movielens 100K. Because each *P*-value is less than 0.05, these tests have enough evidence to reject the null hypothesis $\mathbf{H_0}$. This means the performance of the proposed UC-EAMF is significantly superior to EAMF for MAE and all the topN recommendation.

Table 6. *P*-values for MAE and all the top-*N* recommendations

	MAE	Top 2	Top 4	Top 6	Top 8	Top 10	Top 12
P-value	0.000	0.000	0.000	0.000	0.000	0.000	0.000

4.6 Time Complexity

In order to observe the computational complexity of the proposed method and EAMF, Big-O notation for studying algorithm efficiency is used (Devi *et al.*, 2011) [34]. According to research observations, most irrelevant users and items are eliminated after clustering, which greatly reduces the number of users and items. Therefore, UC-EAMF spends shorter time than EAMF. The following notations are used to measure the complexity:

N: Number of users (s)
M: Number of items (s)
n: Number of users (s) after clustering
m: Number of items (s) after clustering
F: Number of latent factors (s)
P: Population size

Table 7 displays the time complexity of the proposed method and EAMF. We can see that the newly proposed method greatly reduces the amount of data input to EAMF by excluding irrelevant data from user groups. Table 8 displays the time spent by UC-EAMF and EAMF for Movielens 100K and Movielens 1M.

Table 7. Time complexity analysis

Phase	Time Order
EAMF	*O*(NMFP)
User cluster	*O*(N)
UC-EAMF	*O*(nmFP)

Table 8. The computational time of UC-EAMF and EAMF for Movielens 100K and Movielens 1M

Algorithm	Movielens 100K	Movielens 1M
EAMF	22663s	238604s
UC-EAMF	2815s	21521s

5 Conclusions

This study has proposed a method which is an evolutionary-based collaborative filtering system combining user characteristic clustering and matrix factorization, UC-EAMF. The proposed UC-EAMF combined with the user characteristic clustering method can greatly improve the calculation efficiency and improve the recommendation performance. In addition, the proposed UC-EAMF uses exponential ranking selection method that is different from the original EAMF, which can increase the recommendation performance.

With the development of datasets, the data of users and items will become larger and larger. Therefore, more research is needed to improve computational efficiency and reduce the recommended time. There may be better ways to divide the rating matrix to be discovered, so that recommend more accurate and efficient. This experiment studies a method that combines matrix factorization technology and evolutionary algorithms. It may be possible to combine matrix factorization technology with more heuristic algorithms to get better models.

References

[1] J. Bobadilla, F. Ortega, A. Hernando, A. Gutiérrez, Recommender systems survey. *Knowledge-Based Systems*, Vol. 46, pp. 109-132, July, 2013.

[2] U. Liji, Y. Chai, J. Chen, Improved personalized recommendation based on user attributes clustering and score matrix filling, *Computer Standards & Interfaces*, Vol. 57, pp. 59-67, March, 2018.

[3] D. Z. Navgaran, P. Moradi, F. Akhlaghian, Evolutionary based matrix factorization method for collaborative filtering systems, *21st Iranian Conference on Electrical Engineering (ICEE)*, Mashhad, Iran, 2013, pp. 1-5.

[4] J. Lu, D. Wu, M. Mao, W. Wang, G. Zhang, Recommender system application developments: A survey, *Decision Support Systems*, Vol. 74, pp. 12-32, June, 2015.

[5] R. V. Meteren, M. V. Someren, Using content-based filtering for recommendation, *Proceedings of ECML 2000 Workshop: Machine Learning in Information Age*, Barcelona, Spain, 2000, pp. 47-56.

[6] J. L. Herlocker, J. A. Konstan, L. G. Terveen, J. T. Riedl, Evaluating collaborative filtering recommender systems, *ACM Transactions on Information Systems*, Vol. 22, No. 1, pp. 5-53, January, 2004.

[7] P. J. Gai, A. K. Klesse, Making Recommendations more effective through framings: impacts of user- versus item-based framings on recommendation click-throughs, *Journal of Marketing*, Vol. 83, No. 6, pp. 61-75, November, 2019.

[8] M. Jalili, S. Ahmadian, M. Izadi, P. Moradi, M. Salehi, Evaluating collaborative filtering recommender algorithms: a survey, *IEEE Access*, Vol. 6, pp. 74003-74024, November, 2018.

[9] R. Burke, Hybrid web recommender systems, in: P. Brusilovsky, A. Kobsa, W. Nejdl (Eds.), *The Adaptive Web*, Springer, 2007, pp. 377-408.

[10] A. Karatzoglou, B. Hidasi, Deep learning for recommender systems, *Proceedings of the Eleventh ACM Conference on Recommender Systems (RecSys '17). Association for Computing Machinery*, Como, Italy, pp. 396-397, 2017.

[11] N. Idrissi, A. Zellou, A systematic literature review of sparsity issues in recommender systems, *Social Network Analysis and Mining*, Vol. 10, No. 1, Article No. 15, December, 2020.

[12] I. Esslimani, A. Brun, A. Boyer, Densifying a behavioral recommender system by social networks link prediction methods, *Social Network Analysis and Mining*, Vol. 1, No. 3, pp. 159-172, July, 2011.

[13] B. Lika, K. Kolomvatsos, S. Hadjiefthymiades, Facing the cold start problem in recommender systems, *Expert Systems with Applications*, Vol. 41, No. 4, pp. 2065-2073, March, 2014.

[14] K. Yu, A. Schwaighofer, V. Tresp, X. Xu, H. Kriegel, Probabilistic memory-based collaborative filtering, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 16, No. 1, pp. 56-69, January, 2004.

[15] C. Yin, L. Shi, R. Sun, J. Wang, Improved collaborative filtering recommendation algorithm based on differential privacy protection, *The Journal of Supercomputing*, Vol. 76, No. 7, pp. 5161-5174, July, 2020.

[16] Z. Wu, G. Li, Q. Liu, G. Xu, E. Chen, Covering the sensitive subjects to protect personal privacy in personalized recommendation, *IEEE Transactions on Services Computing*, Vol. 11, No. 3, pp. 493-506, May-June, 2018.

[17] Q. Jin, Y. Zhang, W. Cai, Y. Zhang, A new similarity computing model of collaborative filtering, *IEEE Access*, Vol. 8, pp. 17594-17604, January, 2020.

[18] R. Mehta, K. Rana, A review on matrix factorization techniques in recommender systems, *2017 2nd International Conference on Communication Systems, Computing and IT Applications (CSCITA)*, Mumbai, India, 2017, pp. 269-274.

[19] J. Bennett, S. Lanning, The netflix prize, *Proceedings of the KDD Cup Workshop*, San Jose, California, USA, 2007, pp. 3-6.

[20] J. T. Kwok, Z. Zhou, L. Xu, Machine Learning, in: J. Kacprzyk, W. Pedrycz (Eds.), *Springer Handbook of Computational Intelligence*, Springer, 2015, pp. 495-522.

[21] J. M. Pena, J. A. Lozano, P. Larranaga, An empirical comparison of four initialization methods for the k-means algorithm, *Pattern Recognition Letters*, Vol. 20, No. 10, pp. 1027-1040, October, 1999.

[22] Q. Ba, X. Li, Z. Bai, Clustering collaborative filtering recommendation system based on SVD algorithm, *2013 IEEE 4th International Conference on Software Engineering and Service Science*, Beijing, China, 2013, pp. 963-967.

[23] J. H. Holland, *Adaptation in Natural and Artificial Systems*, Ann Arbor: University of Michigan Press, 1975.

[24] N. L. Cramer, A representation for the adaptive generation of simple sequential programs, *Proceedings of the 1st International Conference on Genetic Algorithms and their Applications*, Pittsburg, PA, USA, 1985, pp. 183-187.

[25] I. Rechenberg, *Evolution strategy: optimization of technical systems by means of biological evolution*, Fromman-Holzboog, Vol. 104, 1973.

[26] L. J. Fogel, A. J. Owens, M. J. Walsh, *Artificial*

Intelligence through Simulated Evolution, John Wiley, 1966.
[27] T. Horváth, A. C. P. L. F. de Carvalho, Evolutionary computing in recommender systems: a review of recent research, *Natural Computing*, Vol. 16, No. 3, pp. 441-462, September, 2017.
[28] A. Shukla, H. M. Pandey, D. Mehrotra, Comparative review of selection techniques in genetic algorithm, *2015 International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE)*, Greater Noida, India, 2015, pp. 515-519.
[29] J. H. Holland, *Adaptation in natural and artificial systems*, MIT Press, 1992.
[30] D. E. Goldberg, K. Deb, A comparative analysis of selection schemes used in genetic algorithms, *Foundations of Genetic Algorithms*, Vol. 1, pp. 69-93, 1991.
[31] T. Blickle, L. Thiele, A comparison of selection schemes used in genetic algorithms, TIK Report, December, 1995.
[32] V. Snasel, J. Platos, P. Kromer, Developing genetic algorithms for Boolean matrix factorization, *Proceedings of the Dateso 2008 Annual International Workshop on Databases*, Desna, Czech Republic, 2008, pp. 61-70.
[33] Y. Kilani, A. F. Otoom, A. Alsarhan, M. Almaayah, A genetic algorithms-based hybrid recommender system of matrix factorization and neighborhood-based techniques, *Journal of Computational Science*, Vol. 28, pp. 78-93, September, 2018
[34] S. G. Devi, K. Selvam, S. Rajagopalan, An abstract to calculate big O factors of time and space complexity of machine code, *International Conference on Sustainable Energy and Intelligent Systems (SEISCON 2011)*, Chennai, India, 2011, pp. 844-847.

Biographies

R. J. Kuo received the M.S. degree in Industrial and Manufacturing Systems Engineering from Iowa State University, Ames, IA, USA in 1990 and the Ph.D. degree in Industrial and Management Systems Engineering from the Pennsylvania State University, University Park, PA, USA in 1994. Currently, he is the Distinguished Professor in the Department of Industrial Management at National Taiwan University of Science and Technology, Taiwan. He has published more than 100 papers in international journals including Neural Networks, Information Sciences and Decision Support Systems. His research interests include architecture issues of computational intelligence and their applications to data mining, recommender systems, electronic business, logistics and supply chain management and hearth care.

Zhen Wu received the M.S. degree in Industrial Engineering Department from National Taiwan University of Science and Technology 2021. His research interests include data mining, recommender systems, and industrial management.