Swarm and Evolutionary Computation 14 (2014) 21–30

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

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

An evolutionary clustering algorithm based on temporal features for dynamic recommender systems

Chhavi Rana [a,*], Sanjay Kumar Jain [b]

[a] Department of Computer Science Engineering, University Institute of Engineering and Technology, MD University, Rohtak, Haryana 124001, India
[b] Department of Computer Engineering, National Institute of Technology, Kurukshetra, Haryana 136119, India

ARTICLE INFO

Article history:
Received 15 September 2012
Received in revised form
9 August 2013
Accepted 20 August 2013
Available online 29 August 2013

Keywords:
Evolutionary
Clustering
Algorithm
Recommender systems
Collaborative filtering
Data mining

ABSTRACT

The use of internet and Web services is changing the way we use resources and communicate since the last decade. Although, this usage has made life easier in many respects still the problem of finding relevant information persists. A naïve user faces the problem of information overload and continuous flow of new information makes the problem more complex. Furthermore, user's interests also keeps on changing with time. Several techniques deal with this problem and data mining is widely used among them. Recommender Systems (RSs) assist users in finding relevant information on the web and are mostly based on data mining algorithms. This paper addresses the problem of user requirements changing over a period of time in seeking information on web and how RSs deal with them. We propose a Dynamic Recommender system (DRS) based on evolutionary clustering algorithm. This clustering algorithm makes clusters of similar users and evolves them depicting accurate and relevant user preferences over time. The proposed approach performs an optimization of conflicting parameters instead of using the traditional evolutionary algorithms like genetic algorithm. The algorithm has been empirically tested and compared with standard recommendation algorithms and it shows considerable improvement in terms of quality of recommendations and computation time.

1. Introduction

Recommender systems assist users in navigating through information on the web by giving suggestions regarding their preferences. This is done by generating a user profile based on their past behavior. Although RSs are chiefly applied in the area of e-commerce, their domain areas are constantly enlarging. One of the latest examples is their use in the social networking sites which are widely using recommendations. Different techniques are used in building RSs and they are mainly divided into three categories namely collaborative filtering, content based filtering and hybrid system. Collaborative filtering (CF) works by generating recommendations on the basis of similarity of interest between users. The process identifies users that have similar preferences and then from their ratings of different items it can suggest new items to a particular user. Content based filtering is the technique that uses the content knowledge of the domain of recommendation to predict items that could interest users. The domain knowledge becomes very important in such a scenario that poses as a hindrance to the overall evolution of the RSs. Hybrid techniques combines the above two approaches, however, such a system needs to be designed carefully so that the disadvantages of the selected system are not inherited.

The most widely used technique for Recommender System is collaborative filtering [33]. The success of a collaborative filtering system is highly dependent upon the effectiveness of the algorithm in finding set of users (profile) which are most similar to that of the current user. Various data mining as well as information retrieval technique are applied for this purposed until now. The adaptability of Recommender System to predict the evolving user's needs, which are constantly changing, is a growing area of research in the sphere of recommender system. In the constantly changing environment of web, providing relevant content through a recommendation mechanism can be a difficult task and time could be one of the most important factors [23,24]. Various model based algorithms in RSs are well-known to achieve good prediction performances [23]. However, the major drawback for many such model based methods is that they all require a lengthy training period. In addition, the static nature of such models results in a progressive declining of the prediction accuracy after a period of time because new ratings are not taken into account until re-computation of whole model, which cannot be done very often because of the high computational costs. The problem of matching user profile that evolves with time can be more effectively handled by an evolving mechanism that uses temporal

* Corresponding author. Tel.: +91 9896308747.
E-mail addresses: chhavi1jan@yahoo.com, chhavi.rana@gmail.com (C. Rana), skj_nith@yahoo.com (S.K. Jain).

2210-6502/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.swevo.2013.08.003

dimension. Such mechanism can precisely fulfill this gap during profile matching and thus help in dealing with the dynamics of user profile. Evolutionary clustering based methods adjust the time problem of the conventional method according to an optimization scheme [19]. Thus, the incorporation of temporal changes in user preferences can be effectively handled by an evolutionary clustering mechanism. Moreover, clustering mechanism is proven to be beneficial for removing the scalability and sparsity problems in a recommender system [15,33]. Thus, the usage of evolutionary clustering mechanism in a recommender system can be beneficial in multiple ways.

1.1. Motivation

The major focus of RSs in the real world setting is to increase the accuracy of the recommendations. To maintain accuracy of the system, relevant recommendation needs to be provided to the user. As time passes, the relevancy of items to the user shifts as his preferences changes. This shift in the user profile needs to be done periodically and it requires a constantly evolving mechanism. Thus, the biggest problem in improving the accuracy of the system comes with user changing requirements and content up gradation [23]. To handle the dynamic environment of the web, a RS needs to continuously upgrade the incoming stream of users, items and their corresponding choices. The Dynamism in RSs becomes an important factor that needs to be taken into consideration when traditional retraining methods cannot keep pace with the rate of change with respect to user preferences [31]. Maintaining accuracy together with fulfilling user changing requirements is a difficult task. The inclusion of time dimension in RSs can help in overcoming many problems. In addition to predicting changing user needs and thereby improving accuracy, temporal dimension could also help in improving the scalability of the system. This is done by reducing the number of uses and items for similarity calculation that are divided into clusters using temporal data. The question is how to add time. As Koren [23], taken a perspective where a hybrid approach with 67 parameters and lots of calculation proved time could improve accuracy and won the Netflix prize. Lathia et al. [24] on the other hand have also shown how changes over a period of time can affect the accuracy. Although a deep analysis of temporal dimension in evaluating collaborative filtering is presented in his research yet the challenge to utilize temporal dimension for the improvement and diversification of the system is not addressed. Consequently, very few people have actually tried to correlate the effect of temporal factor in the recommendation process. Even less have applied evolutionary computation to the recommendation process. Chakrabarti et al. [9] gives a very novel concept of evolutionary clustering. The process of clustering can be very effective in finding similar users and in turn better clusters that can improve accuracy in the recommendation process in addition to handling scalability. To incorporate the idea of evolving user′s interest with time and the system content, an evolutionary clustering method could be employed in the recommendation system. In this paper, we have proposed an evolutionary clustering based recommendation algorithm that improves the accuracy as well as computational time.

1.2. Contribution

We propose an evolutionary clustering algorithm that could serve as the core module for matching user profiles and their evolving nature in a recommender systems. It is observed that clustering techniques often lead to worse prediction accuracy even though it improves scalability [33]. This is because, when clustering is finished, the size of a cluster that must be analyzed is much smaller. Consequently, clustering methods can solve the scalability problem in RSs at the cost of accuracy. We propose an approach that improves the prediction accuracy as well alleviates the sparsity and scalability issue with the utilization of temporal dimension. This is done using evolutionary clustering that utilizes time dimension to produce updated groups of user in the form of clusters. In short, the detection of evolving user preferences is formulated as a clustering problem and a solution based on evolutionary clustering is proposed. The better the quality of clusters more will be the recommendation accuracy. Evolutionary clustering based recommendation algorithm will give better quality clusters and thereby improve accuracy of prediction for the users. The evolutionary algorithms that uses genetic algorithm (GA) as the basis to depict evolution are not referred here as our proposed approach uses an optimization procedure rather than a GA based approach. Instead the proposed approach is compared with other evolutionary clustering approaches and widely used model based approaches that use various optimization criteria in the area of RSs application.

The rest of the paper is organized as follows: we present related work in the area of evolutionary clustering with some reference to RSs in Section 2. In Section 3, the concept of evolutionary clustering is presented and our proposed algorithm is explained. In Section 4, the evolutionary clustering based recommendation system is presented. In Section 5, the empirical test results of the proposed algorithm on real life dataset are presented and its comparison with other standard recommendation algorithms is discussed. Finally, Section 5 concludes the paper citing future direction of research in this area.

2. Related work

The most widely used technique for RSs is collaborative filtering [8]. Collaborative filtering systems are usually categorized into two types namely memory based and model based methods. Memory-based methods store the whole rating matrix and predict recommendations on the calculations between the target user and item and the rest of the rating matrix. Model-based methods fit a parameterized model to the given rating matrix and then predict recommendations on the basis of this model. Furthermore, model based methods are becoming more popular as with increase in the number of users and items memory based methods have serious limitation in terms of computation time and accuracy. Also, there are some other major advantages in model based technique over standard memory-based methods namely higher accuracy, constant time prediction, and an explicit and compact model representation. The major model based techniques include cluster-based CF [34,40], Bayesian classifiers [28], regression based methods [41]. Recent classes of successful CF models are based on low-rank matrix factorization. The regularized RSVD method [6] factorizes the rating matrix into a product of two low rank matrices (user-profile and item-profile) that are used to estimate the missing entries. An alternative method is Non-negative Matrix Factorization (NMF) [25] which differs in that it constrain the low rank matrices forming the factorization to have non-negative entries. Recent variations are Probabilistic Matrix Factorization (PMF) [32], and Nonlinear Principal Component Analysis (NPCA) [42].

The proposed approach in this paper implements a model based techniques using evolutionary clustering. Evolutionary clustering is a very novel research area which is first postulated by Chakrabarti et al. [9]. Most of the work in evolutionary clustering focuses on developing evolving communities and clusters of users and this paper propose that such clusters could further be related to user profile and can be used in generating recommendation. To the best of our knowledge, this approach has so far not being used

in the area of RSs research. Our proposed approach is different from the traditional evolutionary computation approaches, yet we have also mentioned work from evolutionary computation based on genetic algorithm and Particle Swarm Optimization approach to present an overall picture of the research area. Shankar et al. [35] extended Chakrabarti et al. [9] framework to frequent item set. They postulated that frequent item set are good candidates for evolutionary clustering as they naturally satisfy two criterion of evolutionary clustering. As frequent item set do not change much there is low history cost as well as there is a smooth transition to a new cluster. The updating of clusters over time are carried out by computing snapshot quality using general measure like FScore [43], NMI [38] whereas history cost is calculated using a score function. Kim and Han [21] presented a novel particle and density based evolutionary clustering method for dynamic networks, which overcome the drawbacks in earlier works of assuming fixed number of clusters over time [10]. This method consider dynamic network as a collection of particles called nanocommunities where each particles contains information about evolution of data and hence guide to find variable number of communities of arbitrary forming and dissolving. Further temporally smoothing is applied through a cost embedding technique. A mapping method based on information theory carried out the process of evolving, forming and dissolving clusters, making sequence of local clusters close to data-inherent quasi l-clique-by-clique.

Tang et al. [39] studied the evolution of a multi-mode network by analyzing it through temporal information. A multimode network is a combination of number of heterogeneous elements between which various kinds of interactions are there. They developed an effective iterative Evolutionary clustering algorithm to depict the evolving nature of communities. The algorithm update the cluster indicator matrix iteratively based on the attributes obtained from the clustering results of related objects and neighboring timestamps. This algorithm requires complex computation, which is a major drawback in its application for large-scale multi-mode network. Further, it requires users to provide weights for different interaction and temporal information as well as number of communities, which is a hurdle in the automatic evolution of the whole process. Lin et al. [27] presented an innovative algorithm that differs from the traditional methods, which discover communities by slicing timeframe and then finding their correspondences. This new framework named Facehet analyzes community evolution by maximizing the fit to observed data and temporal evolution. This framework extends the self-clustering algorithm to dynamic network where community participation is flexible and varied. Moreover, the algorithm also provides mechanism to determine the number of communities as well as the handling process of addition and removal of individual in a dynamic network. Folino and Pizzuti [14] propose a multi-objective approach named DYN-MOGA to discover community in dynamic network by employing genetic algorithm. The two competing objectives of evolutionary clustering snapshot quality and history cost are optimized using an input parameter that controls the preference degree of a user with respect to either one of them. A concept of community score was also introduced to determine snapshot quality, whereas history cost is calculated using traditional NMI measure. Das et al. [12] also presents an automatic clustering model called Multi-Elitist PSO (MEPSO) that employs a kernel-induced similarity measure instead of the conventional sum-of-squares distance. It is based on classical Particle Swarm Optimization (PSO) algorithm and the use of the kernel function makes it possible to cluster data that is linearly non-separable in the original input space into homogeneous groups in a transformed high-dimensional feature space. Senthilnath et al. [37] proposes a Firefly Algorithm (FA) for solving nonlinear optimization problems that overcome local optima problems based on the behavior of social insects. Earlier hybrid evolutionary optimization algorithms based on combining evolutionary methods and *k*-means were used to overcome local optima problems in clustering.

Another category of evolutionary algorithm implements Darwin biological evolution in the context of multiobjective optimization. Demir et al. [13] presents graph based sequence clustering approach that uses multiobjective evolutionary algorithms (MOEA). They have analyzed various MOEA and determine an efficient MOEA to cluster sequence data, which could further be employed in RSs. Fong et al. [15] presented a GA-based approach to analyses how the input variables can be coded into GA chromosomes in various modes for supporting combined modes of collaborative filtering. Silva et al. [36] on the other hand have proposed a graph based friend recommendation algorithm using genetic algorithm in a social network setting. They developed an algorithm that analyses the sub-graph composed by a user and all the others connected people separately by three degree of separation. Ko et al. [22] proposed a hybrid techniques for recommendation that allow the application of machine learning algorithms. The method generates recommendations based on clustering users and categorizing items with feature selection through association word mining by Apriori algorithm. They have used Genetic algorithm to group users based on items categorized by Naïve Bayes classifier. The algorithm recommends web documents to users based on grouped user preference and information of categorized items.

3. Evolutionary clustering

Cluster analysis is an important research field in Data Mining. This data mining technique partition a given dataset into clusters that reflect their natural data structure. Various heuristic or statistical approaches have been developed for formulation of such clusters [26]. However, traditional clustering algorithms do not take into account the existence of temporal dimension and its influence in the grouping of dataset. This will greatly degrade the performance of traditional clustering algorithms as far as the temporal evolution of clusters is concerned and thereby the concerned applicative areas. This issue motivates us to find more effective algorithm to conduct the cluster analysis upon temporal evolution of datasets in the area of recommendation application. The general objective in any clustering algorithm is to find, among all partitions of the data set, the best one according to some quality measure which is usually called an objective function or cost function. Consequently, finding the partitions of the dataset that yield the highest values of the objective function is a challenging optimization problem [2,18]. The traditional clustering algorithms typically fail to optimize any objective function globally [7]. We have proposed a cost function that optimizes the objective function and produce accurate clustering results particularly for evolutionary datasets. Evolutionary clustering is applied in many real world problems such as market segmentation, social network analysis, web mining and bioinformatics. From a user point of view, evolutionary computation produces relevant results that depict changes in user′s preferences over a period by producing clusters that evolves smoothly with time. Thus, Evolutionary clustering is emerging as an important area of research that could preserve quality in the fast changing world.

Evolutionary clustering is often considered as an offshoot of Incremental clustering as well as methods that are used in clustering data streams. Though both of them are similar in the sense that they all deal with data that changes with time, but the difference is well explained by Shankar et al. [35] Data stream clustering focuses on optimizing time and space constraints while

evolutionary clustering is concerned about temporal smoothness. Similarly, Incremental clustering does not maintain relevancy to existing clustering as opposed to evolutionary clustering. It concentrates on computational efficiency at the cost of low quality like DataStream mining. Thus, the two differentiate themselves a lot from the concepts of evolutionary clustering which focuses on the quality of clusters. On one hand, a number of evolutionary algorithms for solving clustering problem have been proposed that treat clustering as NP hard problem and are based on optimization of some objective function [20,30,38,40]. On the other hand, Chakrabarti et al. [9] is taking a completely different outlook. He implemented evolutionary clustering by building a framework termed as temporal smoothness that produces updated clusters from data coming at different time stamps. This method combines two conflicting objectives called snapshot quality and history cost. The clustering algorithm should tradeoff the advantage of maintaining a consistent clustering overtime termed as snapshot quality with the cost of deviating from an accurate representation of current data. The preservation of clustering quality is achieved by not deviating too much from history and the conflicting objectives are achieved simultaneously through the optimization of the whole process. As the objects to be clustered evolve over time, in many real world applications like RSs, a new clustering result is desired at each time step. In such cases, evolutionary clustering outperforms traditional clustering methods by producing clusters that can reflect long-term trends while being robust to short-term variations. Thus, Evolutionary clustering based clustering method given by Chakrabarti et al. [9] attempt to optimize cluster accuracy by maximizing incoming relevant new data at the current time and minimizing clustering drift from the historical data. The optimizing of clustering method needs an input parameter from the user that determines the preference of user regarding the two competing objectives.

We propose an evolutionary clustering algorithm for dynamic recommender systems. The algorithm uses the framework proposed by Chakrabarti et al. [9] and attempt to optimize cluster accuracy by maximizing incoming relevant new data at the current time and minimizing clustering drift from the historical data; we name this algorithm EVAR (Evolution VARiance clustering algorithm). EVAR discovers up to date clusters that are evolving with respect to time for finding new preferences of users in recommender systems. EVAR uses a cost function that optimizes an objective function and produces better clustering results particularly for evolutionary datasets. The conflicting objectives named *snapshot quality* and *history cost* are maximized and minimized simultaneously. The snapshot quality is represented using a new parameter; we call it *variance score*. The variance score determines the intra-cluster difference and places different items into different clusters according to their variance scores. The lower the variance score, the more will be quality of clusters. We use a well-known entropy measure called Non-Mutual Information (NMI) to determine the history cost [38]. NMI measures the similarity of two clusters by evaluating the values of the variance scores, obtained at the current and previous time stamps. EVAR optimizes the two conflicting objectives and their respective functions to determine the best possible quality clusters. EVAR demands an input value from a user to control his/her preference degree with respect to either snapshot quality or history cost.

3.1. Problem formulation

The field of data mining is a combination of various learning algorithms. Clustering algorithms is one such category that is widely used in unsupervised learning. RSs, on the other hand uses a technique called collaborative filtering which gives recommendation by finding similar users and predicting new user preference based on their similarity. Clustering algorithms have been very efficiently applied in the process of collaborative filtering to find similar users by developing clusters of similar users or items for prediction. However, with changing user′s requirement, a new mechanism is required to give accurate recommendations. For implementing such recommendation system, a new method EVAR is proposed here. EVAR uses a framework of Evolutionary clustering introduced by Chakrabarti et al. [9], which clusters data over a period of time. At each time stamp, a new cluster is produced by optimizing two competing parameter named snapshot quality and history cost. Snapshot quality refers to the quality of clusters formed and how accurately they depict data at the current time while history cost implies that the new cluster should not differ dramatically from the earlier one. This framework in fact focuses on smooth transition of clusters, which evolves over time by maximizing snapshot quality and minimizing history cost. The total quality of the sequence is defined as follows:

$$\sum_{t=1}^{T} Sq(C_t M_t) - \sum_{t=2}^{T} C_p\, Hc(C_{t-1}, C_t) \tag{1}$$

where

$Sq(C_tM_t)$ return snapshot quality of cluster C_t at time t w.r.t input m.
$Hc(C_{t-1},C_t)$ return history cost of cluster C_t at time t w.r.t time $t-1$.
C_p denotes the parameter for adjustment of the two objectives.

Let $U=\{1, 2, 3\ldots, n\}$ is the universe of objects to be clustered. At each timestamp t where $1 \le t \le T$, a new set of data arrives to be clustered. We assume that this data can be represented as an $n \times m$ matrix M_t that expresses the relationship between each pair of data objects. The relationship expressed by M_t is based on similarity or based on rating given by a user at timestamp t depending on the requirements of the particular underlying algorithm. We define Evolutionary Clusters as a group of items at a given time stamp t. Let $T=\{1,2,3\ldots,t\}$ be a finite set of time stamps and $I=\{i_1,i_2\ldots, i_{tm}\}$ be a set of m items and $U=\{u_1,u_2,\ldots. u_{tn}\}$ be n users arrived at different time stamps. Let $C=(C_1, C_2, C_t)$ be set of clusters at different timestamps. Let C_1 be the cluster at timestamp T_1. When a new item i_t is added at timestamp t or an old item changes its preference level and variance score, the EVAR produces a new clustering C_t, which optimizes the quality of clusters. A cluster group C_t where $C_t=\{C_{t1},C_{t2},C_{t3}\ldots C_{tk}\}$ (k depicts the number of cluster items) is a group of items at any given time and the partitioning of items is such that each group gives the maximum similarity and minimum variance. This is achieved by clustering algorithm through defining their cost function depicting the quality of clusters. This algorithm takes input $M1,\ldots M_t$ at each timestamp and produces clusters $C1,\ldots,C_t$ for the corresponding timestamp t based on the new matrix and history so far.

Different functions are further proposed by different researchers to determine the cost function. The major contribution of out proposed approach is the formation of a cost function that optimizes the two conflicting parameter specifically for determining change in user interests over a period of time. The cost function is defined as a combination of snapshot cost and history cost in the temporal smoothness framework given by Chakrabarti et al. [9] and further adopted by [14,21,35]. This cost function can tradeoff between history cost and snapshot quality and is defined as follows:

$$CF = \alpha \times Sq + (1-\alpha) \times Hc \tag{2}$$

where α is a parameter taken from the user to prioritize one of the two objectives. When $\alpha=1$, the function returns clustering results

without temporal smoothing. When $\alpha=0$, however, clustering result produced are similar to previous one. For testing purposes the value of α is taken as .5 to remove the bias towards any particular function in the total quality of the clusters.

The above defined cost function is optimized using two competitive objectives the snapshot quality *Sq* and History cost *Hc*. As snapshot quality measures how well the clusters represents the data at time *t*, we have defined this measure using a score called variance score, which minimizes difference within the items in a cluster and maximizes the similarity. The implementation of variance introduced in [1,11] has proved very effective in determining snapshot quality [10,17]. Variance score is the difference between the ratings of items in a particular cluster at a given point of time. Greater the value of the variance score, lower will be snapshot quality. We choose the timestamp *t* whose contribution to this expression is maximal.

$$Sq(C_tM_t) = \sum_{t=1}^{T} (1-VScore(M_t,\ t) \tag{3}$$

where

Sq is the snapshot quality.
VScore is the variance score at time *t* w.r.t *M*$_t$.

$$VScore(M_t,t) = \sum_{t=1}^{T} \frac{\sum_{u'\epsilon K(u)}(R(u',i)-RA(i))^2}{K} \tag{4}$$

where

VScore denotes variance of *K* neighbor rating for item *I* in matrix M_t at timestamp *t*.
K (*u*) denotes *k* neighbors of user *u* who rated item *I* and has the highest similarity sim (*u*, *u*′).
to user *u* at time stamp *t*.
R(*u*′, *i*) denotes the rating of user *u*′ on item *i*.
RA(*i*) denotes the average rating of all *K* neighbors on item *i*.

The history cost is defined using traditional entropy measures NMI [2].The normalized mutual information, NMI (A, B) is defined as:

$$NMI(t,t-1) = -2\frac{\sum_{i=1}^{C_t}\sum_{=j=1}^{C_{t-1}} C_{ij}\ \log(C_{ij}N/C_iC_j\ )}{\sum_{i=1}^{C_t} C_i\ \log(C_i/N)+\sum_{j=1}^{C_{t-1}} C_j\ \log(C_j/N)} \tag{5}$$

where C_t (C_{t-1}) is the number of groups in the partitioning t ($t-1$), C_i. (C_j) is the sum of the elements of *C* in row *i* (column *j*), and *N* is the number of nodes. If $t=t$-1, *NMI* (t, $t-1$)=1. If cluster at timestamp *t* and $t-1$ are completely different, *NMI* (t, $t-1$)=0. Thus, our second objective at a generic time step *t* is to maximize NMI (C_t, C_{t-1}).

4. Evolutionary clustering based recommendation model

We now formulate the recommendation modeling problem in terms of predicting the unknown ratings using a matrix representation by transforming it into a weighted matrix approximation problem and using the evolutionary clustering based approach for solving it. Let $U=\{u\}_{u=i}{}^{n}$ be the set of n users and $I=\{i\}_{i=1}{}^{m}$_ be the set of m items. Let $A=n\times m$ be the ratings matrix such that a_{ij} is the rating of the user u_i for the item i_j.

There are the two phases of our recommendation model based on evolutionary clustering:

(i) Neighborhood computation, which involves clustering the ratings matrix and computing the neighbor of a particular user or item which, could be later used for prediction,

(ii) Prediction, which consists of estimating an unknown rating from the neighborhood calculated above.

4.1. Neighborhood computation

The main objective of this component is to compute all the parameters that are required for fast prediction of the unknown rating. In our evolutionary clustering based approach, this essentially involves calculating clusters of users and items. First, we select the number of user-clusters *k*, considering the effect on the recommendation accuracy and resource requirements. Afterwards, we perform evolutionary clustering on the user-preference data. A model is then built with *k* surrogate users, directly derived from the *k* centroids: $\{c_1, c_2,\ldots, c_k\}$, where each c_i is a vector of size *m*, the number of items. That is, $c_i=(R_{ci,a1}, R_{ci,a2},\ldots, R_{ci,am})$, where $R_{ci,aj}$ is the element in the centroid vector c_i corresponding to the item a_j. Further, since $R_{ci,aj}$ is essentially an average value, it is 0 if nobody in the *i*-th cluster has rated a_j. Lastly, we perform similarity computations in order to choose neighborhood for a particular user through Pearson correlation coefficient [32].

4.2. Prediction

In order to compute the rating prediction $R_{ut,at}$ for the target (user, item) pair (*ut*, *at*), the following steps are taken. Firstly, we take the similarity computation values of the target user with each of the surrogate model users who have rated at using the Pearson correlation coefficient given below and we find up to l surrogate users most similar to the target user:

$$S_{u_t,ci} = \frac{\sum_{a\epsilon i}(R_{u_{t,a}}-RA_{u_t})(R_{c_{i,a}}-RA_{c_i})}{\sqrt{\sum_{a\epsilon i}(R_{u_{t,a}}-RA_{u_t})^2\sum_{a\epsilon i}(R_{c_{i,\ a}}-RA_{c_i})^2}} \tag{6}$$

where *I* is the set of items rated by both the target user and *i*-th surrogate user.

(1) $R_{u_t,a}$ is the rating prediction of user item pair (u_t,a_t).
(2) RA_{u_t} the average rating of user time pair *ut*.
(3) $R_{c_i,a}$ is the rating prediction of user item pair (c_i,a_t).
(4) RA_{c_i} is the average rating of user item pair C_i.

Secondly. we compute prediction using the adjusted weighted average:

$$R_{u_{t,at}} = RA_{u_t} + \frac{\sum_{i=1}^{k}(R_{c_i,at}-RA_{ci})S_{ut,ci}\ .f_{ui}^{\alpha}(t)}{\sum_{i=1}^{l} S_{ut,ci}.f_{ui}^{\alpha}(t)} \tag{7}$$

where $R_{c_i,at}$ is the rating prediction of user item pair (c_i,a_t).

RA_{ci} is the average rating of user item pair C_i.
$S_{ut,ci}$ is the value calculated in the first step.
K is the number of neighbors (clusters).

$$f_{ui}^{\alpha}(t) = \sum e^{-\alpha(t-t_{ui}}\sum RA_{u_t} \tag{8}$$

4.3. Evolution of the clusters

The evolutionary approach proposed in this paper is required to work in dynamic settings where ratings are being continuously updated. The traditional clustering algorithms, though quite efficient, are primarily intended for a static setting where the users, items and the ratings are fixed. This exclusion of new ratings in formulating clusters in traditional algorithms could result in poor performance. This problem is tackled by considering an incremental updation mechanism. As we know, the prediction of

the ratings are done using the matrix values termed as summary statistics, updating these statistics using the new ratings will include the most recent information. Three cases emerge when we include new rating. In the first case, the new rating corresponds to an existing user and item, and here we update the corresponding averages. In the second case, when either the user or item is new, then there is no cluster assignment for this new entity. This new user is then assigned to an intermediary temporary cluster and their corresponding averages are updated. During the next run of the evolutionary clustering algorithm, the users in the temporary user cluster and the items in the temporary item cluster are reassigned to one of regular user and item clusters.

The following pseudo code illustrates the implemented EVAR clustering algorithm:

Algorithm 1. Clusters the user-item rating matrix M_t with t timestamps into k clusters,

1. **procedure** EVAR(M_t, k,T)
2. for $t \leftarrow 0$
3. **repeat**
4. $t \leftarrow t+1$
5 Choose the initial centroid c1 to be m1.
6. Choose the next centroid ci by selecting $c_i = m_t' \in M_t$ which maximizes cost function CF (explained in section above):
$\sum_{t=1}^{T} Sq(C_t M_t) - \sum_{t=2}^{T} C_p\, Hc(C_{t-1}, C_t)$
7. **until** ($t=T$) and k centroids are found
8. **end for**
9. **return** { $c1$, $c2$, … ck } i.e k centroids
10. **end procedure**

Algorithm 2. Predict user rating for an using evolutionary clustering for calculating similar users,

procedure EVARCF(U, I, M_t, k, T)
$M_t \leftarrow$empty
For each node € M
$N \leftarrow$neighborhood(EVAR,M, I,NR)
For each (i, s)€N
For each $r_{ui} \in i_r$
next($RM_{node,u}$) € r_{ui}
For each u€U
Pu ←empty
For each i€I
If r_{ui} ¢ u_r **Then**
EVAR←find-EVAR(M, K, t)
$N \leftarrow$empty
For each n€U
next(N)←similarity($RM_{EVAR,u}$,$RM_{EVAR,n}$) (Eq. (6))
sort(N) in descending order of similarity
$p \leftarrow$predict(u, i,N) (Eq. (7))
next(Pu)←(i, p)
Return ***P***

5. Experiments

This paper tries to understand the process of changes in user preferences and detecting those changes using an evolutionary clustering mechanism in a recommendation model using Matlab toolbox. In particular, we look at the performance of our approach with benchmark system using clustering mechanism on predicting user ratings on MovieLens dataset. By doing so, pros and cons of evolutionary clustering mechanism is investigated to give a full understanding of the advantage of this approach in the area of RSs.

5.1. Dataset

In this experiment, we present a comparative study of clustering technique of data mining with other standard techniques on various parameters using Movielens dataset (www.movielens.umn.edu). MovieLens data sets were collected by the GroupLens Research Project at the University of Minnesota and MovieLens is a web-based research RSs that was released in 1997. This website is visited by hundreds of users each week to rate and receive recommendations for movies [33]. The website has released a number of datasets from time to time. We have chosen specifically the 100 K MovieLens Dataset since it is quite big is size for scalability testing and also it contains timestamps for temporal experimentation. The data set used contained 100,000 time stamped ratings, with a scale of one to five, from 943 users and 1,682 movies, with each user rating at least 20 items. By taking an average of the whole dataset, each user has rated 151 movies, out of these 87 were judged to be relevant. The average score for a movie was 3.58.

5.2. Methodology and metrics

5.2.1. Benchmark CF system

To compare the performance of our clustering algorithm, we also entered the training ratings set into eight other benchmark recommendation engine that includes clustering based approaches as well as standard model based approaches that gives good accuracy. Thus, we have done empirical comparison of our approach with some other clustering methods like regularized RSVD [29], COCL [16], and ECOCL [20] as well as standard model based algorithms like MA[3,4], NNMF [25], PMF [32] and NPCA [40].

5.2.2. Experimental platform

The effectiveness of our clustering algorithm is tested over real life data and the performance is compared with other clustering algorithms. The evolutionary clustering code used here has been implemented in Matlab toolbox. The tests were run on a Pentium 4 2.80 GHz computer with 512 M RAM. Tests were run on Matlab Version 7.01 on Microsoft Windows XP Professional.

5.2.3. Experimental steps

The effectiveness of our proposed clustering approach is evaluated empirically using Movielens dataset. For this, the dataset is divided into ten 80–20% random train-test splits for evaluating the prediction accuracy and then the results are averaged over the various splits. This is done for performing Ten-fold cross validation in which the final results are averaged on these ten sets. For the purpose of comparison, we perform the same experiments using other benchmark recommender models. We use the same train/test ratio x, and number of neighbors. We obtained rating predictions for each sample according to the specific recommendation model. We evaluated the results using the MAE metric and also noted the run time elapsed in milliseconds.

5.3. Results

In this section, the results of experiments performed to evaluate the effectiveness of our proposed clustering approach are presented. As discussed earlier, we have used the Movielens

dataset1 consisting of 100,000 ratings (1–5) by 943 users on 1682 movies. We used mean absolute error (MAE, RMSE and runtime) to evaluate and compare different methods. Six methods were used for comparison excluding the proposed method:

(1) RSVD: Singular value decomposition.
(2) COCL: Co-clustering.
(3) ECOCL: Evolutionary co-clustering
(4) EVAR: Proposed approach.
(5) MA: Memetic algorithm.
(6) NNMF: Non-negative matrix factorization.
(7) PMF: Bayesian probabilistic matrix factorization.
(8) NPCA: Nonlinear Principal Component Analysis.

The experiments compared our time based evolutionary clustering based algorithm with other standard benchmark recommendation algorithms in terms of both run time and prediction accuracy to show the tradeoff between efficiency and effectiveness. In order to show the generality of approach, we tend to compare the proposed approach with other model based algorithms as well as some clustering based recommendation algorithms. The performance comparisons for rating prediction for all the algorithms are summarized in Table 1. Clearly, from Table 1, it can be inferred that all algorithms performs better than COCL [16] in terms of prediction time as well as accuracy. The mean absolute error values of all the other algorithms are rather comparable at $t=1$, but with the $t=10$, there a comparable improvement in the accuracy of the proposed algorithm as rest of the model based algorithms [3,25,32,42] do not recomputed values with the incoming new data due to high computational costs. Moreover our proposed approach requires fewer parameter and less training time compared to rest of the algorithms. This contradicts with the result in [30,34], where the prediction quality is worse in case of the clustering algorithms. The reason is that previous clustering approaches group [16,20] users using only rating data and often result in less personal and worse accuracy than classical CF algorithms. Here we cluster users based on their recent temporal rating (the variance score), so using additional information for clustering has the benefit. Therefore, our proposed evolutionary clustering based RSs could resolve the very large scale dataset problem with high prediction quality and less computation time. The proposed evolutionary clustering approach is similar to the other clustering-based techniques in the sense that neighborhoods are employed for prediction, the main difference being that the clusters are generated on the basis of temporal dimension so that user/item synonymy ceases to be a problem. In addition, the evolutionary clustering algorithm also optimizes the variance of a user rating by reconstruction of the ratings matrix like RSVD and NNMF. However, unlike RSVD and NNMF, the effects of these changes in the ratings matrix are restricted to a particular row/colomn which makes it possible to have incremental updates efficiently.

Table 1
Results.

Algorithm	Training time (milliseconds per rating)	MAE ($t=1$)	MAE ($t=10$)
RSVD	27.05	0.6970	0.7311
COCL	3.45	0.7781	0.7513
ECOCL	2.27	0.7626	0.7336
EVAR	1.88	0.7215	0.7239
MA	28.30	0.8254	0.8869
NNMF	3.07	0.7299	0.7467
PMF	12.08	0.7626	0.7826
NPCA	11.18	0.7776	0.7857

Furthermore, we can see that the models RSVD and NNMF can indeed outperform our EVAR method given that they were updated at every time step (i.e. $td=1$) although their run time at such update frequency were almost 15 times longer than the most of the other method. Also, MA [3,4] based GA recommender are also unsuitable in such an environment as they tend to take large training time as incrementally creating such genetic material (by means of mutation operators and crossover) take a lot of time and make the use of genetic algorithms mostly impractical for on-line recommender systems. This is an issue which is less addressed in the GA-based text filtering literature. The two incremental algorithms RSVD and NNMF use a simple strategy to incrementally maintain their model at each time step given new ratings. They use the parameters in the most recent model to initialize the training of the next model. There is very little change in parameters in frequent updates, so we use a parameter td that controls how frequently the model is updated or retrained. Moreover, we can see that if the rate of updation of RSVD, NNMF and other model based methods like NNMF is reduced i.e. td is increased; their performance tends to decline drastically and became much worse than our proposed approach EVAR while they still took much longer run time than the EVAR. This empirically proved that although the EVAR method yielded suboptimal prediction quality, it was nonetheless the most efficient method, which was nearly giving good prediction quality to a certain extent as well as it is faster than the more complicated models. The applications areas that generate user feedback data at a very high speed, it is very difficult to update complicated models such as RSVD and NNMF frequently to achieve optimal prediction accuracy. But as we have seen, the EVAR model would be ideal in terms of both efficiency and effectiveness in such cases. Finally, although most of these algorithms have comparable accuracy in a stable environment yet computation time for EVAR is much lower. In addition, when the models are periodically updated the accuracy of EVAR algorithms tends to be higher as well.

We now present few other experimental results showing how various parameters affect the prediction accuracy and the average prediction time of various algorithms.

5.3.1. *Prediction accuracy vs. choice of CF-algorithm*

The MAE (mean absolute error) values obtained using the different algorithms is shown in Fig. 1. It can be noted from the table that although the prediction accuracy of NNMF, MA and RSVD is higher than other clustering based as well as model based approaches yet this difference tends to subsidies when the rate of updation of RSVD, MA and NNMF is reduced. Infact, their performance tends to decline considerably and became much worse than our proposed approach EVAR while they still took much longer run time than the EVAR. Therefore, the deciding factor in favor of the evolutionary clustering approach for giving better

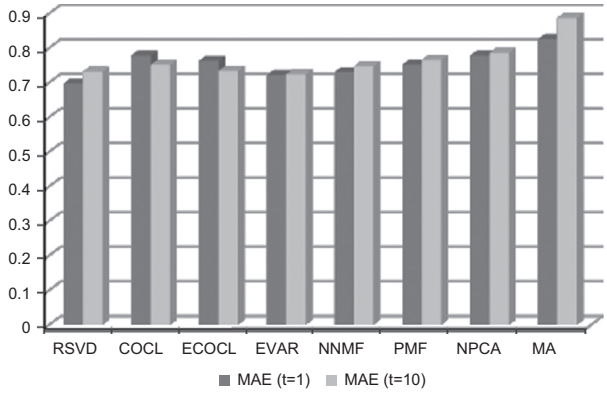

Fig. 1. Prediction accuracy vs. Choice of CF-algorithm.

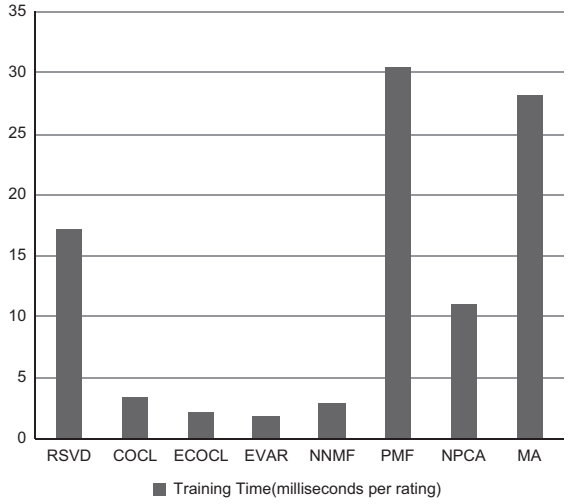

Fig. 2. Prediction time vs. Choice of CF-algorithm.

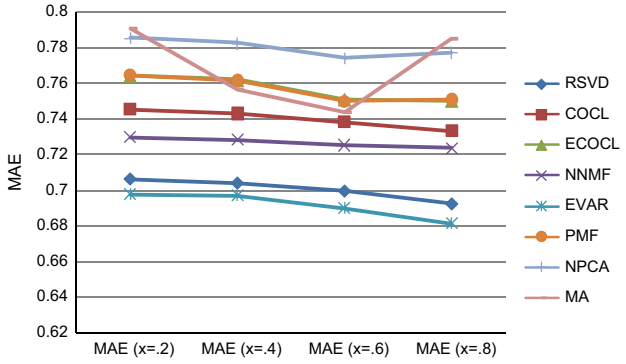

Fig. 3. Sensitivity of different training-test ratio *x*.

performance is that it required fewer calculations and less training time compared to all the other methods while giving optimal prediction accuracy.

5.3.2. *Prediction time vs. choice of CF-algorithm*

The average prediction time for the various recommendation algorithms on the 100 K MovieLens Dataset is shown in Fig. 2. The prediction time is reasonably low in case of evolutionary clustering based approach in comparison with RSVD, MA and PMF methods, although it is comparable with ECOCL,COCL and NNMF methods.

5.3.3. *Sensitivity of different training-test ratio x*

To determine the sensitivity of density of the dataset we carried out an experiment where we varied the value of x from 0.2 to 0.8 in an increment of 0.2. For each of these training-test ratio values we ran our experiments using our proposed algorithm and the other recommendation algorithms. The results are shown in fig. 3. We observe that the quality of prediction increase as we increase x and our proposed CF is better than the traditional methods as depicted by the decreased values of MAE with an increase in training size. The RSVD approach shows better results for low values of x but as we increases the value of x, the quality tends to decrease. This is because of the high computational costs associated with such models that restrict their matrix reconstruction.

5.3.4. *Sensitivity of no. of neighbor/no. of clusters*

We compare the proposed evolutionary clustering method with the other traditional methods w.r.t to no. of neighbors. The size of the neighborhood has a significant effect on the prediction quality. In our experiments, we vary the number of neighbors and compute the MAE. The MA approach shows better results for certain values of K but as we increase or decrease the value of k, the quality tends to decrease. The obvious conclusion from Fig. 4, is that MAE tends decrease with the increasing value of no. of neighbors and the values becomes stable after a while. Thus, an optimum no. of neighbors must be selected to and in this case it is 30.

Our experiments suggest that clustering based neighborhood provides comparable prediction quality as other widely known model based recommendation approach. However, at the same time they tend to significantly improve the online run time performance. This demonstrated the effectiveness of clustering algorithm w.r.t those real life application that require periodic updation In short, most of these recommendation algorithms have comparable accuracy in a stable environment yet computation time for EVAR is much lower. In addition, when the models are periodically updated the accuracy of EVAR algorithm tends to be higher as well. Most of the model based algorithms requires continuous updation to maintain accuracy which is infeasible as it requires very high computation cost. Thus, EVAR tends to produce good accuracy, high scalability and less computation time while updating clusters periodically.

In future, improved clustering algorithms based on temporal dimension as well as enhanced prediction generation schemes can be used to improve the prediction quality further. Clustering techniques should be ideally used as the first step for reducing the candidate set for calculating similar users or for performing neighborhood computation across several recommender engines. While dividing the data into clusters may hurt the accuracy or recommendations to users near the periphery of their assigned cluster, time based clustering methods may bridge the valuable gap between accuracy and throughput.

The dataset MovieLens, has high rating variance among its users and therefore, it is hard to establish statistical significance for behavioral differences among its users. The variance of user rating ranges from some rating couple of movies while others rate hundreds; some logging in once every day while others once per week. Thus, we have chosen some interesting cases i.e users that rated a large number of items over a period of time and the algorithms are tested per user case for such users. For the experiments, we have chosen randomly 10% out of all customers in the dataset as the test customers. For each test customer, we have chosen randomly ten movies that are actually rated by the test customer as the test movies. The final experimental results are averaged over the results of ten different test sets for a statistical significance. We have then computed statistical significance of differences between pairs of algorithms using the Wilcoxon Signed

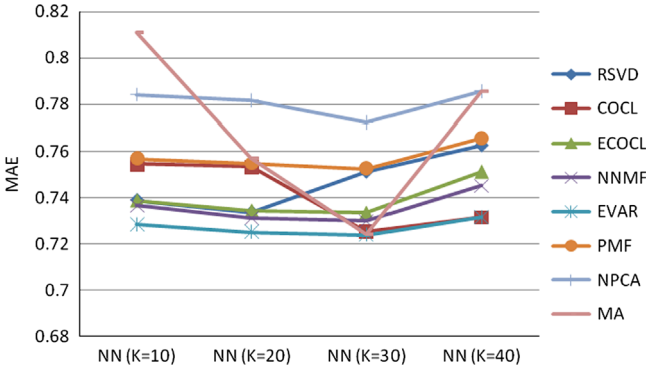

Fig. 4. Sensitivity of no. of neighbor/no. of clusters.

Rank Test [5] with 95% confidence to establish confidence in the analysis of result. This is done in order to test if one algorithm consistently obtains higher metric values than the other. Moreover, we have also calculated standard deviation for our proposed algorithm and the experimental results indicate that it is comparable to the other algorithms in terms of function evaluations and standard deviations. Finally, as we have used only one particular dataset, we must state that in any case that these results are not conclusive. As different data splits used for testing shows variability on the results, therefore further research using additional datasets is needed so as to reach to more robust conclusions. In future we intend to evaluate different approach of generating different time-evolving training sets more deeply, so as to present more vivid explanations.

However, in real applications, we can only consider opinions of a few items in dataset due to the scalability problem, and since our approach yield significantly more accurate prediction than traditional approach when number of items is significantly large, our approach can efficiently contribute to the improvement in the accuracy of rating prediction in real applications. In addition to these dataset, we intend to test the proposed approach with other real life datasets, such as Netflix (dataset available at netflixprize.com) and Yahoo! Movies (ratings publicly displayed at movies.yahoo.com). Currently, though the proposed technique is efficient in terms of accuracy, it shows some restrain in terms of scalability. We hope to present more results of our experiment in the nearest future.

6. Conclusion

An evolutionary algorithm is used to explore the evolving process of any system in general. These methods can be applied in collaborative filtering RSs to incorporate new data over a period of time and update user profile with its current requirements. However, this problem has not been adequately addressed. This paper examined the effectiveness of evolutionary clustering algorithm for generating quality clusters and predicting recommendations based on real life datasets. The evolving clusters are used for depicting user preferences and building an adaptive recommender system. The algorithm uses a score called variance score to calculate the difference of user rating thereby calculating their change in the evolutionary process at each timestamp. Thus, the algorithm provides a cluster, which is refined at each timestamp by two competing score.

Experimental results on large real world data sets demonstrated that our algorithm can provide high quality predictions at a much lower computational cost compared to traditional clustering algorithms and other model based approaches in repeatedly building models. The biggest disadvantage of model based (comparatively slow training) could be improved significantly. Our experimental results on several real life datasets show that this scalability does not come at the cost of quality. The proposed approach is content agnostic and thus can be easily extendible to other domains as well. Our method provides RSs that gives accurate results and updates incrementally without spending time on tuning. In the future, we would consider more complex models involving evolutionary clustering based matrix factorization for implementing the temporal dynamics of user feedbacks. Also, further work involves evaluating the algorithm on different datasets to analyze the efficacy with respect to other domains. The computational efficiency and memory requirement also needs to be studied in detail with adjustment of algorithm parameters such as Cp. The algorithm could also be adjusted for solving many other Information retrieval problem as well as data mining problems.

References

[1] Adomavicius, G., Kwon, Y., 2008. Overcoming accuracy-diversity tradeoff in recommender systems: a variance-based approach. In: Proceedings of 18th Workshop on Information Technology and Systems WITS, Paris, France.

[2] X. Amatriain, A. Jaimes, N. Oliver, J.M. Pujol, Data mining methods for recommender systems, Recommender Systems Handbook (2011) 39–71.

[3] Banati, H., Shikha, M., 2010. Memetic collaborative filtering based recommender system, Second Vaagdevi International Conference on Information Technology for Real World Problems (VCON), pp. 102–107.

[4] H. Banati, S. Mehta, A Multi-perspective evaluation of MA and GA for collaborative filtering recommender system, International Journal of Computer Science & Information Technology (IJCSIT) 2 (5) (2010) 103–122, http://dx.doi.org/10.5121/ijcsit.2010.2508.

[5] D.F. Bauer, Constructing Confidence Sets using Rank Statistics, Journal of the American Statistical Association 67 (339) (1972) 687–690.

[6] Billsus, D., Pazzani, M.J. 1998. Learning collaborative information filters. In: Proceedings of the Fifteenth International Conference on Machine Learning, pp. 46–54.

[7] J.G. Booth, G. Casella, J.P. Hobert, Clustering using objective functions and stochastic search, Journal of the Royal Statistical Society: Series B (Statistical Methodology) 70 (2008) 119–139.

[8] Candillier, L., Meyer, F., Boull, M, 2007. Comparing state-of-the-art collaborative filtering systems, In: Proceedings of the 5th international conference on Machine Learning and Data Mining in Pattern Recognition (MLDM '07), Petra Perner (Ed.), Springer-Verlag, Berlin, Heidelberg, pp 548–562.

[9] Chakrabarti, D., Kumar, R., Tomkins A., 2006. Evolutionary clustering, In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining KDD 06, pp. 554–661.

[10] Chi, Y., Song, X., Zhou, D., Hino, K., Tseng, B.L. 2007. Evolutionary spectral clustering by incorporating temporal smoothness. In: Proceedings of Knowledge Discovery and Data Mining(KDD '07), pp. 153–162.

[11] S. Das, A. Abraham, A. Konar, Automatic clustering using an improved differential evolution algorithm, IEEE Transactions on Systems Man and Cybernetics Part A Systems and Humans 38 (1) (2008) 218–237.

[12] S. Das, A. Abraham, A. Konar, Automatic kernel clustering with a Multi-Elitist Particle Swarm Optimization Algorithm, Pattern Recognition Letters 29 (5) (2008) 688–699.

[13] Demir, G.N., Uyar, A.S., Oguducu, S., 2007. Graph-based sequence clustering through multiobjective evolutionary algorithms for web Recommender Systems. In: Proceedings of 9th annual Conference on genetic and evolutionary computation, pp. 1943–1950.

[14] Folino, F., Pizzuti, C., 2010. Multiobjective evolutionary community detection for dynamic networks. In: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation GECCO2010, pp. 535–536.

[15] Fong, S. Ho, Y., Hang, Y., 2008. Using genetic algorithm for hybrid modes of collaborative filtering in online recommenders. In: Proceedings of the 2008 8th International Conference on Hybrid Intelligent Systems (HIS '08), IEEE Computer Society, Washington, DC, USA, pp 174–179.

[16] George, T., Merugu, S., 2005. A scalable collaborative filtering framework based on co-clustering. In: Proceedings of the IEEE International Conference on Data Mining, pp. 625–628.

[17] J. Handl, J. Knowles, An evolutionary approach to multiobjective clustering, IEEE Transactions on Evolutionary Computation 11 (1) (2007) 56–76.

[18] A. Hatamlou, S. Abdullah, H. Nezamabadi-pour, A combined approach for clustering based on K-means and gravitational search algorithms, Swarm and Evolutionary Computation 6 (2012) 47–52.

[19] E.R. Hruschka, R.J.G.B. Campello, A.A. Freitas, A.C.P.L.F. De Carvalho, A survey of evolutionary algorithms for clustering, IEEE Transactions on Systems Man and Cybernetics Part C Applications and Reviews 39 (2) (2009) 133–155.

[20] Khoshneshin, M., Nick Street, W., 2010. Incremental collaborative filtering via evolutionary co-clustering. In: Proceedings of the Fourth ACM Conference on Recommender Systems (RecSys '10), ACM, New York, NY, USA, pp. 325–328.

[21] M.S. Kim, J. Han, A particle-and-density based evolutionary clustering method for dynamic networks, VLDB 2 (1) (2009) 622–633.

[22] Ko, S.J., Lee, J.H., 2001. Discovery of user preference through genetic algorithm and Bayesian categorization for recommendation, Conceptual Modeling for New Information Systems Technologies, ER 2001 Workshops, LNCS vol. 2465, Springer-Verlag, pp. 471–484.

[23] Koren, Y., 2009. Collaborative filtering with temporal dynamics. In: Proceedings of the 15th ACM Interna tional Conference on Knowledge Discovery and Data Mining, 2009, pp. 447–456.

[24] Lathia, N., Hailes, S., Capra, L. 2009. Temporal collaborative filtering with adaptive neighbourhoods. In: Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval SIGIR, Boston, Massachusetts, pp. 796–797.

[25] D.D. Lee, H.S. Seung, Algorithms for non-negative matrix factorization, in: T. K. Leen, T.G. Dietterich, V. Tresp (Eds.), Lecture Notes in Computer Science, 4029, 2006, pp. 548–562.

[26] Li, T., Anand, S.S., 2008. HIREL: An Incremental Clustering Algorithm for Relational Datasets, ICDM, pp. 887–892.

[27] Lin, Y.R., Chi, Y., Zhu, S., Sundaram, H., B. Tseng, L., 2008. FacetNet: A framework for analyzing communities and their evolutions in dynamic networks. In: Proceedings of 17th International Conference on World Wide Web 2008, pp. 685–694.

[28] Miyahara, K., Pazzani, M.J., 2000. Collaborative filtering with the simple bayesian classifier. In: Proceedings of the 6th Pacific Rim International Conference on Artificial Intelligence, pp. 679–689.
[29] Paterek, A., 2007. Improving regularized singular value decomposition collaborative filtering, In: Proceedings of KDD Cup and Workshop, pp. 39–42.
[30] Papagelis, M., Rousidis, I., Plexousakis, D., Theoharopoulos. E., 2005. Incremental collaborative filtering for highly-scalable recommendation algorithms. In: Proceedings of International Symposium on Methodologies of Intelligent Systems, pp. 553–561.
[31] C. Rana, S.K. Jain, A Study of Dynamic Features of Recommender Systems, Artificial Intelligence Review, Springer, Netherlands, 2012, http://dx.doi.org/10.1007/s10462-012-9359-6 (ISSN: 0269–2821).
[32] R. Salakhutdinov, A. Mnih, Probabilistic matrix factorization, in: J.C. Platt, et al., (Eds.), Learning, vol. 20, 2008, pp. 1–8.
[33] Sarwar, B., Karypis, G., Konstan, J., Riedl, J., 2001. Item-Based collaborative filtering recommendation algorithms. In: Proceedings of the 10th International World Wide Web Conference, pp. 285–295.
[34] Sarwar, B., Karypis, G., Konstan, J., Riedl, J., 2002. Recommender Systems for large-scale e-commerce: Scalable neighborhood formation using clustering. In: Proceedings of the Fifth International Conference on Computer and Information Technology, pp. 158–167.
[35] Shankar, R., Kiran, G.V.R, Pudi, V. 2010. Evolutionary clustering using frequent itemsets, In: Proceedings of the First International Workshop on Novel Data Stream Pattern Mining Techniques Stream in KDD 2010, ACM Press, pp. 25–30.
[36] Silva, N.B., Tsang, I.R. Cavalcanti, G.D.C., Tsang, I.J., 2010. A graph-based friend recommendation system using genetic algorithm. In: Proceedings of the IEEE World Congress on Computational Intelligence (WCCI), Barcelona, Spain, pp.1–7.
[37] J. Senthilnath, S.N. Omkar, V. Mani, Clustering using firefly algorithm: performance study, Swarm and Evolutionary Computation 1 (3) (2011) 164–171.
[38] A. Strehl, J. Ghosh, Cluster ensembles a knowledge reuse framework for combining partitions, Journal of Machine Learning Research 3 (2002) 583–617. (Cambridge, MA, USA).
[39] Tang, L. Liu, H. Zhang, J., Nazeri. Z., 2008. Community evolution in dynamic multimode net works". In: Proceedings of KDD 2008, pp. 677–685.
[40] Ungar, L.H., Foster, D.P., 1998. Clustering methods for collaborative filtering. In: Proceedings of the AAAI Workshop on Recommendation Systems, pp. 1–16.
[41] S. Vucetic, Z. Obradovic, Collaborative filtering using a regression-based approach, Knowledge and Information Systems 7 (1) (2005) 1–22.
[42] Yu, K., Zhu, S. Lafferty, J., Gong. V., 2009. Fast nonparametric matrix factorization for large-scale collaborative filtering. In: Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 211.
[43] Zhao Y., Karypis, G., 2002. Evaluation of hierarchial clustering algorithms for document datasets. In: Proceedings of International Conference on Information and Knowledge Management, pp. 515–524.