Knowledge-Based Systems 131 (2017) 125–134

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

# Memetic algorithm based location and topic aware recommender system

Shanfeng Wang[a], Maoguo Gong[a,*], Haoliang Li[a], Junwei Yang[a], Yue Wu[b]

[a] Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education, International Research Center for Intelligent Perception and Computation, Xidian University, Xi'an, Shaanxi Province 710071, PR China
[b] School of Computer Science and Technology, Xidian University, Xi'an, Shaanxi Province 710071, PR China

ARTICLE INFO

*Article history:*
Received 10 May 2016
Revised 19 May 2017
Accepted 24 May 2017
Available online 7 June 2017

*Keywords:*
Memetic algorithm
Clustering
Latent Dirichlet Allocation
Location-based recommendation

ABSTRACT

Recommender systems based on locations and tags have received a great deal of interest over the last few years. Whereas, recent advances do not transcend limits of recommendation algorithms that solely use geographical information or textual information. In this paper, we propose a novel location and tag aware recommendation framework that uses ratings, locations and tags to generate recommendation. In this framework, all users are partitioned into several clusters by a newly designed Memetic Algorithm (MA) based clustering method. Normal users are recommended items obtained by applying Latent Dirichlet Allocation (LDA) to users within each cluster. For cold-start users, each cluster is viewed as a new user. Each cluster is recommended a list of items by applying LDA to all clusters. The recommendation list to the querying cluster is recommended to all cold-start users in this cluster. Extensive experiments on real-world datasets demonstrate that compared with state-of-the-art location and tag aware recommendation algorithms, the proposed algorithm has better performance on making recommendations and alleviating cold-start problem.



## 1. Introduction

In recent years, we have witnessed the rapid development of mobile Internet, Web 2.0 technology and Social Networking Services, like DoubanEvent, Foursquare and Facebook. Some applications, such as Foursquare, make it easier to check-in spatial locations, bookmark tags and share experience in the physical world. Users' online behaviors (e.g. commenting and rating) are in association with offline activities (e.g. watching movie, outside eating and watching a game). For instance, when people want to watch a movie or dine together with friends nowadays, they prefer making decisions based on the distance to the candidate targets or online tags.

Traditional recommender systems only take rating records into account, so that they are difficult to provide accurate recommendation. Later, many other features of users and items are taken for recommendation. First, an item is usually associated with textual information, such as categories and tags. Second, even two users with similar or even the same semantic topics can rank an item differently if they are in two different regions [1,2]. It is concluded that all ratings, textual information and location are vital information in recommendation.

Various methods have been proposed to learn rating records, textual and geographical information. It will face more challenges on utilizing these information to generate recommendation than traditional recommendation algorithms which just use ratings. In order to utilize textual information, tags associated with items are considered as the most important content features in [3,4]. However, unlike traditional recommendation (i.e., article recommendation [5]), the textual information associated with items is usually incomplete and ambiguous. Studies in [6] also showed that tags relevant to items have a deviation with user's preferences. Traditional recommendation algorithms, like collaborative filtering (CF) etc. have limited abilities to solve this problem. The study in [3] reported the effectiveness of Latent Dirichlet Allocation (or LDA) in alleviating tag incompleteness problem. One of the most popular methods to exploit location information for recommendation is location-based clustering [1,2,7]. Traditional recommendation algorithms, such as LDA [1], collaborative filtering (or CF) [2] and matrix factorization (or MF) [7] etc. have good performance on exploiting geographical information and alleviating cold-start problem, but there are still space for improvement. Clustering algorithms carried out in [1,7] were effective algorithms in utilizing location for recommendation. On the whole, it is still a challenge to

* Corresponding author.
*E-mail address:* gong@ieee.org (M. Gong).

http://dx.doi.org/10.1016/j.knosys.2017.05.030
0950-7051/

generate recommendation considering ratings, textual and location information simultaneously.

MAs are hybrid global-local heuristic search methodologies [8]. The global heuristic search is usually a form of population-based method, while the local search is generally considered as an individual learning procedure for accelerating the convergence to an optimum [8]. In general, the population-based global search has an advantage of exploring the promising search space and providing a reliable estimate of the global optimum [9]. However, the population-based global search is difficult in discovering an optimal solution around the explored search space in a short time. The local search is usually designed for accelerating the search and finding the best solutions in the explored search space. Therefore, this hybridization, which synthesizes the complementary advantages of the population-based global search and the individual-based local search, can effectively produce better solutions [10]. Recent studies have demonstrated that MAs are effective and efficient for clustering [11,12].

In order to combine the very best of the methods introduced before and overcome their deficiencies, we propose a memetic algorithm (or MA) based clustering for recommendation. Our work includes two phases. First, a novel fitness function is designed, taking rating records, tag and location information into consideration. The interest distributions based on rating and tag in the fitness are mined by LDA. All the users are partitioned into different clusters based on the proposed novel fitness function. Then, users are recommended with items obtained by applying rating and tag based LDA within each cluster and cold-start users are recommended with items obtained by applying rating and tag based LDA of every cluster. Specifically, our contributions in this study include:

- We propose a novel memetic algorithm based location and topic aware recommendation framework (MLTRS). Compared with traditional recommendation algorithms, our algorithm makes full use of rating records, tag and location information to make personalized recommendation and alleviate the cold-start problem.
- The LDA model is applied to mine the users' interest distributions based on rating records and tag information. A novel MA based clustering is proposed to partition users into several clusters. We design an objective function which represents users' interests based on users' rating records, tag and location information. Novel crossover, mutation and local search operators are designed, adapting for this clustering task. By combining LDA and memetic-based clustering algorithm, the proposed algorithm can make better recommendation.
- Extensive experiments are conducted on three real-world datasets. Experimental results show that the proposed algorithm outperforms other location-aware and topic-aware recommendation algorithms in recommendation accuracy. Especially, the proposed algorithm has outstanding performance on solving the cold-start problem.

This paper is organized as follows. Section 2 gives the motivations of our work. The proposed MLTRS is described in detail in Section 3. Section 4 shows the extensive experiments to validate the effectiveness of MLTRS. In Section 5, related works are reviewed. We summarize our work and discuss possible further improvements in Section 6.

## 2. Motivations of our work

As introduced in introduction, except for ratings of users on items, location and tag are also important information to improve the recommendation results. How to utilize and balance these information for recommendation is a challenge. In the proposed algorithm, we combine ratings, location and tag to generate recommendation, and then a good recommendation algorithm is needed.

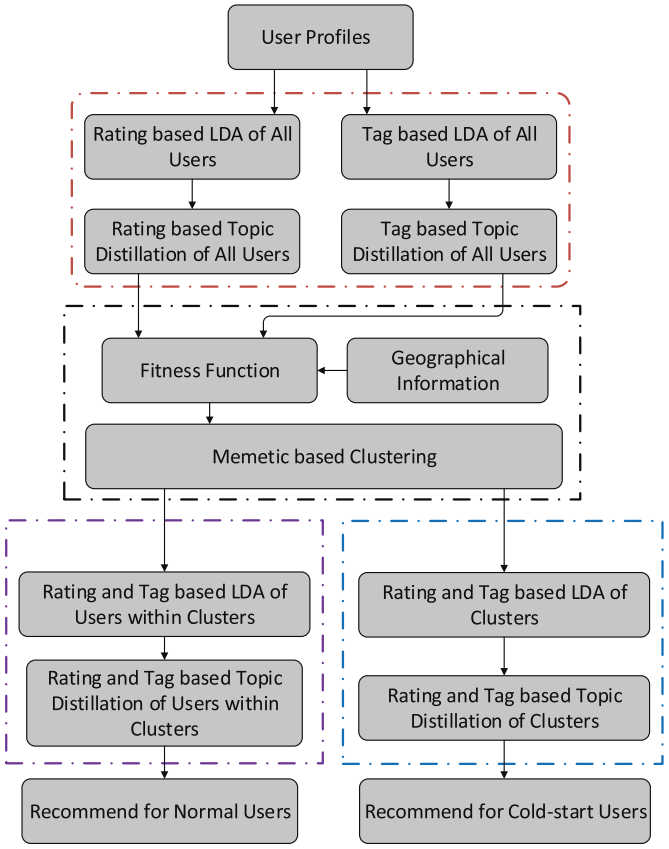



**Fig. 1.** Framework of MLTRS. The red, purple and blue dashed boxes show the process of LDA in different uses. The black dashed box shows the process of MA based clustering. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

LDA is one of the most representative models in recommendation. Not only does LDA have the advantage of preventing overfitting and alleviating data sparsity in rating based recommender systems [1], but also it can be used in location and tag based recommendation [5]. In the proposed algorithm, the LDA model is used to mine the users' interest distributions based on rating records and tag information. Then we use users' interest distributions and geographical information to cluster and calculate the region interest distributions to recommend for the users who have few rating records. In recommendation, memetic algorithm based clustering can alleviate data sparsity and partition users with similar preferences into the same cluster. Compared with traditional clustering methods, the MA based clustering method can find a better clustering result by balancing ratings, tag and location information.

It is concluded that to generate better recommendation, more information in recommendation is taken into consideration in this paper, while in previous studies, not all of these information is used. Second, to mine users' preferences effectively, LDA and memetic based clustering are adopted.

## 3. Memetic algorithm based location and topic aware recommendation system

In this section, we present the details of the proposed algorithm which is a key component of our work. Fig. 1 illustrates the framework of MTLRS. All users are partitioned by a MA based clustering method. We mine interests of users based on ratings and tags with

| user\item | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 4 | 2 | 0 | 2 | 3 | 0 | 4 |
| 2 | 1 | 0 | 2 | 3 | 0 | 4 | 0 | 5 | 0 | 3 |
| 3 | 4 | 2 | 5 | 0 | 5 | 0 | 4 | 1 | 0 | 2 |
| 4 | 0 | 0 | 4 | 1 | 0 | 0 | 1 | 0 | 2 | 2 |
| 5 | 4 | 5 | 0 | 0 | 5 | 5 | 0 | 3 | 0 | 0 |
| 6 | 0 | 0 | 2 | 5 | 1 | 0 | 5 | 2 | 0 | 0 |
| 7 | 1 | 0 | 4 | 2 | 0 | 3 | 0 | 2 | 1 | 0 |
| 8 | 1 | 0 | 5 | 5 | 3 | 0 | 0 | 0 | 0 | 3 |
| 9 | 2 | 0 | 2 | 0 | 0 | 5 | 2 | 0 | 5 | 0 |
| 10 | 4 | 1 | 2 | 0 | 2 | 0 | 5 | 0 | 4 | 1 |

(a)

| users | latitude | longitude |
|---|---|---|
| 1 | 40.81 | -73.04 |
| 2 | 18.16 | -66.72 |
| 3 | 42.11 | -72.53 |
| 4 | 42.12 | -80.08 |
| 5 | 37.67 | -82.06 |
| 6 | 35.8 | -79.72 |
| 7 | 28.92 | -81.92 |
| 8 | 31.8 | -85.96 |
| 9 | 43.74 | -85.14 |
| 10 | 37.77 | -97.46 |

(b)

| items | tags | | |
|---|---|---|---|
| 1 | Animation | Children's | Comedy |
| 2 | Adventure | Children's | Fantasy |
| 3 | Comedy | Romance | |
| 4 | Comedy | drama | |
| 5 | Comedy | | |
| 6 | Action | Crime | Thriller |
| 7 | Comedy | Romance | |
| 8 | Adventure | Children's | |
| 9 | Action | Adventure | Thriller |
| 10 | Comedy | Drama | Romance |

(c)

**Fig. 2.** An example of MLTRS. (a) The ratings matrix of users on items. (b) Location of users. (c) Tags of items.

LDA. The geographical information is added to the fitness function, together with the rating and tag based distillation of users. In our framework, all the users are partitioned into two parts: normal users and cold-start users. Cold-start users are users who have rated no more than 10 items and the other users are normal users. Normal users are recommended with items obtained by applying rating and tag based LDA within each cluster. This recommendation technique makes recommendations by considering only the ratings and tags generated by users who are in the same cluster with the querying user. For cold-start users, the cluster is viewed as a whole, similar to a user. Each cluster is recommended a list of items by applying rating and tag based LDA. The recommendation list of the querying cluster is recommended to all cold-start users in this cluster, meaning that all cold-start users in the same cluster are recommended with the same recommendation list.

In the following, we give an example in introduction of the proposed algorithm to clarify the entire process of MLTRS in Fig. 2. In this example, there are 10 users and 10 items. Fig. 2 (a) is the ratings matrix of users on items. Fig. 2 (c) gives the location of all the users. There are 10 tags of items in this example as shown in Fig. 2 (c).

### 3.1. Rating based topic distillation

In LDA, each document is represented as a probability distribution over a number of words. The rating records in recommender system has a similar format with documents, that users consist of items as documents consist of words. LDA model is applied to rating based recommender system [13]. Given $M$ users containing $|\vec{Z^R}|$ topics expressed over $N$ distinctive items, we can represent $P(i|Z^R)$ with a set of $|\vec{Z^R}|$ multinomial distributions. Here we use the LDA model in maximum likelihood estimates in the context of recommender system. The complete probability model is as follows:

$$\begin{aligned} vec\theta^R &\sim Dir(\vec{\alpha^R}) \\ vecZ^R|\theta^R &\sim Mult(\vec{\theta^R}) \\ vec\phi^R &\sim Dir(\vec{\beta^R}) \\ vecI|\vec{Z^R}, \vec{\phi^R} &\sim Mult(\vec{\phi^R}) \end{aligned} \tag{1}$$

where, $\vec{Z^R}$ stands for a set of hidden topics, $\vec{\theta^R}$ denotes users preference distributions over the topics based rating records and $\vec{\phi^R}$ represents the specific topics $Z^R$s association distributions over items. $\vec{\alpha^R}$ and $\vec{\beta^R}$ are hyper-parameters of the prior of $\vec{\theta^R}$ and $\vec{\phi^R}$. The latent parameters are calculated to predict user behaviors and make recommendations.

| users | topics | | | | |
|---|---|---|---|---|---|
| 1 | 0.198 | 0.278 | 0.246 | 0.139 | 0.139 |
| 2 | 0.231 | 0.233 | 0.202 | 0.174 | 0.161 |
| 3 | 0.172 | 0.230 | 0.204 | 0.207 | 0.187 |
| 4 | 0.210 | 0.143 | 0.206 | 0.298 | 0.143 |
| 5 | 0.218 | 0.246 | 0.163 | 0.107 | 0.265 |
| 6 | 0.133 | 0.201 | 0.297 | 0.244 | 0.126 |
| 7 | 0.150 | 0.184 | 0.148 | 0.281 | 0.237 |
| 8 | 0.160 | 0.189 | 0.195 | 0.313 | 0.149 |
| 9 | 0.176 | 0.210 | 0.156 | 0.208 | 0.249 |
| 10 | 0.197 | 0.137 | 0.241 | 0.210 | 0.215 |

(a)

| users | topics | | | | |
|---|---|---|---|---|---|
| 1 | 0.243 | 0.152 | 0.152 | 0.152 | 0.302 |
| 2 | 0.128 | 0.214 | 0.299 | 0.147 | 0.212 |
| 3 | 0.135 | 0.227 | 0.252 | 0.147 | 0.240 |
| 4 | 0.210 | 0.261 | 0.141 | 0.198 | 0.191 |
| 5 | 0.245 | 0.227 | 0.223 | 0.183 | 0.122 |
| 6 | 0.181 | 0.150 | 0.266 | 0.202 | 0.201 |
| 7 | 0.250 | 0.109 | 0.112 | 0.257 | 0.271 |
| 8 | 0.111 | 0.268 | 0.265 | 0.178 | 0.178 |
| 9 | 0.210 | 0.117 | 0.266 | 0.221 | 0.186 |
| 10 | 0.277 | 0.149 | 0.113 | 0.186 | 0.276 |

(b)

**Fig. 3.** Results of the example in Fig. 2. (a) Rating based topic distillation of all users got in Section 3.1. (b) Tag based topic distillation of all users got in Section 3.2.

In this process, rating based topic distillation can be found. We apply this process to the example in Fig. 2 to show the result. In this example, the number of topics is set 5, and then we can get users preference distributions over the topics based rating records, which is shown in 3 (a).

### 3.2. Tag based topic distillation

We set the tags that summed up based on the users' rating records as the users' tags. Then we get the user-tag and item-tag pairs those have a similar format with document-word pair. Here, we use the example in Fig. 2 to show how to get tags of users. The user 1 has rated items 4, 5, 7, 8 and 10 in Fig. 2. Item 4 is with tags Comedy and Drama, and user 1 rated item 4 with rating 4. Then, tags of user 1 based on item 4 are Comedy, Drama, Comedy, Drama, Comedy, Drama, Comedy and Drama. Similarly, tags of user 1 based on item 5 are Comedy and Comedy. In this way, we sum all the tags of users on all the items and get a user-tag matrix. So we can apply LDA model to user-tag and item-tag pairs as it is used in text corpora. Based on user-tag matrix, we can get the result in Fig. 3 (b). After mining the interests of both users and items based on topic distribution, personalized preference can be computed by a matching score. The similarity in terms of user interest topic distribution $\vec{\theta_m^T}$ and item topic distribution $\vec{\pi_n^T}$ is set as the matching score between user $u_m$ and item $i_n$. Inspired by Liu and Xiong [14], the symmetric Jensen–Shannon divergence between user $u_m$ and item $i_n$ is used:

$$D_{JS}(u_m, i_n) = \frac{1}{2}D(\vec{\theta_m^T}||\vec{M}) + \frac{1}{2}D(\vec{\pi_n^T}||\vec{M}) \tag{2}$$

where $M = \frac{1}{2}(\vec{\theta_m^T} + \vec{\pi_n^T})$ and $D(\cdot\,||\,\cdot)$ is the Kullback–Leibler distance. The matching score is defined as $S(u_m, i_n) = 1 - D_{JS}(u_m, i_n)$.

### 3.3. Memetic algorithm based clustering

Clustering method is used to partition all users into specific number of parts that users have similar preferences within each part. Memetic algorithm consists of some important components: population initialization, genetic operators and the local search operator. In our algorithm, different parts of memetic algorithm should be redesigned in order to make the algorithm applicable for recommendation. In this section, more detailed descriptions about fitness function, initialization procedure and genetic operators in the memetic algorithm based clustering (or MAC) will be given.

#### 3.3.1. Fitness function

In order to take preference locality and users' interests calculated by rating records and tag information into consideration simultaneously for users clustering, a novel distance measure is pro-

posed in this paper. The Euclidean geographic distance between users first proposed in [15] is adopted. The Euclidean geographic distance $dist(i, j)$ between two users $u_i$ and $u_j$ is shown as follow:

$$dist(i, j) = \sqrt{(lon(i) - lon(j))^2 + (lat(i) - lat(j))^2} \times c \tag{3}$$

where $lon(i) \in [-90, 90]$ represents the longitude in location of $u_i$ and $lat(i) \in [-90, 90]$ indicates the latitude in location respectively. Parameter $c$ is a constant converting the unit of degree to meter. In this case, $c = 111261$.

To reweigh the different neighbors inside a neighborhood, the local similarity $LS(i, j)$ between $u_i$ and $u_j$ is calculated as follow:

$$LS(i, j) = 1 - \frac{1}{1 + exp^{-dist(i,j)}} = \frac{1}{1 + exp^{dist(i,j)}} \tag{4}$$

Then, we combine users' preference similarity and users' local similarity. The distance measure is as follow:

$$DM(i, j) = sim(i, j)LS(i, j) \tag{5}$$

where $sim(i, j) = norm(D(\vec{\theta_i^R}||\vec{\theta_j^R})) + norm(S(\vec{\theta_i^T}||\vec{\theta_j^T}))$ in MLTRS, $norm()$ is the normalization function, $D(\cdot \,||\, \cdot)$ is the Kullback–Leibler distance.

Since we get the distance measure in clustering, the fitness function is defined as follow:

$$f = \sum_{c=1}^{K} \frac{\sum_{(i,j)\in Cc} DM(i, j)}{|Cc|} \tag{6}$$

where $K$ is the number of clusters, $Cc$ is the $c$th cluster, $DM(i, j)$ is the distance measure in clustering if user $i$ and user $j$ and $|Cc|$ is the number of users in cluster $Cc$.

### 3.3.2. Problem encoding

The proposed MAC is inspired by the classical grouping encoding initially proposed by Falkenauer [11]. The encoding consists of two parts to represent each individual: $c = [l|g]$ in the algorithm. The first half is the element section, whereas the cluster section is another half. Following the notation, the individual in a solution for a clustering problem with $N$ elements and $K$ clusters will have the following format [16]:

$$l_1, l_2, \ldots, l_N|g_1, g_2, \ldots, g_K$$

where $l_j$ represents the cluster to which $j$th subscript is assigned. In a formal way:

$$l_j = g_i \Leftrightarrow x_j \in C_i$$

The following individual is set as a simple example to fully understand the MAC encoding:

1 3 2 1 4 1 1 2 3 2 1 3 4 2 1 | 1 2 3 4

This individual represents a solution with 4 clusters, and the following shows the clustering: $\{x_1, x_4, x_6, x_7, x_{11}, x_{15}\}$, $\{x_3, x_8, x_{10}, x_{14}\}$, $\{x_2, x_9, x_{12}\}$ and $\{x_5, x_{13}\}$.

### 3.3.3. Genetic operators

*Selection operator.* In this paper, a rank-based wheel selection mechanism similar to the one described in [12] is adopted. First, all the individuals are sorted based on their fitness values in a list. The rank is defined as the position in the individual, and denote $R_i$, $i = \{1, 2, \ldots, N\}$, with $N$ individuals in the population. A ranking rule is proposed that the best individual $x$ is assigned $R_x = N$, the second best $y$, $R_y = N - 1$, and so on. Then the fitness value of each individual is defined as follows:

$$fv = \frac{2 \cdot R_i}{N \cdot (N + 1)} \tag{7}$$

Note that the fitness values depend on the position of individual in the ranking list, normalized between 0 and 1. It is necessary to emphasize that this kind of mechanism is static, because that probabilities of survival only depend on the position of the individual in the ranking list but not the generation. As a simple example, consider a population with 5 individuals, in which individual 1 is the one with best quality ($R_1 = 5$), individual 2 the second best ($R_2 = 4$), and so on. In this case, the fitness values of all the individuals are $\{0.33, 0.26, 0.2, 0.13, 0.06\}$, and the corresponding intervals for the wheel are $\{0 - 0.33,\ 0.34 - 0.6,\ 0.61 - 0.8,\ 0.81 - 0.93,\ 0.94 - 1\}$.

*Crossover operator.* The crossover operator in this paper is designed for the clustering problem. The process follows a two parents one offspring schema, with the following steps in Algorithm 1.

**Algorithm 1** Crossover operator.

**Input:**
Solutions: $X_C$;
Two parent individual solutions: $x_a$ and $x_b$;
Crossover probability: $p_c$;
Solution size: $N_c$;
Iteration number: $r$;
The number of clusters: $K$;
**Output:**
Offspring solution: $X_O$;
**for** each $r \in [1, N_c/2]$ **do**
Randomly select two individuals $x_a$ and $x_b$ ($a \neq b$) from $X_C$ and randomly select a crossing point in the cluster section;
Calculate the fitness values of $x_a$ and $x_b$: $f(x_a)$ and $f(x_b)$;
Suppose $f(x_a) > f(x_b)$:
Insert the elements of $x_a$ into the offspring $x_o$;
Insert the elements of $x_b$ into the offspring $x_o$;
Randomly complete the elements not yet assigned in element section;
Remove empty clusters, if any;
Modify the labels of the current groups in $x_o$ by numerating them from 1 to $K$;
**end for**

Fig. 4 shows a simple example of the crossover operator implemented in this paper. The probability of crossover should be higher in the first stage and moderate in the last stage of the algorithm in order to sufficiently explore the search space and avoid destroying the good solutions we have got. Thus, we define an adaptive crossover probability in the following way:

$$P_c(j) = P_{ci} + \frac{j}{TG}(P_{ci} - P_{cf}) \tag{8}$$

where $P_c(j)$ is the crossover probability used in a given generation $j$, $TG$ stands for the total number of generations of the algorithm, and $P_{ci}$ and $P_{cf}$ are the initial and final values of probability considered, respectively.

*Mutation operator.* In this scenario, a two-step mutation operator is adopted for this clustering problem:

Mutation by cluster splitting. One element is selected and assigned to a new cluster. Note that all of clusters except the cluster consists of the selected element will keep its label, whereas the newly generated cluster will be assigned a new label $(K + 1)$. In other word, the number of clusters is $(K + 1)$ after cluster splitting. The selection probability of the initial cluster to be split is based on the clusters size, larger clusters with more probability to be split. As an example, an illustration of this operator is given in Fig. 5. In Fig. 5, the 10th element is performed mutation by cluster splitting, and then the 10th user is assigned to the new cluster with label 6. The labels of other clusters keep unchanged.

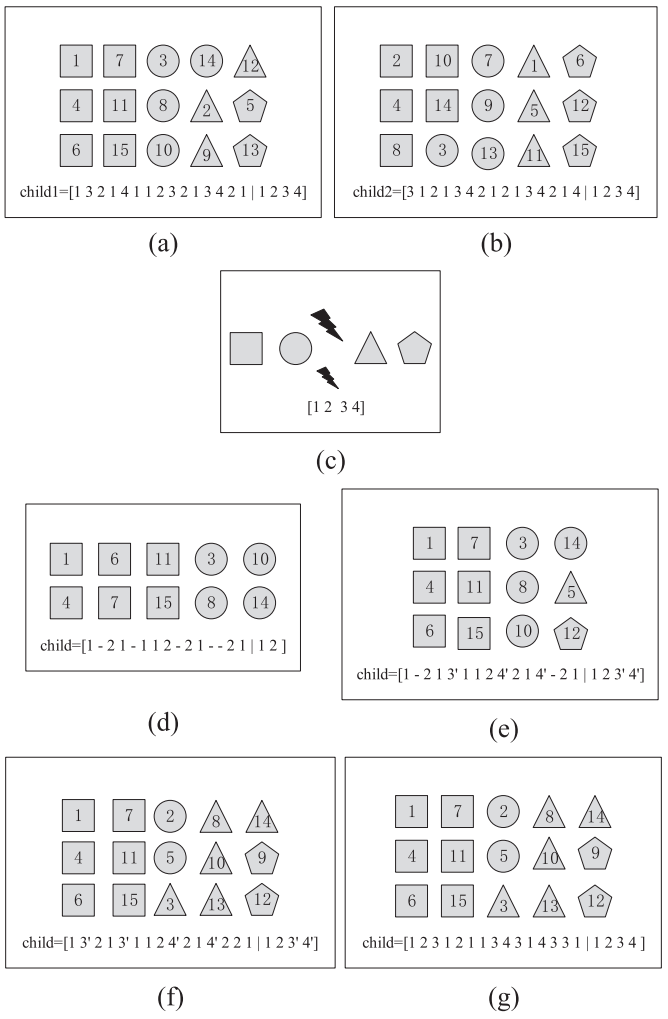



**Fig. 4.** An illustration of the crossover operation. (a) A parent, randomly selected. (b) another parent. (c) Randomly select a crossing point in the cluster section. (d) We assume that the fitness of *parent*1 is higher, so that elements of *parent*1 are inserted in advance. (e) Elements of *parent*2 are inserted. (f) Randomly complete the elements not yet assigned. (g) The produced child.

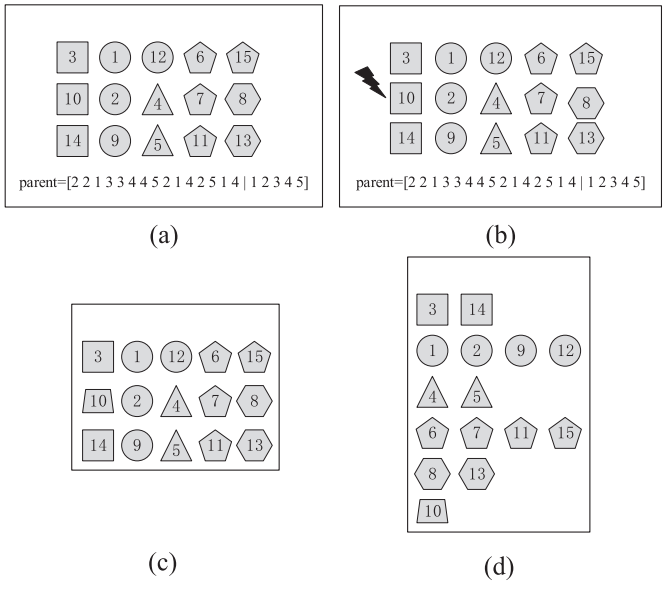



**Fig. 5.** Illustrations of mutation by cluster splitting. (a) The parent, randomly selected. (b) An element is selected randomly. (c) The element section. (d) The cluster section.

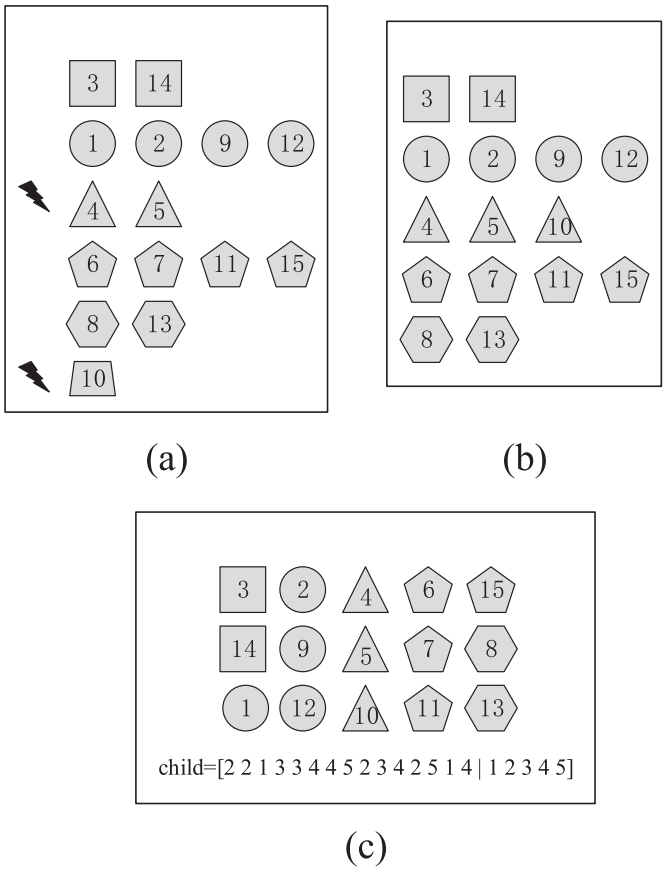



**Fig. 6.** Illustration of mutation by cluster merging. (a) Two clusters are randomly selected. (b) The selected clusters are merged into one. (c) The produced child individual.

Mutation by clusters merging. Different from mutation by clusters splitting, two clusters are randomly selected in this process. The probabilities of choosing the clusters depend on their sizes, but larger clusters are selected with less probability. In this case in Fig. 6, let us suppose that the selected clusters are clusters 3 and 6, and then clusters 3 and 6 are merged into one cluster.

Similarly to the crossover case, we apply the mutation operators described above with an adaptive probability. In this case, in order to avoid converging in local minimum in the last stages of the evolution, the mutation probability is set self-adaptively smaller in the early stages of the algorithm and larger in the last ones.

$$P_m(j) = P_{mi} + \frac{j}{TG}(P_{mf} - P_{mi}) \tag{9}$$

where $P_m(j)$ is the mutation probability used in a given generation $j$, $TG$ stands for the total number of generations of the algorithm, and $P_{mi}$ and $P_{mf}$ are the initial and final values of probability considered, respectively.

### 3.3.4. Local search

In this clustering case, local search works over the cluster section of all individuals. Before implementing local search, we create a mapping table that each user is mapped to the nearest neighbor. The process follows a one parent one offspring schema, as shown in Algorithm 2. Compare the fitness of parent and offspring individuals. If the fitness of offspring individual is higher, the parent individual will be deleted and the offspring individual is inserted into the population, otherwise, nothing will be done.

As shown in Algorithm 2, when there is a change in the local search, it is needed to recalculate fitness $f(x_r)$ and $f(x_r^c)$. The computational complexity of this process is large. Inspired by Ma et al.

**Algorithm 2** Local Search Operator.

**Input:**
Solutions: $X_C$;
Solution size: $N_c$;
Iteration number: $r$;
Individual solutions: $x_r$;
**Output:**
Offspring solution: $x_r^o$;
**for** each $r \in [1, N_c]$ **do**
Calculate the fitness of $x_r$: $f(x_r)$;
Select an element randomly in $x_r$;
Insert the element into the cluster that its neighbor belongs to according to the mapping table, producing $x_r^c$;
Calculate the fitness of the children $x_r^c$: $f(x_r^c)$;
Compare $f(x_r)$ and $f(x_r^c)$. If $f(x_r) < f(x_r^c)$, $x_r^o = x_r^c$, otherwise, $x_r^o = x_r$.
**end for**

[9] to decrease the computational complexity, we adopt a simple way to calculate the difference $\triangle f$ between $f(x_r)$ and $f(x_r^c)$ obtained by moving a user $u_m$ to the cluster $c_n$ that its neighbor belongs to according to the mapping table. The $\triangle f$ can be easily computed by:

$$\begin{aligned}\triangle f &= \left[\sum_{c=1}^{K} \frac{\sum_{(i,j)\in C_1 c} DM(i,j)}{|C_1 c|}\right] - \left[\sum_{c=1}^{K} \frac{\sum_{(i,j)\in C_2 c} DM(i,j)}{|C_2 c|}\right] \\ &= \left[\frac{\sum_{(i,j)\in C_1 c_m} DM(i,j)}{|C_1 c_m|} - \frac{\sum_{(i,j)\in C_2 c_m} DM(i,j)}{|C_2 c_m|}\right] - \\ &\left[\frac{\sum_{(i,j)\in C_1 c_n} DM(i,j)}{|C_1 c_n|} - \frac{\sum_{(i,j)\in C_2 c_n} DM(i,j)}{|C_2 c_n|}\right] \\ &= \frac{\sum_{(i,j)\in C_1 c_m} DM(i,j) - |C_1 c_m| \sum_{j=1}^{C_1 c_m} DM(|C_1 c_m|+1, j)}{|C_1 c_m|(|C_1 c_m|+1)} + \\ &\frac{|C_2 c_n| \sum_{j=1}^{C_2 c_n} DM(|C_2 c_n|+1, j) - \sum_{(i,j)\in C_2 c_n} DM(i,j)}{|C_2 c_n|(|C_2 c_n|+1)} \\ &= \frac{d_{C_1 c_m} - s_{C_1 c_m}}{A} + \frac{s_{C_2 c_n} - d_{C_2 c_n}}{B}\end{aligned}$$

where $C_1$ is the clustering result of the parent, $C_2$ is the clustering result with a slight modification of parent, $c_m$ is $m$th cluster of the specific clustering and $c_n$ is $n$th by the same token, $i$ and $j$ denote the sequence number of users in the specific cluster. From the equation, $d_{C_1 c_m}$ and $d_{C_2 c_n}$ are calculated only one time and can be reused in any calculations in the later. $s_{C_1 c_m}$ and $s_{C_2 c_n}$ are the only parts we need to calculate in every local search. Furthermore, all pair-wise distance measures of all users are calculated and stored in advance.

#### 3.3.5. Replacement and elitism

In the proposed MAC, the population at a given generation $j+1$ is obtained by replacement of the population at generation $j$ after the implement of selection, crossover, mutation and local search operators. An elitist schema is also applied, the best individual in generation $j$ is automatically remained and passed onto the population of generation $j+1$, ensuring that the best solution got from the beginning of the algorithm so far in the evolution is always kept.

### 3.4. Prediction and recommendation

After clustering, we get $K$ clusters composed of users. Different recommendation strategies are applied for normal users and cold-start users, respectively.

*Normal users.* For normal users, rating based and tag based LDA are implemented in all the users within a specific cluster. In order to describe to what extent user $u_m$ prefers item $i_n$. A matching score is designed with the following definition:

$$R_{predict}(u_m, i_n) = norm(\vec{\theta_m^k} \cdot \vec{\phi_n^k}) + norm(S(\vec{\theta_m^t} || \vec{\theta_n^t})) \quad (10)$$

Then items that have not been rated yet by the querying user will be sorted according to the matching scores and then a $top-N$ list is generated. The $top-N$ list is the recommendation list for the querying user.

*Cold-start users.* For cold-start users, every cluster is viewed as a whole, similar to a user. Rating based and tag based LDA are implemented on all these clusters. The matching score is as follows:

$$R_{predict}(c_h, i_n) = norm(\vec{\theta_h^k} \cdot \vec{\phi_n^k}) + norm(S(\vec{\theta_h^t} || \vec{\theta_n^t})) \quad (11)$$

It describes degree of preference of a cluster $c_h$ for item $i_n$. Then all the items will be sorted according to the matching scores and we get the $top-N$ recommendation list for all the cold-start users in this cluster. It is worth noting that the candidate items for cold-start users are all the items in a cluster while the candidate items for normal users are the items in a cluster that has not been rated yet by the querying user.

## 4. Experimental results

In this section, settings of experiments are introduced firstly, including the datasets, comparison algorithms and the evaluation metrics. Then experimental results are reported.

### 4.1. Experimental setup

#### 4.1.1. Datasets

We conduct our experiments on three real datasets: Movielens,[1] Foursquare,[2] and DoubanEvent.[3] All the three datasets are publicly available.

- Movielens. The Movielens 1M dataset used in experiments is crawled from the Movielens developed by the University of Minnesota, which is the real movie rating data. This dataset consists of 1M ratings from 3900 users on 6040 movies. There are also some other additional information, such as the zip code of each user and tags for every movie.
- Foursquare. Foursquare is a location-based social networking website where users can share their locations by checking-in. The Foursquare dataset used in our experiments is a publicly available check-in dataset, which consists of a total 483814 check-ins from 4392 users on 36963 venues. Each check-in is stored as a tuple (user-id, time, venue-id, venue-name, venue-location, venue-category).
- DoubanEvent. DoubanEvent is the largest event-based social network application in China. On DoubanEvent, a social event is specified by when, where, and what. Then, the created event is declared publicly in the application. Other users interesting in this event may join the event by check-in online. This dataset consists of 100,000 users, 300,000 events and 3,500,000 check-ins. All users provide their home location information across 533 cities in China in this dataset.

[1] http://grouplens.org/datasets/movielens/.
[2] www.foursquare.org.
[3] www.datatang.com/Member/67341.

#### 4.1.2. Comparison approaches

We compare the performance of MLTRS and MLRS with four state-of-the-art recommendation methods, LARS [2], TL-PMF [14], ULA-LDA [1], LDA [6], K-means and MRS [17].

- LARS. LARS is a location-aware recommendation algorithm that takes location into consideration to produce recommendations. It exploits user locations to partition all the users into a number of clusters with different sizes, and generates recommendations within all the regions. The recommendation to the querying user is produced only with ratings provided by users who are located at the same region with the querying user. LARS implements a travel penalty method to exploit item locations so that the querying user will be given items close in travel distance as recommendation candidates. For cold-start users, LARS recommends the most popular items within the region the current user belongs to.
- TL-PMF. TL-PMF, whose full title is a topic and location-aware probabilistic matrix factorization method, is based on the learned user and topic distribution, and simultaneously incorporating location information. A unique perspective of this proposed method is to consider both the extent to which a user interest matches the item in terms of topic distribution and the word-of-mouth opinions of the item.
- ULA-LDA. ULA-LDA assumes that the behavior of a user is determined not only by individual intrinsic interest, but also local preferences, which are the general public from the same home location with the user. The model mainly discovers: (1) users personal interest distribution over latent geo-topics; (2) the local preference distribution over latent topics; and (3) an item generative distribution for each topic.
- LDA. Following the previous works of Krestel and Fankhauser [6], a LDA-based method is implemented in recommender system, similar to the way it is used in the semantic analysis. Each user is viewed as a document, and the items rated by the user are viewed as the words appearing in the document. It should be noted that this method does not exploit either user geographical information or item tag information.
- MLRS. In order to compare the effectiveness of MLTRS with other methods, especially the location-based ones, we propose a MLRS, which has a similar framework with MLTRS but does not take tag information into consideration compared to MLTRS. The definition of similarity in MLRS is $sim(i, j) = D(\theta_i^{\bar{k}} || \theta_j^{\bar{k}})$.
- K-means. In our algorithm, many elements are considered. To show the effectiveness of memetic algorithm based clustering, we adopt k-means to partition users into different clusters. In this algorithm, the objective function is also the one proposed in this paper.
- MRS. MRS is also a memetic based clustering for recommendation. This algorithm is adopted to show that different elements or the proposed objective function is effective for recommendation.

#### 4.1.3. Evaluation metrics

The following evaluation method is designed to make an objective and overall evaluation of the effectiveness of the proposed MLTRS. For all rating records, we randomly mark off 20% as test set and the rest 80% as train set to evaluate the effectiveness of all the recommendation methods including MLTRS, MLRS and other comparison methods. That is, the user rating records $S$ are split into the training dataset $S_{training}$ and the testset $S_{test}$. The aim of the evaluation method is to evaluate how accurately items in $S_{test}$ are recommended by the recommendation algorithms. For each user, a $top - N$ list is generated. For this $top - N$ recommendation task, we follow measurement *Accuracy@N*. For each test case in $S_{test}$:

(1) Ranking scores of items unrated by $u$.

**Table 1**
The experimental parameters used in memetic algorithm.

| *Symbol* | *Description* | *Value* |
|---|---|---|
| *pop_size* | The size of population | 100 |
| *G* | The evolutional generation | 200 |
| $p_c$ | Crossover probability | 0.9 |
| $p_m$ | Mutation probability | 0.1 |

(2) All these items are sorted according to their ranking scores.
(3) A $top - N$ recommendation list is generated by picking the top $N$ ranked items for each user. If the item recommended for the current user is in the rating records in test set, we have a hit. Otherwise, we have a miss.

The *Accuracy@N* is adopted as the measurement to show the effectiveness of recommendation algorithms. For each user in test set, The *hit@N* is set as 1 if the item $i$ in test set appears in the $top - N$ recommendation list, or as 0 if otherwise. Then the *Accuracy@N* is defined as the average *hit@N* over all test cases, which is computed as follows:

$$Accuracy@N = \frac{\#hit@N}{|S_{test}|} \tag{12}$$

where *#hit@N* denotes the sum number of hits for each user in the test set, and $|S_{test}|$ is the number of test cases in the test set $S_{test}$.

Besides *Accuracy@N*, mean average precision (MAP) is also adopted to test the performance of the proposed algorithm for recommendation. MAP is computed as follows:

$$MAP = \frac{AP}{|U|} \tag{13}$$

where *AP* is average precision of users, $|U|$ is the number of users and the precision is computed by *Accuracy@N*.

#### 4.1.4. Parameters

In this section, we show the parameters used in the experiments.

*LDA hyperparameters.* All the parameters used in LDA model were analyzed in detail in [18]. Good model quality has been reported for $\alpha = 1$ and $\beta = 0.01$, which $\alpha$ is hyperparameter on the mixture proportions and $\beta$ is hyperparameter on the mixture components.

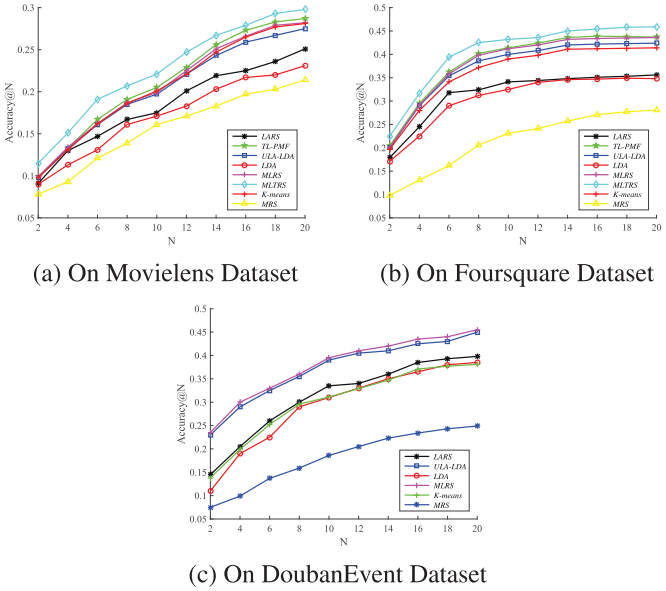
*Memetic algorithm hyperparameters.* The parameters used in memetic algorithm of MLTRS and MLRS are listed in Table 1.

### 4.2. Experimental results

In this section, we first report the optimal performance with well-tuned parameters and then study the impacts of parameters on the effectiveness of the algorithm. It should be noted that not all algorithms are applied on all datasets. We do not conduct experiments of MLTRS and TL-PMF on DoubanEvent dataset because there are no tags on this dataset. But MLRS is based on MA and has the same framework with MLTRS so that the performance of MLRS can reflect that of MLTRS. The Movielens dataset is not tested for cold-start problem, as few users complying with the strict definition of cold-start users.

#### 4.2.1. Effectiveness of $top - N$ recommendation

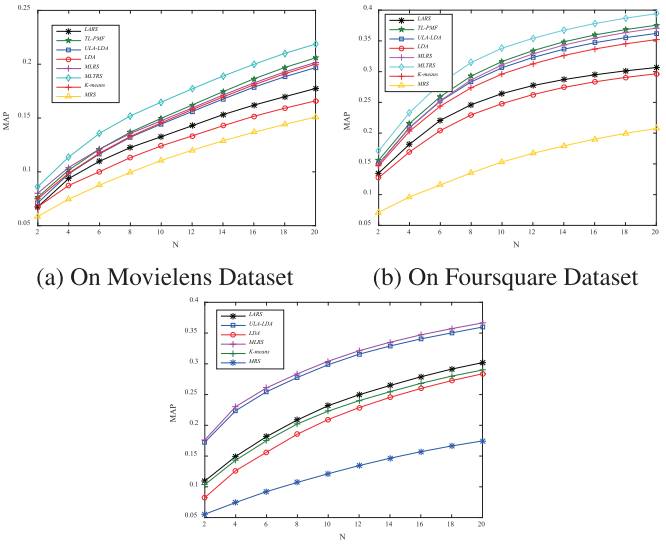
Fig. 7(a) reports the accuracy of the recommendation algorithms on the Movielens dataset. We show only the performance where $N$ is in the range of $\{2, 4, 6, 8, 10, 12, 14, 16, 18, 20\}$, because a larger value of $N$ is usually ignored for a typical $top - N$ recommendation task. We compare MLRS, MLTRS with LARS, TL-PMF,




(a) On Movielens Dataset (b) On Foursquare Dataset

(c) On DoubanEvent Dataset

**Fig. 7.** The accuracy of $Top-N$ recommendation.




(a) On Movielens Dataset (b) On Foursquare Dataset

(c) On DoubanEvent Dataset

**Fig. 8.** MAP of $Top-N$ recommendation.

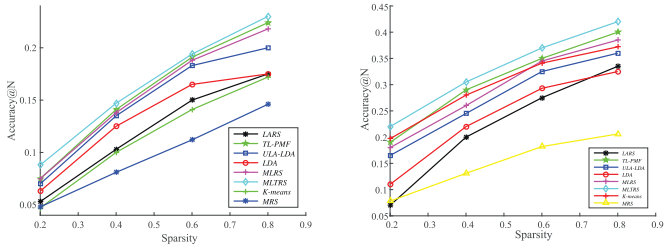
ULA-LDA, LDA, K-means and MRS. It is apparent that the algorithms have significant performance disparity in terms of $top-N$ accuracy. The accuracy value of MLTRS, for instance, is about 0.211 when $N = 10$, and 0.295 when $N = 20$. From the result, we have several observations: 1) MLTRS, TL-PMF, MLRS, ULA-LDA and LARS perform better than LDA, showing the advantage of exploiting user home location information; 2) MLTRS outperforms MLRS. TL-PMF outperforms LDA and MLRS because MLTRS and TL-PMF exploit the tag information. 3) MLRS and ULA-LDA outperform LARS because MLRS and ULA-LDA is latent class model-based methods while LARS is a memory-based CF. Recently, literature [19] has demonstrated the superiority of model-based approaches to memory-based methods as well. 4) MLRS performing better than ULA-LDA shows the effectiveness of memetic based clustering method. Moreover, MLRS and ULA-LDA adopt clustering, which is very helpful to overcome the data sparsity when users have few ratings. 5) K-means is not very good, which is because K-means is not a good clustering algorithm. 6) MRS is with a bad result, because it does not utilize enough elements for recommendation and the Movielens dataset used in this experiments is not very sparse. The results of all algorithms on the Movielens and the Foursquare datasets are shown in Fig. 7 (a) and (b), respectively. From the figures, we can see that the trend of comparison results are similar to that represented in Fig. 7 (c). The main difference is that all algorithms achieve higher accuracy on the Foursquare and DoubanEvent datasets. This is probably because users' offline activities (e.g., attending events or visiting venues) are limited by traveling distance while users online behaviors (e.g., watching videos or movies) are not affected distinctly by the location factor. Thus, for the Movielens dataset, it is likely that there are more available candidate items for the test users and that these potentially relevant items will have prior rankings (i.e., larger ranking scores), which degrades the *Accuracy@N* performance. Specifically, LARS and LDA fail to recommend items that do not have any ratings or only have few ratings, while MLRS and ULA-LDA can still work well in these situations. This is because MLRS and ULA-LDA captures category locality and geographical clustering. Thus, the latent topics discovered by MLRS and ULA-LDA are not only semantic-coherent, but also geographically proximate. Moreover, MLRS only performs slightly better than ULA-LDA because the DoubanEvent dataset takes cities as for a category so that the clustering method in ULA-LDA can partition the users according to the cities similar to the memetic based clustering method. And MLRS and ULA-LDA both consider the user home location information so that those two methods have good performance in recommending items to users with few ratings.

Fig. 8 reports the MAP of the recommendation algorithms on the different datasets. It is shown that MAP of algorithms on different datasets are similar with Accuracy in Fig. 7. Then, it is concluded that our algorithms are with good performance in recommending accurate items.

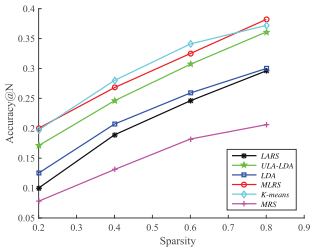
Overall, from Figs. 7 and 8, it is observed that: (1) the proposed MLTRS consistently outperforms the competitor approaches, TL-PMF, LARS, ULA-LDA and LDA, on all datasets. MLRS is only worse than TL-PMF, bacause in MLRS, tags are not considered. (2) in the comparison between LARS and LDA, LARS consistently performs better than LDA on the three data sets since LARS exploits the location information. (3) MLTRS consistently outperforms MLRS and TL-PMF consistently outperforms LDA because exploiting the tag information can improve the recommendation results significantly. The above experimental results demonstrate that exploiting the location information and tag information can significantly improve recommendation accuracy.

#### 4.2.2. Effectiveness of sparsity

The following experiments are designed to study the effectiveness of recommendation algorithms dealing with the sparsity problem. We test the performance of our algorithm and all comparison approaches by varying the sparsity of all datasets (e.g., sparsity = 0.2, 0.4, 0.6, 0.8.) in Fig. 9 (a)–(c), in which $N$ is assigned 10. From the figures, it is observed that: 1) the Accuracy values of all approaches increase with the increase of sparsity on all datasets; 2) MLTRS, MLRS, ULA-LDA perform better than LDA under any sparsity and our MLTRS consistently performs best with any sparsity on the and the Foursquare dataset and DoubanEvent dataset; 3) the Accuracy values of MLTRS, MLRS and ULA-LDA decrease more slowly than LDA and TL-PMF with the decrease of sparsity on all datasets which demonstrates that clustering method is an effective way of alleviating data sparsity. LARS decreases rapidly because of the CF-based method suffers from the severe data sparsity. 4) K-means is with good results on Foursquare and Douban dataset, while it is with a bad result on Movielens. 5) MRS

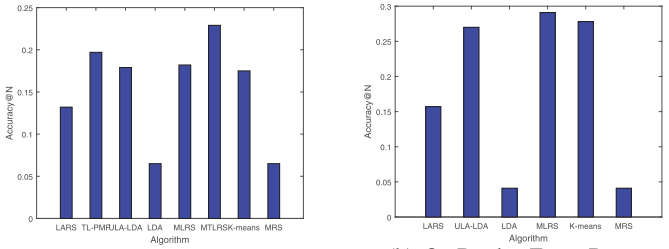



(a) On Movielens Dataset (b) On Foursquare Dataset




(c) On DoubanEvent Dataset

**Fig. 9.** Accuracy of alleviating sparsity problem.




(a) On Foursquare Dataset (b) On DoubanEvent Dataset

**Fig. 10.** *Top* − *N* accuracy for cold-start users.

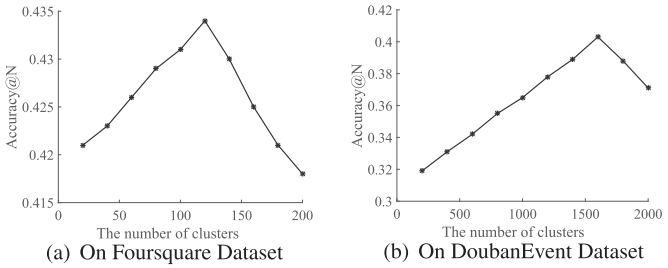
is also with the best results in different datasets, because it takes more elements into consideration.

From Fig. 9, it is concluded that our algorithms are with good results in alleviating data sparsity, because they takes more elements and an effective clustering algorithm is used.

#### 4.2.3. Test for cold-start problem

This section studies the effectiveness of different recommendation algorithms for the cold-start problem and the effect of the number of clusters on the algorithm performance. Our strategy alleviating cold-start problem is to recommend *Top* − *N* items within the cluster $C_k$ the user $u_i$ belongs to by applying rating and tag based LDA in all clusters, as described in Section 3. Therefore, in these experiments, we determine the length of recommendation $N = 10$.

*Results of cold-start.* The same training set is used as before, and those users who have less than 10 ratings in the training set are considered as cold-start users. Fig. 10 (a) and (b) show the accuracy of recommendation algorithms for cold-start users on Foursquare and DoubanEvent datasets, respectively. By comparison among Fig. 7, Fig. 8 and Fig. 10, it is observed that, although the recommendation accuracy of all algorithms decreases to certain degrees, for cold-start users, the proposed MLTRS still performs the best in both datasets. Note that the cold-start users preferences are hard to be exploited as items rated by cold-start users are few. MLTRS overcomes the lack of users ratings by exploiting graphical information and tag information because these two kinds of features may supply many valuable references and clues. So MLTRS




(a) On Foursquare Dataset (b) On DoubanEvent Dataset

**Fig. 11.** Impact of the number of clusters on *Top* − *N* recommendation accuracy.

performs significantly better than LDA, TL-PMF, showing the advantages of exploiting user home location. Another observation is that LARS performs poorly in this scenario, while it performs better than LDA. This is because LARS is a CF-based method and suffers from the severe data sparsity, while all other recommendation methods are latent class models that can alleviate the data sparsity problem to some extent.

*Effect of the number of clusters.* Experiments in this section are designed to test the effect of the number of clusters (i.e., K). The performance of MLTRS is tested by varying the number of clusters (e.g., K = 20, 40, 60, 80, 100, 120, 140, 160, 180, 200.) on Foursquare in Fig. 11 (a). The performance of MLRS on the DoubanEvent dataset is shown in Fig. 11 (b) with different number of clusters (e.g., K = 200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000.). From the figures, it is observed that: 1) the Accuracy values of MLTRS and MLRS increase with the increase of the number of clusters until reaching a peak and decrease with the increase of the number of clusters; 2) MLTRS consistently performs better than MLRS under any number of clusters since MLTRS exploits the tag information. It should be noted that the performance in Fig. 11 (a) and (b) is achieved with $K = 100$ on the Movielens dataset, $K = 120$ on the Foursquare dataset and $K = 1600$ on the DoubanEvent dataset.

## 5. Related works

*Generic recommender system.* Recommender systems are software tools and techniques providing suggestions to a user [20]. From a data mining perspective, these systems take as input a set of users' ratings, and aim to predict the preference scores of uses on items that have not been rated yet [1]. Approaches for this problem are normally divided in two broad categories: content-based and collaborative filtering approaches [19]. The general principle of content-based (or cognitive) approaches is to identify the common characteristics of items that have received a favorable rating from a user *u*, and then recommend the user *u* new items that share these characteristics. Unlike content-based approaches, which use the content of items previously rated by a user *u*, collaborative (or social) filtering approaches [21,22] rely on the ratings of the user *u* as well as ratings of other users in the system. Recently, topic-based approaches, such as MF [23] and LDA, are getting more popular. Probabilistic topic models are based on the idea that documents are mixtures of topics, where a topic is defined as a probability distribution over words.

*Location based recommendation.* Location data is used in recommender systems in different ways, such as user profiles and visited locations. Different methods are adopted to mine user preferences for different types of location data. The most used methods for location based user rating profiles are location based clustering [1,2,7,24]. A location aware probabilistic mixture generative model was proposed in [1]. This mixture model considers users' interest

and the influence of local preferences. The local preference is got by a location based clustering method. LARS, proposed in [2] partitions users into a number of regions through a location based clustering method. Then, it makes recommendations only considering the ratings rated by users in the same cluster with the querying user. Location based *K*-means clustering algorithms were adopted in [7] and [24]. A novel distance metric was designed in [24], taking both rating records and geographical information into consideration, while Euclidean distance of two locations was applied directly in [7].

*Tag based recommendation.* Tag based recommendation [6,25,26] is an important kind of recommender system. In [6], a mixture of simple language models and LDA was employed to estimate the probability of new tags based on the already assigned tags of a resource and a user. The idea of the proposed system in [26] was to perform Latent Semantic Analysis (or LSA) and Probabilistic Latent Semantic Analysis (or pLSA) on the short letters, users writing to the radio program, to search for similar letters. Recommendation was made by this letters. The problem of automatic annotation of metadata records was addressed in [25]. The goal is to build a fast and robust system that annotates a given keyword library. The idea is to annotate a poorly annotated records that it was most similar with.

Based on the previous review, ratings, location and tag are important information to improve recommendation quality, while none of algorithms has taken all of these information into account for recommendation. Then in our algorithm, recommendation is generated by considering ratings, location and tags. LDA is used to mine interests of users based on ratings and tags and generate recommendation based on the clustering results. Memetic based clustering is used to partition users into different clusters by balancing ratings, location and tag. The combination of LDA and memetic algorithm based clustering can alleviate data sparsity, mine users preferences accurately and recommend items to cold start users with better results than other algorithms.

## 6. Conclusion and future research

This paper studies how to leverage rating records, geographical information and textual information to make recommendations. All users are partitioned into several clusters by a MA based clustering method. We mine interests of users based on ratings and tags with LDA. The geographical information is added to the fitness function, together with the rating and tag based distillation of users mined by LDA. Normal users are recommended with items obtained by applying rating and tag based LDA within each cluster and cold-start users are recommended with items obtained by applying rating and tag based LDA of every cluster. There are several advantages of the proposed recommendation method. First, although the tags associated with items are usually incomplete and ambiguous, the proposed MLTRS can learn the users' interests from those insufficient information and have a good recommendation accuracy. Second, we apply the memetic algorithm to this application scenario to make full use of three kind of features while other algorithm have difficulty in alleviating this problem. It provides a way to match the user interests. Compared with other location based recommender system, the proposed algorithm performs excellently in alleviating the cold-start problem. Last but not least, the proposed algorithm is flexible and could be extended to incorporate other types of context aware information to enhance the recommendation performance.

In the future, our research will be extended for exploiting other kind of features, such as friendship relations to generate recommendation.

## Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant no. 61422209), the National Program for Support of Top-notch Young Professionals of China and the National Key Research and Development Program of China (Grant no. 2017YFB0802200).

## References

[1] H. Yin, B. Cui, L. Chen, Z. Hu, C. Zhang, Modeling location-based user rating profiles for personalized recommendation, ACM Trans. Knowl. Discov. Data (TKDD) 9 (3) (2015) 19.
[2] J.J. Levandoski, M. Sarwat, A. Eldawy, M.F. Mokbel, Lars: a location-aware recommender system, in: Data Engineering (ICDE), 2012 IEEE 28th International Conference on, IEEE, 2012, pp. 450–461.
[3] B.-X. Wu, J. Xiao, J.-M. Chen, Friend recommendation by user similarity graph based on interest in social tagging systems, in: Advanced Intelligent Computing Theories and Applications, Springer, 2015, pp. 375–386.
[4] J. Huang, Recommendation with Contextual Information, Ph.D. thesis, Drexel University, 2014.
[5] C. Wang, D.M. Blei, Collaborative topic modeling for recommending scientific articles, in: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2011, pp. 448–456.
[6] R. Krestel, P. Fankhauser, Personalized topic-based tag recommendation, Neurocomputing 76 (1) (2012) 61–70.
[7] P. He, J. Zhu, J. Xu, M.R. Lyu, A hierarchical matrix factorization approach for location-based web service QoS prediction, in: Service Oriented System Engineering (SOSE), 2014 IEEE 8th International Symposium on, IEEE, 2014, pp. 290–295.
[8] Y.-S. Ong, M.H. Lim, X. Chen, Research frontier-memetic computation past, present & future, IEEE Comput. Intell. Mag. 5 (2) (2010) 24.
[9] L. Ma, M. Gong, J. Liu, Q. Cai, L. Jiao, Multi-level learning based memetic algorithm for community detection, Appl. Soft Comput. 19 (2014) 121–133.
[10] X. Chen, Y.-S. Ong, M.-H. Lim, K.C. Tan, A multi-facet survey on memetic computation, IEEE Trans. Evol. Comput. 15 (5) (2011) 591–607.
[11] E. Falkenauer, The grouping genetic algorithms-widening the scope of the gas, Belgian J. Oper. Res. Stat. Comput. Sci. 33 (1) (1992) 2.
[12] T.L. James, E.C. Brown, K.B. Keeling, A hybrid grouping genetic algorithm for the cell formation problem, Comput. Oper. Res. 34 (7) (2007) 2059–2079.
[13] G. Xu, Y. Zhang, X. Yi, Modelling user behaviour for web recommendation using LDA model, in: Web Intelligence and Intelligent Agent Technology, 2008. WI-IAT'08. IEEE/WIC/ACM International Conference on, 3, IEEE, 2008, pp. 529–532.
[14] B. Liu, H. Xiong, Point-of-interest recommendation in location based social networks with topic and location awareness., in: SDM, 13, SIAM, 2013, pp. 396–404.
[15] R. Srikanth, R. George, N. Warsi, D. Prabhu, F.E. Petry, B.P. Buckles, A variable-length genetic algorithm for clustering and classification, Pattern Recognit. Lett. 16 (8) (1995) 789–800.
[16] L. Agustı, S. Salcedo-Sanz, S. Jiménez-Fernández, L. Carro-Calvo, J. Del Ser, J.A. Portilla-Figueras, et al., A new grouping genetic algorithm for clustering problems, Expert Syst. Appl. 39 (10) (2012) 9695–9703.
[17] M. Ukrit, T.U. Nipon, A. Sansanee, Applying memetic algorithm-based clustering to recommender system with high sparsity problem, J. Cent. S. Univ. 21 (9) (2014) 3541–3550.
[18] G. Heinrich, Parameter Estimation for Text Analysis, Tech. Rep, University of Leipzig, 2008.
[19] P.B. Kantor, L. Rokach, F. Ricci, B. Shapira, Recommender Systems Handbook, Springer, 2011.
[20] P. Resnick, H.R. Varian, Recommender systems, Commun. ACM 40 (3) (1997) 56–58.
[21] M. Balabanović, Y. Shoham, Fab: content-based, collaborative recommendation, Commun. ACM 40 (3) (1997) 66–72.
[22] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Item-based collaborative filtering recommendation algorithms, in: Proceedings of the 10th International Conference on World Wide Web, ACM, 2001, pp. 285–295.
[23] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer (8) (2009) 30–37.
[24] J. Yin, W. Lo, S. Deng, Y. Li, Z. Wu, N. Xiong, Colbar: a collaborative location-based regularization framework for QoS prediction, Inf. Sci. 265 (2014) 68–84.
[25] S. Tuarob, L.C. Pouchard, C.L. Giles, Automatic tag recommendation for metadata annotation using probabilistic topic modeling, in: Proceedings of the 13th ACM/IEEE-CS Joint Conference on Digital Libraries, ACM, 2013, pp. 239–248.
[26] Z. Hyung, K. Lee, K. Lee, Music recommendation using text analysis on song requests to radio stations, Expert Syst. Appl. 41 (5) (2014) 2608–2618.