Ant Collaborative Filtering Addressing Sparsity and Temporal Effects

XIAOFENG LIAO 1,2,3 , HU WU 4 , AND YONGJI WANG 4

1School of Management, Nanchang University, Nanchang 330031, China
2Postdoctoral Research Station of Management Science and Engineering, Nanchang University, Nanchang 330031, China
3Informatics Institute, University of Amsterdam, 1098XH Amsterdam, The Netherlands
4State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China

Corresponding author: Xiaofeng Liao (xfliao@ncu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61762062 and Grant 61862042, in part by the Science and Technology Innovation Platform Project of Jiangxi Province under Grant 20181BCD40005, in part by the Major Discipline Academic and Technical Leader Training Plan Project of Jiangxi Province under Grant 20172BCB22030, in part by the Primary Research and Development Plan of Jiangxi Province under Grant 20192BBE50075 and Grant 20181ACE50033, and in part by the Jiangxi Province Natural Science Foundation of China under Grant 20192BAB207019.

ABSTRACT

Though collaborative filtering (CF) is a popular and successful recommendation technique, it still suffers from the data sparsity and users’ evolving taste over time. This paper presents a new collaborative filtering scheme: the Ant Collaborative Filtering. With the mechanism of pheromone transmission between users and items, the proposed method can pinpoint most relative users and items even in the case of the sparsity situation. Also, by virtue of the evaporation of existing pheromone, the proposed method captures the evolution of user preferences over time. Experiments are performed on the standard, public datasets and two real corporate datasets, which cover both explicit and implicit rating data. The results illustrate that the proposed algorithm outperforms current approaches in terms of accuracy and changing data.

INDEX TERMS

Ant collaborative filtering, ant colony optimization, collaborative filtering, recommender systems.

I. INTRODUCTION

Though widely adopted for a recommendation, collaborative filtering still lacks capabilities of handling two fundamental issues: (a) data sparsity and (b) evolving preference.

Data sparsity is one of the difficult problems limiting Collaborative Filtering, makes the calculated similarity among users and items inaccurate, further rendering CF ineffective. Two aspects mainly cause the data sparsity problem. First, a user’s ratings toward items are always small comparing to the enormous number of items exist in a system. Second, ratings toward an item are also small comparing to the enormous amount of users exist. This problem gets even worse when it applies to new users and items, which is referred to as the cold-start problem.

In a real-world recommender system, another practical but often overlooked issue is to consider the evolution of the user interests over time. For example, a customer may like to see recommendations about digital cameras if she/he plans to buy one. However, after purchasing, she/he may be no longer interested in the recommendations on buying a new digital camera. Therefore, the time factor is of vital importance for the success of recommender systems in many applications, such as e-commerce, advertisement, and news services.

Ant colony optimization (ACO) borrows the idea from real ant colonies behaviour. The ants can easily find the best path for foods via using the pheromone as an intermediate communicator. ACO algorithms successfully exploit this characteristic to solve, for example, discrete optimization problems [29], [32].

To recommend using the sparse and evolving preference data, this paper proposes a novel collaborative filtering algorithm, named the Ant Collaborative Filtering (ACF). Similar to other swarm intelligence algorithms, the ACF can handle very sparse (rating) data by virtue of pheromone transmission. Pheromone is a chemical signal that triggers a response in another agent. Pheromone concentration increases the probability that an ant will follow a path. We make an analogy of users to ants that carry specific pheromone, initially. When the user rates a movie or simply reads a piece of news on the Web, our algorithm links the user and the item by the mechanism of pheromone transmission from the user to the
item and vice versa. Therefore, the types of pheromone and the amounts, constitute a clear clue of historical preferences, which turns out to be strong evidence for finding similar users and items.

An essential property of the deposited pheromone in real ant environment is the pheromone evaporation over time. Pheromone evaporation allows ants to adapt to changing environments, also allows ants to avoid being trapped in a suboptimal solution. By virtue of the evaporation of existing pheromone, the proposed method captures the evolution of user preferences over time.

The paper is structured as follows. Section II investigates the existing work. Section III details the ACF algorithm. Section IV represents and analyses the experiment results. Section V concludes the paper and discuss future works.

II. RELATED WORK
As indicated in the title, our proposed method targets two problems in the recommendation, i.e., sparsity and time effect. Therefore, we present related work respectively below from these two aspects.

A. SPARSITY
Various approaches and techniques have been proposed targeting eliminating the sparsity problem. Among them, most achieve good recommendations. However, it is still hard to reach a general solution for tackling with sparsity [13], [20], [33].

The most common technique is applying dimension reduction techniques towards the user-item matrix [10], [64]. The dimension reduction technique tackles the sparsity problem by removing low representative or noise data to condense the matrix.

Other techniques, such as Latent Semantic Indexing (LSI) from information retrieval, are also often adopted [23], [35]. One drawback of these approaches is that potentially useful information might be unavoidably removed during the reduction process.

Other approaches tackling sparsity include utilizing associative retrieval techniques toward the bipartite graph consist of items and users [19], [41], content-boosted CF [9], and using item-based similarity to replace user-based similarity [43].

Instead of using dimension reduction towards the user-item matrix, this work presents an Ant Colony Optimization (ACO) based approach for tackling the sparsity problem. ACO algorithms are proposed after observing the automatic accumulation or communication phenomena common to the ant colony. The ACO has been widely applied in various domains, such as clustering and information retrieval [24], web search [3], [26], [38], [44] and social network [4], [21], [30], [40], [46], [47]. The ACO belongs to a more general group of simulation algorithms named the swarm intelligence algorithms [31].

Among which, the recommender system is one of the many applications [34], [45]. Reference [50] used the particle swarm optimization algorithm to model every user as a particle in a multi-dimensional space with each dimension representing a movie genre. The advantage of such an approach is that every particle has a unique position and velocity. Therefore, the system is dynamic and to a certain extent probabilistic in nature, which is preferable in recommender systems. Reference [2] introduced a notion of dynamic trust among users to help select user neighbour based on that. Reference [6] treated recommendation as a ranking problem, factorize the user-item matrix into feature vectors first, then learn a linear ranking model via optimizing a mean average precision metric. Reference [62] used binary particle swarm optimization to achieve the best performing set satisfying contextual constraints. In [1], the particle swarm optimization algorithm was adopted to estimate the weights of the users and items. The weight space can be quickly searched using this algorithm and lead to a converge at a nearly global optimal value. After that, the optimal value was used to improve the prediction model.

In recent years, deep learning-based recommendation methods have increased exponentially. However, regarding the sparsity problem, the deep neural network still needs to rely on additional information or architecture or implicit data. More specifically, [48] incorporates side information, such as user profiles and item descriptions, to mitigate the sparsity and cold start influence. DeepCoNN [60] adopts two parallel CNNs to alleviate the sparsity problem by exploiting rich semantic representations of review texts with CNNs. Zheng et al. [61] proposed incorporating implicit feedback to overcome the sparsity problem of the rating matrix.

To the best of our knowledge, there are only a few works that directly relate to our model in the CF domain.

B. CONSIDERING TIME EFFECTS
The data in recommender systems keep changing [37]. Not only new users and new items are continuously added to the system, the preferences of existing users and the features of existing items also change over time [36]. To make better recommendations with the temporal diversity [24] into consideration, recommendation algorithms should update the learned model efficiently and appropriately.

Methods used in earlier years include: [65] tackled the time effect by encoding time order information into the data, transforming to a univariate time series problem, and adopting a decision-tree learning process. During computing the weights for different items, [7] adopted the strategy of decreasing the weights of the old data. However, this approach made an assumption that the old data was trivial and only new data was important, which however was not always true. Reference [18] proposed to use the concept of life cycles to formalizes a user’s long-term behaviour and determine the user’s stage, which also used lately in [28]. Taking this method a step further, a session-based temporal graph was proposed in [54]. The advantage of this approach is that it can model the long- and short-term factors at the same time. Reference [22] analyzed the evolution of ratings in the Netflix
movie recommender system and proposed two CF algorithms that considered the time effects, i.e. item-item neighbour method and rating matrix factorization method. Both of the methods were elaborately tailored for the Netflix movie rating data and achieved the best results thus far reported.

Reference [39] introduce a novel time-aware recommendation algorithm that is based on identifying overlapping community structure among users. Their strategy is to detect time evolution by the same time to minimize the sparsity effects, to identify overlapping community structure amongst the users.

Other recent approaches dealing with time effect include Hidden Markov Chain [57], fusion model [52], probabilistic model [27], latent factor [56]. Among these methods, [5]'s work is similar to our idea in that it assumes a hidden network structure exists among the items, and each user tracks a sequence of items in this network. The difference is that in their work, the dependencies between the items are modelled based on statistical diffusion models, and the parameters are obtained through the maximum-likelihood estimation. In contrast, in our proposal, the time effect is modelled via modelling the evaporation of pheromone, which is a more natural design.

We particularly notice a category of work based on deep learning. According to a latest comprehensive survey [59], among the many kinds of neural networks, Recurrent Neural Network (RNN) is suitable for sequential data processing. As such, RNN becomes a natural choice for dealing with the temporal dynamics of interactions. Some examples include [8], [15], [53], [55].

As pointed in [22], the preference evolution is subtle and delicate. For a single user, we can only use a few preference instances inundated by millions of non-relevant data. The methods mentioned above have been designed and tested for particular recommendation scenarios. However, in a more general sense, a robust recommendation algorithm considering time evolution for a broader range of applications including both rating-based and ranking-based recommender systems is still absent and is the main focus of this paper.

By virtue of the evaporation of existing pheromone [49], which naturally reflects the changing preference of existing users and existing items, our method adopting ACO considers the evolution of user preferences over time in a more natural manner.

Along the research line of adopting ACO based methods, we also identified some related works. One closely similar work with ours is [25] which also based on ant colony algorithm. They also take into account the user access time and frequency. In addition, our work not only assumes the user but also the items have the pheromone, which is exchangeable between users and items.

Another system utilizing trust information among ant colony is [11], their STARS system incorporates temporal information by exploiting timestamps of ratings, which uses an user-item rating matrix to store the ratings made on and before a time slot. Instead, our method does not need to maintain such a matrix, which may pose another sparse matrix. In addition, they use time values with day granularity, which is not a problem in our system as the evaporation feature of pheromone mimics the dynamicity.

To summarize, based on the examination of existing work regarding the two aspects: sparsity and temporal effects, we can see a need for a general approach that can resolve these two problems in an integrated manner. Thus we propose the ACO based recommendation system. The ACO algorithm has several enticing properties. The most prominent one is the dynamic and self-organizing nature of the algorithms [12]. Therefore, the problems that are complex and dynamic to be solved using other machine learning methods can directly be optimized for some utility function.

III. ANT COLLABORATIVE FILTERING

In the typical settings of ACO, every ant is identified by some kind of indicators for communication, often referred to as pheromone. The pheromone is used as an indirect communication medium. For example, consider the Traveling Salesman Problem (TSP) [16] that finds the shortest path on a graph. In TSP, each ant walks on the graph and leaves a pheromone on the path. Shorter paths will leave stronger signals. Other ants, when deciding the path to take, tend to choose paths with stronger pheromones with a higher probability, so that shorter paths are found.

Before proceeding to a more detailed description, the notations used are defined in Table 1.

The intuition of our ACF algorithm is that: given a pheromone that represents a user or a group of users, the item may share the user’s pheromone when the item is rated. Meanwhile, an item transfers the pheromone that is already attached to the user. After a while, similar items receive similar patterns of pheromone, and then the users with similar choices become identical with respect to the pattern of the pheromone. The pheromone transmission process is illustrated in Fig. 1. We must note that the ratings are learned one by one in the original order of time. Recommendations can be generated in two strategies: (a) Provided similar users and similar items are identified, we can estimate the current user rating on the items that have not been seen before by employing memory-based CF methods. (b) We can rank the items according to the similarity of the pheromones to the users. The above two strategies correspond to the rating-based and the ranking-based recommendation systems, respectively.

A. TRAINING

First, we initialize the user pheromone by allocating every single user a unique pheromone with a value of 1. The pheromone for each item is empty. After user \( u_i \) provides rating \( r_{ij} \) towards item \( v_j \), the pheromones are exchanged between each other. That is to say, the user updates its pheromone by adding the item’s pheromone times by rating adjustment and a constant \( \gamma \), which is introduced to control the spreading rate. Similarly, the item’s pheromone is updated by adding the user’s pheromone times by rating adjustment.
TABLE 1. Notations used in this work.

| Notation   | Meaning                                                                 |
|------------|-------------------------------------------------------------------------|
| $u_i, v_j$ | A user and an item                                                      |
| $r_{i,j}$  | The rating that user $u_i$ gives to item $v_j$, 0 if unknown           |
| $\bar{r}, \bar{r}_u, \bar{r}_v$ | The average rating for all the ratings, user $u_i$ and item $v_j$ |
| $\mathcal{X}, K$ | Total number of types of pheromone and maximum of types of pheromones attached to a user or an item, $\mathcal{X} \geq K$ |
| $C(u_i)$  | Neighborhood user set for user $u_i$                                    |
| $C(v_j)$  | Neighborhood item set for item $v_j$                                    |
| $Ph(u_i), Ph(v_j)$ | The pheromones attached to user $u_i$ and item $v_j$, they are vectors of pheromones |
| $\gamma, \lambda, \sigma$ | Parameters in our model, controlling the transmission rate, evaporating rate and disappearing rate |
| TopN      | The recommended top $N$ items                                           |
| Hitting   | The subset that user likes in the recommended list                      |
| Likes     | The subset that user likes in the training set                          |
| rank($v_j$) | The final rank position for item $v_j$ in rank-based recommendation    |

FIGURE 1. Illustration showing the pheromone transmission between the users and the items. The user pheromone is contained inside the triangles, and the item pheromone are contained in the rectangles. The upper sub-fig shows the initial state of the training process, where each user contains its unique pheromone, and each item’s pheromone is left empty. The bottom sub-fig shows the pheromone transmission between users and items after several steps. The numbers in the circle indicate the timestamp. The item pheromone and user pheromone are respectively updated using Eq. 1 and Eq. 2.

and a constant $\gamma$. The rating adjustment is the gap between the rating and the average rating. The bigger it is, the stronger is the user’s preference. If the rating is much higher than the average, then the transmission will be a strong positive “plus”. Moreover, if the rating is much less than the average, then the transmission will be a strong negative “minus”, which means that the user and the item are unalike, resulting in the negative values for the pheromones.

As mentioned previously, the user interests may change over time. Older interests fade out, and new interests develop. We capture this evolution by the mechanism of pheromone evaporation. Before the pheromone exchange between the user and the item is executed, the existing pheromones evaporate at a pre-specified rate. For our research, we consider the ratio of the current pheromone amount to the highest pheromone concentration among all the $K$ types existing pheromones as the rate.

For the rating-based recommendation scenario, the complete pheromone update formulae for the item and the user are described in Eq. 1 and Eq. 2.

$$Ph^{(t+1)}_{v_j} = Ph^{(t)}_{v_j} \times \exp \left( \frac{\text{amount}_{v_j,k} + \lambda}{\text{Max}_{k \in K}(\text{amount}_{v_j,k}) + \lambda} - 1 \right) + (r_{i,j} - \bar{r}_u) \times \gamma \times Ph^{(t)}_{u_i}$$

(1)

$$Ph^{(t+1)}_{u_i} = Ph^{(t)}_{u_i} \times \exp \left( \frac{\text{amount}_{u_i,k} + \lambda}{\text{Max}_{k \in K}(\text{amount}_{u_i,k}) + \lambda} - 1 \right) + (r_{i,j} - \bar{r}_v) \times \gamma \times Ph^{(t)}_{v_j}$$

(2)

Similarly, for the 0/1 preference data, we rewrite the pheromone update formula as follows:

$$Ph^{(t+1)}_{v_j} = Ph^{(t)}_{v_j} \times \exp \left( \frac{\text{amount}_{v_j,k} + \lambda}{\text{Max}_{k \in K}(\text{amount}_{v_j,k}) + \lambda} - 1 \right) + \gamma \times Ph^{(t)}_{u_i}$$

(3)

$$Ph^{(t+1)}_{u_i} = Ph^{(t)}_{u_i} \times \exp \left( \frac{\text{amount}_{u_i,k} + \lambda}{\text{Max}_{k \in K}(\text{amount}_{u_i,k}) + \lambda} - 1 \right) + \gamma \times Ph^{(t)}_{v_j}$$

(4)

To keep our model simple and robust against the rating noise, after evaporation and transmission, we delete the pheromones with the value less than a threshold value $\sigma$. This process is referred to as the threshold cut off. In our simulation experiments, $\sigma$ is set to 0.01, and the training algorithm is shown in Alg. 1.

B. RECOMMENDATION

We provide recommendation algorithms for both the explicit rating prediction and implicit relevance ranking tasks.
Algorithm 1 Training Phase of ACF

| Input: Ratings, U, V |
|----------------------|
| Output: Updated user pheromone and item pheromone |

//Initialization
for $u_i \in U$ do
    \[ P_{h_{ui}}^{(0)} = \{u_i : 1.0\}; \]
end
for $v_j \in V$ do
    \[ P_{h_{vj}}^{(0)} = \{\}; \]
end

//Training
for rating $\in$ Ratings do
    $u_i = rating[\text{user}]$;
    $v_j = rating[\text{item}]$;
    value $= rating[\text{value}]$;
    // Update user pheromones
    for Pheromone $\in P_{h_{ui}}^{(t)}$ do
        // Evaporation and Transmission
        Pheromone $= Pheromone \times \exp\left(\frac{Pheromone_{ui,k} + \lambda}{\lambda \cdot \max_{u_i} Ph_{ui,k}} - 1\right) + \gamma \times (value - \overline{r_{ui}}) \times P_{h_{vj,k}}^{(t)}$
        // Cut off
        if $\text{abs}(\text{Pheromone}) < \sigma$ then
            Pheromone $= 0$;
        end
        \[ P_{h_{ui}}^{(t+1)} \leftarrow \text{Pheromone}; \]
    end
    // Update item pheromones
    for Pheromone $\in P_{h_{vj}}^{(t)}$ do
        // Evaporation and Transmission
        Pheromone $= Pheromone \times \exp\left(\frac{Pheromone_{vj,k} + \lambda}{\lambda \cdot \max_{v_j} Ph_{vj,k}} - 1\right) + \gamma \times (value - \overline{r_{v_j}}) \times P_{h_{ui,k}}^{(t)}$
        // Cut off
        if $\text{abs}(\text{Pheromone}) < \sigma$ then
            Pheromone $= 0$;
        end
        \[ P_{h_{vj}}^{(t+1)} \leftarrow \text{Pheromone}; \]
    end
end

Moreover, we can also calculate the following three types of similarities through pheromone comparison:

- **User-User similarity, $s(u_i, u_k)$**: To what extent are any two given users $u_i$ and $u_k$ alike? (This is computed by comparing the user pheromones.)
- **Item-Item similarity, $s(v_j, v_l)$**: To what extent are any two given items $v_j$ and $v_l$ alike? (This is computed by comparing the item pheromones.)
- **User-Item similarity, $s(u_i, v_j)$**: To what extent is a given user $u_i$ and a given item $v_j$ alike? (This is computed by comparing the corresponding user and item pheromones.)

For rating prediction, we employ memory-based methods. That is to say that the user $u_i$’s rating toward the item $v_j$, is predicted based upon the ratings, which are given by users similar with $u_i$ toward items similar with $v_j$. The central problem lies in respectively identifying the neighbors of users and items that share similar rating patterns.

Now, we give the rating-based recommendation in Alg. 2. The relevance ranking tasks are concerned with the relevance of the current user and an item. The problem is to rank the items according to the similarities of the current user. The detailed algorithm is shown in Alg. 3.

**C. COMPLEXITY ANALYSIS**

The ACF algorithm’s time complexity is $O(K \times \#\text{ratings})$, where $K$ is the maximum number of the types of pheromones that a user and an item carry. Typically, we have $k \ll \#\text{users}$.

For online updates, we can easily determine that an update complexity is only $O(K)$. In the recommendation phase, for the rating-based recommendation, the time complexity is...
For the ranking-based recommendation, the time complexity is $O(\#\text{users} + \#\text{items})$. However, the computations could be significantly reduced, if we maintain user neighbors and item neighbors explicitly in the memory or in a database. For the ranking-based recommendation, the time complexity is $O(\#\text{items})$.

Algorithm 2 Rating-Based Recommendation

```
Input: $u_i$, $v_j$, $Ph(U)$, $Ph(V)$
Output: $r_{i,j}$
// Find user neighbors
for user $\in U$ do
    $s_{u_i, \text{user}} = \frac{Ph_{u_i} \circ Ph_{\text{user}}}{|Ph_{u_i}| \times |Ph_{\text{user}}|}$;
end
// Sort users according to their similarity to the current user in descendant order
users = sort($s_{u_i, \text{user}}$);
$C(u_i) = \text{users}[0 : \text{Neighborhood\_size}]$;
// Find item neighbors
for item $\in I$ do
    $s_{v_j, \text{item}} = \frac{Ph_{v_j} \circ Ph_{\text{item}}}{|Ph_{v_j}| \times |Ph_{\text{item}}|}$;
end
// Sort items according to their similarity to the current item in descendant order
items = sort($s_{v_j, \text{item}}$);
$C(v_j) = \text{items}[0 : \text{Neighborhood\_size}]$;
// Fusing ratings
user\_prediction = 0;
user\_similarity = 0;
for user $\in C(u_i)$ do
    if user has rated item $v_j$ then
        user\_prediction += abs($s(u_i, user)) \star (r_{u_i, v_j} - r_{user})$; 
        user\_similarity += abs($s(u_i, user))$;
    end
end
item\_prediction = 0;
item\_similarity = 0;
for item $\in C(v_j)$ do
    if $u_i$ has rated item then
        item\_prediction += abs($s(item, v_j)) \star (r_{u_i, item} - r_{item})$; 
        item\_similarity += abs($s(item, v_j))$;
    end
end
prediction = $r + user\_prediction + item\_prediction$;
```

Algorithm 3 Ranking-Based Recommendation

```
Input: $u_i$, $Ph(U)$, $Ph(V)$
Output: Recommendation list
// Calculate recommendation list
for item $\in I$ do
    $s(u_i, item) = \frac{Ph_{u_i} \circ Ph_{\text{item}}}{|Ph_{u_i}| \times |Ph_{\text{item}}|}$;
end
// Ranking
return Top N list of items ranked by $s(u_i, item)$ in descendant order
```

IV. EXPERIMENTAL EVALUATION

As the collaborative filtering methods follow two different strategies: (a) rating prediction and (b) top-N ranking, we benchmark our proposed methods on both scenarios.

A. RATING-BASED RECOMMENDATIONS

We use a popular movie recommendation dataset, the MovieLens for the benchmarking of the rating prediction algorithms. The MovieLens data consists of more than 1,000,000 ratings from 6,040 users of 3,706 movies. The ratings are made on a five-star scale, and 90% are used for the model training, and the rest of the 10% constitute the test set.

The evaluation metric is the Root Mean Square Error (RMSE), which is given as:

$$RMSE = \sqrt{\frac{1}{n} \sum_{(u_i,v_j) \in \text{TestSet}} (r_{i,j} - \text{estimated\_rating})^2}.$$
The parameters in our algorithm are: (a) transmission rate $\gamma$, (b) evaporation rate $\lambda$. The transmission rate $\gamma$ controls the speed of pheromones that are transferred from the user to an item and vice versa. The bigger the value of $\gamma$, the faster the transmission. The evaporation rate $\lambda$ controls the speed of the pheromone evaporation. The bigger the value of $\lambda$, the slower the pheromone evaporates. In our simulations, we empirically choose the value of $\gamma$ and $\lambda$ to be 0.2 and 1.

We compare our proposed methodologies with two memory-based methods, namely: (a) classic user-based CF and (b) item-based CF [42], two model-based methods: (a) Probabilistic Latent Semantic Analysis (PLSA) [17] and (b) Non-negative Matrix Factorization (NMF) [58], and with one activation spreading method: (a) Rating Similarity Matrix (RSM) [14]. The comparative results are shown in Table 2.

Although the MovieLens data has timestamped information, we do not consider the time effects because the timestamp does not reflect the evolution of the user interests at all. Therefore, all of the algorithms are trained in a batched manner without particular time considerations. However, we will see the influence of time in the following simulations.

### B. RANKING-BASED RECOMMENDATION

To examine the performance of our methodology for a ranking-based recommendation scenario, we performed simulations on two real-world recommender systems: (a) book recommendation and (b) music recommendation. The book recommendation data was crawled from the largest Chinese book recommendation website: Douban (http://www.douban.com). For simulation, we used part of the readers and books as our training data and the rest as the testing dataset. The music recommendation data was crawled from the largest online music recommender system: Last.FM (http://www.last.fm). We used the aforementioned datasets because (a) They are of higher quality than experimental dataset in terms of reflecting real user preferences. (b) They symbolize two types of popular recommender systems. (c) Both of them contain timed implicit ratings (which is of high interest to us). Table 3 shows some statistics of the datasets.

As one of the flaws of ranking-based recommendation is the lacking of evaluation, we use two metrics to be less subjective, namely: Precision [51] and Ranking Accumulation (RA) [63], which is defined as follows.

\[
\text{Precision} = \frac{\# \text{Hitting}}{N}, \quad \text{Recall} = \frac{\# \text{Hitting}}{\# \text{Likes}}, \\
\text{RA} = \frac{\sum_{\text{item} \in \text{Hitting}} \text{rank}(\text{item})}{N} + \frac{\sum_{\text{item} \notin \text{Hitting}} N + 1}{N}.
\]

It is noteworthy to mention that the higher value of $\text{Precision} \in [0, 1]$ is considered better for the system, and the lower the value of $\text{RA} \in \left[\frac{N+1}{2}, N+1\right]$ is deemed appropriate for the system. We also must take notice of the fact that the Hitting set only contains items that are of interest to both of the users and is already present in the test set. We compare our methods with the three ranking-based CF methods reported in [14], [51], [63], respectively. The classical user- and item-based algorithms are also considered for comparisons. The methods proposed in [63] and [14] are denoted as NBI and RSM,\(^1\) respectively. We also implemented the BM25-Item algorithm in [51]. For our simulations, we hold 10% data for testing. All of the results are obtained by averaging the results of five-fold cross-validation. The value of recall of various methods is also provided along with the precision, as shown in Table 4.

The ACF with and without considering time sequences are referred to as the time and timeless versions. We choose

\(^1\)The RSM algorithm in the previous section is the original algorithm, and RSM algorithm in this section is the 0/1 preference implementation.
15 time points within the time span, denoted as $1 \sim 15$ from 2006.3.3 to 2009.6.22. The results are shown in the Fig. 2. An apparent increase can be observed from the results in the precision as time goes on, which means that when the users keep using the recommender systems, the recommendation will be more accurate.

V. CONCLUSION AND FUTURE WORKS

This paper proposed a novel CF algorithm by simulating the ant colony behaviour. By pheromone transmission between users and items, and evaporation of the pheromones on both users and items over time, the ACF algorithm could flexibly reflect the latest user preference and thus can make the most reliable recommendations.

We summarize our major contributions as:

- We introduced the concepts in Ant Colony Optimizations, such as Pheromone and Evaporation into the recommendation domain and proposed a scalable collaborative filtering algorithm that can gracefully handle sparse and evolving rating data.
- The algorithm proposed could recommend with both strategies: rating-based and ranking-based recommendation, which are used in explicit user preference and implicit user preference scenarios, respectively.
- ACF algorithm is easy to be deployed on distributed computational resources, even in a peer-to-peer environment, which means higher scalability and more importantly, user privacy protection.

There are some unexplored possibilities to improve the algorithm proposed in this paper. First, the initialization of pheromones does affect the final recommendation results. Second, the robustness of the ACF algorithm needs to be further explored while running on a larger dataset. The possible mechanism that applications should find the most suitable recommendations.

REFERENCES

[1] A. Abdelwahab, H. Sekiya, I. Matsuba, Y. Horiiuchi, and S. Kuroiwa, “Feature optimization approach for improving the collaborative filtering performance using particle swarm optimization,” J. Comput. Inf. Syst., vol. 8, no. 1, pp. 435–450, 2012.
[2] P. Bedi and R. Sharma, “Trust based recommender system using ant colony for trust computation,” Expert Syst. Appl., vol. 39, no. 1, pp. 1183–1190, Jan. 2012.
[3] S. Chawla, “Web page ranking using ant colony optimisation and genetic algorithm for effective information retrieval,” Int. J. Swarm Intell., vol. 3, no. 1, pp. 58–76, 2017.
[4] B. Chen and L. Chen, “A link prediction algorithm based on ant colony optimization,” Appl. Inf. Technol., vol. 41, no. 3, pp. 694–708, Oct. 2014.
[5] S. M. Daneshmand, A. Javari, S. E. Abtahi, and M. Jalili, “A time-aware recommender system based on dependency network of items,” Comput. J., vol. 58, no. 9, pp. 1955–1966, Sep. 2015.
[6] E. Díaz-Aviles, M. Georgescu, and W. Nejdl, “Swarming to rank for recommender systems,” in Proc. 6th ACM Conf. Rec. Syst., 2012, pp. 229–232.
[7] Y. Ding and X. Li, “Time weight collaborative filtering,” in Proc. 14th ACM Int. Conf. Inf. Knowl. Manage., 2005, pp. 485–492.
[8] T. Donkers, B. Loep, and J. Ziegler, “Sequential user-based recurrent neural network recommendations,” in Proc. 11th ACM Conf. Rec. Syst. (RecSys), 2017, pp. 152–160.
[9] A. Gautam, P. Chaudhary, K. Sindhwani, and P. Bedi, “CBCARS: Content boosted context-aware recommendations using tensor factorization,” in Proc. Int. Conf. Adv. Comput., Comput. Informat. (ICACCI), Sep. 2016, pp. 75–81.
[10] A. K. Ghosh and A. Chakrabarty, “Use of EM algorithm for data reduction under sparsity assumption,” Comput. Statist., vol. 32, no. 2, pp. 387–407, Jun. 2017.
[11] F. S. Gohari, H. Haghif, and F. S. Aliee, “A semantic-enhanced trust based recommender system using ant colony optimization,” Appl. Intell., vol. 46, no. 2, pp. 328–364, Mar. 2017.
[12] D. Green, A. Aleti, and J. Garcia, “The nature of nature: Why nature-inspired algorithms work,” in Nature-Inspired Computing and Optimization, Cham, Switzerland: Springer, 2017, pp. 1–27.
[13] G. Guo, H. Qiu, Z. Tan, Y. Liu, J. Ma, and X. Wang, “Resolving data sparsity by multi-type auxiliary implicit feedback for recommender systems,” Knowl.-Based Syst., vol. 138, pp. 202–207, Dec. 2017.
[14] P. Han, B. Xie, F. Yang, and R.-M. Shen, “An adaptive spreading activation scheme for performing more effective collaborative recommendation,” in Database and Expert Systems Applications. Berlin, Germany: Springer, 2005, pp. 95–104.
[15] H. Hidasi and A. Karatzoglou, “Recurrent neural networks with top-k gains for session-based recommendations,” in Proc. 27th ACM Int. Conf. Inf. Knowl. Manage. (CIKM), 2018, pp. 843–852.
[16] K. L. Hoffman, M. Padberg, and G. Rinaldi, “Traveling salesmen problem,” in Encyclopedia of Operations Research and Management Science, S. I. Gass and M. C. Fu, Eds. Boston, MA, USA: Springer, 2013.
[17] T. Hofman, “Latent semantic models for collaborative filtering,” ACM Trans. Inf. Syst., vol. 23, pp. 89–115, Jan. 2004.
[18] W. Hong, L. Li, and T. Li, “Product recommendation with temporal dynamics,” Expert Syst. Appl., vol. 39, no. 16, pp. 12398–12406, Nov. 2012.
[19] X. Hu, Z. Mai, H. Zhang, Y. Xue, W. Zhou, and X. Chen, “A hybrid recommendation model based on weighted bipartite graph and collaborative filtering,” in Proc. IEEE/WIC/ACM Int. Conf. Web Intell. Workshops (WIW), Oct. 2016, pp. 119–126.
[20] H. Hwangbo and Y. Kim, “An empirical study on the effect of data sparsity and data overlap on cross domain collaborative filtering performance,” Expert Syst. Appl., vol. 89, pp. 254–265, Dec. 2017.
[21] S. H. S. Javadi, S. Khadivi, M. E. Shiri, and J. Xu, “An ant colony optimization method to detect communities in social networks,” in Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining (ASONAM), Aug. 2014, pp. 200–203.
[22] K. Koren, “Collaborative filtering with temporal dynamics,” Commun. ACM, vol. 53, no. 4, p. 89, Apr. 2010.
[23] B. Kumar, P. K. Bala, and A. Srivastava, “Cosine based latent factor model for precision oriented recommendation,” Int. J. Adv. Comput. Sci. Appl., vol. 7, no. 1, pp. 451–457, 2016.
[24] N. Lathia, S. Hailes, L. Capra, and X. Amatriain, “Temporal diversity in recommender systems,” in Proc. 33rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval, 2010, pp. 210–217.
[25] Z. L. Li, C. X. Hu, X. Y. Wei, T. F. Zou, H. R. Zhang, and G. C. Yang, “Enhancing collaborative filtering recommendation by utilizing improved ant colony optimization algorithm,” Appl. Mech. Mater., vol. 556–562, pp. 3793–3799, May 2014.
[26] Z. Lin, Q. Xue, and Y. Lai, “PACOKS: Progressive anti-colony-optimization-based keyword search over relational databases,” in Web-Age Information Management, B. Cui, N. Zhang, J. Xu, X. Lian, and D. Liu, Eds. Cham, Switzerland: Springer, 2016, pp. 107–119.
[27] Z. Lin and H. Chen, “A probabilistic model for collaborative filtering,” in Proc. 9th Int. Conf. Web Intell., Mining Semantics (WIMS), 2019, pp. 1–8.
[28] Y. Liu, F. Cai, Q. Sun, and Y. Zhu, “Collaborative filtering recommendation considering seed life cycle for BT download Websites,” in Computational Intelligence and Intelligent Systems, K. Li, W. Li, Z. Chen, and Y. Liu, Eds. Singapore: Springer, 2018, pp. 562–575.
[29] M. López-Ibáñez, T. Stützle, and M. Dorigo, “Ant colony optimization: A component-wise overview,” in Handbook of Heuristics A. K. Ghosh and A. Chakraborty, “Use of EM algorithm for data reduction under sparsity assumption,” Comput. Statist., vol. 32, no. 2, pp. 387–407, Jun. 2017.
[30] X. Liao, C. Li, and S. Yang, “A survey of swarm intelligence for dynamic optimization: Algorithms and applications,” Swarm Evol. Comput., vol. 33, pp. 1–17, Apr. 2017.
[31] X. Liao et al.: ACF Addressing Sparsity and Temporal Effects
[32] S. Mirjalili, “Dragonfly algorithm: A new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems,” *Neural Comput. Appl.*, vol. 27, no. 4, pp. 1053–1073, May 2016.

[33] N. Mishra, S. Chaturvedi, V. Mishra, R. Srivastava, and P. Bargarh, “Solving sparsity problem in rating-based movie recommendation system,” in *Computational Intelligence in Data Mining*, H. S. Behera and D. P. Mohapatra, Eds. Singapore: Springer, 2017, pp. 111–117.

[34] L. Peška, T. M. Tashu, and T. Horváth, “Swarm intelligence techniques in recommender systems—A review of recent research,” *Swarm Evol. Comput.*, vol. 48, pp. 201–219, Aug. 2019.

[35] X. Pu, R. Jin, G. Wu, D. Han, and G.-R. Xue, “Topic modeling in semantic space with keywords,” in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, 2015, pp. 1141–1150.

[36] C. Rana and S. K. Jain, “An evolutionary clustering algorithm based on temporal features for dynamic recommender systems,” *Swarm Evol. Comput.*, vol. 14, pp. 21–30, Feb. 2014.

[37] C. Rana and S. K. Jain, “A study of the dynamic features of recommender systems,” *Artif. Intell. Rev.*, vol. 43, no. 1, pp. 141–153, Jan. 2015.

[38] A. Rashno, S. Sadri, and H. SadeghianNejad, “An efficient content-based image retrieval with ant colony optimization feature selection schema based on wavelet and color features,” in *Proc. Int. Symp. Artif. Intell. Signal. Process. (AISP)*, Mar. 2015, pp. 59–64.

[39] F. Rezaeinmehr, P. Moradi, S. Ahmadian, N. N. Qader, and M. Jalili, “TCAIRS: Time- and community-aware recommendation system,” *Future Gener. Comput. Syst.*, vol. 78, pp. 419–429, Jan. 2018.

[40] L. B. Romdhane, Y. Chaabani, and H. Zardi, “A robust ant colony optimization-based algorithm for community mining in large scale oriented social graphs,” *Expert Syst. Appl.*, vol. 40, no. 14, pp. 5709–5718, Oct. 2013.

[41] C. Sardianos, I. Varlamis, and M. Eirinaki, “Scaling collaborative filtering to large-scale bipartite rating graphs using lenskit and spark,” in *Proc. IEEE 3rd Int. Conf. Big Data Comput. Service Appl. (BigDataService)*, 2017.

[42] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, “Item-based collaborative filtering recommendation algorithms,” in *Proc. 10th Int. Conf. World Wide Web (WWW)*, 2001, pp. 285–295.

[43] Q. Shambour, “A user-based multi-criteria recommendation approach for personalized recommendations,” *Int. J. Comput. Sci. Inf. Secur.*, vol. 14, no. 12, p. 657, 2016.

[44] P. Sharma, J. Kaur, V. Arora, and P. S. Rana, “Information retrieval in Web crawling using population based, and local search based meta-heuristics: A review,” in *Proc. 6th Int. Conf. Soft Comput. Problem Solving*, K. Deep, J. C. Bansal, K. N. Das, A. K. Lal, H. Garg, A. K. Nagar, and M. Pant, Eds. Singapore: Springer, 2017, pp. 87–104.

[45] J. Sheeba and S. P. Devaneyan, “Recommendation of keywords using swarm intelligence techniques,” in *Proc. Int. Conf. Informat. Anal.*, 2016, p. 8.

[46] B. Sherkat, M. Rahgozar, and M. Asadpour, “Structural link prediction based on ant colony approach in social networks,” *Phys. A, Stat. Mech. Appl.*, vol. 419, pp. 80–94, Feb. 2015.

[47] M. Soleimani-Pouri, A. Rezvanian, and M. R. Meybodi, “‘An ant based particle swarm optimization algorithm for maximum clique problem in social networks,’” in *State of the Art Applications of Social Network Analysis Appl.* Phys. A, Stat. Mech., vol. 14, pp. 21–30, Feb. 2014.

[48] F. Strub, R. Gaudel, and J. Mary, “Hybrid recommender system based on coautocoders,” in *Proc. 1st Workshop Deep Learn. Rec. Syst.*, 2016, pp. 11–16.

[49] S. Tabakhi, P. Moradi, and F. Akhlaghian, “An unsupervised feature selection algorithm based on ant colony optimization,” *Eng. Appl. Artif. Intell.*, vol. 32, pp. 112–123, Jun. 2014.

[50] S. Ujjin and P. J. Bentley, “Particle swarm optimization recommender system,” in *Proc. IEEE Swarm Intell. Symp.*, Mar. 2003, pp. 124–131.

[51] J. Wang, S. Robertson, A. P. de Vries, and M. J. T. Reinders, “Probabilistic relevance ranking for collaborative filtering,” *Inf. Retr.*, vol. 11, no. 6, pp. 477–497, Dec. 2008.

[52] X. Wang, P. Yin, Y. Gao, and L. Guo, “A dynamic recommender system with time-based collaborative user preferences,” *Comput. Intell. Inform.*, vol. 23, no. 1, pp. 25–33, 2019.

[53] C.-Y. Wu, A. Ahmed, A. Beutel, J. A. Smola, and H. Jing, “Recurrent recommender networks,” in *Proc. 10th ACM Int. Conf. Web Search Data Mining (WSDM)*, 2017, pp. 495–503.

[54] L. Xiang, Q. Yuan, S. Zhao, L. Chen, X. Zhang, Q. Yang, and J. Sun, “Temporal recommendation on graphs via long- and short-term preference fusion,” in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2010, pp. 723–732.

[55] H. Ying, F. Zhuang, F. Zhang, Y. Liu, G. Xu, X. Xie, H. Xiong, and J. Wu, “Sequential recommender system based on hierarchical attention networks,” in *Proc. 27th Int. Joint Conf. Artif. Intell. (IJCAI)*, Jul. 2018, pp. 3926–3932.

[56] F. Zafar, I. Moser, and T. Baarslag, “Modelling and analysis of temporal preference shifts using a component-based factorised latent approach,” *Expert Syst. Appl.*, vol. 116, pp. 186–208, Feb. 2019.

[57] H. Zhang, W. Ni, X. Li, and Y. Yang, “Modeling the heterogeneous duration of user interest in time-dependent recommendation: A hidden semi-Markov approach,” *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 48, no. 2, pp. 177–194, Feb. 2018.

[58] S. Zhang, W. Wang, J. Ford, and F. Makedon, “Learning from incomplete ratings using non-negative matrix factorization,” in *Proc. SIAM Int. Conf. Data Mining*, Apr. 2006, pp. 549–553.

[59] S. Zhang, L. Yao, A. Sun, and Y. Tay, “Deep learning based recommender system: A survey and new perspectives,” *ACM Comput. Surv.*, vol. 52, no. 1, pp. 1–38, Feb. 2019.

[60] L. Zheng, V. Noroozi, and P. S. Yu, “Joint deep modeling of users and items using reviews for recommendation,” in *Proc. 10th ACM Int. Conf. Web Search Data Mining*, 2017, pp. 425–434.

[61] Y. Zheng, C. Liu, B. Tang, and H. Zhou, “Neural collaborative filtering for implicit feedback,” in *Proc. 1st Workshop Deep Learn. Rec. Syst. (DLRS)*, 2016, pp. 2–6.

[62] Y. Zheng, R. Burke, and M. Bolasheer, “Optimal feature selection for context-aware recommendation using differential relaxation,” in *Proc. 4th Int. Workshop Context-Aware Recommender Syst. (RecSys)*, 2012.

[63] T. Zhou, J. Ben, M. Medo, and Y.-C. Zhang, “ Bipartite network projection and personal recommendation,” *Phys. Rev. E, Stat. Phys. Plasma Fluids Relat. Interdiscip. Top.*, vol. 76, no. 4, Oct. 2007, Art. no. 046115.

[64] Y. Zhu, X. Shen, and C. Ye, “Personalized prediction and sparsity pursuit in latent factor models,” *J. Amer. Stat. Assoc.*, vol. 111, no. 531, pp. 241–252, Jan. 2016.

[65] A. Zimdars, D. M. Chickering, and C. Meek, “Using temporal data for making recommendations,” 2001, arXiv:1301.2320. [Online]. Available: https://arxiv.org/abs/1301.2320

---

XIAOFENG LIAO received the Ph.D. degree in computer science from the Chinese Academy of Sciences, in 2012. He was supported by the Chinese Scholarship Council as a Visiting Scholar at the University of California, Berkeley, from 2014 to 2015. He is currently with Nanchang University. He works as a Postdoctoral Researcher with the University of Amsterdam (UVA). He works in several EU projects, including ENVRIPLUS and ENVRI-FAIR. His research interests include machine learning, data mining, big data, semantic Web, and information retrieval.

HU WU received the Ph.D. degree from the Institute of Software, Chinese Academy of Sciences, in 2009. He is currently a Freelance Researcher. His main research interests include recommender system, machine learning, and data mining.

YONGJI WANG received the B.S. and M.S. degrees from the Beijing University of Aeronautics and Astronautics, in 1984 and 1987, respectively, and the Ph.D. degree from the University of Edinburgh, U.K., in 1995. He is currently a Professor with the Institute of Software, Chinese Academy of Sciences. His main research interests include real-time scheduling theory, non-linear control, real-time and hybrid control theory, network congestion control, and robot path planning.

---