A Two-phase Evolutionary Algorithm for Solving the Accuracy-diversity Dilemma in Recommendation

Zhipeng Hou
School of Artificial Intelligence
Xidian University
Xi'an, China
hou_zhipeng@outlook.com

Jing Liu
School of Artificial Intelligence
Xidian University
Xi'an, China
neouma@163.com

***Abstract*—Generating both accurate and diverse recommendations is required for modern recommender systems. Accurate recommendations can well meet the needs of users, while diverse recommendations can bring users novelty, discover the unknown preferences of users, and can also alleviate the long tail problem. However, these two metrics are conflicting and it is very hard to improve both at the same time. Thus, a compromise needs to be made. The easiest way is using hyperparameters to combine different objective functions or the recommendations of different recommendation algorithms, but it is difficult to determine parameters. Another common strategy is applying multi-objective evolutionary algorithm to provide multiple recommendations for each user, but how to choose the final recommendation becomes a new problem. To this end, we propose a two-phase evolutionary algorithm-based recommendation framework, named EARF. In EARF, a novel fitness function is designed, which converts the evaluation of accuracy and diversity into a single objective and automatically trades off these two goals. Based on the property of recommendation problem and fitness function, the genetic representation and operators are redefined. The experiments on real-world rating datasets indicate that the EARF is effective and the proposed evolutionary algorithm can achieve a good balance between accuracy and diversity.**

***Index Terms*—recommender system, evolutionary algorithm, accuracy, diversity**

I. Introduction

With the rapid development of information technology and the Internet, we human beings gradually enter into the era of information overload from the era of lack of information. Facing with such a huge amount of data, it is impossible to go through all the content, let alone find what we are interested in. A possible way to solve this problem is using the recommender system (RS), which tries to learn each user's preference from their personal information and historical activities, to discover what the users might like in the future [1]. Since the RS can bring economic benefits and improve the user experience, it is already an indispensable part of current Internet applications and is widely used in various scenarios. Such as, Netflix movie website [2], the e-commerce website in Amazon and Taobao [3], [4], YouTube video website [5], Spotify music system [6], and Google apps store system [7]. Therefore, the RS has high research value no matter in academia or business.

Traditional RSs mainly focus on accuracy, and believe that maximizing the accuracy as much as possible can better match the needs of users and make users satisfy with the system. Since the accuracy-focused recommendation methods pay much more attention to user's historical preferences, it always recommends the same type of items and cannot explore more types of items, which result in a very single recommendation list for each user and cause serious long tail problem [8], [9]. After a period, users will get tired of the recommendations. Therefore, the modern RSs require that the generated recommendations should be both accurate and diverse. However, increasing the diversity means giving up on accuracy and adding variety types of items to the recommendation list, which is obviously a trade-off between these two objectives [10].

Certainly, various recommendation algorithms which take into account both accuracy and diversity have been proposed. A hybrid algorithm was proposed in [10] by combining accuracy-focused and diversity-focused algorithms using weighted linear aggregation. Multi-objectives evolutionary algorithm is used in [11] to find several hybridization parameters to combine different algorithms. Reference [12] proposed a synthetically collaborative filtering model, which combines the user-based and item-based collaborative filtering techniques by using the prevalence rate and novelty rate parameters. Reference [13] introduced a multi-objective evolutionary algorithm to find a set of recommendations for each user by optimizing accuracy and diversity simultaneously. Similarly, both [14] and [15] proposed a multi-objective framework to optimize the two designed objective functions, which describe the abilities of RS to recommend accurate and unpopular items, respectively. The algorithms mentioned above can be roughly divided into two types. One is using hyperparameters to combine different objective functions or the recommendations of different recommendation algorithms. Although simple and effective, it is difficult to determine parameters and those parameters are found by trial and error. The other is using multi-objective evolutionary algorithm to provide multiple recommendations for each user. However, in the real RS, usually only one recommendation list is required for each user. How to choose the final recommendation becomes a new problem.

To alleviate the above problems, we propose a two-phase evolutionary algorithm-based recommendation framework (EARF). First, existing accuracy-focused and diversity-focused recommendation algorithms are applied to acquire the

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

recommendations with different preferences for each user. On this basis, an evolutionary algorithm with carefully designed fitness function and redefined genetic operators is used to automatically trade off the obtained recommendations and generate both accurate and diverse recommendations. Simulation results show that the EARF is effective and the proposed evolutionary algorithm can greatly increase diversity while ensuring the high quality of recommendations.

The main contributions of this work can be summarized as follows:

- We propose a two-phase evolutionary algorithm-based recommendation framework to generate both accurate and diverse recommendations.
- A novel fitness function is designed, which could simultaneously evaluate the accuracy and diversity of recommendations and automatically trade off these two goals.
- Based on the property of recommendation problem and fitness function, the genetic representation and operators are redefined.
- Experiments using real-world rating datasets show that the EARF is effective and the proposed evolutionary algorithm can achieve a good balance between accuracy and diversity.

The rest of this paper is organized as follows. Section II briefly reviews the related work. The proposed EARF is introduced in detail in Section III. The experiments on real-world rating datasets are given in Section IV. Finally, Section V summarizes the work in this paper.

II. Related Work

In our method, recommendation algorithm and evolutionary algorithm are integrated to generate both accurate and diverse recommendations. Therefore, it is necessary to give a brief introduction to these algorithms before delving into details of the proposed framework.

A. Recommendation Algorithm

The most basic recommendation algorithm is collaborative filtering-based algorithms, which give recommendations by calculating similarities between users (user based) [16] or items (item based) [3], [17]. After that, due to the excellent performance in Netflix competition, the Singular Value Decomposition (SVD) based algorithms [18] become popular, which takes the advantages of SVD in linear algebra. To gain better performance and smoothen disadvantages of individual technique above, hybrid algorithms combining two or more recommendation methods are proposed [19], [20]. In addition, others techniques are also applied in RSs, for example, graph-based [10], [21], knowledge-based [22], factorization machines-based [23], and neural network-based [24], [25].

Although, various techniques are used for recommendation, the key processes of recommendation algorithm are the same, which can be summarized as follows: First, for each unrated item of each user, based on the user's historical activities and additional information, predict the user's rating of the item or the probability that the user will like the item in the future. Then, the unrated items are sorted according to the estimated value and the top N items are recommended to the user.

B. Evolutionary Algorithm

The evolutionary algorithm (EA) is a powerful heuristic random search model [26], which has been successfully applied to solve many difficult real-world problems [27]–[30]. In the previous research, a number of recommendation methods based on EA are proposed, in which EAs are used to assist in generating recommendations or optimize the recommendation list. For example, use an EA to cluster users [31], find the similarity weight of each user [32], or estimate similarity matrix [33]. More EA-based recommendation methods can be found in [34], [35].

III. Proposed Framework

To acquire both accurate and diverse recommendations, the most intuitive way is to optimize the recommendations generated by existing accuracy-focused and diversity-focused recommendation algorithms. As shown in Fig. 1, the proposed EARF has a similar design idea, which consists of two phases:

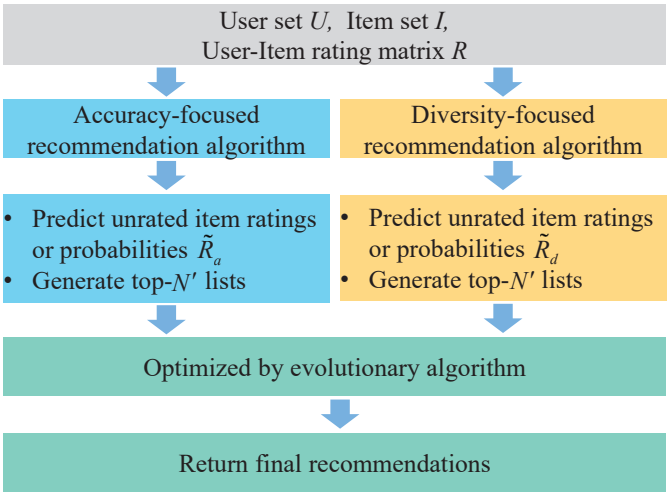

Fig. 1: The proposed two-phase evolutionary algorithm-based recommendation framework.

(1) First, choose suitable accuracy-focused and diversity-focused recommendation algorithms. Then, as we summarized in Subsection II-A: Given the user set U, item set I and user-item rating matrix R, the selected recommendation algorithms will predict the user's unknown ratings or the probability that the user will like the unrated item in the future and generate the recommendation list for each user. In the end, the user-item predict value matrix of accuracy-focused and diversity-focused algorithms can be obtained, which is represented by $\tilde{R}_a$ and $\tilde{R}_d$, respectively. And, each user will have two Top-N' recommendation lists given by the two algorithms.

(2) The evolutionary algorithm is used to combine and optimize the two recommendation lists generated in the first phase and give the final Top-N recommendation list of each user. It is worth mentioning that in the first phase, the length

of recommendation list N' is greater than N. That is because we hope to get more items with different preferences in the first phase to enrich the optimization space. Thus, the recommendation list in the first phase is also called the candidate recommendation list. In the following subsections, the components of the proposed evolutionary algorithm are described in detail.

A. Representation and Initialization

Directly, the Top-N recommendation list is encoded by a vector of integer values, each of which represents the corresponding item ID. Since we need to provide recommendations for all users, the individual is represented as a matrix with the shape of $|U| \times N$, where $|U|$ is the number of users, N is the length of recommendation list, and each row in the matrix represents the Top-N recommendation list of each user. A toy example of the individual representation is given in Fig. 2. In some previous works, the individual is used to encode hyperparameters of each user to combine the recommendations generated from different algorithms, which requires additional decoding steps to get the final recommendation lists. Compared with this method, the representation we adopt is more direct, does not require additional operations and can bring more variety of combination.

	Item ID 1	Item ID 2	…	Item ID N
User 1	45	6		87
User 2	71	34	…	25
⋮		⋮		
User $\|U\|$	11	90		23

Fig. 2: A toy example of the individual representation.

The initialization is simple. To take advantage of recommendations with different preferences, half of the individuals in the population are randomly initialized from the result of the accuracy-focused algorithm, and the other half are randomly initialized from the result of the diversity-focused algorithm. Specifically, the Top-N recommendation list of each user in each individual is randomly selected from the Top-N' candidate recommendation list. Note that each item can only be recommended once for each user, which means that there is no duplicate item in each user's recommendation list.

B. Fitness Function

To provide a fitness function which could converts the evaluation of accuracy and diversity into a single objective and automatically trade off these two goals, there are two problems to be solved. First, we need to redesign the objective functions that measure accuracy and diversity to let the values of these two objectives are no longer negatively correlated. Second, these two goals should be combined appropriately to automatically find a well balance between them rather than simply using hyperparameter.

In terms of accuracy, since the use of test data is prohibited during the training phase, it is impossible to use commonly used accuracy metrics such as Precision, Recall and F1 value (see (6), (7) and (8)) as the objective function. Considering that the recommendation list of each user in the accuracy-focused algorithm is generated by sorting the unrated items in descending order according to the predicted values in $\tilde{R}_a$, to some extent the values predicted by the accuracy-focused algorithm can represent the accuracy of items. Therefore, we use the sum of the predicted value in $\tilde{R}_a$ of each item in the recommendation list of each user to approximately estimate the recommendation accuracy. The objective function to measure accuracy is:

$$\text{Maximize } O_1 = \sum_{u=1}^{|U|} \sum_{i=1}^{N} \tilde{r}_{ui\_a} \tag{1}$$

where $\tilde{r}_{ui\_a}$ represents the predicted rating or probability of item i given by user u in $\tilde{R}_a$. A higher value of this objective indicates the recommendation is more accurate. And, the recommendation generated by the accuracy-focused algorithm can achieve the maximum value of this objective.

In terms of diversity, since it can be directly calculated in the training stage, the diversity metric like Coverage and Personality (see (9) and (10)) is used in many existing studies. But during the experiment, we find that the accuracy objective is indirect and the diversity objective is direct, which will make it easy to improve the diversity of recommendations. And because the accuracy and diversity are contradictory, this will bring great difficulties to find a balance between these two objectives. To alleviate the above problems, similar to accuracy objective O_1, the values predicted by the diversity-focused algorithm in $\tilde{R}_d$ can be used to evaluate the diversity of the items. And the objective function to measure diversity can be calculated as follows,

$$\text{Maximize } O_2 = \sum_{u=1}^{|U|} \sum_{i=1}^{N} \tilde{r}_{ui\_d} \tag{2}$$

where $\tilde{r}_{ui\_d}$ represents the predicted rating or probability of item i given by user u in $\tilde{R}_d$. A higher value of this objective indicates the recommendation is more diversity. And, the recommendation generated by the diversity-focused algorithm can achieve the maximum value of this objective. In this way, the value of these two objectives are no longer negatively correlated and can be increased together.

Inspired by the F1 value using harmonic mean to balance the Precious and Recall metrics (see (6), (7) and (8)), the same combination strategy is used to trade off the two designed objectives O_1 and O_2. In such way, the values of these two objectives could be as high as possible and as close as possible. Since the range of predicted value is different in different algorithms, the normalization should be performed first. The

normalization procedure is given in (3) and (4), and the fitness function is given in (5):

$$O_{1\_norm} = O_1 / O_{1\_max} \tag{3}$$

$$O_{2\_norm} = O_2 / O_{2\_max} \tag{4}$$

$$\text{Maximize } fitness = \frac{2 \times O_{1\_norm} \times O_{2\_norm}}{O_{1\_norm} + O_{2\_norm}} \tag{5}$$

where O_{1_max} and O_{2_max} are the maximum values of objective O_1 and O_2, respectively. The recommendation which can maximize objective O_1 comes from the Top-N recommendation of the accuracy-focused algorithm. Similarly, the Top-N recommendation of the diversity-focused algorithm can maximize objective O_2. To achieve this objective, the easiest way is to combine and optimize the recommendations generated from different algorithms, which is obviously a combination optimization problem. Since the number of users and items in the system is usually very large, the search space is also very huge. Compared with other search algorithm, EA can better solve this kind of problem. Thus, EA is used in our framework. It can be seen in the experimental section that this fitness function can lead the EA find a good balance between accuracy and diversity.

C. Genetic Operators

The genetic operators used in our EA include crossover and mutation, which decide the algorithm performance and convergency time. In the following subsections, detailed descriptions about these two operators are given.

1) Crossover: The procedure of crossover operator used in this paper can be summarized as follows: First, the 2-tournament selection is used to randomly select two parents. Then, for each row of the two parent matrices, a random number in [0, 1] is produced. If the number is less than the crossover probability P_c, these two rows perform the uniform crossover. Thus, P_c determines the percentage of users whose recommendation list may change. Since duplicate items are not allowed in user's recommendation list, additional operations are introduced to avoid generating invalid solutions. An example of the uniform crossover is shown in Fig. 3. First, the common items from two parents are identified and passed to the child. Then, the remaining items perform crossover. A random number in [0, 1] is produced for each remaining position in the child. If the number is larger than 0.5, the child receives the item from the first parent. Otherwise, it receives the item from the second parent. The reason why we adopt this crossover operator is because this method is simple and easy to implement, can generate any combination of parent chromosomes and guarantee the generated solutions are all reasonable.

2) Mutation: To make the recommendation lists not limited to the items included in the initialization, improve the fitness value of the individual and accelerate the convergence speed, the mutation operator is carefully designed. The probability of whether each row performs mutation is determined by the mutation probability P_m. The key idea of mutation is that under a certain judgement criterion, one item can be found and replaced with another item to improve the accuracy or diversity. Detailed description is given in Algorithm 1.

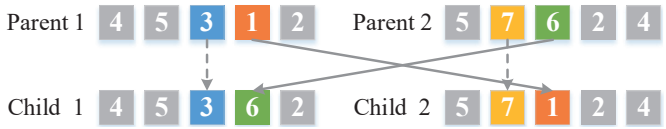

Fig. 3: An example of the uniform crossover. Only the positions not in grey perform crossover. Two generated random numbers are 0.2 and 0.8, respectively.

Algorithm 1 Mutation

Input: P_{child}: The offspring generated in crossover operator;
P_m: Mutation probability;
Output: P_{child}:The individual generated by P_{child} through mutation;

for each user $u \in U$ **do**
  if $rand(0,1) < P_m$ **then**
    L_u=Get recommended items of user u in P_{child};
    $item_1$=In L_u, find the item with the lowest predicted value in $\tilde{R}_a$;
    C_1=Find the candidate set that can replace $item_1$, in which the item is unrated by user u, does not appear in L_u and has higher predicted value in $\tilde{R}_a$ than $item_1$;
    $Prob_1$=Sort the items in C_1 in ascending order according to the predicted value in $\tilde{R}_a$ and record the ranking of each item as the probability of being selected;
    $item_2$=In L_u, find the item with the lowest predicted value in $\tilde{R}_d$;
    C_2= Find the candidate set that can replace $item_2$, in which the item is unrated by user u, does not appear in L_u and has higher predicted value in $\tilde{R}_d$ than $item_2$;
    $Prob_2$=Sort the items in C_2 in ascending order according to the predicted value in $\tilde{R}_d$ and record the ranking of each item as the probability of being selected;
    if $item_1 = item_2$ **then**
      Take $Prob_1$ and $Prob_2$ as probabilities, randomly select an item from C_1 and C_2 to replace $item_1$ in L_u. If C_1 and C_2 have a common item, the probability of the item is the sum of the corresponding value in $Prob_1$ and $Prob_2$;
    else
      repeat
        $item'_1$=Take $Prob_1$ as probabilities, randomly select an item from C_1;
        $item'_2$=Take $Prob_2$ as probabilities, randomly select an item from C_2;
      until $item'_1 \neq item'_2$;
      Replace $item_1$ and $item_2$ in L_u with $item'_1$ and $item'_2$, respectively;
    end if
  end if
end for

IV. Experimental Results

A. Datasets

In our experiments, we use three datasets collected from real systems. An overview of these datasets is given in Table I. The ml-1m and hetrec-ml are two benchmark datasets in the recommender system. The first one is collected by GroupLens from MovieLens over various periods of time and the hetrec-ml is released at the 5th ACM Conference on Recommender Systems as an extension of ml-10m dataset. Since both of them describe the user ratings (1-5) about movies, they are used to test the performance of algorithms at different scales. There are more users in ml-1m, while the hetrec-ml has more items. These two datasets can be downloaded from GropuLens web site[1]. The last dataset[2], which describes the user ratings (1-10) of books, is a part of the dataset released by Kaggle to test the performance of the algorithms in different scenarios.

TABLE I: Statistics of datasets used in our experiments.

Dataset	No. of users	No. of items	No. of ratings
ml-1m	6040	3706	1000209
hetrec-ml	2113	10109	855598
goodbooks	3369	9993	376332

B. Metric

To evaluate the performance of the recommendation algorithm, the dataset is usually split into two parts, namely the known part D and unknown part $\tilde{D}$. The known part is visible to the algorithm and the recommendation is given based on it, while the unknown part is regarded as a validation set to measure the quality of the generated recommendations and the items with high ratings in it are considered to be the items that users will like in the future.

The Precision, Recall and F1 value are three commonly used metrics to evaluate the accuracy of recommendations. Precision measures the fraction of items that user likes among top N recommended items, while Recall measures the fraction of items that can be found in top N recommendation list among all items that the user likes. Since Precision and Recall are often negatively correlated and rely on the number of items recommended, for fairness, the F1 value is used as the accuracy metric in this paper, which considers both of the Precision and Recall metrics and is a harmonic mean of them. The F1 value can be calculated as follows,

$$Precision = \frac{1}{|U|}\sum_{i=1}^{|U|}\frac{|L_i \cap \tilde{D}_i|}{N} \tag{6}$$

$$Recall = \frac{1}{|U|}\sum_{i=1}^{|U|}\frac{|L_i \cap \tilde{D}_i|}{|\tilde{D}_i|} \tag{7}$$

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{8}$$

[1]https://grouplens.org/datasets/

[2]https://www.kaggle.com/zygmunt/goodbooks-10k

where $|U|$ is the number of users, L_i is the top N recommendation list of user i and $\tilde{D}_i$ is the set of items with high ratings rated by user i in $\tilde{D}$. The higher the F1 is, the more accurate the recommendation results are.

The diversity metrics consist of Coverage and Personality. Coverage measures the fraction of items that can be covered in the top N recommendation list of all users among all items in the system, which is defined as follows,

$$Coverage = \frac{|L_1 \cup L_2 \cup \cdots \cup L_{|U|}|}{|I|} \tag{9}$$

where $|I|$ is the number of users A higher value of coverage indicates the algorithm can cover more broadly items, and can better solve the long tail effect. Personality measures the difference of top N recommended list between each pair of users. A higher value of Personality means the recommendation is more diversity and personalized. The Personality can be calculated as follows,

$$Personality = \frac{1}{|U|\cdot(|U|-1)}\sum_{i\in U}\sum_{j\in U}(1-\frac{|L_i \cap L_j|}{N}),(i \neq j) \tag{10}$$

C. Baseline Methods

To verify the performance of our method, we compare it with three hybrid recommendation algorithms designed to trade off the accuracy and diversity. Since two of these three algorithms use the results generated by the user-based and item-based collaborative filtering algorithms (UCF and ICF) to provide the final recommendations, in EARF we choose UCF as the accuracy-focused algorithm and ICF as the diversity-focused algorithm. The details about these algorithms are given as follows,

- UCF [16]: A widely used algorithm in RS, which tends to recommend popular items to acquire high recommendation accuracy. Since it is not only simple and easy to implement, but also generates accurate recommendations, it is still used as prototype or component of many modern recommendation systems.
- ICF [3]: It has a similar process to the UCF, but give recommendations by calculating similarities between items. The ICF tends to recommend long-tail items (unpopular or less rated items) to improve the recommendation diversity.
- Top UCF+ICF: A simple way to combine the recommendations generated by UCF and ICF, which alternately selects items from top N recommendations of UCF and ICF to generate the final recommendation list. If the item is already in the final recommendation list, skip the item and select the next one. If there are no duplicate items, the final recommendation list is equal to directly concatenate the top $N/2$ recommendations of UCF and ICF.
- Probs+Heats [10]: Both ProbS and HeatS are graph-based recommendation algorithms. The former is accuracy-focused and the latter is diversity-focused. The

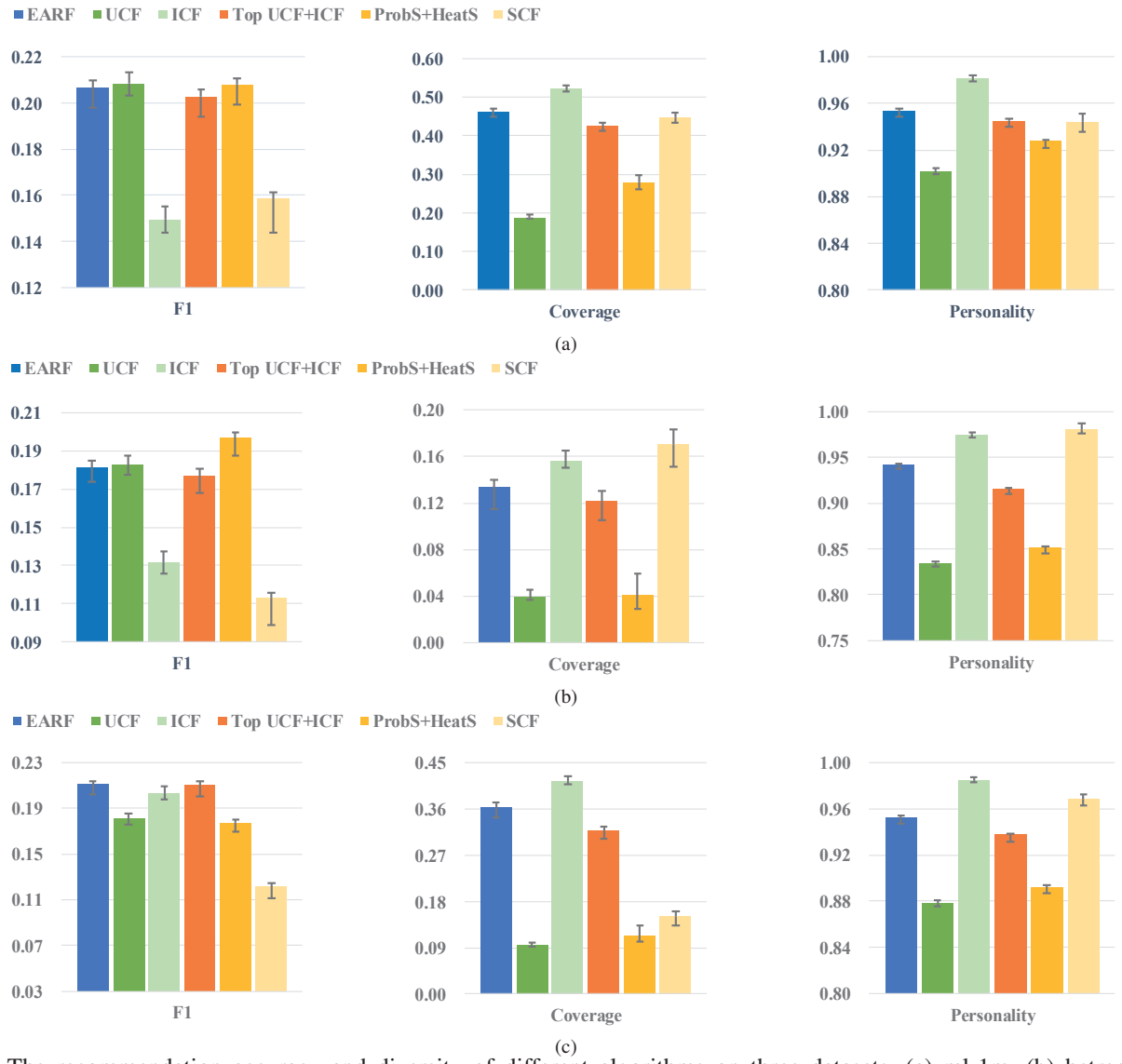

Fig. 4: The recommendation accuracy and diversity of different algorithms on three datasets. (a) ml-1m. (b) hetrec-ml. (c) goodbooks.

ProbS+HeatS uses weighted linear aggregation to combine the functions in the two algorithms to obtain both accurate and diverse recommendations.

- SCF [12]: A synthetically collaborative filtering model, which uses the prevalence rate and novelty rate parameters to combine the recommendations of UCF and ICF.

D. Result

In our experiments, for all datasets, we randomly sample 80% rated items of each user as the known part and evaluate the performance of the algorithm on the remaining 20% rated items. In EARF, the length of candidate recommendation list N' is set to 50, the population size is set to 50, the number of generations is set to 100, the crossover probability P_c is set to 0.8, and the mutation probability P_m is set to 0.5. In UCF and ICF, Cosine is used as the similarity metric and the number of nearest neighbors k is set to 50. In ProbS+HestS, the weight parameter λ is set to 0.2. In SCF, the prevalence rate α and the novelty rate β are set to default values 0.3, the items with less than five ratings are regard as long tail items. For all algorithms, the length of final recommendation list N is 10.

The recommendation accuracy and diversity of different algorithms on three datasets are given in Fig. 4, where the leftmost blue bar is the result of our algorithm, the two green bars in the middle are the result of the accuracy-focused algorithm and diversity-focused algorithm used in our framework, and the three orange bars in the rightmost are the result of the three other hybrid algorithms. Considering the randomness of data splitting and the stochastic nature of evolutionary algorithm, we use five-cross validation and run EARF 30 times for each experiment. Then, the maximum, average and minimum values of each algorithm on each metric are plotted. It can be seen

from the figures that no matter for the datasets at different scales (Fig. 4a and 4b) or different scenarios (Fig. 4c), EARF can always balance the accuracy and diversity very well and generate both accurate and diverse recommendations, which implies that the effectiveness of designed fitness function and the robustness of using evolutionary algorithm. In terms of accuracy (F1 value), the result of EARF is very close to the result of the accuracy-focused algorithm UCF used in our framework, and even improved in some scenarios (Fig. 4c). In terms of diversity (Coverage and Personality), the result of EARF is between the results of the accuracy-focused algorithm UCF and the diversity-focused algorithm ICF used in our framework. Compared with the three other hybrid algorithms, with the same accuracy, our algorithm can provide the most diverse recommendations. All in all, we can draw a conclusion that our framework is effective and the proposed evolutionary algorithm can greatly increase diversity while ensuring the high quality of recommendations.

To further verify that EARF can well combine the recommendations generated from different algorithms, we use ProbS and HeatS as the accuracy-focused and diversity-focused algorithms in our framework, respectively, then compare the results of EARF and using weighted linear aggregation (ProbS+HeatS). The accuracy and diversity of using different ways to combine the recommendations generated by ProbS and HeatS on the goodbooks dataset is given in Fig. 5. It is obvious that EARF outperformers ProbS+HeatS, which also demonstrates the superiority of using EA.

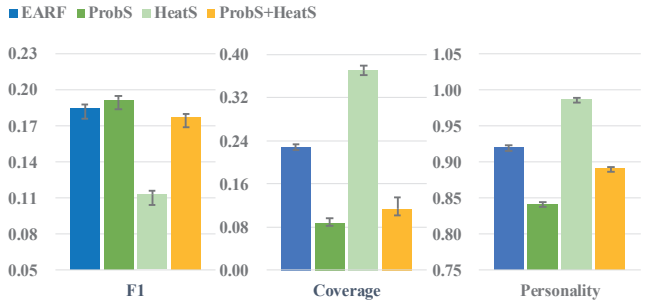

Fig. 5: On the goodbooks dataset, the accuracy and diversity of using different ways to combine the recommendations generated by Probs and Heats.

V. Conclusions

We propose EARF, a novel two-phase evolutionary algorithm-based recommendation framework to generate both accurate and diverse recommendations. In EARF, a general EA is introduced, which can optimize the recommendations generated by existing accuracy-focused and diversity-focused recommendation algorithms. To automatically trade off the accuracy and diversity, the fitness function is carefully designed. In addition, the genetic representation and genetic operators are also redefined to better adapt to the recommendation problem and the fitness function. Extensive experiments on real-world rating datasets with different scales and scenarios demonstrate that the framework is effective and the proposed EA can greatly increase diversity while ensuring the high quality of recommendations.

In the future, we will try to add local search or use other more complex EAs to further improve the framework. In this work, we only focus on how to use EA to combine recommendations of two algorithms. Whether using EA to optimize recommendations of more than two algorithms can achieve better results is still a problem that can be further explored.

Acknowledgments

This work was supported in part by the Key Project of Science and Technology Innovation 2030 supported by the Ministry of Science and Technology of China under Grant 2018AAA0101302 and in part by the General Program of National Natural Science Foundation of China (NSFC) under Grant 61773300.

References

[1] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, “Recommender systems handbook”, Springer, 2015.

[2] C. A. Gomez-Uribe and N. Hunt, “The netflix recommender system: algorithms, business value, and innovation,” ACM Trans. on Management Information Systems, vol. 6, no. 4, pp.13, 2016.

[3] G. Linden, B. Smith, and J. York, “Amazon.com recommendations: item-to-item collaborative filtering,” IEEE Internet computing, no. 1, pp.76-80, 2003.

[4] X. Ma, L. Zhao, G. Huang, and Z. Wang, “Entire space multi-task model: an effective approach for estimating post-click conversion rate”, In: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, pp.1137-1140, 2018.

[5] P. Covington, J. Adams, and E. Sargin, “Deep neural networks for youtube recommendations”, In: Proceedings of the 10th ACM conference on recommender systems, pp.191-198, 2016.

[6] C.-W. Chen, P. Lamere, M. Schedl, and H. Zamani, “Recsys challenge 2018: automatic music playlist continuation”, In: Proceedings of the 12th ACM Conference on Recommender Systems, pp.527-528, 2018.

[7] H.-T. Cheng, L. Koc, J. Harmsen, and T. Shaked, “Wide & deep learning for recommender systems”, In: Proceedings of the 1st workshop on deep learning for recommender systems, pp.7-10, 2016.

[8] A. Bellogin, P. Castells, and I. Cantador, “Precision-oriented evaluation of recommender systems: an algorithmic comparison”, In: Proceedings of the fifth ACM conference on Recommender systems, pp.333-336, 2011.

[9] S. M. McNee, J. Riedl, and J. A. Konstan, “Being accurate is not enough: how accuracy metrics have hurt recommender systems”, In: CHI’06 extended abstracts on Human factors in computing systems, pp.1097-1101, 2006.

[10] T. Zhou, Z. Kuscsik, J.-G. Liu, and M. Medo, “Solving the apparent diversity-accuracy dilemma of recommender systems,” In: Proceedings of the National Academy of Sciences, pp.4511-4515, 2010.

[11] M. T. Ribeiro, A. Lacerda, A. Veloso, and N. Ziviani, “Pareto-efficient hybridization for multi-objective recommender systems”, In: Proceedings of the sixth ACM conference on Recommender systems, pp.19-26, 2012.

[12] J. Wang and J. Yin, “Combining user-based and item-based collaborative filtering techniques to improve recommendation diversity”, In: The 6th International Conference on Biomedical Engineering and Informatics, pp.661-665, 2013.

[13] Y. Zuo, M. Gong, J. Zeng, and L. Ma, “Personalized recommendation based on evolutionary multi-objective optimization,” IEEE Computational Intelligence Magazine, vol. 10, no. 1, pp.52-62, 2015.

[14] S. Wang, M. Gong, H. Li, and J. Yang, “Multi-objective optimization for long tail recommendation,” Knowledge-Based Systems, vol. 104, no. pp.145-155, 2016.

[15] L. Cui, P. Ou, X. Fu, and Z. Wen, “A novel multi-objective evolutionary algorithm for recommendation systems,” Journal of Parallel and Distributed Computing, vol. 103, no. pp.53-63, 2017.
[16] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl, “An algorithmic framework for performing collaborative filtering”, In: The 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp.230-237, 1999.
[17] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl, “Item-based collaborative filtering recommendation algorithms,” WWW, vol. 1, no. pp.285-295, 2001.
[18] Y. Koren, R. Bell, and C. Volinsky, “Matrix factorization techniques for recommender systems,” Computer, vol. no. 8, pp.30-37, 2009.
[19] R. Burke, “Hybrid recommender systems: Survey and experiments,” User modeling and user-adapted interaction, vol. 12, no. 4, pp.331-370, 2002.
[20] R. Burke, “Hybrid web recommender systems,” The Adaptive Web, pp. 377–408, 2007.
[21] T. Zhou, J. Ren, M. Medo, and Y.-C. Zhang, “Bipartite network projection and personal recommendation,” Physical review E, vol. 76, no. 4, 2007.
[22] R. Burke, “Knowledge-based recommender systems,” Encyclopedia of library and information systems, vol. 69, no. 32, pp.175-186, 2000.
[23] X. He, J. Pan, O. Jin, and T. Xu, “Practical lessons from predicting clicks on ads at facebook”, In: Proceedings of the Eighth International Workshop on Data Mining for Online Advertising, pp.1-9, 2014.
[24] H. Guo, R. Tang, Y. Ye, and Z. Li, “DeepFM: a factorization-machine based neural network for CTR prediction,” arXiv preprint, 2017.
[25] G. Zhou, N. Mou, Y. Fan, and Q. Pi, “Deep interest evolution network for click-through rate prediction”, In: Proceedings of the AAAI Conference on Artificial Intelligence, pp.5941-5948, 2019.
[26] D. E. Goldberg and J. H. Holland, “Genetic algorithms and machine learning,” 1988.
[27] S. Wang and J. Liu, “Enhancing the robustness of complex networks against edge-based-attack cascading failures”, In: IEEE Congress on Evolutionary Computation, pp.23-28, 2017.
[28] Z. Yang, J. Liu, and K. Wu, “Learning of boosting fuzzy cognitive maps using a real-coded genetic algorithm”, In: IEEE Congress on Evolutionary Computation, pp.966-973, 2019.
[29] K. Wu and J. Liu, “Classification-based optimization with multi-fidelity evaluations”, In: IEEE Congress on Evolutionary Computation, pp.1126-1131, 2019.
[30] Z. Ren and J. Liu, “Extracting information cores with multi-property using a multi-objective evolutionary algorithm”, In: IEEE Congress on Evolutionary Computation, pp.1014-1021, 2019.
[31] K.-j. Kim and H. Ahn, “A recommender system using GA k-means clustering in an online shopping market,” Expert systems with applications, vol. 34, no. 2, pp.1200-1209, 2008.
[32] J. Bobadilla, F. Ortega, A. Hernando, and J. Alcalá, “Improving collaborative filtering recommender system results and performance using genetic algorithms,” Knowledge-Based Systems, vol. 24, no. 8, pp.1310-1316, 2011.
[33] B. Alhijawi and Y. Kilani, “Using genetic algorithms for measuring the similarity values between users in collaborative filtering recommender systems”, In: IEEE/ACIS 15th International Conference on Computer and Information Science, pp.1-6, 2016.
[34] T. Horváth and A. C. de Carvalho, “Evolutionary computing in recommender systems: a review of recent research,” Natural Computing, vol. 16, no. 3, pp.441-462, 2017.
[35] M. Sadeghi and S. A. Asghari, “Recommender Systems Based on evolutionary computing: a survey,” Journal of Software Engineering and Applications, vol. 10, no. 05, pp.407, 2017.