Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 132 (2018) 1795–1803

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on Computational Intelligence and Data Science (ICCIDS 2018)

An Improved Collaborative Filtering Based Recommender System using Bat Algorithm

Sambhav Yadav[a], Vikesh[a], Shreyam[a], Sushama Nagpal[a] *

[a]Netaji Subhas Institute of Technology, Sector-3, Dwarka, Delhi 110078, India

Abstract

Recommender Systems have proven to be of great aid in dealing with the issue of Information Overload by improving the user experience through quality recommendations. In recent times, heuristic techniques have been employed by researchers in recommender systems along with traditional methods of collaborative and content based filtering. On the same account, in this work a Bat algorithm based heuristic technique has been used to compute the weights of items (features) so as to find better neighbourhood for the active user. We argue and also prove using the results that this technique of giving weights to items using heuristic methods helps in achieving better personalized recommendations. The performance of this system was also compared to that of Artificial Bee Colony based system (ABC). The results indicated that BA performed 6.9% better than ABC in terms of Mean Absolute Error and F1 Score using our technique.

Keywords: Bat Algorithm; Recommender System; Artificial Bee Colony algorithm; Swarm Intelligence; Collaborative Filtering;

1. Introduction

Finding the right item/information that might interest the user has become a troublesome task, of late, as data on the web has grown to a massive level throughout the past couple of decade. To void out this problem Recommender Systems (RSs) have been developed over the years to suggest items to the user that they would most likely prefer.[1] Exempli gratia: - Amazon recommends items to user in the section '*People who purchased this also purchased these items*'. Traditionally RSs have been bracketed as Collaborative filtering (CF), Content based filtering (CBF), Demographic based and Hybrid recommender systems.

* Corresponding author. Tel.: +91-9910460599 ;
E-mail address: sushmapriyadarshi@yahoo.com

10.1016/j.procs.2018.05.155

CF is based on the idea that people who consented in the past will also consent in the future. It tries to find out similar users on the basis of the past knowledge of user's behaviour and recommend items from the set of items liked by those similar users. CBF is based on the specifications of the items. This type of RS tries to suggest items, alike to the items which the user has previously liked. The problem associated with content based filtering is that if the content doesn't contain sufficient information to discriminate the items then recommendations won't be good enough [1]. Demographic based RSs exploit users' demographic data (Ex: Age, Gender, Occupation) to find similar users and then make recommendations. Hybrid RSs are those that combine two or more of the above techniques for recommendations. CF techniques have been found to give most promising results from the above kinds. However still, there are certain deficiencies in CF techniques viz. Cold start, Sparsity, lack of customized recommendations and generating context aware recommendations [2].

Researchers have strived to eliminate the effect of these problems to improve upon the quality of recommendations using various techniques. Guo et al. [3] used TrustSVD to reduce the degradation of recommendations by Sparsity and cold start. Trust information between users was used by [4] to better find the set of users with similar taste and enhance the recommendations. Liu Xiaojun [5] showed how clustering can be used to improve CF. However clustering based CF techniques fails to achieve higher accuracies as they tend to give equal weight-age to all the items while forming the clusters of similar users whereas Matrix factorization based techniques fails to generalize new information.

Heuristic techniques have seen a huge advancement in the past decade. The advantage of these techniques is their ability to improve the efficiency of search process, possibly by sacrificing claims of completeness. Meta-heuristic technique Swarm Intelligence (SI) which includes Cuckoo Search, Artificial Bee Colony(ABC) [3], Bat algorithm (BA) have been widely maneuvered by researchers to search for the optimal solution in a search space of solutions [6]. SI techniques tend to produce the optimized solution by iteratively improving the candidate solutions. The use of SI techniques to learn the optimal weight of each feature so as to form better clusters of similar users was the motivation behind conceptualizing and then later validating this work. Thence the objective of this work is twofold:

1. To produce customized recommendations for each user. For this purpose different weight vectors of features are used for different users.
2. To improve upon the accuracies of recommendations by combining SI techniques with traditional CF techniques.

Bat algorithm which is a relatively less explored meta-heuristic method has been used to compute the weights of the features. Xin-She [7] showed in his work that BA outperforms PSO and genetic algorithms on certain standard functions. BA provides very quick convergence by reflexive switching between exploitation and exploration because of its certain parameters discussed in section 3 [7]. In this work we compared BA in RS with ABC and concluded that BA outperforms ABC in generating more personalized and accurate recommendations. The approach used to revamp the recommendations was authenticated by the experimental results obtained on the Jester dataset.

The remainder of writing is ordered as follows: Portion 2 presents the associated work in the domain of RSs and use of CI techniques in RSs. The approach along with the algorithm has been reported in section 3. Section 4 explains the various experiment conducted and section 5 elaborates the results by unveiling better insights into them. Section 6 concludes the paper and discusses the future work.

2. Related Work

RSs are intelligent systems that help users in providing information useful to them. These have seen a remarkable growth in past two decades in providing better recommendations to the users. This section surveys the (1) Conventional CF based recommender systems and (2) Incorporation of Nature inspired Swarm techniques in RSs.

2.1 Conventional CF based RSs

CF based RSs are the most versatile recommender systems. These are generally implemented using two methodologies: Model based approach[8, 9] and Nearest-neighbourhood approach. Model based appeal use the ratings matrix to train a models like SVD[8] and provide good accuracy but generally are not flexible to adapt with new data (ratings or items) easily. Nearest-neighbour approaches on other hand rely on user's cooperation and try to recommend items (movies, music etc.) based on finding users having items pattern similar to that of active user. The CF based RSs were first used in Tapestry weave systems where the authors used it to find similar users[10]. Since then these have been used in different contexts and have proved to be successful to a great extent like these have been used by Amazon for recommending books, Jester System[11] for recommending jokes.

With time, researchers have also incorporated other techniques like the use of demographic data, proposing a hybrid RS based on content based and collaborative filtering, incorporation of implicit and explicit trust[1] into CF based RS. Penncock et al.[12] in their work used CF for finding the probability of a user liking a new item based on their previous items history. Traditional CF methods however suffer from a number of limitations[2]. In order to tackle these limitations, and to improve the accuracy of RSs, people have tried to incorporate swarm intelligent techniques in RSs.

2.2 Swarm Techniques in Recommender Systems

Swarm intelligent (SI) techniques are a set of optimization techniques which tend to provide feasible solutions to highly complicated problems[13]. Many researchers have combined traditional RSs with SI techniques and their results have outdone conventional RSs. Ujjin and Bentley[14] utilised Particle Swarm Optimization (PSO) technique to generate a set of weights for a user's features and use a modified Euclidean function in order to generate recommendations. Their approach showed considerable improvement over the Pearson algorithm and the Genetic algorithm.

Katarya and Verma[15] utilised K-Means to provide initial parameters to the PSO algorithm. PSO was then used to optimize Fuzzy C Means Clustering and the experiment conducted on the MovieLens dataset indicated some enhancement over existing methods. Choudhary et al. [6] compared the Gravitational Search Algorithm (GSA) with PSO on Jester dataset and found that GSA performs significantly better than PSO. The paper[16] proposed a hybrid recommender system combining K-Means and ABC (ABC-KM) in order to generate recommendations. Results found using MovieLens dataset indicated that the new algorithm boosted scalability and also helped reduce the problem of cold start. Chen et al. [17] used artificial immune system with collaborative filtering on MovieLens and EachMovie datasets and achieved good results. Wasid and Kant [18] fuzzified user features and generated user weights using PSO technique. Their hybrid RS (FPSO-CF) performed better than existing CF and fuzzy RSs. The paper also showed that the performance of FPSO-CF achieved better results as compared to a hybrid of Genetic algorithm and fuzzy features. Janusz Sobecki [19] compared a number of SI algorithms for the purpose of recommending courses to students. Prediction Accuracy indicated a difference of only 0.1 between all the five algorithms thus establishing the usefulness of swarm techniques.

Recently, authors have tried to enhance the effectiveness of RSs by adding trust to RSs. Bedi and Sharma [4] combined the Ant Colony optimization (ACO) technique with trust. After initially constructing a trust graph using a harmonic mean of similarity and confidence of users, the authors utilised ACO as a means to update trust values between users and thereby provide recommendations. A study conducted on Jester and MovieLens datasets indicate good improvement in performance. Kant and Bharadwaj[20] improved recommendations by exploiting fuzzy models for trust and distrust in RSs.

The above works proved that a hybrid of SI and CF techniques are helpful in achieving better personalized recommendations for users. Therefore, continuing this work further, in this paper, we have attempted to use Bat algorithm (BA)[7], which is a relatively lesser explored technique in RS domain, to generate feature weights, and Pearson Correlation Coefficient measure to find neighbour of active user. Finally, top-N recommendations have been generated for the active user. Based on a number of evaluation metrics, we have compared our results with the ABC algorithm.

3. Our Approach

Traditionally CF works by finding the set of alike users for the active user (user for whom recommendations are to be created) and then emulating the behaviour of those similar users for the active user. Several similarity measures which include Cosine similarity and Pearson correlation coefficient (PCC) have been used by researchers to assess how far two users are correlated or not related at all. These similarity measures evaluate similarity between two users by giving equal importance to all the items which doesn't always produce good results. To better understand this argument, consider a user-item rating matrix with 3 users and 2 items as shown below:

$$\begin{bmatrix} & \mathbf{I1} & \mathbf{I2} \\ \mathbf{U1} & 2 & 4 \\ \mathbf{U2} & 2 & 5 \\ \mathbf{U3} & 2 & 4 \end{bmatrix}$$

User-Item rating Matrix

$$\begin{bmatrix} & \mathbf{U1} & \mathbf{U2} & \mathbf{U3} \\ \mathbf{U1} & 1 & 0.83 & 1 \\ \mathbf{U2} & 0.83 & 1 & 0.83 \\ \mathbf{U3} & 1 & 0.83 & 1 \end{bmatrix}$$

Similarity Matrix

Presume that, if a user U1 likes an item he rates it a 4 and if he doesn't like it he rates it a 2. Thus from above matrix it is clear that U1 likes I2 but doesn't like I1. But if some other user U2 likes item I2 equally as much as U1 likes I2, he rates it a 5. Also user U2 dislikes I1 and rates it 2. Thus the similarity between U1 and U2 should be 1 as they both like I2 and dislike I1, but this is not the case as is evident from the similarity matrix. This contradiction occurred because the criteria for rating good items by user U1 and U2 was different.
On the same note, if user U3 rates an average item (I1) 2 and a bad item 1, the similarity between U1 and U2 shouldn't be 1.

Certain researchers have used fuzzy logic to try and overcome this problem [20]. In this work, the authors present a different scheme to conquer this problem by giving weights to items whilst evaluating the similarity between users. The weights are assigned and iteratively improved using the algorithms mentioned. The authors used BA because of its superiority to give better results over other swarm based techniques as explained by [7]. The equations used in Bat algorithm are explained below and then later on the algorithm to predict ratings is presented formally.

3.1 Bat Algorithm

BA[7] is based on the echolocation behaviour of micro bats with differing pulse rate emission and loudness. Bats use SONAR to detect food/prey and circumvent barriers. Bats emit high frequency sound waves and when these waves hit any obstacle they reflect back to the bat, the time after which the bats receive the wave and the difference in time of receiving the wave in each ear help bats to locate objects.

Each bat flies randomly in the search space at position x_i having velocity v_i with a fixed wavelength λ and a varying frequency f_i having loudness A_0 to search for the optimal solution. Since frequency wavelength product is constant, we can also keep f fixed and λ varying. The bats move during every iteration to generate new solution according to the following equations:

$$f_i = f_{min} + (f_{max} - f_{min})\beta \qquad (1)$$

$$v_i^{t+1} = v_i^t + (x_i^t - x_{best})\, f_i^{(t)} \qquad (2)$$

$$x_i^{t+1} = x_i^t + v_i^t \qquad (3)$$

Where, the superscript $t + 1$ in each equation represents the current iteration and t the previous iteration. The subscript i represents the i^{th} bat in a population of N bats. β is a uniformly distributed random vector in [0, 1]. The current global best position is stored in x_{best} which is updated by comparing the solutions of all bats. f_i is the randomly initialised pulse frequency for x_i^{th} bat lying between $[f_{min}, f_{max}]$, where f_{min} and f_{max} are minimum and maximum frequency range in which the bats can change their frequency and are determined from the problem size. Random walk is used to locally create new solutions for each bat according to the following equation:

$$x_{new} = x_{old} + \epsilon A^t \tag{4}$$

Where A^t is the average of the loudness of each bat during the t^{th} iteration and ϵ the scaling factor in the range [-1, 1]. Also as each bat gets closer to the optimal solution (food) the loudness amplitude decreases and rate of pulse emission r increases according to the following equations:

$$\mathrm{A}_i^{t+1} = \alpha \mathrm{A}_i^t \tag{5}$$

$$r_i^{t+1} = r_i^0(1 - e^{(-\gamma t)}) \tag{6}$$

We have used $\alpha = \gamma = 0.9$ for our simulations as suggested by [7]. Now we present the formal description of BA used by us.

Algorithm 1: Bat Algorithm

```
Input:-
        N: Bats population size          r : Pulse rate
        Iter: Iterations in each run     A : Loudness
        fmin: Minimum frequency       fmax: Maximum frequency

Output:- Optimized feature weights

Initialize N bats' velocity $V_i$ and position $X_i$ randomly. (i= 1,2...N)
Initialize the rate of pulses $r_i$ and loudness amplitude $A_i$
Set frequency of pulse $f_i$ at $X_i$
for t=0 to Iter
    Generate new solutions (move bats) by updating frequency using equation 1,
    velocity using equation 2, and position using equation 3
    if ( $r_i$ < random)
        Choose a bat from the best bats/solutions
        Produce an internal solution around the chosen bat using equation 4
    end if
    Assess the new solutions using the fitness function.

    fit_value(i) = fitness_fun($X_i$)
    if (solution improves for any bat and rand < $A_i$ )
        best_sol(i) = $X_i$
        fitness(i) = fit_value(i)
        Decrease $A_i$ and increase $r_i$ using equation 5 and 6.
    end if
    Revise the current global best solution from the best_sol.

end for
```

For the fitness function of the BA, authors have used Mean Absolute Error (MAE) of the actual and predicted ratings of the items. Thus the goal of BA is to find those sets of feature weights which minimize MAE.

$$MAE = \frac{\sum(actual\ rating - predicted\ rating)}{R}$$

where R = the total number of readings over which MAE is calculated.

Karaboga [21] showed superiority of ABC over PSO for a variety of functions, and PSO has also been widely used in the domain of RS. So in this work we compare the results of BA with that of ABC. We do not describe the ABC algorithm here, it can be studied from the work of [21].

3.2 How recommendations are generated?

When a new active user enters the system for which the suggestions are to be created, BA/ABC iteratively optimizes the weights of the features (items) by searching in the m (total items) dimensional search space of weights. After the weights of features are learnt then the neighbourhood of the active user is formed using PCC as measure of distance between two similar users. PCC is modified wherein the ratings of items are considered after multiplying the actual ratings with the weights calculated by BA.

$$s_{u,v} = \frac{\sum_i (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_i (r_{u,i} - \bar{r}_u)^2}\sqrt{\sum_i (r_{v,i} - \bar{r}_v)^2}} \quad (7)$$

$r_{u,i}, r_{v,i} =$ Item i's rating given by user u and v respectively	$\bar{r}_u =$ Mean rating of user u
$\bar{r}_v =$ mean rating of user v	$S_{u,v} =$ PCC among user u and v

To predict ratings $P_{a,k}$ of item $\boldsymbol{k}$ for active user $\boldsymbol{a}$

$$P_{a,k} = \frac{\sum_{i \in U_M \cap U_{Rk}} S_i * R_{ik}}{\sum_{i \in U_M \cap U_{Rk}} S_i} \quad (8)$$

U_M = set of users in the neighbourhood of user $\boldsymbol{a}$	U_{Rk} = set of users who rated item'k'
S_i = PCC between user 'i' and $\boldsymbol{a}$	R_{ik} = Rating given by user 'i' to item 'k'

Equation 8 is used to forecast the ratings of items which the active user hasn't rated so far. The $top - N$ highly rated items are then endorsed to the active user.

4. Experimental Setup

The $Jester$ dataset-1 (http://eigentaste.berkeley.edu/dataset/) have been used to demonstrate the experimental work. The Jester dataset comprises of 4.1 million ratings given by 73k users for 100 jokes. Ratings provided by the first 1000 users were utilized while conducting the experiment. All these users have rated more than 36 jokes. The first column of the dataset consists of the number of jokes rated by the user. Subsequent 100 columns contain the ratings for the 100 jokes in the range of -10.0 to +10.0 with unrated jokes given a rating of 99.

The experiments were performed using five-fold cross validation technique. This ensured that bias was not present. For calculating precision and recall, any joke with a positive rating was considered as relevant. For both BA and ABC, three sets of experiments were carried out by randomly picking 10 active users (k) in the first experiment, 15 active users in the second experiment, and 20 active users in the third experiment. All these experiments were performed for topN = 5, 10, and 15. The parameters used while applying BA and ABC are provided in table 1. The same parameter values have been used by [6] [18]. Moreover on increasing the swarm size or the number of iterations, no substantial boost in performance was found. It was only increasing the computational time.

Table 1. Specifications for BA and ABC

Parameter	Parameter Value	Description
Swarm Size	10	No. of bats/bees
Iterations	30	Iterations for which BA/ABC is executed
Runs	10	Total runs of algorithm

In case of ABC, the swarm size is equally divided among employed bees and onlooker bees. The Limit[21] in ABC is taken as $0.6 * No. of\ onlooker\ bees * Dimension$ where it provides an estimation of the number of trials after which unimproved state is abandoned.

Mean Absolute error (MAE), Precision (Pre), Recall (Rec), and F1 Score (F1) have been used to evaluate the proposed approach. The key findings are depicted in the subsequent sections. The experiments have been performed on a standard machine with an i5 processor, having a clock speed of 2.2GHz and 8 GB of RAM.

5 Results and Discussions

This section describes the experimental results and also describes the findings of the work. The values of Mean absolute error, Precision, Recall, and F1 Score obtained using BA and ABC is displayed in table 2. Figure 1 graphically shows the variation of MAE for different numbers of randomly selected active users using both the algorithms. From the figure 1 and the table 2 it can be interpreted that BA outperforms ABC 67% of time.

Table 2. Results obtained using BA and ABC on Jester Data set

Active Users(k)	Algorithm	topN = 5				topN = 10				topN = 15			
		MAE	**Pre**	**Rec**	**F1**	**MAE**	**Pre**	**Rec**	**F1**	**MAE**	**Pre**	**Rec**	**F1**
10	BA	2.82	56.0	44.9	49.8	3.11	60.6	78.7	68.5	3.06	59.6	94.9	73.2
	ABC	3.23	53.2	37.1	43.7	2.97	62.6	67.2	64.8	3.04	55.7	86.5	67.8
15	BA	2.74	68.6	41.3	51.6	3.63	63.6	74.0	68.4	3.12	60.1	88.7	71.7
	ABC	3.01	57.2	36.6	44.6	4.13	52.3	71.5	60.4	3.76	58.5	81.9	68.3
20	BA	3.37	59.0	35.4	44.2	3.08	62.1	72.2	66.8	3.64	58.7	90.4	71.2
	ABC	3.90	53.8	31.6	39.8	3.75	56.7	70.9	63.0	3.19	59.3	92.1	72.1

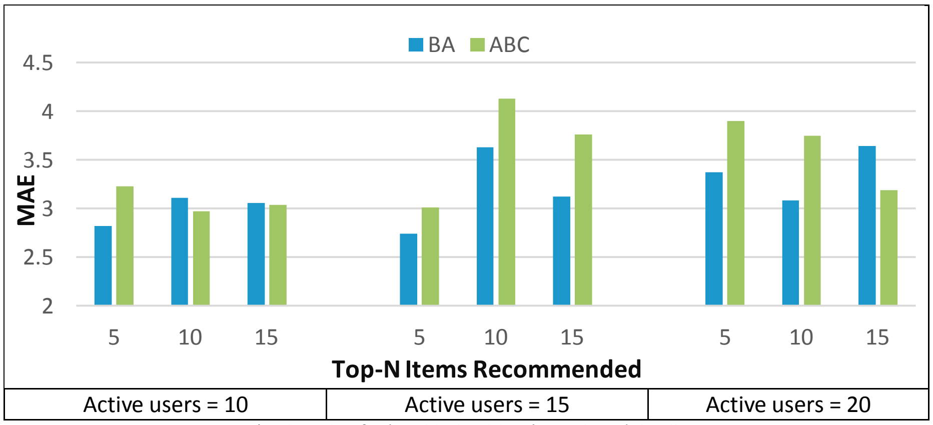

Fig. 1. MAE for k = 10, 15, 20 using BA and ABC

Root mean squared error (**RMSE**) is also a good measure of evaluating the algorithms' performance by penalizing large errors more. Fig. 2 depicts the variation of RMSE by varying the active users for top 10 recommendations. From the figure we can deduce that BA outperforms ABC in most of the cases.

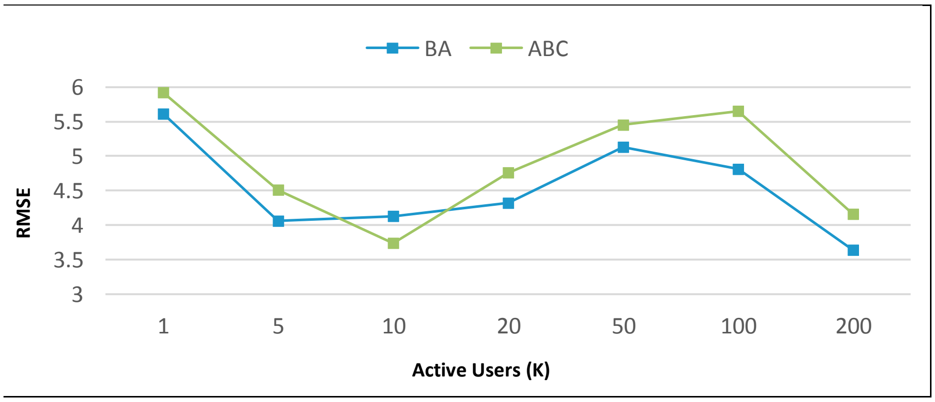

Fig. 2. RMSE for top 10 Recommendations

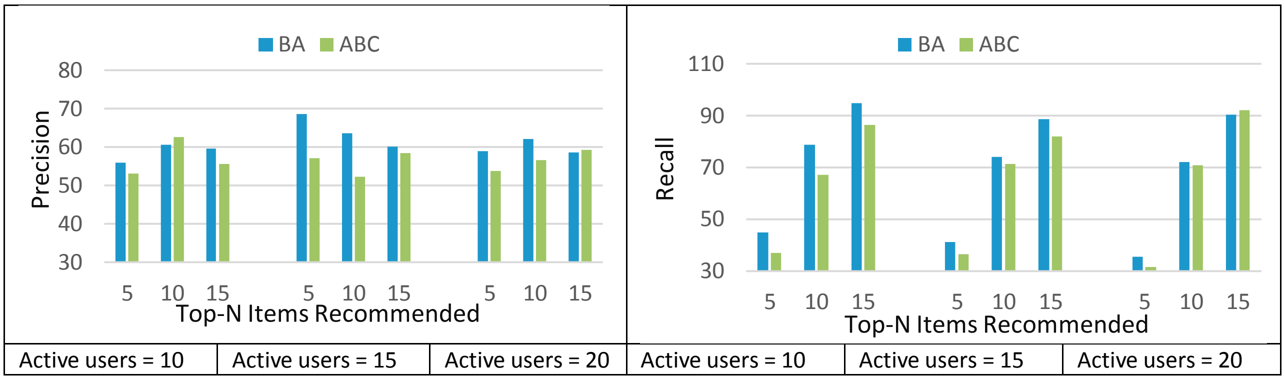

Fig. 3 Precision and Recall for k = 10, 15, 20 using BA and ABC

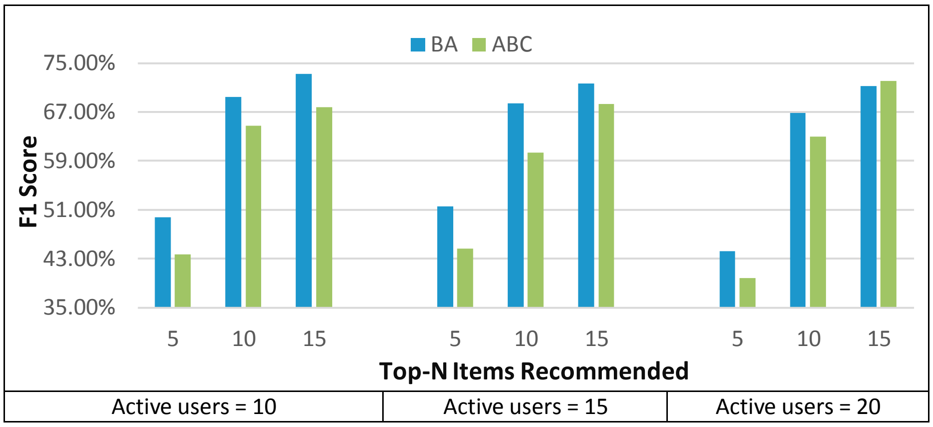

Fig. 4 F1 Score for k = 10, 15, 20 using BA and ABC

Fig. 4 shows F1 score for different sets of active users using both BA and ABC. From the F1 score it can be inferred that the BA not only minimizes the MAE but also provides quality recommendations to users in terms of precision and recall. From the results the application of BA in the field of RSs for feature weighting was confirmed, as BA performed 6.9% better than the ABC algorithm on the Jester Data set.

6. Conclusions and Future Work

In this work, we proposed an approach to give weights to the items in the user-item rating matrix so as to find the better neighborhood of the active user using the BA. This method helped in providing personalized recommendations to all users as it generated a different set of weights for each user. The results also confirmed that BA achieves a lower MAE compared to other traditional collaborative filtering methods. BA also performs better than other swarm based techniques (PSO, ABC) in RSs.

To extend this work, the authors will fuzzify the ratings into good, bad and average to further improve the recommendations. Social Network data can also be used to learn personal traits and likes of users'.

7. References

[1] Donovan, John O', and Barry Smyth. (2005) "Trust in recommender systems" in *Proceedings of the 10th international conference on Intelligent user interfaces*, 167-174, California, USA .

[2] Gupta, Swati, and Sushama Nagpal. (2015) "Trust Aware Recommender Systems: A Survey on Implicit Trust Generation Techniques" *International Journal of Computer Science and Information Technologies* **6**(**4**): 3594-

3599.

[3] Guo, Guibing, Jie Zhang, and Neil Yorke-Smith. (2016) "A Novel Recommendation model regularized with user trust and item ratings" *IEEE transaction on Knowledge and Data Engineering* **28**(**7**) : 1607-1620.

[4] Bedi, Poonam, and Ravish Sharma. (2012) "Trust based recommender system using ant colony for trust computation" *Expert Systems with Applications* **39**(**1**) : 1183-1190.

[5] Xiaojun, Liu. (2017) "An improved clustering based collaborative filtering recommendation algorithm" *Cluster Computing* **20**(**2**) : 1281-1288.

[6] Choudhary, Vedant, Dhruv Mullick, and Sushama Nagpal. (2017) "Gravitational Search Algorithm in Recommendation Systems", in *International Conference in Swarm Intelligence* : 597-607, Japan.

[7] Yang, Xin-She. (2010) "A New Metaheuristic Bat-Inspired Algorithm" *Nature Inspired Cooperative Strategies for Optimization* **284** : 65-74.

[8] Koren, Yehuda, Robert Bell, and Chris Volinsky. (2009) "Matrix Factorization Techniques for Recommender Systems" *Computer* **42**(**8**) : 30-37.

[9] Rennie, Jasson, and Nathan Srebro. (2005) "Fast maximum margin matrix factorization for collaborative prediction", in *ICML '05 Proceedings of the 22nd international conference on Machine learning* 713-719, Bonn Germany.

[10] Goldberg, David, David Nichols, Brian M. Oki, and Douglas Terry. (1992) "Using collaborative filtering to weave an information tapestry," *Communications of the ACM - Special issue on information filtering* **35**(**12**) : 61-70.

[11] Goldberg, Ken, Theresa Roeder, Dhruv Gupta, and Chris Perkins (2001) "Eigentaste: A constant Time Collaborative Filtering Algorithm" *Information Retrieval* **4(2)** 133-151.

[12] Pennock, David, Eric Horvitz, Steve Lawrence and CLee Giles (2000) "Collaborative Filtering by Personality Diagnosis: A Hybrid Memory- and Model-Based Approach" in *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence* **16:** 473-480, California USA.

[13] Deepa, and Senthilkumar (2016) "Swarm Intelligence from natural to artificial systems : Ant Colony Optimization," *International Journal on Applications of Graph Theory in Wireless Ad hoc Networks and Sensor Networks(GRAPH-HOC)* **8(1):** 9-17.

[14] Ujjin, Supiya, and Peter. J. Bentley (2003) "Particle swarm optimization recommender system" in *Swarm Intelligence Symposium*, Indianapolis.

[15] Katarya, Rahul and Om P. Verma (2016) "A collaborative recommender system enhanced with particle swarm optimization technique" *Multimedia Tools and Applications* **75(15):** 9225–9239.

[16] Katarya, Rahul (2018) "Movie recommender system with metaheuristic artificial bee" *Neural Computing and Applications* **1(1):** 1-8.

[17] Chen, Meng Hui, Chin H. Teng, and Pei Chann Chang (2015) "Applying artificial immune systems to collaborative filtering for movie" *Advanced Engineering Informatics* **29(4):** 830-839.

[18] Wasid, Mohammed, and Vibhor Kant (2015) "A Particle Swarm Approach to Collaborative Filtering based Recommender Systems through Fuzzy Features" in *International Conference on Communication Networks* **54:** 440-448, Bangalore India.

[19] Sobecki, Janusz (2014) "Comparison of Selected Swarm Intelligence Algorithms in Student Courses Recommendation Application" *International Journal of Software Engineering and Knowledge Engineering* **24(1):** 91-109.

[20] Kant, Vibhor, and Kamal K. Bharadwaj (2013) "Fuzzy Computational Models of Trust and Distrust for Enhanced Recommendations," *International Journal of Intelligent Systems* **28(4):** 332-365.

[21] Karaboga, Dervis (2005) "An Idea based on Honey bee swarm for Numerical Optimization" *Technical Report - TR06* Erciyes University Kayseri/Türkiye.