Improving Results and Performance of Collaborative Filtering-based Recommender Systems using Cuckoo Optimization Algorithm

Majid Hatami
Faculty of Electrical and Computer Engineering
University of Tabriz, Tabriz, Iran

Saeid Pashazadeh
Faculty of Electrical and Computer Engineering
University of Tabriz, Tabriz, Iran

ABSTRACT
An approach for improving quality and performance of collaborative filtering-based recommender systems is proposed in this paper. A slight change on similarity metric is proposed. To obtain more accurate similarity measurement between two users, similarity measurement method needs a well-chosen weight vector. Different weight vectors could be employed based on the recommender system and the taste of users, but only some of them are suitable. To obtain the best results we have to find the most suitable weight vector among all possible ones. A meta-heuristic algorithm has been introduced to find near optimal weight vector. Cuckoo optimization algorithm is used to obtain optimized weight vector. The results are promising and satisfactory. Our results are compared with the results of previous approaches to verify effectiveness of new proposed method.

General Terms
Artificial Intelligence, Data Mining, Recommender System, Evolutionary Algorithm.

Keywords
collaborative filtering, recommender systems, similarity measurement, cuckoo optimization algorithm, performance

1. INTRODUCTION
Recommender systems (RS) are built to help users prevent information overload problems which makes finding desired information more time-consuming and sometimes exhausting process. They assist users when they are not experienced enough to make a choice among all alternatives[1]. There are two categories of RS in general: collaborative filtering (CF) and content-based filtering (CB)[2].

CF simply refers to collaboration of people when they help each other with performing filtering by recording their reactions to the items they have faced[3]. CF uses some of the information filtering techniques based on the users' reactions to items or their history of purchases, but CB uses the items content and users' preferences to find more items of interest.

Hybrid RS has been introduced for combining two or more recommendation techniques to obtain better results with fewer drawbacks which each technique normally exhibits when is used individually[4]. One of the most common hybrid models is combination of CF and CB.

Much research has been done on CF area in recent years. As described earlier, CF methods reflect real human's social behavior from this view point that people have a tendency to make collective decisions instead of individual decisions (like choosing a book to read, a brand to buy a product from, etc.). People usually base their collective decisions on those with whom they have something in common.

For instance, machine learning papers could be recommended to the persons who have shown interest in RS related papers. The reason would be the high interests of RS paper readers in machine learning field. In this situation, the common point is reading RS papers. Recommender systems are continuously observing users' behavior to discover items that they are interested in or feel distaste for. Having more things in common increases the similarity of two users. Providing a good similarity measurement method will results in finding more similar users which causes making better recommendations.

Making suggestions (recommendations), using k-nearest neighbor (kNN) approach in CF for certain user whom we call 'active user' consists of two main steps:

1. Finding K most similar users to the active user (who the RS should make recommendation for).
2. Reasoning the most desirable unseen products as recommendation to the active user, based on collective knowledge of the K most similar users.

Finding similarity between users is one of the most important parts of a kNN CF method. Researchers usually use Pearson correlation metric as a basis for comparison to show that how much they have improved RS qualities[5].

In the literature, different kinds of similarity measurement metrics have been reported, such as Cosine, Pearson Correlation and Mean Squared Differences. Recently a novel metric has introduced by Bobadilla et al. in [5] that takes advantage of Genetic algorithm to find near optimal similarity function. Their similarity method with slight changes is used in this paper. An enhanced prediction formula that we introduced in[6] along with using Cuckoo Optimization Algorithm (COA) instead of Genetic Algorithm (GA) is used in this paper. Based on the experiments made by [7], COA has shown to be faster and more effective in some optimization areas in comparison with GA and Particle Swarm Optimization (PSO), specifically when we are dealing with floating point numbers.

2. RELATED WORKS
There have been many similarity measurement methods reported in the literature. The majority of these methods are used in kNN method. Some of the most commonly used traditional methods to obtain similarity are Pearson correlation (COR), cosine (COS), adjusted cosine (ACOS), constrained correlation (CCOR), Mean Squared Differences (MSD) and Euclidean distance (EUC) [8].

Bobadilla et al. [9] introduced a new metric by combining Jaccard and MSD. It not only uses numerical information from the ratings (through MSD), but also uses the non-numerical information via Jaccard. Bobadilla et al. [10] introduced a new method that uses information of all users' votes instead of restricting their method to use comparison of two users (user to user) or two items (item to item). Bobadilla et al. [5] proposed a metric based on a model obtained using genetic algorithm.

With emerge of web 2.0 new metrics have been introduced, in which new social information such as friends, followers, followings, etc. are used. Most of these methods take advantage of trust, reputation and credibility [8]. These new metrics are able to improve RS qualities, however not all of the RS could provide necessary social information, so using these metrics are not always feasible. There are similarity methods which are able to draw trust and reputation information out of users' ratings, however these social information are not rich enough [8].

3. SIMILARITY METHOD

The ability to recommend items in a kNN approach is accomplished by being able to find similarities between users. A similarity measure represents that how well two users resemble each other based on their ratings.

In this paper we used the result of Bobadilla et al. [5] in the Eq. (1).

$$Sim_w = \frac{1}{M-m+1}\sum_{i=0}^{M-m} w^{(i)} v_{x,y}^{(i)} \quad (1)$$

Where variables M and m referred to the maximum and minimum possible rating that each user could give to a particular item, respectively. RS rates typically are in the range of $\{1, \dots, 5\}$ (therefore $M = 5$ and $m = 1$). Vector v is just another representation for two specific users' ratings, so we consider it as a fixed vector. Thus, the only variable in the formula is the weight vector, w. The length of both w and v are equal to the recommender system rating range length, that is $M - m + 1$ (which in this case is 5).

Modified form of Eq. (1) is used in this paper. Similarity functions normally return a real number in the range of $[-1, 1]$, which -1 means lack of similarity between two users, while 1 means a complete similarity. The output range of Eq. (1) is $[-\frac{1}{M-m+1}, \frac{1}{M-m+1}]$, so we expanded it to the standard range by eliminating the first part of the formula. As a result, our final similarity function will be as follows:

$$Sim_w = \sum_{i=0}^{M-m} w^{(i)} v_{x,y}^{(i)} \quad (2)$$

Each element of vector w should lie in the range of $[-1, 1]$. As is obvious, we can have many different variations of weight vector, however not all of them are useful or optimized. Even though we can estimate the area where the optimized vector may lie, it will not be a small area yet. Bobadilla et al. made use of a genetic algorithm to find the best value for each element of vector w.

Where the optimized weight vector may have lain, depends on the users' taste and their ratings. We are not able to determine a global or overall weight vector for all RS. Bobadilla et al. specified the boundary of optimal weight vector as follows:

$$w_0 \in [1, 0.6] \quad w_1 \in [0.6, 0.2] \quad w_2 \in [0.2, -0.2]$$
$$w_3 \in [-0.2, -0.6] \quad w_4 \in [-0.6, -1]$$

4. CUCKOO OPTIMIZATION ALGORITHM

In COA each individual is called "*habitat*", while it's known as "*chromosome*" in GA. Bobadilla et al. used GA to find the optimal weight vector. Their GA chromosomes were represented in binary form as strings of 0s and 1s. A conversion method has been used to convert weight vectors with real numbers into binary strings, and vice versa.

COA has been proposed as a new evolutionary algorithm by Rajabioun [7]. In his experiments, COA has shown effective results in comparison with some other algorithms such as GA and PSO. So, in this paper we made use of COA instead of GA to find the best weight vector, w. COA uses floating point number array for habitats. Thus, we can use weight vectors directly as our habitats without any conversion. Our habitats look like the following:

w_0	w_1	w_2	w_3	w_4
$\in [-1, 1]$	$\in [-1, 1]$	$\in [-1, 1]$	$\in [-1, 1]$	$\in [-1, 1]$

Figure 1 shows the flowchart of COA.

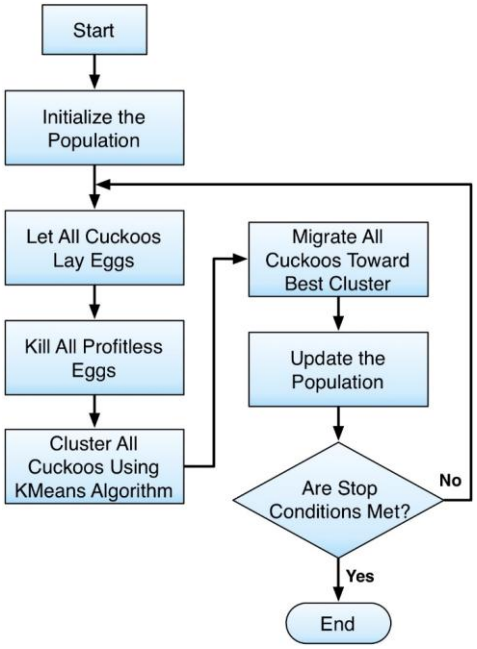


Fig 1: COA Flowchart

4.1 Initial Population

Bobadilla et al. seeded the 50% of the GA initial population with random values, and the remainder 50% with the values in the area where optimal vector, w, might lie. The same assumptions are made in our COA initial population.

4.2 Profit Function

Each individual has a value of profitability which is known as '*fitness*' in GA or '*profit*' in COA. Recommender systems usually tend to have more accurate predictions, in other words, less Mean Absolute Error. So, the most profitable individual (i.e. least MAE) among all cuckoos is of interest. As a consequence, we used the MAE of the whole recommender system to assign the profit value of each individual. First, we set the vector w of the similarity function (Eq. (1)) to the individual habitat which represents a weight

vector, afterwards we calculate the MAE of the RS using the modified similarity function.

Our profit function is as follows:

$$profit = MAE = \frac{1}{\#U}\sum_{u\in U}\frac{\sum_{i\in I_u}\left|p_u^i - r_u^i\right|}{\#I_u} \quad (3)$$

where $\#U$ and $\#I_u$ represent the number of training users and the number of training items rated by the user u, respectively. p_u^i represents our prediction of the item i for the user u, while r_u^i stands for the real rate. To obtain p_u^i, we use the enhanced prediction formula which is introduced in [6] as follows:

$$p_u^i = \lambda + \frac{\sum_{x\in K_u}[Sim(u,x)\times(r_u^i - \bar{r_x})]}{\sum_{x\in K_u} Sim(u,x)} \quad (4)$$

where $\bar{r_x}$ is the average of the ratings given by the user x and λ is described by the following equations:

$$\lambda = \alpha \times \bar{r^i} + (1-\alpha)\gamma \quad (5)$$

$$\gamma = \beta\,(\frac{1}{K}\sum_{x\in K_u}\bar{r_x}) \; + \; (1-\beta)\bar{r_u} \quad (6)$$

where γ represents the collective average of active user's and all other similar users' ratings. λ is a combination of the γ and the average of rates given to the item i by all users. α and β determine how the obtained averages should be mixed together and how much each one should affect the prediction. We have set $\alpha = 0.25$ and $\beta = 0.5$ as default values, however these values could be varied based on the nature of RS. In this way, all possible values for α and β in the range of $[0, 1]$ and in steps of 0.01 will be examined when the training is done and we have obtained the optimized weight vector.

4.3 Properties

As other evolutionary algorithms, cuckoo optimization algorithm has different parameters which needs to be tuned. We have set all these parameters as follows:

- **Initial Cuckoos Count = 5**
  The population size in the beginning.
- **Population Size = 10**
  The population size during each generation.
- **KMeans Cluster Count = 1**
  Number of clusters that should be made using KMeans algorithm.
- **Cuckoos Minimum Eggs = 3**
  Minimum eggs that each cuckoo could lay.
- **Cuckoos Maximum Eggs = 5**
  Maximum eggs that each cuckoo could lay.
- **Migration Coefficient = 9**
  The coefficient which is used in migration formula.
- **Egg Laying Radius = 3**
  Maximum distance that all eggs should be laid within.

We have achieved these values by performing many experiments, however COA algorithm has shown to be rather stable against slight variation in most of these parameters.

4.4 Reproduction

In our implementation we kept the number of individuals immutable through generations. Taking into consideration the fact that each cuckoo could lay 3 to 5 eggs, the total number of cuckoos to be created in each generation varies between 30 and 50 (as the population size is 10). We keep the top 10 suitable individuals as the new population for the next generation. In the terms of time, as the most expensive operation in evolutionary algorithms is evaluating the profit (or fitness) of individuals, we made our comparison based on the number of calls to this function. The GA method evaluate 100 individuals fitness, while in our proposed method, we evaluate 30 to 50 individuals profit. Consequently, our method is proved to be at least 2 times faster than the GA method.

5. EXPERIMENTS

Thus far, different metrics to measure the quality of recommender systems have been introduced. Bobadilla et al. in[8][11] provided different evaluation metrics. We have employed some of the most commonly used metrics such as MAE, Coverage, Precision and Recall to compare our proposed method with GA and Pearson Correlation. We also carried out our experiments on different datasets as specified in Table 1.

Table 1. Specifications of the datasets used in our experiments

Quantity	MovieLens 100K	MovieLens 1M
Users	943	6040
Movies	1682	3952
Ratings	100,000	1,000,209

6. RESULTS

In this section, we show our experiments results. Fig. 2 and Fig. 3 show different evaluations which have been made on the MovieLens 100K and 1M, respectively. In all these figures we considered the following constrains:

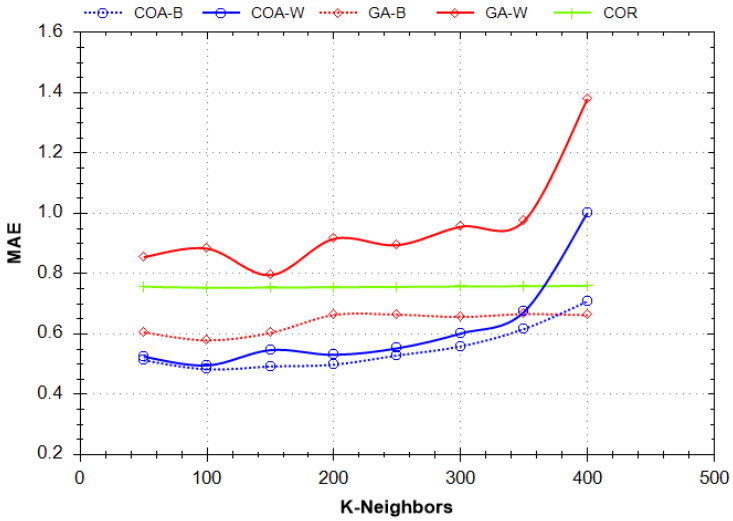
- Experiments have been done on Pearson Correlation (COR), Genetic Algorithm (GA) and our proposed COA method.
- As evolutionary algorithms usually do not return constant results on different runs of the algorithm, the COA and GA methods have been run 10 times with 10 iterations for each run to evaluate the best and the worst cases. In all figures COA-B and GA-B refer to the best result of COA and GA, and COA-W and GA-W refer to the worst results of COA and GA.
- The constant K varies between 50 and 400 in steps of 50.
- To measure Precision and Recall, we have set: $\theta = 4$, N (the number of recommendations) varying in the range of $\{2, \dots, 30\}$ in steps of 2, $K = 100$ for MovieLens 100K, $K = 150$ for MovieLens 1M dataset.
- The weight vector is obtained only by using 80% of the whole datasets as training users and items. The remaining 20% is used to measure different qualities and obtaining the diagrams.

As is clear from Fig. 2(a) and Fig. 3(a), the error in proposed COA algorithm is lower in comparison to other methods. The best and worst cases are closer to each other in comparison with GA. In most cases, even the worst COA case is better than the best case of GA. The best result is achieved when we used K=100 in MovieLens 100K and K=150 in MovieLens 1M.

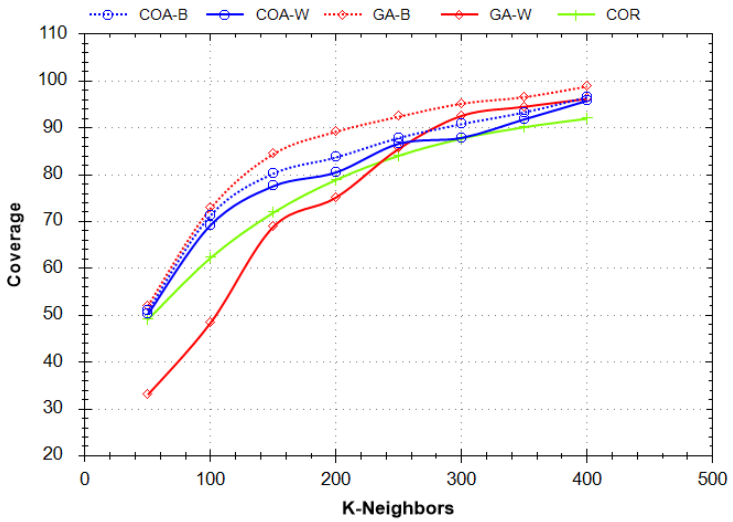
The best and worst cases in coverage (Fig. 2-3(b)) are close together which means good stability. In MovieLens 100K,

COA coverage is higher than COR and is close to GA-B, but in MovieLens 1M it's closer to the GA-B, even though the COR is the best when K is less than 200.

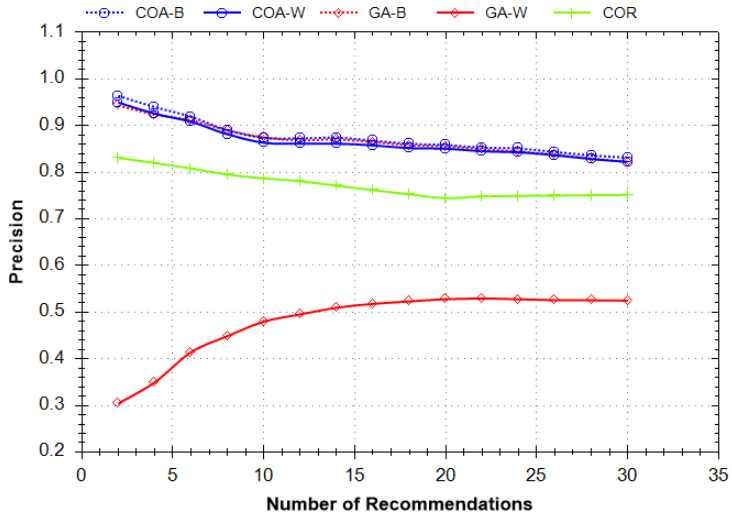
Fig. 2(c-d) and Fig. 3(c-d) show that our proposed method gives best results for precision and recall, in all cases. So, we can conclude that our method results in the best and worst cases are always better than GA and COR, however the GA-B is so close to COA. COA-B and COA-W are almost overlapped which means complete stability.



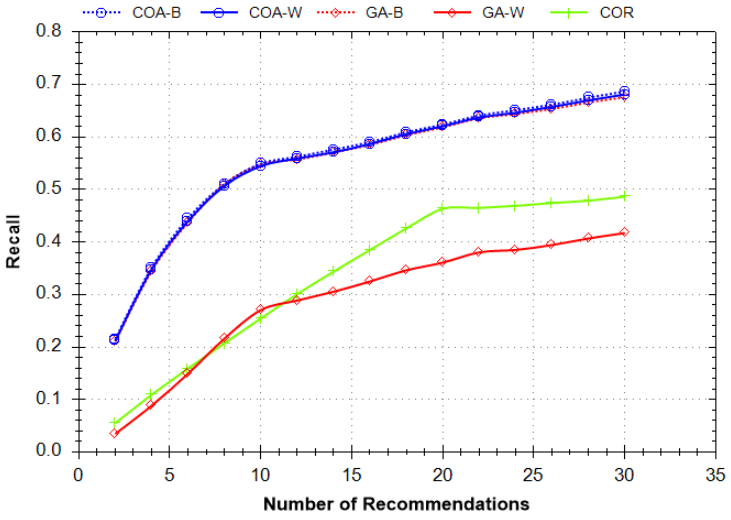
a) MAE for MovieLens 100K



b) Coverage for MovieLens 100K

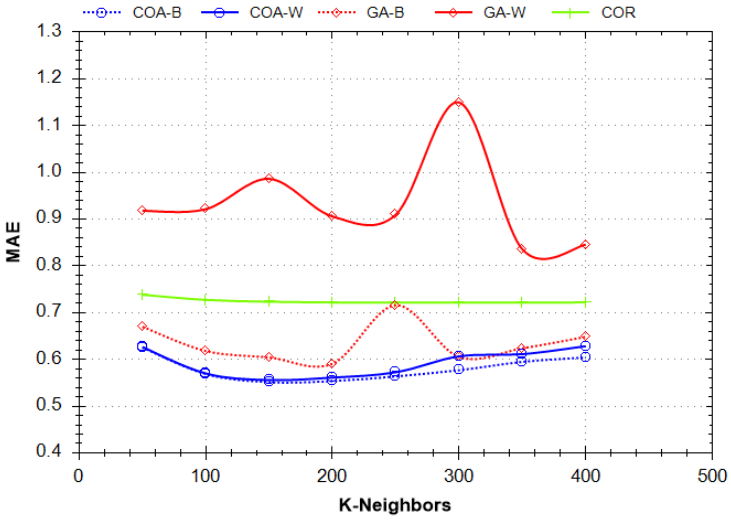


c) Precision for MovieLens 100K

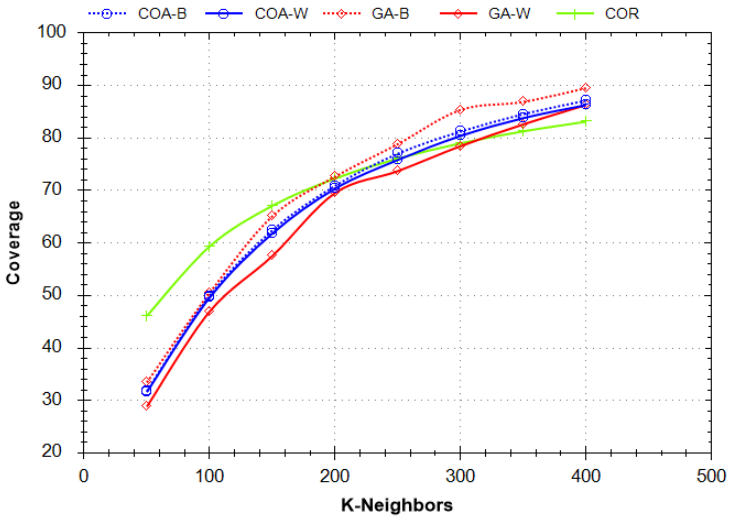


d) Recall for MovieLens 100K

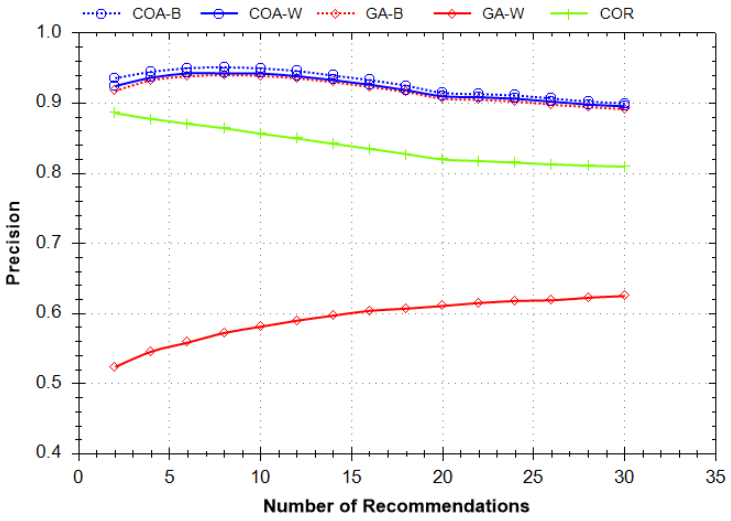
Fig 2: Comparison of results using MovieLens 100K dataset for COA, GA and Pearson Correlation methods



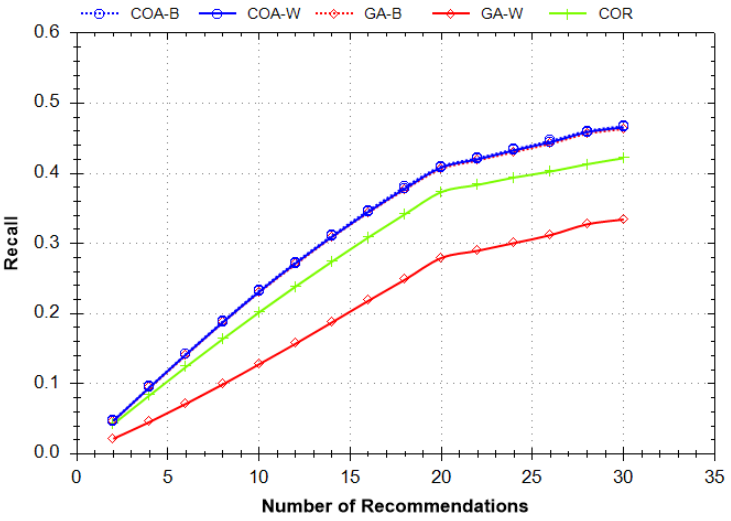
a) MAE for MovieLens 1M



b) Coverage for MovieLens 1M



c) Precision for MovieLens 1M



d) Recall for MovieLens 1M

Fig 3: Comparison of results using MovieLens 1M dataset for COA, GA and Pearson Correlation methods

7. CONCLUSIONS

In this paper we have proposed a new approach to find near optimal similarity function and employed the enhanced prediction formula, which is resulted in better quality of results in shorter time. Our approach reaches to the high quality results 2 times faster than the prior method that is implemented with genetic algorithm.

Evolutionary algorithms produce different results on different runs. In real applications with large amount of data, speed is a crucial factor. Our experiments have shown that the proposed COA method is much more stable than the GA method and give qualified answers in fewer runs and shorter time.

8. REFERENCES

[1] Resnick, P., & Varian, H. R. 1997. Recommender systems. Communications of the ACM, 40(3), 56-58.

[2] Park, D. H., Kim, H. K., Choi, I. Y., & Kim, J. K. 2012. A literature review and classification of recommender systems research. Expert Systems with Applications, 39(11), 10059-10072.

[3] Goldberg, D., Nichols, D., Oki, B. M., & Terry, D. 1992. Using collaborative filtering to weave an information tapestry. Communications of the ACM, 35(12), 61-70.

[4] Burke, R. 2002. Hybrid recommender systems: Survey and experiments. User modeling and user-adapted interaction, 12(4), 331-370.

[5] Bobadilla, J., Ortega, F., Hernando, A., & Alcalá, J. 2011. Improving collaborative filtering recommender system results and performance using genetic algorithms. Knowledge-based systems, 24(8), 1310-1316.

[6] Hatami, M., Pashazadeh, S. 2014. Enhancing prediction in collaborative filtering-based recommender systems. International Journal of Computer Sciences and Engineering, 2(1), 48-51.

[7] Rajabioun, R. 2011. Cuckoo optimization algorithm. Applied Soft Computing, 11(8), 5508-5518.

[8] Bobadilla, J., Ortega, F., Hernando, A., & Gutiérrez, A. 2013. Recommender systems survey. Knowledge-Based Systems.

[9] Bobadilla, J., Serradilla, F., & Bernal, J. 2010. A new collaborative filtering metric that improves the behavior of recommender systems. Knowledge-Based Systems, 23(6), 520-528.

[10] Bobadilla, J., Ortega, F., & Hernando, A. 2012. A collaborative filtering similarity measure based on singularities. Information Processing & Management, 48(2), 204-217.

[11] Bobadilla, J., Hernando, A., Ortega, F., & Bernal, J. 2011. A framework for collaborative filtering recommender systems. Expert Systems with Applications, 38(12), 14609-14623.