Clustering Dependant Recommender Systems

Mohd-Abdul Hameed, Sirandas Ramachandram
Dept. of CSE
University College of Engg
Osmania University
Hyderabad 500007, AP, India
researcher.hameed@gmail.com
schandram@gmail.com

Omar Al-Jadaan
Medical and Health Sciences University
Ras Al-Khaimah
United Arab Emirates
o_jadaan@yahoo.com

***Abstract* - A number of clustering dependant recommender systems (CD_RS) have been consider in this paper including IGCRGA (Information Gain Clustering through Rank Based Genetic Algorithm), IGCEGA (Information Gain Clustering through Elitist Genetic Algorithm), IGCGA (Information Gain Clustering through Genetic Algorithm), IGCN (Information Gain Clustering Neighbor), among others. Non clustering heuristics used in recommender system - namely, entropy and popularity - were also considered for comparison purposes. The two evaluation metric, Mean Absolute error (MAE) and Expected Unity (EU) used in this work show that, firstly, CD_RS out perform their counterparts which used Non clustering heuristics; secondly, IGCRGA emerged vector amongst the CD_RS.**

Keywords* - *expected utility(), mean absolute error, recommendation system, collaborative filtering (CF), popularity, entropy, bisecting k-mean algorithm, genetic algorithm (GA), elitist genetic algorithm (EGA), and rank based genetic algorithm (RGA).

I. INTRODUCTION

Al Mamunur Rashid et al [2] proposed IGCN (Information Gain Clustering Neighbor), a recommender system which use IG (Information Gain) and clustering in a bid to generate recommendations. This coined the beginning of CD_RS (clustering dependant recommender systems).

Mohd Abdul Hameed et al [1], [10], [11] developed this concept to a higher level by proposing and developing a number of CD_RS - namely, IGCGA, IGCEGA and IGCRGA - explored in this paper. Other name coined for these breeds of RS include IG dependant heuristics and Hybrid systems. The latter name was suggested because these systems - briefly explained in section II - use both IG (Information Gain) and CF in the process of providing a recommendation to a user.

Non clustering heuristics used in recommender system - namely, entropy and popularity - were also explored for comparison purposes. These heuristics are briefly explained in section III

These various systems were implemented and experimented upon and the results are discussed in section V of this paper.

II. CLUSTERING DEPENDANT RECOMMENDER SYSTEMS (CD_RS)

The CD_RS explored in this section in chronological order include IGCN, IGCGA, IGCGA, and IGCRGA. Table I shows a comparison of CD_RS along the various dimensions.

TABLE I A COMPARISON OF IG DEPENDANT HEURISTICS

Features	IGCN	IGCGA	IGCEGA	RBGA
Global Minima	Results do not attain	Results attain	Results attain	Results attain
Initialization value	Dependant	Independent	Independent	Independent
Time complexity	Low	High	High	High
Initial / Parent Chromosomes	Not applicable	Not Preserved	Preserved	Preserved

A. *IGCN: Information Gain through Clustered Neighbours*

As an information theoretic measure, one advantage of IGCN over popularity, entropy and its variants is that it takes into account the user's historical ratings and thereby, more adaptive to user's rating history.

IGCN works by repeatedly computing information gain of items, where the necessary ratings data is considered only from those users who match best with the target user's profile [4], [13]. Users are considered to have labels corresponding to the clusters they belong to; and the role of the most informative item is treated as the most useful to the target user in terms of reaching the appropriate representative cluster.

The few assumptions taken into consideration while developing IGCN include:

- Goal of building profiles is to find right neighborhoods
- Neighborhoods correspond to user clusters

IGCN consists of two sessions, namely, the clustering session in which bisecting k-mean algorithm is applied, and the recommendation or profiling session which precedes with evaluation of IG (Information Gain) and is further subdivided into non-personalization and personalization stages. The Non personalization stage is used to address the cold start problem, in other words, new users, while the personalization stage solves problems associated with users whose profiles are known. Algorithm 1 shows the pseudo codes for IGCN.

Algorithm 1: IGCN Algorithm

- Create c user clusters using **bisecting k-mean**
- Compute information gain (IG) of the items
- ***Non-personalized step:***

/* The first few ratings to build an initial profile */

Repeat
- Present next top n items ordered by their IG scores
- Add items the user is able to rate into her profile

Until the user has rated at least i items

- ***Personalized step:***

/* Toward creating a richer profile */

Repeat
- Find best l neighbors based on the profile so far
- Re-compute IG based on the l users' ratings only
- Present next top n items ordered by their IG scores
- Add items the user is able to rate into her profile

Until best l neighbors do not change

B. IGCGA: Information Gain - Clustering through Genetic Algorithm

In normal k-means clustering, centers are randomly selected as initial seeds and clustering proceeds in steps / Phases. In each step, points are reassigned to the nearest cluster. This process has the inherent ability to keep the cluster centers generated in each step to be very close to the initial chosen random centers. As such friend centers of this clustering technique are heavily dependent upon initial choice of centers, which is random. Due to this uncertainty attributed to the random initialization, it is desirable to introduce some heuristic to make sure that the clustering finally reflects optional clusters (as measured by some clustering metric). GA and K-means based GA are such technique introduced to target and optimize the aforementioned fallback.

The searching capability of GAs is used in IGCGA for purposes of appropriately determining a fixed number K of cluster centers in R^N, thereby, appropriately clustering the set of n unlabeled points. The clustering metric that is adopted is the sum of the Euclidean distances of the points from their respective cluster centers. Mathematically, the clustering metric M for the k clusters $C_1, C_2 \ldots C_K$ is given by

$$M(C_1, C_2 \ldots C_K) = \sum_{i=1} \sum_{x_j \in C_i} \| x_j - z_i \|$$

The task of the GA is to search for the appropriate cluster centers $z_1, z_2 \ldots z_k$ such that the clustering metric M is minimized.

In view of this, Abdula Hameed et al [1], proposed IGCGA, which uses GA in the clustering process instead of K-mean bisecting algorithm used in IGCN. K-mean Bisecting Algorithm, as explained above, does not guarantee the attainment of global minima, in other words, the algorithm sometimes locks up in local mimima depending on the initial random centre values chosen. However, on the other hand GA ensues that global minima is attained. The effect of this has produced good result in terms of better recommendation for IGCGA as compared to IGCN. This fact is supported by the two evaluation metrics discussed in section IV

In GA, randomly generated solutions (centers) are populated and in each step of the process, are evaluated for their fitness weight giving greater emphasis to solutions offering greater fitness; and by so doing, there is surety that only good solutions are influenced in the final clusters. Moreover, the crossover and mutation phases ensure production of better solution based on previous solution. Generally, the technique, iterated over several generations, ensures that most of the points in the solution space become randomly selected potential initial centers and are evaluated in the next steps. This leaves no room for any uncertainty raised due to the initial selection. The whole solution space is traversed in search of a potential center, and hence the possibility of ensuring a global maxima is high. This is the main advantage of using GA over normal k-mean algorithm and this benefit has been fully exploited in IGCGA.

The pseudo codes for IGCGA and GA are shown below.

Algorithm 2: IGCGA

- Create c user clusters using **GA**
- Compute information gain (IG) of the items
- ***Non-personalized step:***

/* The first few ratings to build an initial profile */

Repeat
- Present next top n items ordered by their IG scores
- Add items the user is able to rate into her profile

Until the user has rated at least i items

- ***Personalized step:***

/* Toward creating a richer profile */

Repeat
- Find best l neighbors based on the profile so far
- Re-compute IG based on the l users' ratings only
- Present next top n items ordered by their IG scores
- Add items the user is able to rate into her profile

Until best l neighbors do not change

Algorithm 3: GA

1: Generate the Initial Population Randomly.
2: Evaluate the Individuals in the Population and Assign a fitness value to each individual.
3: repeat
4: Selection Operation.
5: Crossover Operation.
6: Mutation Operation.
7: Until Stopping Criteria is met

C. *IGCEGA: Information Gain - Clustering through Elitist Genetic Algorithm*

Problem with information theoretic measures like entropy, popularity, among others, is that they are static by nature, i.e. they are unable to adapt to the rating history of the users. As such, informativeness of items not rated so far is the same for all users of the system regardless of their rating history; however, perfect personalization of a user needs a dynamic algorithm, which has the ability to adapt to continuously changing user rating pattern / style, which lead to better selection of the best neighbor. In IGCN, a previous approach, the computation of IG of each item is repeated for each iteration. Based on previous rating users are clustered using bisecting k-mean approach. In IGCEGA, bisecting k-mean is replaced by EGA to eliminate the local minima sensitivity of k-mean algorithm and focus on global minima. The basic feature of clustering is to group the users such that similarity is maximized in intra cluster and minimized in inter cluster users. Based on the above similarity function, the clusters are regarded as chosen user neighborhood, the required neighbors of a user may join from any cluster.

The formed user clusters are used for profiling, the best approach applied in this situation is ID3 algorithm, which presents results in the form of a decision tree that holds cluster numbers at leaf nodes and each internal node represents a test on an item indicating the possible way the item can be evaluated by the user. The item which holds the maximum IG is taken as the root node. The IG of an item a_t is evaluated using equation below.

$$IG(a_t) = H(C) - \sum_r \frac{C_{at}^r}{|C|} H(C_{at}^r)$$

The basic steps of clustering by EGA are listed in below.

1) String representation:
2) Population initialization:
3) Fitness computation:
4) Selection:
5) Crossover:
6) Mutation:
7) Elitism:
8) Termination criterion

The pseudo code for EGA are shown in Algorithm 5. Both EGA and GA are used for clustering purposes. A comparison between the two shows that EGA has an Elitism stage which GA does not incorporate; and it is this Elitism stage which has given EGA an advantage over GA in that it tends to preserve good solution which would have been lost during mutation stage, the preceding stage. Because of this inherent advantage of EGA, IGCEGA has outperformed IGCGA - which uses GA during clustering - in term of producing better recommendation to the end user.

Algorithm 4 shows the pseudo codes for IGCEGA , which are no different from those of IGCGA and IGCN except in IGCEGA, EGA is used as a clustering algorithm.

Algorithm 4: IGCEGA

1: Create c user clusters using **EGA**
2: Compute information gain (IG) of the items
3: Non-personalized step:
4: **repeat**
5: Present next top n items ordered by their IG value
6: Add items the user is able to rate into his profile
7: **until** the user has rated at least i items
8: Personalized step:
9: /* Toward creating a richer profile */
10: **repeat**
11: Find best l neighbors based on the profile so far
12: Re-compute IG based on the l users' ratings only
13: Present next top n items ordered by their IG values
14: Add items the user is able to rate into his profile
15: **until** best l neighbors do not change.

Algorithm 5: EGA

1: Generate the Initial Population Randomly.
2: Evaluate the Individuals in the Population and Assign a fitness value to each individual.
3: repeat
4: Selection Operation.
5: Crossover Operation.
6: Elitism Operation
7: Mutation Operation.
8: Until Stopping Criteria is met

D. *IGCRGA (Information Gain Clustering through Rank Based Genetic Algorithm)*

IGCRGA is a novel CD_RS for solving personalization problems. In a bid to improve the quality of recommendation of RS and to alleviate the problem associated with personalization heuristics, which use fitness value in the clustering process, Abdula Hameed et al [11] proposed IGCRGA .

IGCRGA using the technique of global minima still resolves the problem associated with IGCN (Information Gain Clustering Neighbor) which sometime traps the algorithm in local clustering centroids. Although this problem was alleviated by both IGCGA (Information Gain Clustering through Genetic Algorithm) and IGCEGA (Information Gain Clustering through Elitist Genetic Algorithm), IGCRGA solves the problem even better and this fact is supported by Mean Absolute Error (MAE) and Expected utility(EU), the two evaluation metric used in this work.

Personalization heuristics (for example IGCGA, IGCEGA, among others), which use fitness value in the selection phase of clustering process, are associated with the problem of lack of diversity in the selection due to biasness evaluated as the absolute difference between two fitness values.

Both RGA and EGA are GA used in clustering and therefore have a lot in common including the stages involved in their algorithms. However the difference between the two is implicit and lies in the selection stage. RGA uses rank based selection while EGA uses proportional selection in reference tofitness value. This difference has transformed into a tremendous boost in terms of better recommendation for IGCRGA.

RGA, the clustering algorithm used in IGCRGA, is a GA therefore the solutions provided by IGCRGA still converge to a global minima and therefore, still solves the problem associated with IGCN whose solution sometime converge to local minima.

The basic steps of clustering by RGA are described in details below:

1) String representation: Each string is a sequence of real numbers representing the K cluster centers. For an N-dimensional space, the length of a string is N*K, where the first N positions (or, genes) represent the N dimensions of the first cluster center, the next N positions represent those of the second cluster center, and so on.
2) Population initialization: K cluster centers encoded in each string are initialized to K randomly chosen points from the dataset. This process is repeated for each of the P chromosomes in the population, where P is the size of the population.
3) Fitness computation: The fitness computation process consists of two phases. In the first phase, the clusters are formed according to the centers encoded in the string under consideration. This is done by assigning each point x_i, i = 1, 2 … n, to one of the clusters C_j with center z_j such that

   $||x_i - z_j|| < ||x_i - z_p||$, where p = 1, 2 …k and j ≠p

   All ties are resolved arbitrarily. After the clustering is done, the cluster centers encoded in the string are replaced by the mean points of the respective clusters. In other words, for cluster C_i, the new center z_i^* is computed as in equation (2). These z_i^* s replace the previous z_i s in the string. Subsequently, the clustering metric M is computed by the follow equation (3)

   $$z_i^* = \frac{1}{n_i} \sum_{x_j \in C_i} x_j, \quad j=1, 2 \ldots k \qquad (2)$$

   $$M = \sum_{i=1}^{k} m_i, \text{ where } m_i = \sum_{x_j \in C_i} \|x_j - z_i\| \qquad (3)$$

   The fitness function is defined as $f = 1/M$, such that maximization of the fitness function leads to the minimization of M.

   The points are sorted in ascending order as per their fitness value, and consequently each is given a rank.
4) Selection: The chromosomes are selected from the mating pool according to their ranks [9]. In the proportional selection method adopted in this paper, a string is assigned a number of copies, which are proportional to their rank in the population, and are sent to the mating pool for further genetic operation. Roulette wheel selection [6], [7] is one common technique that implements the proportional selection method.

TABLE II RGA PARAMETERS

Population Size	Max. Gen	Mutation Prob.	Crossover Prob.
25	100	0.09	0.9

5) Crossover: Crossover is a probabilistic process that exchanges information between two initial chromosomes for generating two resultant chromosomes. In this paper a single point crossover with a fixed crossover probability is used. For chromosomes of length l, a random integer, called the crossover point, is generated in the range [l, l-1]. The pair of chromosomes is broken at the crossover point and the four resultant pieces are interchanged.
6) Mutation: Each string undergoes mutation with a fixed probability. For binary representation of chromosomes, a bit position (or gene) is mutated by simply flipping its value. Since we are considering floating representation in this paper, we use the following mutation. A number in the range [0, 1] is generated with uniform distribution. If the value at a gene position is v, after mutation it becomes

   $$v \pm 2 * \delta * v, \quad v \neq 0 \qquad (4)$$

   $$v \pm 2 * \delta, \quad v = 0 \qquad (5)$$

   The + or - sign occurs with equal probability. Note that mutation can be implemented as:

   $$v \pm \delta * v \qquad (6)$$

   However, one problem with this form is that if the values at a particular position in all the chromosomes of a population become positive (or negative), then it is impossible to generate a new string having a negative (or positive) value at that position. In order to overcome this limitation, a factor of 2 is incorporated while implementing mutation. Other forms like

   $$v \pm (\delta * E) * v \qquad (8)$$

   where $0 < \varepsilon < 1$ would also have been satisfied. One may note in this context that similar sort of mutation operators for real encoding have been used mostly in the realm of evolutionary strategies [8], Chapter 8. The remaining parameter values are listed in table 2.
7) Elitism: Elitism, a new operation, has been added to improve the quality of GA results and guarantee the convergence to global solution, where the good

solutions are not lost during the other genetic operations. In this phase, the two populations, parent and children population, are put together, then sorted based on their fitness and the best N solutions are selected and incorporated in the new generation, where N is the population size.

8) Termination criterion: In this phase, the processes of fitness computation, selection, crossover, and mutation are executed for a maximum number of iterations. The best strings identified up to the last generation provides the solution to the clustering problem.

The pseudo codes for IGCRGA and RGA are shown in Algorithm 4 and 5 respectively.

Algorithm 4: IGCRGA

1: Create c user clusters using **RGA**
2: Compute information gain (IG) of the items
3: Non-personalized step:
4: **repeat**
5: Present next top n items ordered by their IG value
6: Add items the user is able to rate into his profile
7: **until** the user has rated at least i items
8: Personalized step:
9: /* Toward creating a richer profile */
10: **repeat**
11: Find best l neighbors based on the profile so far
12: Re-compute IG based on the l users' ratings only
13: Present next top n items ordered by their IG values
14: Add items the user is able to rate into his profile
15: **until** best l neighbors do not change.

Algorithm 5: RGA

1: Generate the Initial Population Randomly.
2: Evaluate the Individuals in the Population and Assign a fitness value to each individual.
3: repeat
4: Selection Operation.
5: Crossover Operation.
6: Elitism Operation
7: Mutation Operation.
8: Until Stopping Criteria is met

III. NON CLUSTERING HEURISTICS USED IN RECOMMENDER SYSTEM

Two heuristics are considered in this category of non CD_RS, namely popularity and entropy.

A. Popularity

Popularity of an item indicates how frequently users rated the item. Popular items may be good at connecting people with each other as co-raters, since many people are likely to rate popular items. [2]

One disadvantage of using Popularity measure to elicit preferences, as pointed out by [2], is the possibility of worsening the prefix bias - that is, popular items garnering even more evaluations. Unpopular items, lacking enough user opinions, may be hard to recommend. This situation would not improve if the system keeps asking opinions on popular items.

B. Entropy

Entropy of an item represents the dispersion of opinions of users on the item. Considering a discrete rating category, entropy of an item a_t, is given by equation: $H(a_t) = -\sum_{i=1}^{n} pi \log_2 pi$

where p_i denotes the fraction of a_t's ratings that is equal to i. Notably, the logarithm to base 2 is used because entropy is a measure of the expected encoding length expressed in bits.

One limitation of entropy is that it often selects very obscure items leading to senseless information on items which are rated by a very small number of people, in which situation the rating frequencies or popularities cannot be inferred.

In general, it is not possible to infer the rating frequencies or popularities of items from their entropy scores. A plot (graph) between entropy and popularity (rating frequency, to be exact) of items, shows that entropy and popularity are only slightly correlated (correlation coefficient is only 0.13) [2].

A few other researchers who employed entropy as a measure for informativeness on other domains also report mixed results. For example, in their work on using information theoretic measures such as entropy to find informative patterns in data, Al Mamunur et al, observed that in addition to picking informative patterns, entropy suggests "garbage" (meaningless or not useful) patterns to be useful as well [3] .

Other variants of entropy, such as Entropy0, are not discussed and considered in this work.

IV. EXPERIMENTATION

For purposes of experimentation, the data-set obtained from Movie Lens database was used in this work. The base table - containing 100,000 records and 3 attributes namely user_Id, movie_id and rating - was used. The System interacting with this dataset and applying the heuristics discussed in section II and III above was implemented through Java JDK 6, a java programming language flavour.

The MAE (Mean Absolute Error) and EU (Expected Utility) - used as evaluation metrics - corresponding to each heuristic and to the size of the recommendation was computed and tabulated as shown in tables 3 and 4 respectively. The result of the tables is represented in graph form in figure 1and II.

Mean Absolute Error (MAE), a widely used metric in the CF domain, is the metric used for evaluating the quality of the recommendation. The formula below is used for computing MAE.

MAE=$\frac{1}{n}\sum_{i=1}^{n}|fi - yi|$ where n represents the number of movies recommended, f_i - expected value of rating and

TABLE III MEAN ABSOLUTE ERROR (MAE) FOR THE PERSONALISATION HEURISTICS

Sample size	15	30	45	60	75	Mean
Popularity	0.8762	0.8039	0.7813	0.7075	0.6842	0.7706
Entropy	0.6945	0.6798	0.6703	0.6539	0.6320	0.6661
IGCN	0.7561	0.7133	0.6856	0.6598	0.6341	0.6898
IGCGA	0.6811	0.6492	0.6174	0.6014	0.5713	0.6241
IGCEGA	0.6883	0.6447	0.6121	0.5972	0.5684	0.6221
IGCRGA	0.6708	0.6128	0.5812	0.5725	0.5547	0.5984

y_i - actual recommendation generated. In other words, MAE is a measure of deviation between the recommendation and the corresponding actual ratings.

However, a limitation of MAE is that it only considers absolute differences. MAE sees no difference between two pairs of (actual rating, recommendation) for a movie that are rated (1, 5) and (5, 1), although users may be more unhappy about the former pair. Therefore, another accuracy metric, Expected Utility [14] is used that tries to penalize false positives more than false negatives.

The formula used for evaluating EU is given below:

$$EU = \sum_{1}^{i} \sum_{1}^{j} U(Ri,,Rj)*P(Ri|Rj)$$

where $U(Ri,,Rj) = Rj - 2*|Ri - Rj|$. and $P(Ri|Rj)$ is probability of occurrence estimated using an m-estimate Cestnik [15] smoothing. The m-estimate can be expressed as following:

$$p = \frac{r + m * P_i}{n + m}$$

where n is the total number of examples, r is the number of times the event for which the probability is estimated occurs, m is a constant, and P_i is the prior probability.

V. RESULTS AND DISCUSSION

In reference to EU results, CD_RS have a competitive advantage over non CD_RS in terms of producing better quality of recommendation. This suggests that clustering plays a major role in improving the quality of recommendation.

The result generated by the experimentation (figures 3 and 4) show that IGCRGA provides the best recommendation; and this fact is supported by MAE and EU, the two evaluation metrics used in this work.

There is a remarkable improvement in terms of quality of recommendation between IGCRGA and the pervious personalization heuristics, especially ICGN the base CD_RS; that is to say, in terms of EU there is an improvement of 17%.

TABLE IV EXPECTED UTILITY (EU) FOR THE PERSONALISATION HEURISTICS

Sample size	15	30	45	60	75	Mean
Popularity	0.1768	2.8901	3.5518	4.3977	5.2303	3.2493
Entropy	4.9265	5.3309	5.5086	5.7813	6.2943	4.5095
IGCN	3.2010	5.4128	6.0115	6.2548	6.5219	5.4804
IGCGA	5.2243	6.0410	6.1954	6.5949	6.9610	6.2033
IGCEGA	5.2924	6.0089	6.3072	6.5574	7.0868	6.2505
IGCRGA	5.8993	6.5249	6.7113	6.9018	7.0107	6.6096

Although there is a slight overhead in terms of evaluating the ranks by the sorting algorithm, IGCRGA resolves the problem of biasness and lack of diversity by using rank as a selection criterion in the selection stage / phase of the clustering session; this in turn translates into better recommendation. This points to the fact that whenever the quality of clustering is improved so is the quality of recommendation.

It is worthwhile noting that the quality of recommendation is directly proportional to the size of recommendation and this fact is supported by the two evaluation metrics and is true for all the recommendation heuristics explored in this work.

Though entropy has a lower MAE as compared to IGCN, its performance is undesirable in dynamic environment, that is to say, in environment where the historical data or the rating pattern / style of a visitor / user is continuously changing (dynamic). In contrary, Entropy works well in static environment, (in other words, where the historical data of the visitor is not changing (static)). Notably, lower MAE does not translate into better quality of recommendation. Still, IGCN produces better recommendation than entropy especially when the size of recommendation is slightly large (precisely, greater than 30); and this fact is supported by EU.

VI. CONCLUSION

Clustering has a bearing to the quality of recommendation, and therefore Improving the quality of clustering plays a major role in improving the quality of recommendation.

In reference to cold start and non cold start problems, IGCRGA produces the best result in terms of generating the best recommendation for both static and dynamic environments.

Generally, the personalization heuristic investigated show that the quality of recommendation improves with increase in recommendation size.

IGCRGA can be applied in any type of personalization problem, let it be web or non web personalization.

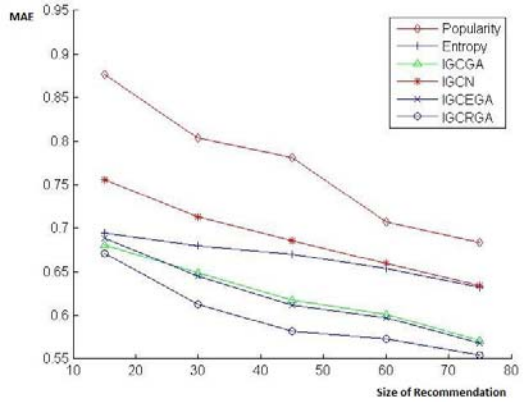

Figure 1 Graphic Representation of MAE Vs Size of Recommendation for the Personalization Heuristics

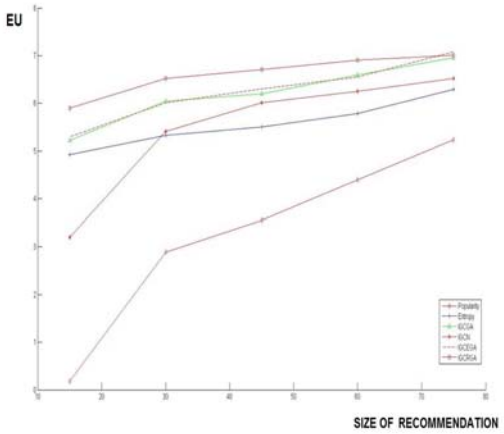

Figure 1I Graphic Representation of EU Vs Size of Recommendation for the Personalization Heuristics

REFERENCES

[1] Mohd Abdul Hameed, OmarAl Jadaan, and S. Ramachandaram, "Information theorectic aproach to cold start problem using Genetic Algorithm", IEEE, 2010, ISBN: 978-0-7695-4254-6.

[2] Al Mamunur Rashid, Gerge Karypis and John Riedl, "Learning Preferences of new users in Recommender System: An Information Approach.", SIGKDD Workshop on Web Mining and Web Usage Analysis (WEBKDD) , 2008.

[3] Al Mamunur Rashid, Istvan albert, Dan Cosley, Shyong K. Lam, Sean MCNee, Joseph A. Konstan, and John Riedl, "Learn New User Preferences in Recomender Systems, 2002 international Conference on Intelliganent User interfaces. pp. 127-134.

[4] Isabelle Guyon, Nada Matic, and Vladimir Vapnik, "Discovering Informative Patterns and data cleaning" , 1996.

[5] Mitchell Thomas M., "Machine Learning", McGraw-Hill Higher Education, 1997.

[6] Omar Al Jadaan, Lakshmi Rajamani, and C. R. Rao, "Improved selection operator for Genetic Algorithm", Journal of Theoretical and Applied Information Technology, Vol. 4, No. 4, pp. 269-277, 2008.

[7] Omar Al Jadaan, Lakshmi Rajamani, and C. R. Rao, "Parametric study to enhance genetic algorithm performance, using ranked based Roulette Wheel Selection method", International Conference on Multidisciplinary Information Sciences and Technology (InSciT2006), volume 2, pp. 274-278, Merida, Spain, 2006.

[8] Daniel Billsus and Michael J. Pazzani, "Learning collaborative information filters", Proc. 15th International Conference on Machine Learning, pages web personalization problem 46 -54. Morgan Aufmann, San Francisco, CA, 1998.

[9] Omar Al Jadaan, Lakishimi Rajamani, and C. R. Rao, "Improved selection operation for Genetic Algorithm", Journal of Theoretical and Applied information Technology.

[10] Mohd Abdul Hameed, S. Ramachandram, and Omar Al Jadaan, " IGCEGA, an acronym for Information Gain Clustering through Elitizt Genetic Algorithm", 2011 International Conference on Communication Systems and Network Technologie, 2011

[11] Mohd Abdul Hameed, S. Ramachandram, and Omar Al Jadaan, "IGCRGA: A Novel Heuristic Approach for Personalization of Cold Start Problem", 2011 Fifth Asia Modelling Symposium ,2011

[13] Thomas M. Mitchell. Machine Learning. McGraw- Hill Higher Education, 1997.

[14] Al Mamunur Rashid, Shyong K. Lam, George Karypis, and John Riedl. Clustknn: A highly scalable hybrid model- & memory-based cf algorithm. WEBKDD 2006: Web Mining and Web Usage Analysis, 2006.

[15] Cestnik, B. 1990. "Estimating probabilities: A crucial task in machine learning." Proc. Ninth European Conference on Artificial Intelligence. 147_149.