Using Genetic Algorithm for Hybrid Modes of Collaborative Filtering in Online Recommenders

Simon Fong, Yvonne Ho, Yang Hang
Faculty of Science and Technology, University of Macau, Macau SAR
ccfong@umac.mo

Abstract

Online recommenders are usually referred to those used in e-Commerce websites for suggesting a product or service out of many choices. The core technology implemented behind this type of recommenders includes content analysis, collaborative filtering and some hybrid variants. Since they all have certain strengths and limitations, combining them may be a promising solution provided there is a way of overcoming a large amount of input variables especially from combining different techniques. Genetic algorithm (GA) is an ideal optimization search function, for finding a best recommendation out of a large population of variables. In this paper we presented a GA-based approach for supporting combined modes of collaborative filtering. In particular, we show that how the input variables can be coded into GA chromosomes in various modes. Insights of how GA can be used in recommenders are derived through our experiments with the input data taken from Movielens and IMDB.

1. Introduction

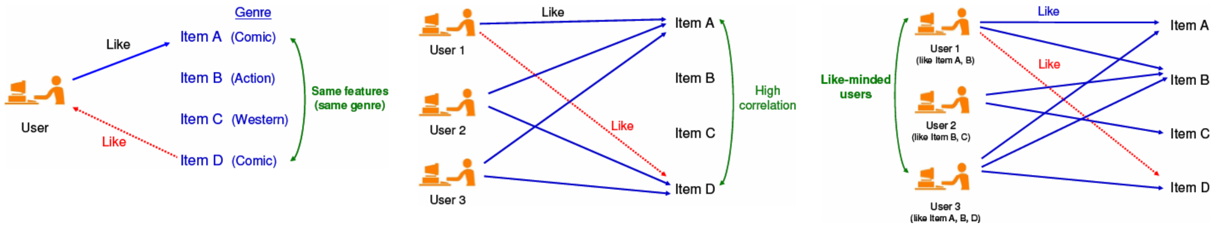
Recommender systems have been widely adopted by e-Commerce websites to suggest products or services to customers. In general, they assist users to narrow down their choices for making a purchase decision from a large pool of items. One simple way of offering recommendation is based on the top selling products. This however does not differentiate customers who may have different tastes. The other approach is known as one-to-one marketing, which takes into account the information about a particular user and tries to find a personal match of product that is predicted to best suit his or her flavor. Generally, there are three methods of the recommendation systems for one-to-one marketing: Content-based Filtering, Collaborative Filtering and Hybrid Model.

Content-based Filtering selects items based on the correlation between the content of the items and user preferences. Current search engines are based on automatic analysis of the content of documents and the content of user's query. As an example shown in Figure 1a, that is a movie recommendation application, in order to recommend movies to a user, the content-based recommender system tries to understand the commonalities among the movies that the user has rated highly in the past (specific actors, directors, genres, subject matter, etc.). Movie item D is hence recommended because it is has a certain high degree of similarity to Item A by its content.

For another example, a Personalized Recommender System [1] creates dynamic hyperlinks on a web site that contains a collection of advises about do it yourself home improvement. The advices are recommended based on the similarity that they have, to the ones the user has highly rated in the past. In addition to hypertext content, the same concept on content analysis was extended in [2] for recommending multimedia products such as mp3 music on a recommender website.

One common problem with the Content-based recommendation system is that it can only recommend items scoring highly against the user profile; so the user is restricted to see the items similar to those already rated, new items will never be recommended due to the working on an individual user.

Collaborative Filtering (CF) based on the similarity between currently active user and other users, finds new items the active has never seen before but they were guessed to be interested by him because the other users who have similar interest to his have seen/liked. The similarity can either be measured by the same item which known as item-based CF or by the same type of user, known as user-based CF. This method is to suggest new items or to predict the utility of a certain item for a particular user based on his previous likings or the opinions of other like-minded users.



(a) Content-based Filtering (b) Item-based Collaborative Filtering (c) User-based Collaborative Filtering

Figure 1. Examples of different Filtering

With item-based collaborative filtering as shown in Figure 1b, if many users (User2 and User3 in this example) like Item A and Item D, we assume that Item A is highly correlated with Item D. If User1 likes Item A, he should also like Item D given the strong link of association between Items A and D.

Item based Collaborative Filtering is quite common for current recommendation systems, which have been wildly used by Movie Lens and Yahoo. According to papers [3][4], item-based techniques first analyze a user-item matrix to identify relationships between different items, and then use these relationships to indirectly compute recommendations for users.

With user-based collaborative filtering as shown in Figure 1c, the taste of User3 is very similar to that of User1 because they have preferred Item A and Item B in common. They are deemed to be like-minded users. Hence User 3 likes Item D so will User1.

Furthermore, it was argued in [5] that in real life the way in which two people are said to be similar is not based solely on whether they have complimentary opinions on a specific subject, e.g., movie ratings, but also on other factors, such as their background and lifestyles. Therefore, when doing the profile matching, issues such as age, gender and preferences of movie genres must also be taken into account.

A novel framework for user-based collaborative filtering is proposed in [6] that enables recommendation by groups of closely related individuals. By using the rating information from a group of closely related users, unrated items of the individual user in a group can be predicted.

1.1. Hybrid Model

Several recommendation systems (e.g. [7][8]) use a hybrid approach by combining collaborative and content-based methods, which helps to avoid certain limitations of content-based and collaborative systems. Different ways to combine collaborative and content-based methods into a hybrid recommender system can be classified as follows:

1. Implementing collaborative and content-based methods separately and combining their predictions,
2. Incorporating some content-based characteristics into a collaborative approach,
3. Incorporating some collaborative characteristics into a content-based approach, and
4. Constructing a general unifying model that incorporates both content-based and collaborative characteristics.

One thing in common across the above four fusion approaches is that a large amount of input data is required. Combining several filtering approaches implies more than one aspect of input data might need to be incorporated into the recommender system, such as the users' information, item contents and users' ratings. A table summarizing what input data is required for the filtering approach to work on is shown below.

Table 1: Input data required by different CF

Filtering approach	Input data
Content-based analysis	Item contents + User ratings (by a single user)
Item-based CF	Item contents + User ratings (by multiple users)
User-based CF	User demographic information + User ratings (by multiple users) +
Hybrid CF (e.g. GA-based CF)	User demographic information + Item contents + User ratings (by multiple users)

During filtering, the input data are processed as 'features' by some heuristic algorithms. In some hybrid recommendation systems, all features are included. But not all features contribute significantly to the quality of recommendation output; some are noises that bring down the prediction accuracy. Moreover, every user places a different importance priority on each feature. These priorities are enumerated as referred to feature weights. For example, if a mature user prefers to be given recommendations based on the opinions of other mature users, then his feature weight for age would be higher than other features.

1.2. Research Motivation

The main objective of this research work is to design a recommender that exploits advantages of hybrid CF for high quality prediction and recommendation, at the same time overcoming the limitations inherited from hybrid CF techniques.

Genetic algorithm-based approach is proposed here for it is one of the most powerful search techniques for finding an optimized solution of a matching item to be recommended to the user. In order to support an extensive range of input data required for Hybrid CF (c.f. Table 1), the GA approach allows the data to be encoded as 'features' into relatively elongated chromosomes.

With the chromosomes ready, GA searches for a solution that has the highest value of fitness function (to be defined later). The feature weights which are relevant remain, noises are eliminated. During the early period of usage, there wasn't enough rating data which is known as the data sparsity problem [9]. We define a way of solving this problem in our GA recommender by coding an additional 'preference' component in the chromosome. So during the cold boot-up stage, the preference component will dominate the search function in the GA operation.

2. GA-based Collaborative Filtering

GA is a heuristic search method, based on the mechanics of natural selection and genetic evolution, introduced by John Holland in 1975 [10]. It maintains a population of computing feature transformation matrices. By using selection, crossover and mutation methods of GA, it finds the fitness value for picking the fittest individuals. A collection of individuals are represented by chromosomes which are coded in numeric real-value.

2.1. Chromosome Encoding

In order to implement a recommendation system as a case study of Movie recommender, we apply Genetic Algorithm to find the fine-tuned feature weights so that each feature used in CF can be characterized; the chromosome structure is presented as in Figure 2.

This GA chromosome structure is designed in mind that it could embrace the full range of input data from Table 1 for supporting hybrid CF. Optionally the chromosome structure can be downsized to leave out some data components such as user demographic data or item contents, in cases of Item-based CF or User-based CF were in use respectively.

Feature 1 is the movie rating. Features 2-4 represent the user profile from MovieLens (source: movielens.umn.edu); age, gender and occupation are the most influential demographic attributes that describe the background characteristic of a user. Features 5-12 represent the additional profile features calculated in our work; they are the extra component added into the chromosome. So they can be used in the initial period of usage when the movie ratings are scarce. Features 13-30 represent the movie genres from MovieLens; features 3-37 represent the movie attributes from IMDb (source: imdb.com). They are the component of item content that describes about the movie. Both data from MovieLens and IMDb are used in order to prevent any bias in any set of the data, for a fairer evaluation in our experiments.

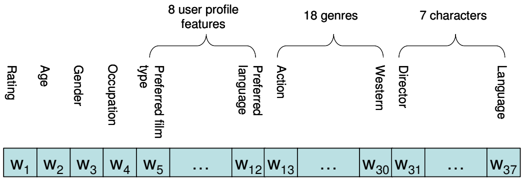


Figure 2. Chromosome of full features

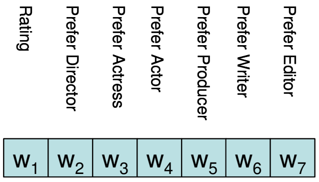


Figure 6. Chromosome of 7 features

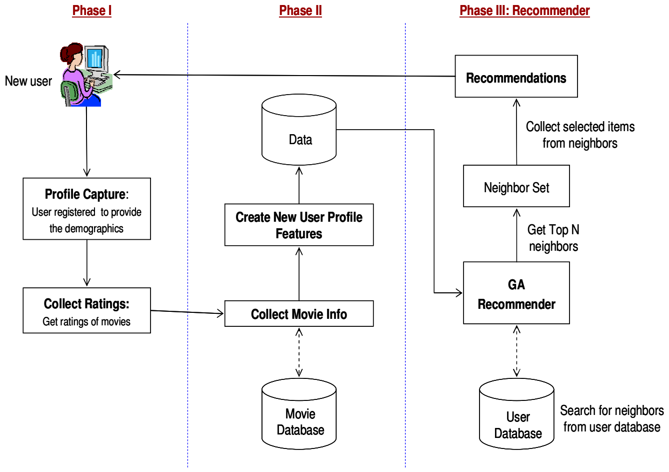

Phase I
Phase II
Phase III: Recommender
New user
Recommendations
Data
Collect selected items from neighbors
Neighbor Set
Profile Capture: User registered to provide the demographics
Create New User Profile Features
Get Top N neighbors
GA Recommender
Collect Ratings: Get ratings of movies
Collect Movie Info
Movie Database
User Database
Search for neighbors from user database

Figure 3. GA-based Recommendation System

2.2. GA Recommender System Architecture

We constructed a GA-based Recommendation System as a prototype for conducting experiments. The experimental protocol is capable of gathering, disseminating, and using ratings from some users to predict other users' interest in movies. Figure 3 shows the architecture of our system that can be deployed as an online hybrid CF application implemented in GA algorithm. The process is divided into three phases:

- ***Phase I: Collect User Information***

For a new visitor, the system requests him to register an account as to collect his personal particulars as well as the ratings of a set of movies. Each time the user has watched or purchased a movie, he would be asked to rate how good it is.

- ***Phase II: Create User Profile features***

For those movies the user rated, we search for the corresponding genres and characters from the movie database. In addition to using users' ratings, we try to capture what types of items are most preferred by the user. In the case of movie recommendation, the preferred types of items are the movie genres such as the following, *Preferred Film Type, Preferred Director, Preferred Actress, Preferred Actor, Preferred Producer, Preferred Writer, Preferred Editor and Preferred Language.* They form the additional 8 features to be stored in the user profile (c.f. Figure 2), which will be coded into the GA chromosome.

Instead of relying on users' ratings alone, an explicit abstract of what the user prefers often offers a good feature weight that helps in the recommendation search process. This is particularly effective when the rating records were in scarce during the early stage.

This is how it works on extracting the 8 features: with respect to each user, we sum up the total number (frequency) appears on each genre, the one with maximum number is assumed to be most favoured by the user and marked as *Prefer_FilmType*. If the same number of genres exists, we would get the latest one as we suppose that the last record reflects the latest user preference. A flowchart that shows how *Prefer_FilmType* is computed is shown in Figure 4. The same logic applies to the other features.

For example, in a user's record, the most frequent type is Musical; then we assume that the preferred film type of this user is Musical. In another example, if most of the movies that the user watched are found to be directed by 'Steven Spielberg', then we assume the preferred director of the user is 'Steven Spielberg'.

This concept of extracting user preferences from the majority of items that he frequented, should be generic enough to apply on other recommendation domains than movie advisor.

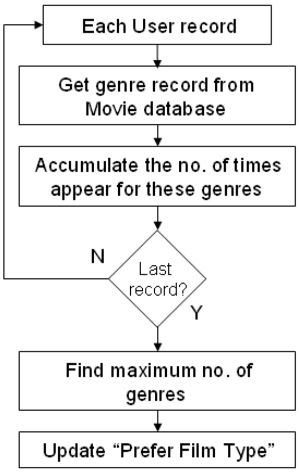

Each User record
Get genre record from Movie database
Accumulate the no. of times appear for these genres
N
Last record?
Y
Find maximum no. of genres
Update "Prefer Film Type"

Figure 4. Flow chart of creating a feature

- ***Phase III: GA Recommender***

This phase uses GA functions to search for the appropriate recommendation by selecting and weighing the features. Figure 5 shows the workflow of the recommender that the best subset can be gained from the full data set by processing *Feature Selection*, *Feature Weighting* and *Recommendation Generation*.

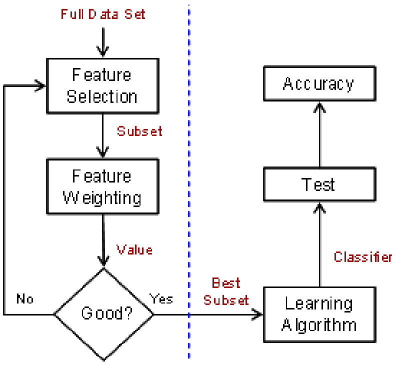


Figure 5. Workflow of GA Recommender

Feature selection is the process that chooses an optimal subset of features according to a certain criterion. As there are thousands of features in the database, selection can reduce the dimensionality, eliminate noise and save search time.

The first step in the GA Recommender phase is to prepare the features that are needed. For hybrid GA-based CF, all the 37 features should be included. However, from our evaluation experiments, we found that some of the user profile attributes are more effective than the others. Good quality of recommendation can be produced by using a minimum 12 features out of 37: *Rating, Age, Gender, Occupation, Prefer FilmType, Prefer Director, Prefer Actress, Prefer*

Actor, Prefer Producer, Prefer Writer, Prefer Editor and Prefer Language.

Feature weighting is the next step after feature selection. Each gene in the chromosome is represented by a feature weight *w* in real value. The heavier the weight the more important the feature is, so that the value can reflect the feature importance to the user. For example, the weight of the feature *Prefer Director* is the highest, that indicates the user favors over his choice of movies by certain movie directors. We programmed in GALib (lancet.mit.edu/ga) to find out the feature weight *w,* distance measure *d* and obtain a group of neighbor set for the active user by choosing half of the top scores from the users of similar taste.

After we obtained the neighbor set, we can provide the active user a list of recommendations by summarizing the neighbor's rated movies which have not been rated by the active user. Also we can predict the votes of those movies. Predicting of the votes helps guiding the user to choose his favorites by ranking the votes. The vote for movie *i* for active user *a* can be predicted by:

$$predict\_vote(a,i) = \bar{v}_a + k\sum_{j=1}^{n} euclidean(a,j)(v_{j,i} - \bar{v}_j)$$

where: $\bar{v}_a$ is the mean vote of active user *a*

k is the normalizing factor

n is the size of neighbor set

$v_{j,i}$ is the actual vote of neighbor *j* on movie *i*

As we can also measure the predict vote for those movies that the active user rated before, we can compare it with the actual vote to cross-check the fitness rate for user *a* on movie *i*:

$$fitness(a,i) = \left| \frac{predict\_vote(a,i) \times 100\%}{actual\_vote(a,i)} \right|$$

2.3. Similarity Measure

For obtaining the neighbor set of the active user, similarity measure is used as to reflect the affinity between users. Firstly, we collect those users who shared common movies with the active user. Then the distance measure d and feature weight w between the active user and his neighbors can be found by the Euclidean Distance function:

$$d(a,j) = \sqrt{\sum_{i=1}^{z}\sum_{f=1}^{n} w_f\left(v_{a,i,f} - v_{j,i,f}\right)^2}$$

where

a is the active user, *j* is the neighbor, and $a \neq j$

i is the common movie between *a* and *j*

z is the total number of common movies

f is the feature number

n is the total number of features

w_f is the weight of feature *f* for user *a*

$v_{a,i,f}$ is the value of feature *f* on movie *i* for user *a*.

Sum of the weights should be $W = \sum_{i=1}^{n} w_i = 1$ where *n* represents the total number of features; *W* represents the sum of all the weights in a chromosome. As the calculation goes, infeasible chromosomes will appear. To solve this problem, we applied the *Repair Algorithm* by Okan Yilmaz [11].

3. Experiments

We want to evaluate the effectiveness of our GA-based recommender, by comparing the traditional Collaborative Filtering method using Pearson Coefficient and our proposed hybrid schemes using Genetic Algorithm. We repeated the experiments by using different sets of features as to show how and which feature weights affect the fitness and performance.

3.1. Experiment I - Features Weights

We calculated the weights for each feature, and took 10 users to test the importance of features to each user. In Figure 7, we show the average weights for each feature on different users. The average feature weights are about 0.2649. The separation of two groups of lines is evident that the weights of user profile features are much higher than the other movie attributes. Especially the features *prefer director, prefer actress, prefer actor, prefer producer, prefer writer, prefer editor;* all of their feature weights are over 0.3. By this observation, we confirm that *rating, age, gender, occupation, prefer film type, prefer director, prefer actress, prefer actor, prefer producer, prefer writer, prefer editor* are more relevant to the user preference than the other features.

3.2. Experiment II - Fitness Accuracy

For testing the fitness accuracy of Pearson Algorithm and Genetic Algorithm for CF, and also the effectiveness of different features on GA, we configured a variety of five different types of chromosomes in this experiment and applied them on 50 random users respectively. The chromosome feature compositions of the five types are shown in Table 2. The different combinations represent on how different types of filtering as shown in Table 1 can be made up by including various input data (such as Item Contents, User Ratings and User Demographics).

One sample chromosome with combination of 7 features that is constructed by preprocessing the user profile data pertaining to the movie attributes is shown in Figure 6.

From Figure 8, we can observe apparently that the accuracy of prediction of GA is much higher than that of Pearson. The average fitness of Pearson is about 69.2%, whereas the average fitness for GA with various combinations of features ranges from 81.28% to 81.77%. In particular, the fitness of 'GA UserPro12' is the highest, about 18.21% better than that of Pearson.

Figure 9 shows the processing time for running the GA algorithm on different features. 'GA UserPro7' that is GA with seven features out-performs the other four in terms of speed. Its average processing time is 19 seconds. This is a 67.53% reduction over GA with 22 features which implements Content Analysis.

The longest time taken is by 'GA UserPro37' and the shortest time is 'GA UserPro7'. This reinforces the belief that when more features are integrated into the recommender a longer processing time it takes.

3.3. Experiment III - Neighbor Set

We tested the performance on different group sizes of neighbor set, from 10 to 100 respectively.

3.3.1. Process time vs. Neighbor Set. Figure 10 shows the performance of different features running on GA with different sizes of neighbor sets. At the beginning, their performances are close. As the neighbor set size increases, the processing times for 'GA', 'GA Merge', 'GA UserPro37' increase quite sharply. The additional features they have in common are the 18 movie genres.

As we can see, 'GA UserPro37' has 37 features, the process time increases gradually as the neighbor set expands; whereas for 'GA UserPro7' with 7 features, the process time increases slowly.

3.3.2. Fitness vs. Neighbor Set. In Figure 11, the fitness of different features rise up gradually as the neighbor set size expands. Interestingly when the neighbor set reaches over the size of 35, the fitness continues to stay constant. As indicated by the dotted line in the chart, the fitness approaches at approximately 95% at the turning point. Further increase on the neighbor set size has no effect.

4. Conclusion

We proposed and evaluated using GA for supporting hybrid CF to recommend new items to a particular user based on his previous likings or the opinions of other like-minded users. GA-based CF provides reasonably good recommendation accuracy. The performance can be further improved by incorporating user profile features in the chromosomes. As our experiments show, GA offers more accurate recommendation than that of Pearson Algorithm. By applying user profile features that are more significant than other features such as movie genres on GA, the similarity measure finds the neighbors with similar taste to the user; as a result, the user preference can be better predicted. And when user profile features are used alone, the process speeds up. Our experiment also shows the GA fitness keeps constant when neighbor set size increases. This implies a positive prospect in the scalability and speed issues of GA recommender.

5. References

[1] R. Meteren, M. Someren, "Using Content-based Filtering for Recommendation", *Proceedings of MLnet / ECML2000 Workshop*, 2000

[2] K. Iwahama, Y. Hijikata, S. Nishida, "Content-based Filtering System for Music Data", *Proceedings of the 2004 International Symposium on Applications and the Internet Workshops*, pp. 480 - 487, 2004

[3] B. Sarwar, G. Karypis, J. Konston and J. Riedl, "Item-Based Collaborative Filtering Recommendation Algorithms", *Proceedings of the 10th International Conference on World Wide Web*, pp. 285 - 295, 2001

[4] M. Deshpande, G. Karypis, "Item-Based Top-N Recommendation Algorithms", *ACM Transactions on Information Systems*, Volume 22, Issue 1 (January 2004), pp. 143 – 177, 2004.

[5] S. Ujjin, J. P. Bentley, "Particle swarm optimization recommender system", *IEEE International conference on Evolutionary Computation*, pp. 124-131, 2003

[6] G. Xue, C. Lin, Q. Yang, W. Xi, H. Zeng, Y. Yu and Z. Chen, "Scalable Collaborative Filtering Using Cluster-based Smoothing", *International ACM SIGIR Conference on Research and Development in Information Retrieval SIGIR '05*, pp. 114 – 121, 2005

[7] Y. Shih, D. Liu, "Hybrid Recommendation Approaches: Collaborative Filtering via Valuable Content Information", *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, Track 8 - Volume 08, pp. 217-220, 2005

[8] M. Claypool, A. Gokhale, T. Miranda, "Combining Content-Based and Collaborative Filters in an Online Newspaper", *ACM SIGIR Workshop on Recommender Systems*, August 19, 1999, USA

[9] E. Han, G. Karypis, "Feature-Based Recommendation System", *In Proceeding of ACM CIKM'05*, pp. 446-452, October 2005, Bremen, Germany

[10] S. Ujjin, P. J. Bentley, "Learning User Preferences Using Evolution", *4th Asia-Pacific Conference on simulated Evolution and Learning*, Singapore.

[11] O. Yilmaz, L. Tuaf, "Data Mining Feature Subset Weighting and Selection using Genetic Algorithms", Thesis, Air Force Institute of Technology, pp. 55-59

Table 2. Compositions of chromosome types

Chromosome Type	Chromosome Features				
	User Rating	User Demographics (3 features)	User Preferences (8 features)	Item Contents	
				18 genres from MovieLens	7 characters from IMDb
Pearson Algorithm (Pearson)	✓	✓		✓	✓
Genetic Algorithm with 7 features (GA User Profile 7)	✓		✓		
Genetic Algorithm with 12 features (GA User Profile 12)	✓	✓	✓		
Genetic Algorithm with 22 features (GA)	✓			✓	
Genetic Algorithm with 29 features (GA Merge)	✓	✓		✓	✓
Genetic Algorithm with 37 features (GA User Profile 37)	✓	✓	✓	✓	✓

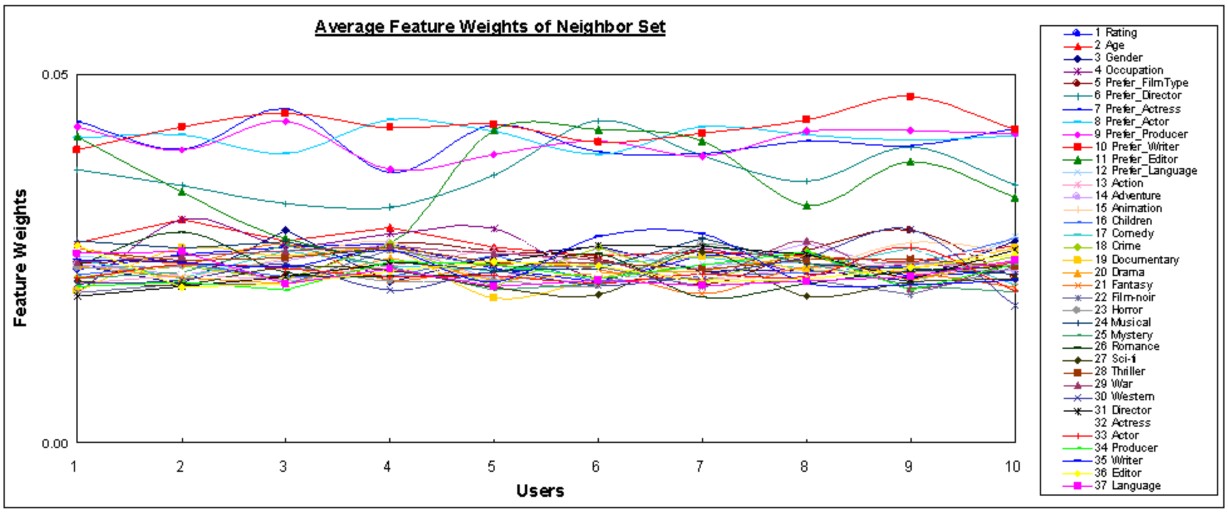


Figure 7. Feature Weights Chart of Neighbor Set

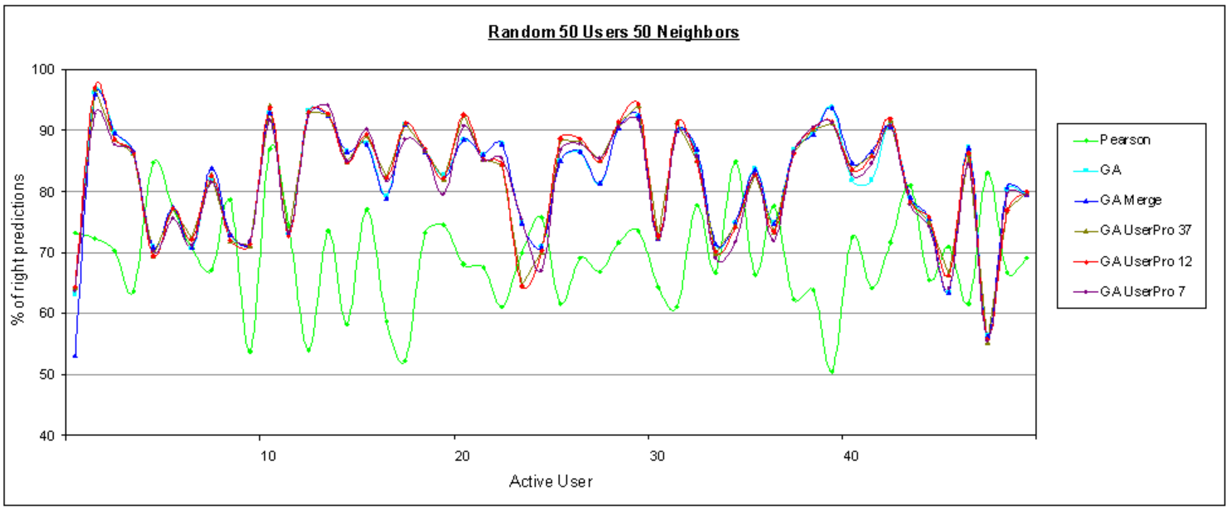


Figure 8. Fitness Accuracy Chart for random 50 Users

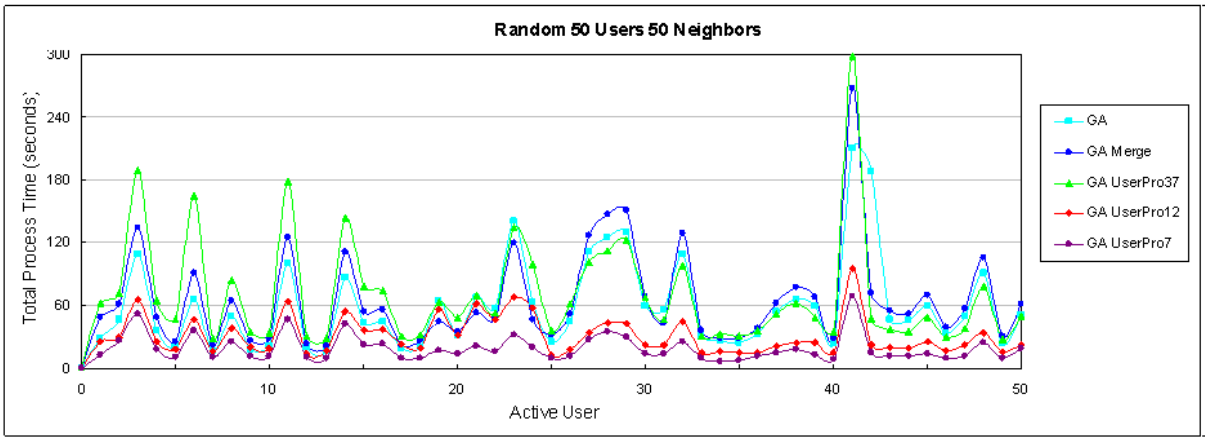


Figure 9. Performance Chart for Random 50 Users

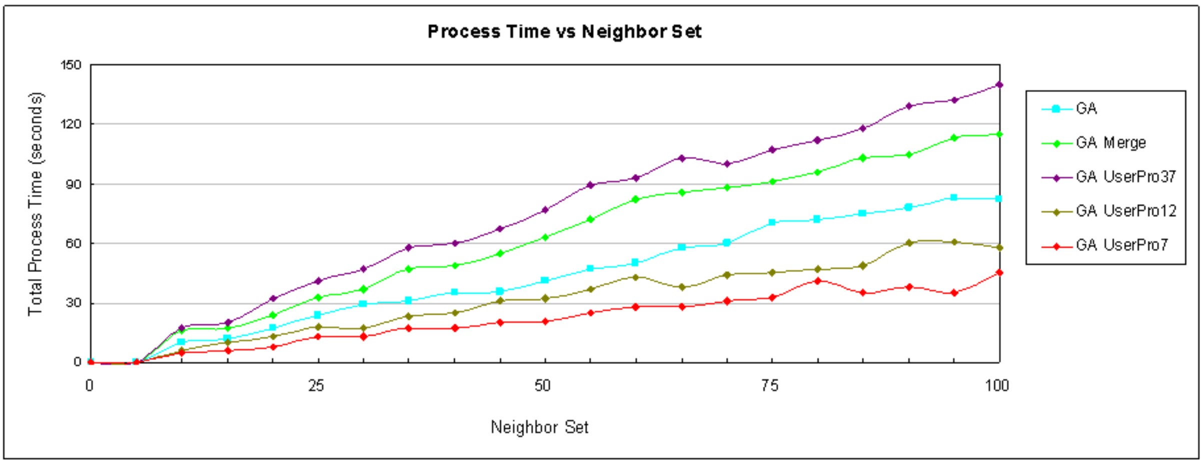


Figure 10. Processing Time verse Neighbor Set Chart

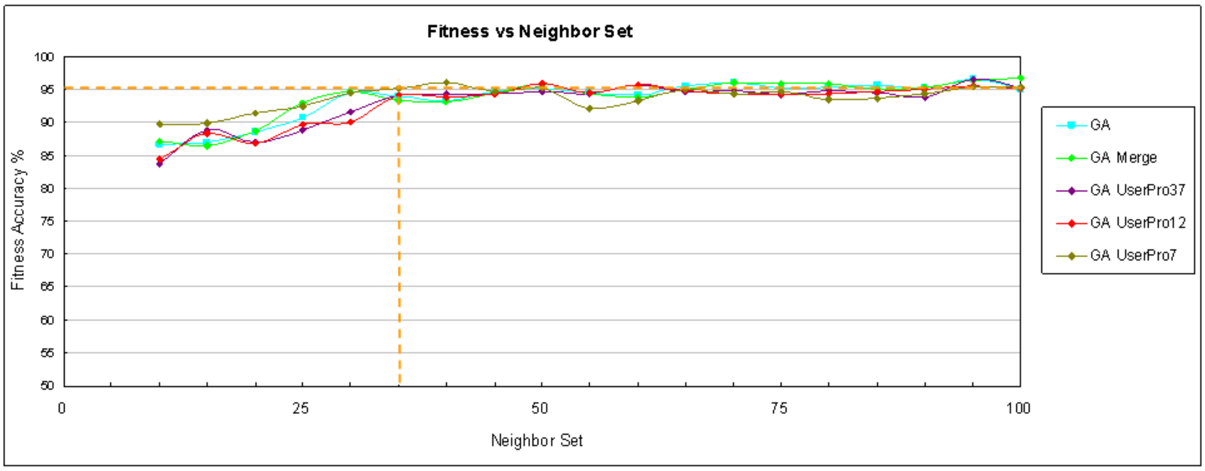


Figure 11. Fitness verse Neighbor Set Chart