On Improving GA-based Collaborative Filtering for Online Recommender

Yvonne Ho, Simon Fong, Zhuang Yan
Faculty of Science and Technology
University of Macau
ccfong@umac.mo

Abstract

Collaborative filtering (CF) is popularly used in recommendation systems that assist users to find items of their interest by offering them personalized suggestions. This is done by CF rating servers that predict scores based on the heuristic of similarity of other peoples' taste to that of the user. However most of the current recommendation systems take all users with common items as neighbors in their measurement. Hence some of the suggested items become noises. In this paper, we proposed an extension to a Genetic Algorithm-based CF scheme in order to improve the accuracy of prediction in a recommendation system. It selectively chooses the top similarity measure values, and codes the user's profile into a chromosome. Our experiments demonstrated that this new scheme gives relatively high accuracy rate in CF.

Keywords: Recommendations, Collaborative Filtering, Genetic Algorithm, Feature Selection, Feature Weight

1. Introduction

Prevalent database and internet technologies enable users to easily search for information or shop online nowadays. Quite often the users face an overwhelming number of choices where a recommendation system is desired to help finding and evaluating items of interest. Such a recommendation system offers the user personalized information regarding the content of recommended items that is derived from the opinions of other individuals who share similar tastes.

Generally, there are three common methods applied in recommendation systems for predicting the users' items of interest, they are *Collaborative Filtering, Content-based Filtering,* and *Hybrid Model*.

Collaborative Filtering (CF) method [1] is based on the similarity between currently active user and other users. It can either be measured by the same item which is known as item-based CF or by the same type of user, known as user-based CF. The goal of this method is to suggest new items, 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. It presents the users a highly relevant set of items.

Content-based Filtering selects items based on the correlation between the content of the items and user preferences. It recommends the items similar to those a given user has liked in the past. Current search engines are based on this retrieval approaches - based on automatic analysis of the content of documents and the content of user's query. But there are some short-comings while applying Content-based Filtering technique. Firstly, for text documents, the system can only capture certain aspects of the content, so that only a very shallow analysis of certain kinds of content can be supplied. Secondly, the system 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 seldom be recommended. Finally, the user's own rating is the only factor influencing the future performance. *Hybrid model* combines the *Collaborative Filtering* and *Content Based Method*.

In some recommendation system, all features will be included when making a suggestion. Some features however are noises that damper the prediction accuracy of the recommendation in the calculation. This is one of the main concerns in current recommendation systems. Customers may not want to use the system again if the recommendations do not reflect their likes.

By using the Hybrid model method, the recommendation system needs large amount of data to build a satisfactorily accurate model. Also it is not incremental that the model must be re-built whenever the data is updated, this process is time consuming.

In comparison *Collaborative Filtering* would be a better choice for recommendation systems. Thus it becomes one of the most popular personalization techniques in online applications recently. Nevertheless we raise a question; how can we achieve high prediction accuracy in a recommendation system? In the traditional CF method, the system only searches for users who contain common items with the active user. In this case, only the top popular items will be selected. As a few common movies are insufficient to reflect a user's taste, the prediction will not be very accurate.

In this paper, we experimented on a movie recommendation system by extending a Genetic Algorithm (GA) for CF [2]. By measuring the feature weights, we show that user profile features are most important to the target user in relative to other movie attributes which may turn into noises. Moreover, the similarity measure can collect the neighbor sets with a similar taste; in this way the recommendations will be more adaptable. From the results of our experiments which we show in the latter sections, we show that GA combined with some fine-tuned User Profiles features is a good candidate for recommendation systems.

2. Background Technology

2.1. Genetic Algorithm

GA is a heuristic search method, based on the mechanics of natural selection and genetics, introduced by John Holland in 1975. 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.

In order to implement a movie recommendation system as a case study, we apply Genetic Algorithm to find the fine-tuned feature weights so that the user preference on each feature can be manifested, the chromosome structure is presented as in Figure 1.

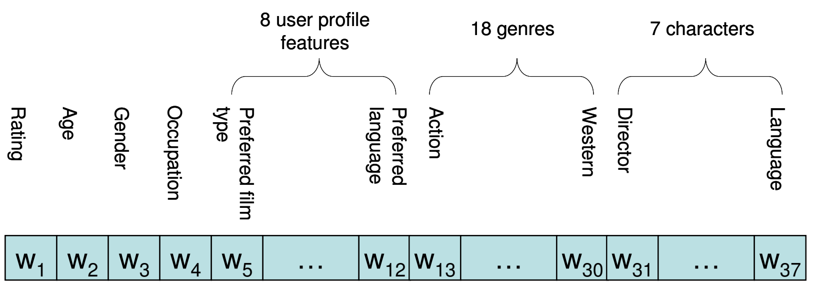
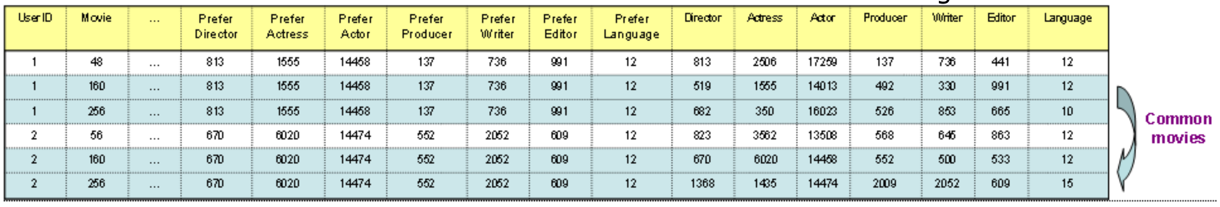


Figure 1. Chromosomes of feature weights

Table 1. Common movies between active user and neighbors



User ID	Movie	…	Prefer Director	Prefer Actress	Prefer Actor	Prefer Producer	Prefer Writer	Prefer Editor	Prefer Language	Director	Actress	Actor	Producer	Writer	Editor	Language
1	48	…	813	1555	14458	137	736	991	12	813	2506	17259	137	736	441	12
1	160	…	813	1555	14458	137	736	991	12	519	1555	14013	492	330	991	12
1	256	…	813	1555	14458	137	736	991	12	682	350	16023	526	853	665	10
2	56	…	670	6020	14474	552	2052	609	12	823	3562	13508	568	645	863	12
2	160	…	670	6020	14474	552	2052	609	12	670	6020	14458	552	500	533	12
2	256	…	670	6020	14474	552	2052	609	12	1368	1435	14474	2009	2052	609	15

Features 1-4 represent the user profile from MovieLens [3]; features 5-12 represent the additional profile features calculated in our work; features 13-30 represent the movie genres from MovieLens; features 3-37 represent the movie attributes from IMDb [4].

2.1.1. Feature Selection and feature weight. *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 and eliminate noise. If all the features are selected without filtering, the performance will indeed be very slow and costly.

On the other hand, *feature weight* is the extension of *feature selection*, which represents the real value of a feature. Thus the value reflects the importance to an item. Basically, it divides into two kinds: binary and real-valued. Binary feature with vector 1 participates in classification, feature with vector 0 does not participate and it will be ignored.

2.1.2. Similarity Measure. In order to gain 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. For example in Table 1, user 1 and user 2 have two movies ID 160 and 256 in common.

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 make use of the *Repair Algorithm* by Okan Yilmaz [5].

2.2. Pearson Algorithm

In order to evaluate the performance of our method, we also implemented the Pearson Algorithm as a comparison reference in our prototype recommendation system. *Pearson Algorithm* is a traditional method used for recommendation systems. It calculates the similarity measure using the ratings of the active user and others, and then classify them into groups of similar tastes.

Consider the movie ratings for each user, the similarity weights w (correlation coefficient) of other users, by *Pearson Algorithm* can be measured to gain the neighbor set as follow:

$$w(a,i)=\frac{\sum_j\left(v_{a,j}-\bar{v}_a\right)\left(v_{i,j}-\bar{v}_i\right)}{\sqrt{\sum_j\left(v_{a,j}-\bar{v}_a\right)^2\sum_j\left(v_{i,j}-\bar{v}_i\right)^2}}$$

where a is the active user, i is the neighbor, where $a \neq I$; j is the number of common movies for a and I; $v_{a,j}$ is the actual vote of movie j for active user a; $\bar{v}_a$ is the mean vote of active user a. The predict vote for active user a on movie item j can also be found as:

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

where n is the total number of common movies between a and i. The fitness then can be measured by comparing the predict vote and the active vote.

3. GA-based Recommendation System

Based on the techniques mentioned in the previous section, 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 2 shows the architecture of our system that can be deployed as an online application. The process flow is divided into three phases:

- *Phase I: Collect User Information*
  For a new comer, the system requests him to register as to collect his basic personal particulars as well as the ratings of a set of movies.

- *Phase II: Create User Profile features*
  For those movies the user rated, we search for the corresponding genres and characters from the movie database. For example, on the rating records of user ID 1, the most frequent type is Musical, then we assume that the preferred film type of this user is Musical. Or, most of the movies that the user rated are directed by 'Steven Spielberg', then we assume the preferred director of the user is 'Steven Spielberg'. In this way, we create 8 additional user profile features, *Preferred Film Type, Preferred Director, Preferred Actress, Preferred Actor, Preferred Producer, Preferred Writer, Preferred Editor and Preferred Language.*

- *Phase III: GA Recommender*
  This phase has the following GA related functions that search for the appropriate recommendation by selecting and weighing the features.

3.1. Feature Selection

The first step in the GA Recommender phase is to prepare the features that are needed. From the experiments we observed that some of the user profile attributes are more effective than the others. So we only select 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.*

3.2. Feature Weighing

The structure of the chromosomes is similar to Figure 1. There are totally 12 genes. Each gene represents by a feature weight w in real value. The heavier the weight the more important the feature is, so that the value can represent 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 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.

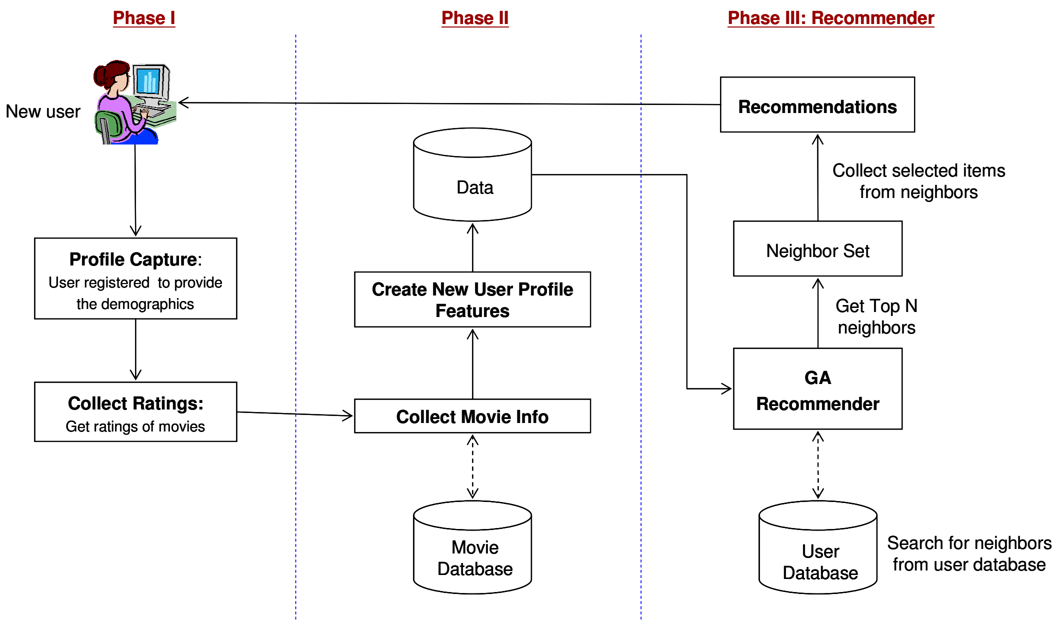

Phase I
Phase II
Phase III: Recommender
New user
Recommendations
Data
Collect selected items from neighbors
Neighbor Set
Profile Capture: User registered to provide the demographics
Create New User Profile Features
Get Top N neighbors
GA Recommender
Collect Ratings: Get ratings of movies
Collect Movie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

Figure 2. GA-based Recommendation System

3.3. Recommendations

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, that 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*:

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

4. Experiments

In order to evaluate the effectiveness of our work, we would compare the traditional Collaborative Filtering method using Pearson Coefficient and our proposed schemes by using Genetic Algorithm. We repeated the experiments by using different features as to show how the feature weights affect the fitness accuracy and performance.

4.1. 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 4, we show the average weights for each feature on different users. The average feature weights is about 0.2649.

The separation of two groups of line is evident that the weights of user profile features are much higher than 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 assume 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.

4.2. Experiment II - Fitness Accuracy

For testing the fitness accuracy of Pearson Algorithm and Genetic Algorithm, and also the effectiveness of different features on GA, we configured a variety of six components in this experiment and applied them on 50 random users respectively. The chromosome feature compositions of the five components are shown in Table 2.

The last component is constructed by preprocessing the user profile data pertaining to the movie attributes, that has a set of 7 unique features as shown in Figure 3.

Table 2. Compositions of chromosome types

Chromosome Type	Chromosome Features			
	1 movie rating and 3 user's particulars	8 user profile features	18 movie genres	7 movie characters
Pearson Algorithm (Pearson)	✓		✓	✓
Genetic Algorithm with 22 features (GA)	✓		✓	
Genetic Algorithm with 29 features (GA Merge)	✓		✓	✓
Genetic Algorithm with 37 features (GA User Profile 37)	✓	✓	✓	✓
Genetic Algorithm with 12 features (GA User Profile 12)	✓	✓		

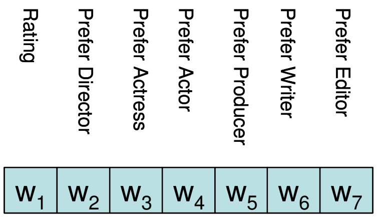


Figure 3. Chromosome of 7 features

From Figure 5, we can observe apparently that 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 range 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 6 shows the process time for running GA on different features. 'GA UserPro7' that is GA with 7 features out performs the other 4 in terms of speed. Its average process time is 19 seconds. This is a 67.53% reduction over 'GA'.

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

4.3. Experiment III - Neighbor Set

For a particular user, we tested the performance on different group sizes of neighbor set, from 10 to 100 respectively.

4.3.1. Process time vs. Neighbor Set. Figure 7 shows the performance of different features running on GA with different neighbor sets. At the beginning, their performances are close. As the neighbor set size increases, the process time 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' for 37 features, the process time increases gradually as the neighbor set expands; where for 'GA UserPro7' with 7 features, the process time increases slowly.

4.3.2. Fitness vs. Neighbor Set. In Figure 8, the fitness of different features rise up gradually as the neighbor set enlarges. Interestingly when the neighbor set reaches over the size of 35, the fitness continues to stay constant. As indicated by the dotted line in chart, the fitness approaches 95% at the turning point. Further increase on the neighbor set size has no effect on it.

5. Conclusion

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 valuable 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 some positive elements in the scalability and speed issues of GA online recommendation systems.

6. References

[1] J. Breese, D. Heckerman, and C. Kadie, "Empirical Analysis of Predictive Algorithms for Collaborative Filtering", *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, 1998.

[2] S. Ujjin, P. J.Bentley, "Evolving Good Recommendations", *Genetic and Evolutionary Computation Conference (GECCO)*, 2002.

[3] MovieLens, http://www.movielens.umn.edu

[4] IMDb, http://www.imdb.com/

[5] O. Yilmaz, L. Tuaf, "Data Mining Feature Subset Weighting and Selection using Genetic Algorithms", Thesis, Air Force Institute of Technology, Air University, 2002.

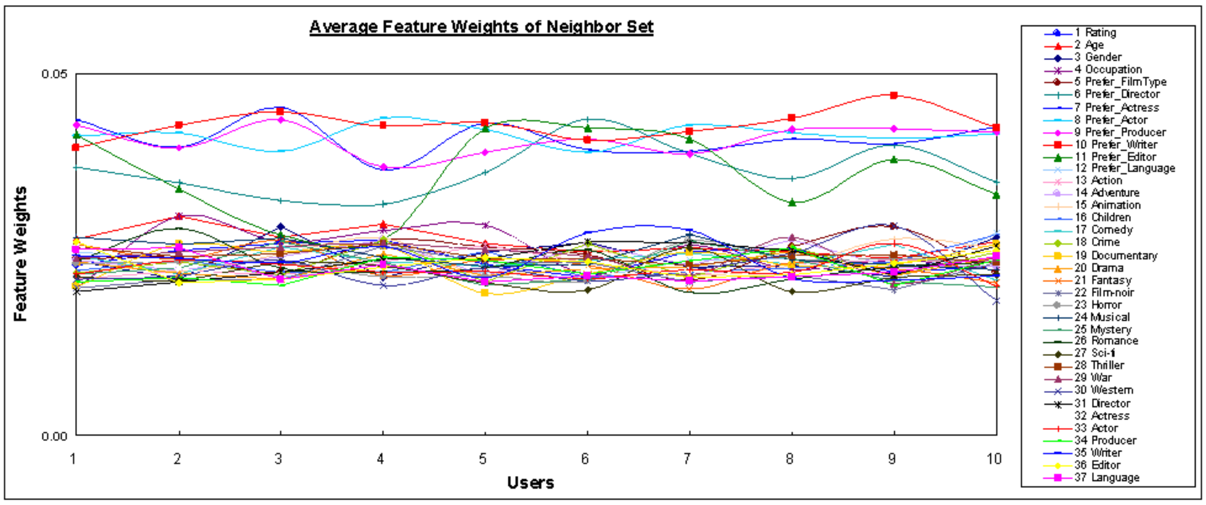


Figure 4. Feature Weights Chart of Neighbor Set

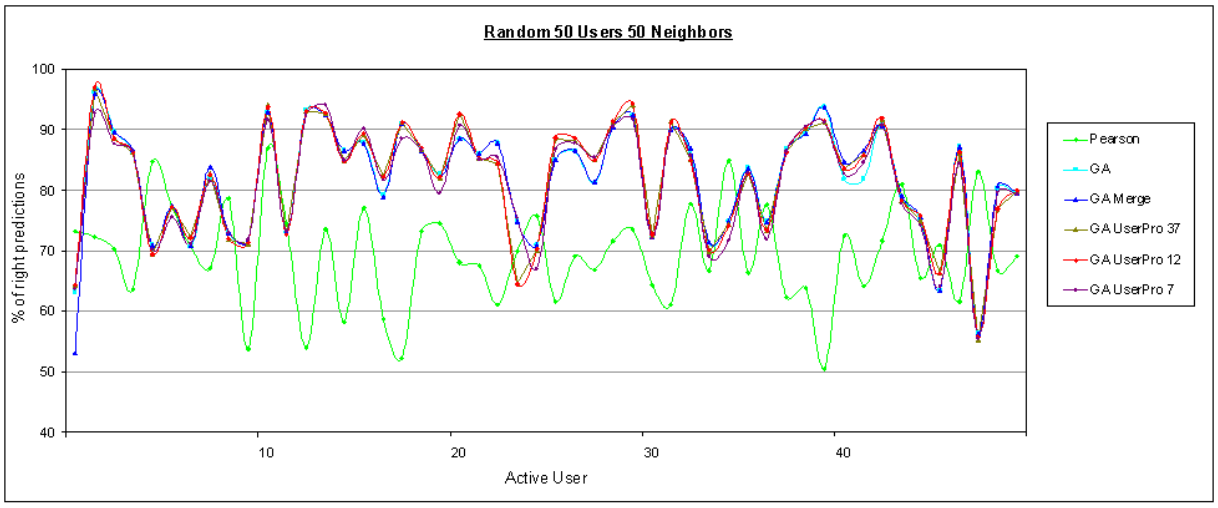


Figure 5. Fitness Accuracy Chart for random 50 Users

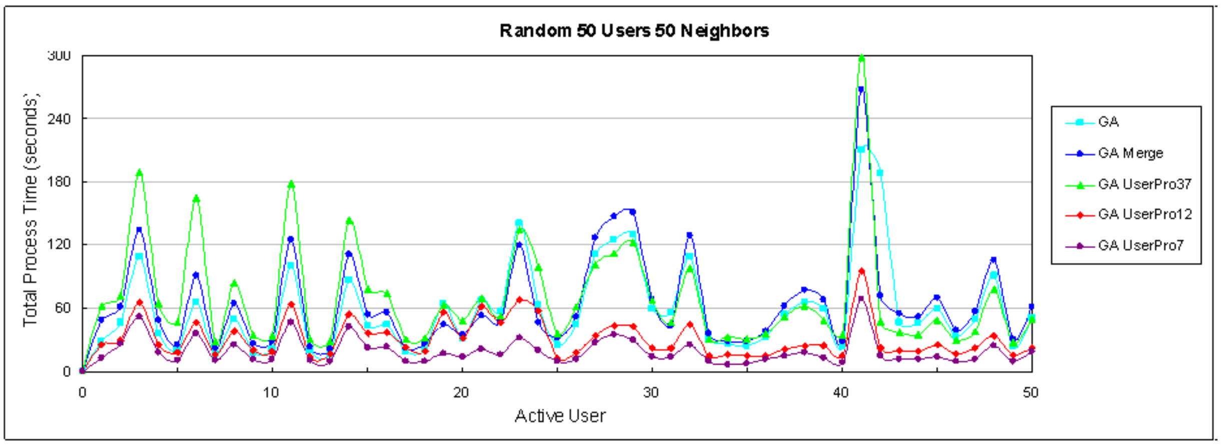


Figure 6. Performance Chart for Random 50 Users

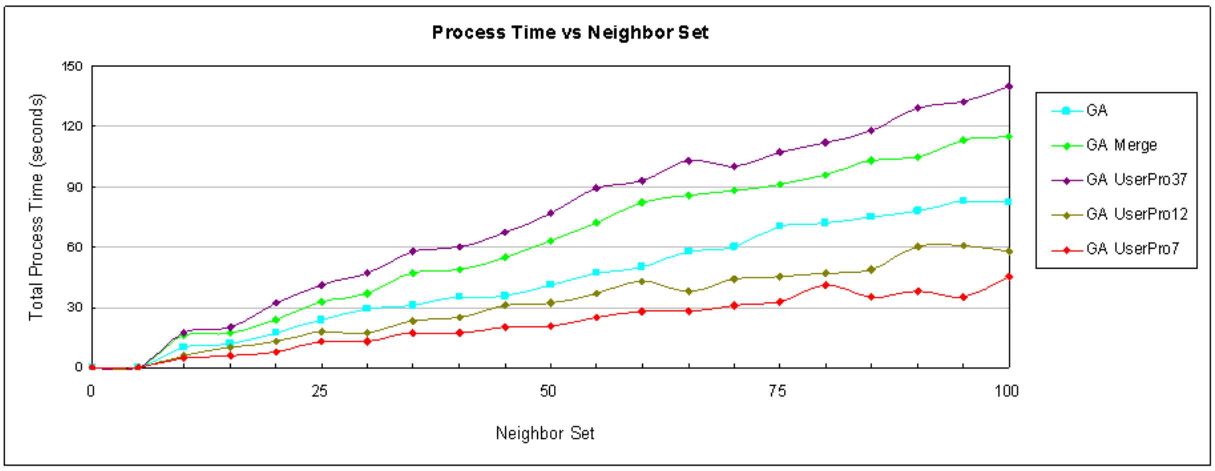


Figure 7. Process Time vs. Neighbor Set Chart

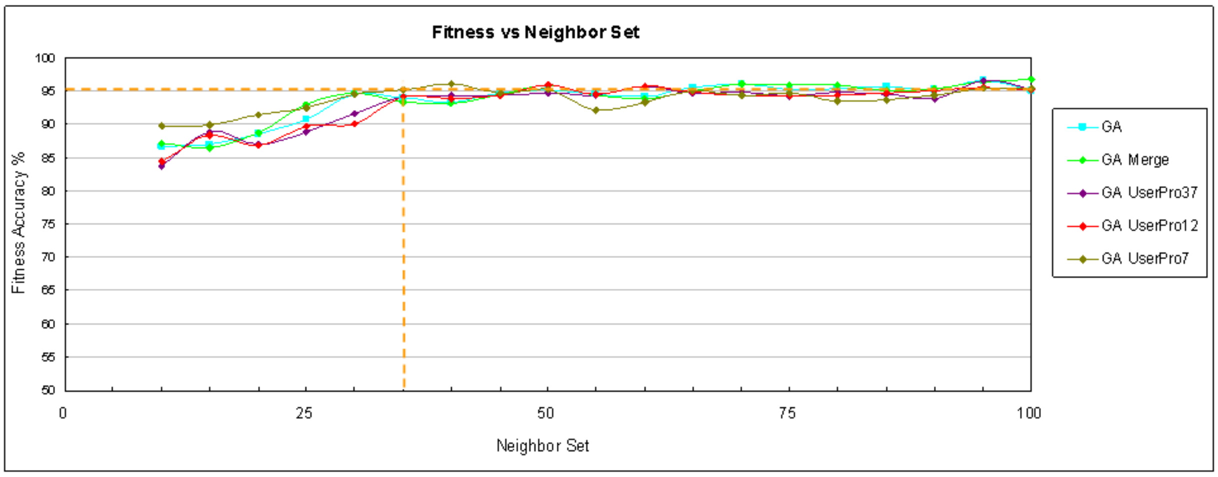


Figure 8. Fitness vs. Neighbor Set Chart