A NEW COLLABORATIVE FILTERING ALGORITHM USING NEAR-CLIQUE BIPARTITE GRAPH CLUSTERS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

HÜSNÜ YILDIZ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

SEPTEMBER 2014

Approval of the thesis:

A NEW COLLABORATIVE FILTERING ALGORITHM USING NEAR-CLIQUE BIPARTITE GRAPH CLUSTERS

submitted by **HÜSNÜ YILDIZ** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen
Dean, Graduate School of **Natural and Applied Sciences** _______________

Prof. Dr. Adnan Yazıcı
Head of Department, **Computer Engineering** _______________

Prof. Dr. İsmail Hakkı Toroslu
Supervisor, **Computer Engineering Department, METU** _______________

Examining Committee Members:

Assoc. Prof. Dr. Pınar Karagöz
Computer Engineering Department, METU _______________

Prof. Dr. İsmail Hakkı Toroslu
Computer Engineering Department, METU _______________

Prof. Dr. Ahmet Coşar
Computer Engineering Department, METU _______________

Dr. Onur Tolga Şehitoğlu
Computer Engineering Department, METU _______________

Dr. Güven Fidan
AGMLAB _______________

Date: _______________

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name:	HÜSNÜ YILDIZ
Signature :	

ABSTRACT

A NEW COLLABORATIVE FILTERING ALGORITHM USING NEAR-CLIQUE BIPARTITE GRAPH CLUSTERS

Yıldız, Hüsnü
M.S., Department of Computer Engineering
Supervisor : Prof. Dr. İsmail Hakkı Toroslu

September 2014, 53 pages

Recommendation systems are becoming increasingly crucial for everyday tasks such as choosing movies, discovering new songs, connecting to other people. These systems try to give the best recommendations as quickly as possible. In order to achieve this target,they employ similarity metrics and clustering for better suggestions, parallel algorithms and dimensionality reduction for fast running time. In this study, we propose prediction algorithms that complete missing values using former user preferences and user information. Our algorithms utilize hierarchical clustering with bottom-up approach to find nearly complete bipartite graphs(near-clique). Near-clique graphs indicate strong connectivity between users and items. However, finding complete bipartite graph is an NP-Complete problem. Therefore, hierarchical clustering and similarity metrics are used for detecting near-clique graphs as much as possible. Predictions are made by using near-clique graphs. To evaluate the algorithms performance, the experiments are held on the MovieLens dataset. The results show that, we achieved high accuracy for overall predictions and especially initial predictions are remarkable.

Keywords: Recommendation Systems, Collaborative Filtering, Matrix Completion, Hierarchical Clustering, Complete Bipartite Graph

ÖZ

İKİ KISIMLI ÇİZGE KÜMELEMELERİNE YAKIN ÇİZGELERİ KULLANARAK YENİ İMECELİ FİLTRELEME ALGORİTMASI

Yıldız, Hüsnü
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi : Prof. Dr. İsmail Hakkı Toroslu

Eylül 2014 , 53 sayfa

Öneri sistemleri film seçmek, yeni şarkılar bulmak, yeni insanlarla iletişim kurmak gibi günlük kullanımlarımız için giderek daha önemli bir hal almaktadır. Bu sistemler mümkün olduğu kadar çabuk bir şekilde en iyi önerileri vermeye çalışır. Bunu başarmak için çeşitli yöntemler kullanılır. Benzerlik metrikleri ve gruplandırma daha iyi öneriler elde etmek için kullanılırken, paralel algoritmalar ve boyut azaltma yöntemleri daha hızlı sonuç almak için kullanılır. Bu çalışmada kullanıcıların bilgilerini ve geçmiş tercihlerini kullanarak, yeni önerileri yüksek doğrulukla bulmaya çalışan öngörü algoritmaları tasarladık. Algoritmalarımız tam iki kısımlı çizgelere yakın çizgeleri bulmak için sıradüzensel gruplandırmayı kullanır. Bu çizgeler kullanıcılar ve öğeler arasında güçlü bağlantıların olduğunu gösterir. Ancak, tam ikili çizgelerin var olup olmadığını bulmak NP-Tam problemdir. Bu yüzden, sıradüzensel gruplandırma ve benzerlik metrikleri kullanılarak tam iki kısımlı çizgelere yakın çizgeler belirlenir. Algoritmaların performansı MovieLens veri kümesi kullanılarak değerlendirildi. Sonuçlar gösteriyor ki, bütün veri kümesi için yüksek doğrulukla sonuçlar elde edilirken, özellikle ilk kısımlarda yapılan tahminlerin doğruluğu daha da yüksektir.

Anahtar Kelimeler: Öneri Sistemleri, İmeceli Filtreleme, Matris Tamamlama, Sıradüzensel Gruplama, Tam İki Kısımlı Çizge

To undying hopes

ACKNOWLEDGMENTS

I would like to express my great appreciations to my supervisor Professor İsmail Hakkı Toroslu for his endless support and guidance during my studies. Without his help and encouragement, this work would never have happened.

I would like to express my thanks to in particular Nurdan Doğruyol and Önder Gülbeyaz, Savaş Ilıkkanlı, Evren Koç, Zeynep Durgut, Seda Günay, Engin Erdemir, Hakan Aytuttu, Murat Kırmızı, Sercan Cihanbilir, Hayri Ersoy, Özgür Alkan and Sevcan Doğruyol, for their help, support, friendship and encouragement. Without them, it would not be possible to finish this work.

I would also like to express my thankfulness to Ali Aralıoğlu, Haydar Güzel, Abdullah Yahşi, Ferhat Yaşar and Murat Aldoğan for all times especially Saturday nights.

I am deeply grateful to my collegagues and friends, Merve Aydınlılar, Özgür Kaya, Gökdeniz Karadağ, İlkcan Keleş and Alperen Eroğlu for their support and encouragement.

I would like to extend my sincerest thanks to my family for their believing and support in every moment of my life.

TABLE OF CONTENTS

ABSTRACT v

ÖZ vi

ACKNOWLEDGMENTS viii

TABLE OF CONTENTS ix

LIST OF TABLES xii

LIST OF FIGURES xiii

LIST OF ABBREVIATIONS xv

CHAPTERS

1 INTRODUCTION 1

2 RELATED WORK AND BACKGROUND 3

2.1 Related Work 3

2.1.1 Collaborative Filtering 3

2.2 Background 6

2.2.1 Clustering 6

2.2.1.1 Hierarchical Clustering 6

2.2.2 Recommendation Systems 7

2.2.2.1 Content-Based Filtering 8

2.2.2.2 Collaborative Filtering 8

2.2.3 Hybrid Recommendation Systems 9

2.2.4 Complete Bipartite Graph 10

2.2.5 Similarity Metrics 11

2.2.5.1 Cosine Similarity 11

2.2.5.2 Pearson Product-Moment Correlation Coefficient 11

2.2.5.3 Euclidean Distance 12

2.2.6 Singular Value Decomposition 12

2.2.7 Accuracy Metrics 13

2.2.7.1 Mean Absolute Error 13

2.2.7.2 Root Mean Square Error 13

3 DATA DEFINITION 15

3.1 MovieLens Data 15

4 OUR APPROACH 17

4.1 Preprocessing 17

4.2 Recommendation Algorithms Using Near-Clique 20

4.2.1 Initialization 20

4.2.2 Simultaneous Matrix Completion Algorithm 20

4.2.3 Stepwise Matrix Completion Algorithm 24

5 EXPERIMENTAL RESULTS 27

5.1 Experimental Setup 27

5.2 Simultaneous Matrix Completion Algorithm 28

5.3 Stepwise Matrix Completion Algorithm 33

5.4 Demographic Experimental Results 38

5.4.1 Simultaneous Matrix Completion Algorithm 38

5.4.2 Stepwise Matrix Completion Algorithm 41

5.5 General Comparison 46

6 CONCLUSION AND DISCUSSION 49

REFERENCES 51

LIST OF TABLES

TABLES

Table 3.1 Film types 16

Table 3.2 Occupations 16

Table 4.1 Distance array representation 19

Table 4.2 Intervals for age 20

Table 4.3 Two dimensional array contains clusters and their distances. C_i, m and $d(,)$ denotes clusters, number of users and distance between clusters, respectively. 21

Table 4.4 Hash table for unrated elements 23

Table 4.5 The distances between users 23

Table 4.6 Array that contains clusters hierarchically 24

Table 5.1 Simultaneous Matrix Completion Algorithm results 32

Table 5.2 Stepwise Matrix Completion Algorithm results 37

Table 5.3 SIMCA and 2 results with demographic features 45

Table 5.4 The results for linkage criteria: single and average 46

Table 5.5 The comparative results 47

LIST OF FIGURES

FIGURES

Figure 2.1 Dendrogram 7

Figure 2.2 Biclique 10

Figure 4.1 Hash table representation 18

Figure 4.2 Experimental data includes ratings 22

Figure 4.3 Near-clique representation of an example 23

Figure 5.1 MAE results of the first step of SIMCA 29

Figure 5.2 MAE results of the second step of SIMCA 29

Figure 5.3 MAE results of the last step of SIMCA 30

Figure 5.4 RMSE results of the first step of SIMCA 30

Figure 5.5 RMSE results of the second step of SIMCA 31

Figure 5.6 RMSE results of the last step of SIMCA 31

Figure 5.7 MAE results of the second step of STMCA 33

Figure 5.8 MAE results of the second step of STMCA 34

Figure 5.9 MAE results of the last step of STMCA 34

Figure 5.10 RMSE results of the first step of STMCA 35

Figure 5.11 RMSE results of the second step of STMCA 35

Figure 5.12 RMSE results of the last step of STMCA 36

Figure 5.13 MAE results of the second step of STMCA 38

Figure 5.14 MAE results of the second step of STMCA 39

Figure 5.15 MAE results of the last step of STMCA 39

Figure 5.16 RMSE results of the first step of STMCA 40

Figure 5.17 RMSE results of the second step of STMCA 40

Figure 5.18 RMSE results of the last step of STMCA 41

Figure 5.19 MAE results of the second step of STMCA 42

Figure 5.20 MAE results of the second step of STMCA 42

Figure 5.21 MAE results of the last step of STMCA 43

Figure 5.22 RMSE results of the first step of STMCA 43

Figure 5.23 RMSE results of the second step of STMCA 44

Figure 5.24 RMSE results of the last step of STMCA 44

LIST OF ABBREVIATIONS

SIMCA	Simultaneous Matrix Completion Algorithm
STMCA	Stepwise Matrix Completion Algorithm
MAE	Mean Absolute Error
RMSE	Root Mean Square Error
PPMCC	Pearson Product-Moment Correlation Coefficient
SVD	Singular Value Decomposition

CHAPTER 1

INTRODUCTION

In recent years, recommendation systems have begun widely to be used by social networking sites, video and music sharing sites, marketing sites in order to suggest new musics, films, videos, books etc. These websites collect information via users' likes, dislikes or ratings parameters. The collected information is processed with various types of recommendation methods. Although these information generally remain sparse, some methods that have remarkable results with high accuracy values and fast running times have been designed.

Recommendation system algorithms can be mainly explained in four parts, similarity metrics, clustering, dimensionality reductions and parallel algorithms. User to user or item to item similarity calculations effect other parts extremely because the operations start with calculating similarities between user or items. Since existing similarity metrics do not cover the all the situations, new similarity metrics had been designed. In addition to that, clustering process is the most important part of the recommendation systems. Grouping of the data directly effects predictions and suggestions. Thus, clustering algorithms such as k-nearest-neighbour and k-means are used and modified in various ways. Moreover, dimensionality reduction techniques are used to eliminate data which involves similar items and very few information about the user or the item. By removing these redundancies, better results are obtained in faster way. The recommendation system algorithms usually take long time. The use of parallel algorithms make the recommendation process faster.

In our algorithms, we use connectivity among the users or items. In other words in order to make prediction or suggestion, at least one related item must be in the data. If the connection is not found, The algorithm can not perform any prediction about that item. Also, relations between users and items determine the ratio of the completion of the matrix. Therefore, our algorithms are much more reliable than other algorithms which do not use these relations. Moreover, to enforce these relations, complete bipartite graphs(biclique) are used in this study.

Rather than k-means and k-nearest-neighbour that depends on the parameter selection for clustering and classification, we used hierarchical clustering that is a top-down or bottom-up approach for clustering. The benefit of the hierarchical clustering for this study is finding near-cliques using bottom-up approach. With this clustering method, the best results for our algorithm are obtained at initial phases of clustering. The main reason for the best results is that near-cliques are found in this phase. Since our algorithms employ heuristic approaches, the parameter selection is important to enhance the results. Also, while computing predictions using ranking information, demographic features of users may be integrated to the algorithms.

Recommendation methods can be time consuming. Therefore, recommendation system algorithms are designed to get faster solutions. Since the data we used has sparseness problem, the hash table is used in our algorithm to produce predictions faster.

In this study, the experiments indicate that our algorithms produce more accurate predictions, especially for more similar users when compared to other recommendation system algorithms.

The outline of this study is summarized as the following:

- *In chapter 2* Literature survey about collaborative filtering algorithms and background information for used methods and algorithms in this study are presented in detail.
- *In chapter 3* MovieLens dataset features is explained.
- *In chapter 4* contains our heuristic prediction algorithms in detail. Also, examples are included to the chapter in order to clarify.
- *In chapter 5* the results of experiments are presented.
- *Inchapter 6* concludes this study and suggests possible enhancement ways

CHAPTER 2

RELATED WORK AND BACKGROUND

2.1 Related Work

In this section, we provide information about collaborative filtering algorithms. We also summarize the important parts of the each algorithm.

2.1.1 Collaborative Filtering

Collaborative filtering algorithms gain importance in parallel with the spread of the internet. There are many algorithms proposed for collaborative filtering. Generally, they focus on sparseness and scalability. There are some algorithms which are based on prediction algorithms [3, 9, 10, 15, 18, 22] In addition, there are some algorithms which employ graph-based algorithms [10, 13]. Some of the proposed algorithms introduce new similarity metrics [2,4,5,8,11]. Also, collaborative filtering algorithms can be time-comsuming. Thus, faster solutions proposed by using parallel algorithms [16, 27]. Moreover, dimensionality reduction algorithms in some of the words [12, 25, 27].

In [15], Linden et al. propose item to item collaborative filtering, they build similar-item tables for scalability and effectiveness. Then, using this table, they calculate the similarity of each item pairs for predictions. All operations are performed offline except recommendation. Since only recommendation phase which uses the created table is a real time process, this algorithm produce recommendations faster.

In [18], Melville et al. use Bayesian text classifier to create the recommendation model.They make use of the user profiles which are learned from the data. They divide data into two parts: texts and labels. Texts fields contain content information and labels fields contain ratings. Then, the model is used to predict ratings with neighborhood-based algorithm, which creates subset of similar users using the model with pearson correlation similarity metric.

In [16], Luo et al. propose an algoritm which uses parallel regularized matrix factorization. To parallelize the algorithm, they firstly eliminate dependency problem between users and items. Then, a model is trained by the data to make predictions. Although their algorithm finds similar accuracy ratios, it has less execution times. In [27], Zhou et al. use Alternating-Least-Square(ALS) algorithm to find low-rank matrix which is decomposed by SVD. However, ALS may result in the overfitting of the data. To overcome of this problem, they use Tikhonov regularization with their regularization matrix. In their algorithm, users and items have a characteristic vector. After applying ALS with regularization, dot product of the vectors give predictions. Moreover, they parallelized their algorithms to get faster results.

In [26], Xue et al. use smoothing and clustering methods together. Firstly, the data is divided into k clusters using k-means. Then, smoothing method is applied to unrated items. And then, for an active user, similar users and the neighbour of this user are chosen. Similarity between an active user and these neighbours are computed once again using different metric with weights which are generated from their rules. Finally, the most similar users are chosen and prediction is computed by using weights and similarities.

In [22], Sarwar et al. present a survey that compares item-based similarity algorithms with user-based algorithms. They present two algorithms to make predictions by using different similarity metrics: weighted sum and regression. The first algorithm sums the number of similar items which are rated and divides its to the number of total similar items. The second algorithm use linear regression method which is similar to the first algorithm. The difference is that second algorithm uses approximate values due to the limitations of the similarity metrics. These approximate values are generated by using their model of the ranked items.

In [3], Bell et al. normalize the data to remove negative effects for clustering. After this operation, data is divided into clusters by using k-nearest-neighbour. And then, all nearest neighbours are weighted to correlate between users concurrently. They point out that normalization and weighting operations improve the prediction accuracy.

In [10], Gao et al. propose a model that builds a weighted connective graph using user rankings. This graph is used by their proposed algorithm, which is named as user-rank. Their algorithm computes similarity coefficient using weighted graph. These coefficients are used to find predictions and item similarities. The outcomes is used to find predictions and item similarities.

In [25], Vozalis et al. use Singular Value Decomposition(SVD) with demographic information. In the beginning, they create the matrix that contains demographic informations. Then, they perform SVD on the matrix to find the minimal set of attributes which constitutes the demographic correlation. Predictions are produced by using this correlation. In [12], Kim et al. propose iterative principal component analysis(PCA). They firstly reduce the dimensions of the data using SVD. Then, they find the principal components of the reduced data. This operation proceeds until the algorithm reaches to the ideal number of principal components. By using the principal components obtained, their algorithm clusters the data with k-means and recursive rectangular clustering. Finally, they predict rankings of the unrated data.

In [2], Ahn defines the new heuristic similarity metric, PIP(Proximity - Impact - Popularity), since cosine and pearson similarity metrics remain incapable at some points. The PIP also gives better results for cold-start problems. It mainly focuses on punishment for calculations using based on the differences and selection of the items according to the users' tastes.

In [13], Kiraly et al. propose methods for completion or reconstruction of the matrix under different conditions. They use graphs for relations between rows and columns. To make prediction, there must be at least one connected edge to rows or columns. If any connected edge is not found, prediction can not be done since any relation index does not exist. Thus, it always may not be possible to find fully completed matrix. Based on this theory, if you obtain better results, you should find near-clique that contains all of the matching edges between two set of vertices except one or two edges.

2.2 Background

In this section, we explain the methods and algorithms in a detailed way to provide background information for our algorithms.

2.2.1 Clustering

Clustering is the process of grouping objects such that similar objects fall into same groups but dissimilar ones distributed to the other groups [6]. There are lots of clustering algorithms which are proposed for different conditions and states. They are used in many fields such as data mining, statistical data analysis, pattern recognition, machine learning etc. In this study, hierarchical clustering is used for our algorithms.

2.2.1.1 Hierarchical Clustering

Hierarchical clustering is one of the clustering methods and it divides the data with respect to similarity measures hierarchically [20]. Clusters are represented by tree which is called as dendrogram and an example dendrogram is shown in Figure 2.1.

Hierarchical clustering generally has two types, agglomerative and divisive. The former is a bottom-up approach. At first, each data has its own cluster. Then, at each iteration, two clusters are merged in the new cluster by using similarities. This process continues until one cluster is formed with all of the objects in the data. The latter, divisive is a top-down approach where data is clustered until each of them is separated into different clusters. Since we use agglomerative hierarchical clustering in this work, the details of the agglomerative hierarchical clustering algorithm is given in Algorithm 1.

While merging clusters, the type of linkage criteria plays an important role for hierarchical clustering. There are several types of linkage criteria such as single, average, complete, ward, centroid and median. We use three of them, single, average and complete whose details are given below:

- *single:* minimum distance from one cluster to the other cluster.

- *average:* average distance from one cluster to the other cluster.
- *complete:* maximum distance from one cluster to the other cluster.

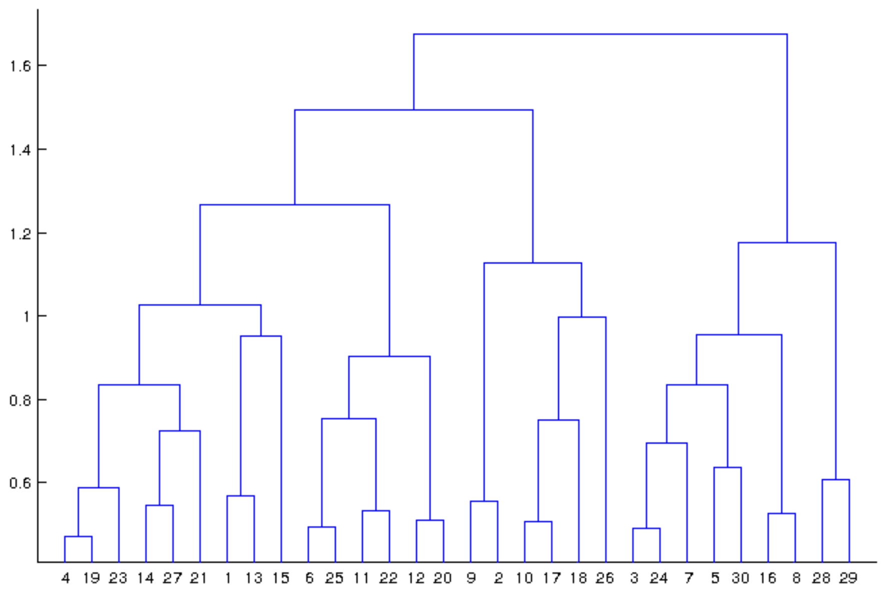


Figure 2.1: Dendrogram

2.2.2 Recommendation Systems

Recommendation system is an approach to suggest new items such as movie, book and music to the users using their former preferences [7,21]. User preferences, which can be ratings, likes or dislikes, are collected via websites. These preferences are processed using some methods like clustering and similarity measures. After applying these methods, the most similar items to the former user preferences are suggested to the users. The aim of recommendation system is to find the best item - user relation. There are mainly three types of recommendation systems:

- Content-based filtering
- Collaborative filtering
- Hybrid recommendation methods.

Algorithm 1 Hierarchical Clustering

```
Input: $data$
output: $clusters$
1: function HIERARCHICALCLUSTERING($data$)
2:     each element of data has its own cluster $C_1, \ldots, C_n$
3:     $k \leftarrow n + 1$
4:     $distanceArray \leftarrow$ find similarities using distance metrics between $data$
5:     repeat
6:         Find nearest cluster between $C_i$ and $C_j$ using $distanceArray$
7:         $C_k \leftarrow$ Merge $C_i$ and $C_j$
8:         $k \leftarrow k + 1$
9:     until One cluster is remained
10: end function
```

2.2.2.1 Content-Based Filtering

Content-based filtering makes suggestions by using past preferences of the users [21]. It analyzes past preferences such as movie, sound, book etc with respect to the users' behaviors, likes and dislikes. Then, according to item types, similar choices are recommended. Suppose that the user likes science fiction films. Content-based filtering analyzes data using the science fiction film properties, then it makes new suggestions with the best matching.

2.2.2.2 Collaborative Filtering

Collaborative filtering matches similar domains that users have. Then, based on this similarity, it makes recommendation to the users [24]. Firstly, collaborative filtering collects information via websites that users rank the movies, songs etc. Secondly, collected rankings are compared with all of the other users rankings to find neighbours that have similar preferences. New recommendations generated by using neighbours ranking results. Collaborative filtering is used by Amazon.com, Netflix and Movielens. In addition, if the user has less information about preferences, collaborative filtering may not provide consistent results. This is due to the fact that there is not enough information to match user preferences with other users' preferences. This situation refers to the cold start problem.

Collaborative filtering is divided into two methods :

- Memory-based methods use whole domain which is formerly rated to make suggestion.
- Model-based trains a model using the user preferences data. Then, it makes recommendations with respect to this model.

Moreover, collaborative filtering has basically two forms: item-based and user-based. Item-based collaborative filtering uses item-item relationships and user-based collaborative filtering uses user-user similarities.

2.2.3 Hybrid Recommendation Systems

Hybrid recommendation systems combine recommendation techniques. The aforementioned techniques, content-based filtering which is explained in 2.2.2.1 and collaborative filtering which is explained in 2.2.2.2, are combined with other methods to gain better performance for predictions.

Hybrid recommendation systems provide an advantage for recommendation techniques that each of them has specific problems in their domains. Different approaches can solve other recommendation system approaches' problems. They are divided into seven categories [6]:

- Weighted
- Switching
- Mixed
- Feature combination
- Cascade
- Feature augmentation

- Meta-level

2.2.4 Complete Bipartite Graph

Complete bipartite graph or biclique is a special type of graph such that each vertex of the graph set is linked to all vertices of the other graph set [19].

Let A and B are different set of vertices. A biclique between A and B has $|A| + |B|$ vertices and $|A| \times |B|$ edges. Determining whether the set of vertices contain biclique or not is an NP-complete problem.

Figure 2.2 shows biclique representation.

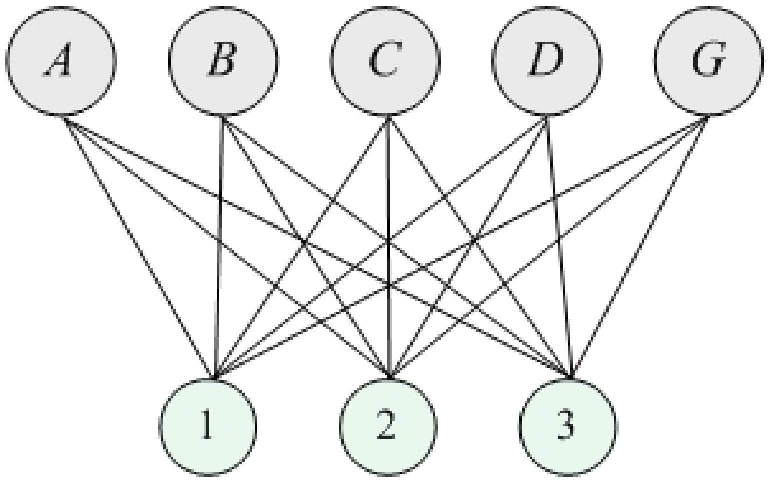


Figure 2.2: Biclique

In this study, our motivation is to find near-cliques that have some missing edges compared to the complete biclique. We try to increase the accuracy values of predictions by using this strong connectivity.

2.2.5 Similarity Metrics

In this section, similarity metrics, used in our algorithms, are explained. Similarity metrics are one of the important parts of clustering methods since they are used to determine whether users are close to each other or not.

2.2.5.1 Cosine Similarity

Cosine similarity is a measure between two vectors to find how much two vectors are similar using the cosine value of the angle between them. [22]. To calculate cosine similarity, dot product and magnitudes of two vectors are used.

Let A and B are two vectors of space, cosine similarity formula is in Equation 2.1:

$$\cos(\theta) = \frac{A \cdot B}{||A||\,||B||} = \frac{\sum\limits_{i=1}^{n} A_i \times B_i}{\sqrt{\sum\limits_{i=1}^{n} (A_i)^2} \times \sqrt{\sum\limits_{i=1}^{n} (B_i)^2}} \tag{2.1}$$

The result of $\cos(\theta)$ is in the range of [-1,1]. -1 represents that two vectors are opposite, where as 1 means that these vectors are the same. 0(zero) represents independent vectors.

2.2.5.2 Pearson Product-Moment Correlation Coefficient

Pearson product-moment correlation coefficient (PPMCC) is a measure to find how much two variables are similar like cosine similarity [14]. It associates linear relationship between variables using standart deviation and covariance. Let A and B are two vectors of space, the PPMCC formula can be seen in Equation 2.2

$$r = \frac{\sum\limits_{i=1}^{n} (A_i - \bar{A})(B_i - \bar{B})}{\sqrt{\sum\limits_{i=1}^{n} (A_i - \bar{A})^2} \sqrt{\sum\limits_{i=1}^{n} (B_i - \bar{B})^2}} \tag{2.2}$$

The value of r can be changed between -1 and 1. If r is eqaul to 1, this implies that a perfect matching is found between variables X and Y. If it is equal to -1, it implies negative correlation between variables and if equals to 0 this means that there is no correlation.

2.2.5.3 Euclidean Distance

Euclidean distance is a metric which finds distance between two points. Euclidean distance is defined as in Equation 2.3: Let p and q are two vectors of space,

$$d(p,q) = \sqrt{\sum_{i=1}(q_i - p_i)^2} \tag{2.3}$$

2.2.6 Singular Value Decomposition

Singular value decomposition (SVD) is a method which is used for matrix factorization [23]. Let X is a $m \times n$ matrix. SVD of X is defines as in Equation 2.4:

$$SVD(X) = U \times S \times V^T \tag{2.4}$$

where U and V are left and right singular vectors, U is an $m \times m$ and V is an $n \times n$ matrix. S is the singular values matrix which has the dimensions of m and n. Singular values are sorted decreasingly on diagonal. The non-diagonal parts of S matrix is zero. Singular value decomposition can be used for matrix similarity and dimensionality reduction of the matrix. To reduce dimension of the matrix, non-zero singular values are used which called as rank. Let r is a rank of X. Using the r singular values, whole matrix of X can be spanned. Also part of the matrix can be spanned with $t \leq r$. The equation 2.5 shows that coverage ratio CR of the matrix using singular values:

$$CR = \frac{\sum_{i=1}^{t} S_{i,i}^2}{\sum_{i=1}^{r} S_{i,i}^2} \tag{2.5}$$

The dimensionality reduction of the matrix using SVD is performed by retaining the first t column of U,the first t singular values of S and the first t rows of V. Then U, S and V are multiplied and X_t is obtained matrix with rank t.

SVD is also used to find matrix similarity. Let user denote rows of X and items denote columns of X. U represents user-user similarity and V represents item-item similarity. Using distance metrics such as cosine similarity, pearson product-moment correlation coefficient and euclidean distance etc., the similarity between rows and columns can be determined.

2.2.7 Accuracy Metrics

In this work, we use two accuracy metrics: mean absolute error and root mean square error. They are used to show that the accuracy of proposed method suggestions.

2.2.7.1 Mean Absolute Error

Mean Absolute Error(MAE) is computed by summing absolute of the real values and proposed prediction values. Then it is divided by the number of total matrix elements [17]. MAE is formulized as in Equation 2.6. In the equation, A is a real matrix, B is proposed matrix and N is the total number of the elements in the matrix.

$$MAE = \frac{\sum_{i,j=1}^{N} |A_{i,j} - B_{i,j}|}{N} \tag{2.6}$$

2.2.7.2 Root Mean Square Error

Root mean square error is the other accuracy metric used in this work. It computes sum of the squared residuals which is divided by the total number of the elements. Then, it takes square root of the result [17]. Let A be the real matrix, B be the proposed matrix and let N stands for the total number of the elements in the matrix. Then, RMSE is defined as in the Equation 2.7.

$$RMSE = \sqrt{\frac{\sum_{i,j=1}^{N} (A_{i,j} - B_{i,j})^2}{N}} \tag{2.7}$$

CHAPTER 3

DATA DEFINITION

In this chapter, we provide information about MovieLens dataset in a detailed way.

3.1 MovieLens Data

In this work, MovieLens dataset [1] is used for our experimental results. MovieLens dataset had been collected by GroupLens Research between 1997, September and 1998, April. For recommendation systems, MovieLens is the most preferred dataset with the Netflix. MovieLens data are classified according to the rating counts, 100.000, 1 million and 10 million ratings. We used MovieLens data with rating count 100.000 ratings dataset for our algorithms.

MovieLens dataset features:

- 1682 movies and 943 users
- 100.000 ratings between 1 and 5
- Minimum 20 movies had been evaluated by each user.
- Demographic features
- Movie information
- 20% of the data is given as test data and the remaining is given as training data.

Movie information includes the attributes: are movie id, movie title, release date, video release date, IMDB url and film types 3.1. Film type attribute can contain more than one film type.

Table 3.1: Film types

Unknown	Action	Advanture	Animation
Children's	Comdey	Crime	Documetary
Drama	Fantasy	Film-Noir	Horror
Musical	Mystery	Romance	Science-Fiction
Thriller	War	Western	

Rating files contain four fields which are user id, movie id, rating and timestamp. Our algorithms use all of them except timestamp.

Demographic features contain age, gender, occupation and zipcode information. All of them except zipcode is used in our algorithms. Users' occupations are listed in the table 3.2.

Table 3.2: Occupations

Administrator	Artist	Doctor
Educator	Engineer	Entartainment
Executive	Healthcare	Homemaker
Lawyer	Librarian	Marketing
None	Other	Programmer
Retired	Salesman	Scientist
Student	Technician	Writer

In this thesis, we use ratings and demographic features.

CHAPTER 4

OUR APPROACH

In this chapter, we provide information regards our proposed algorithms in detail. In this work, we aim to increase the speed and accuracy while designing algorithms. These algorithms mainly focus on the near-cliques. Actually, finding bicliques is an NP-complete problem. Thus, to find near-cliques, we decided to use hierarchical clustering and similarity metrics. Similarity metrics, especially cosine similarity and PPMCC, and linkage criterias for hierarchical clustering play an important role in the prediction phase.

4.1 Preprocessing

In our algorithms, due to the sparseness problem, the data is stored in the hash table which uses keys to index items and each key has its own list. Hash table allows fast access to the data. In our structure, movies and users are used as both keys and items in the hash tables. Figure 4.1 shows how we store the data. Also, values which are ratings are stored in an $m \times n$ matrix, m and n denotes rows and columns, respectively.

In this part, firstly, data is read from file and stored into hash table and matrix. Secondly, using hash table and matrix, distance between each user is calculated and stored into $\frac{m \times (m-1)}{2}$ array. To illustrate, assume that data has four rows and table 4.1 shows pair representation. Finally, the distance array is used for hierarchical clustering and preprocessing is finished. Before going into the details of the preprocessing algorithm, we need to define the following parameters that will be used in the algorithm:

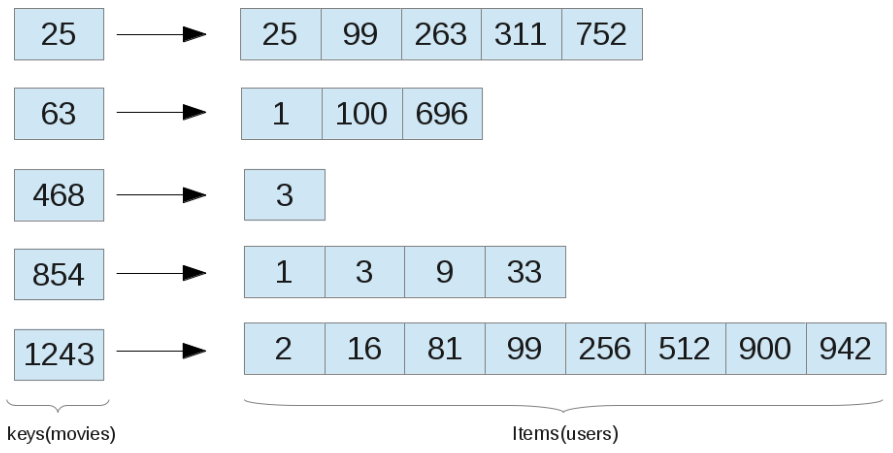


Figure 4.1: Hash table representation

- *dataFile:* File which contains users, movies and ratings information,
- *demFile:* File contains users demographic information.
- *demTable:* it denotes hash table contains demographic information.
- *hTable:* It denotes hash table to store sparseness data.
- *ratingsArray:* it is a matrix and contains ratings.
- *demographic:* it denotes demographic information of users.
- *UseDemographicInformation:* it denotes boolean operator that determine whether demographic information are used or not.
- *hClusters:* Results of the hierarchical clustering.
- *d(pairs):* it denotes distance between pairs.
- *linkageCriteria:* it denotes computations of the distances between clusters.
- *cThreshold:* denotes how many elements must be in clusters.
- *distanceThreshold:* it is a limitation to use the more similar items for prediction.

Table 4.1: Distance array representation

index	0	1	2	3	4	5
distance	$d(m_0,m_1)$	$d(m_0,m_2)$	$d(m_0,m_3)$	$d(m_1,m_2)$	$d(m_1,m_3)$	$d(m_2,m_3)$

Algorithm 2 PreProcessing Algorithm

Input: $dataFile$

output: $hclusters$

```
 1: function PREPROCESSING($dataFile$)
 2:     while not end of the $dataFile$ do
 3:         $htableandratings \leftarrow$ read $dataFile$
 4:     end while
 5:     if $UseDemographicInformation == True$ then
 6:         while not end of the $demFile$ do
 7:             $demtable \leftarrow$ read $demFile$
 8:         end while
 9:     end if
10:     for all $userPair \in ((m_0, m_1), (m_0, m_2), (m_0, m_3), \therefore, (m_{m-1}, m_m)$ do
11:         if $UseDemographicInformation == True$ then
12:             $distanceArray \leftarrow d(userPair) \cup d(demographic(userPair))$
13:         else
14:             $distanceArray \leftarrow d(userPair)$
15:         end if
16:     end for
17:     $clusters \leftarrow HierarchicalClustering(distanceArray)1$
18: end function
```

In addition, we also use demographic information, age, gender and occupation, while calculating distances. We combine the ratings of the users and the demographic features of the users to find similar users. While using demographic features, ages are divided into intervals 4.2. Moreover, users are evaluated with respect to demographic features using equivalence between values.

Table 4.2: Intervals for age

1	age <18
2	17 <age <25
3	24 <age <35
4	34 <age <45
5	44 <age <55
6	49 <age <56
7	age >55

4.2 Recommendation Algorithms Using Near-Clique

In this section, we present two proposed prediction algorithms. Before going into the details of the prediction algorithms, we explain the initialization part which contains clustering operations.

4.2.1 Initialization

After preprocessing part 4.1, we have two dimensional array that contains cluster information with bottom-up approach. The first elements of the array stores the most similar clusters and the last elements of the array stores the least similar clusters. Thus, we start from the beginning of the array. The array contains cluster information and distances between these clusters as shown in the following table 4.3:

In this part, since we aim to find near-cliques, the most similar clusters are taken one by one and near-cliques are formed.

4.2.2 Simultaneous Matrix Completion Algorithm

Simultaneous Matrix Completion Algorithm (SIMCA) completes missing values of the specific columns for all users simultaneously. In the beginning, clusters are combined step by step. At each step, their unrated elements are determined by the algorithm using hash table. Then, unrated elements are collected into the list that contains only missing values indexes in the clusters. In order to reach the prediction phase,

Table 4.3: Two dimensional array contains clusters and their distances. C_i, m and $d(,)$ denotes clusters, number of users and distance between clusters, respectively.

index	cluster1	cluster2	distance
0	C_1	C_2	$d(C_1,C_2)$
1	C_3	C_4	$d(C_3,C_4)$
2	C_5	C_6	$d(C_5,C_6)$
...			
...			
...			
m-4	C_{i-6}	C_{i-5}	$d(C_{i-6},C_{i-5})$
m-3	C_{i-3}	C_{i-2}	$d(C_{i-2},C_{i-3})$
m-2	C_{i-1}	C_i	$d(C_{i-1},C_i)$

the number of the rated elements must be greater than the threshold. If they satisfy the condition, they can be completed using average of the rated elements. In order to obtain better results, the algorithm can be repeated many times. For greater than one call, the algorithm uses two conditions: the number of the limited unrated elements and the number of the limited rated elements. The former determines unrated elements using hash table. If a few unrated elements exist, predictions are made by using average of the rated elements. We use this condition to find near-cliques as much as possible. The latter is used at the last call since the former limitation does not cover all of the situations to complete the matrix. It uses the limited rated elements to make prediction. The details of the algorithm is given in Algorithm 3.

To clarify the algorithm, we present an example run here. Figure 4.2 shows the ratings of the users for the movies. Rows denote users, columns denotes movies and indexes denote ratings. "Question marks(?)" denote unrated items. In the beginning, hash table is created. It is shown in table 4.4. Also figure 4.3 represents near-cliques representation.

Then, the distances are computed for all user pairs. Table 4.5 shows distances where the less distance the more similarity among users.

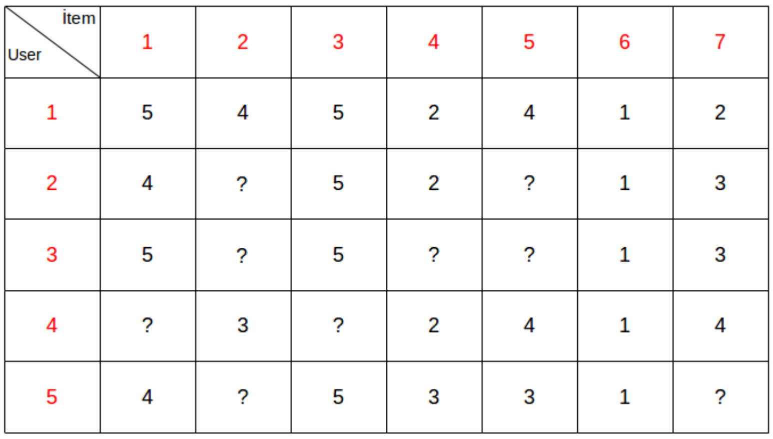
And then, hierarchical clustering is performed by using these similarities. This operation creates an array that contains clusters and their distances.

Algorithm 3 Simultaneous Matrix Completion Algorithm

Input: $dataFile$, $cThreshold$

output: $predictions$

```
1:  function ALGORITHM1($dataFile$)
2:      for $i := 1 \leftarrow 0, repetition$ do
3:          $clusterArray \leftarrow$ PREPROCESSING($dataFile$)
4:          for $j := 1 \leftarrow 0, len(clusterArray)$ do
5:              $list \leftarrow unratedItems(clusterArray[j])$
6:              for all $iteminlist$ do
7:                  if $i\ != repetitions$ and $len(list(item)) < threshold$ then
8:                      $item \leftarrow$ average of rated elements
9:                  else if $len(ratedElements) > cThreshold$ then
10:                     $item \leftarrow$ average of rated elements
11:                 end if
12:             end for
13:         end for
14:     end for
15: end function
```



User \ item	1	2	3	4	5	6	7
1	5	4	5	2	4	1	2
2	4	?	5	2	?	1	3
3	5	?	5	?	?	1	3
4	?	3	?	2	4	1	4
5	4	?	5	3	3	1	?

Figure 4.2: Experimental data includes ratings

It can be seen in table 4.6.

Table 4.4: Hash table for unrated elements

1	$\rightarrow$	4		
2	$\rightarrow$	2	3	5
3	$\rightarrow$	4		
4	$\rightarrow$	3		
5	$\rightarrow$	2	3	
7	$\rightarrow$	5		

Table 4.5: The distances between users

	1	2	3	4
2	0.21			
3	0.23	0.04		
4	0.37	0.67	0.75	
5	0.14	0.16	0.23	0.64

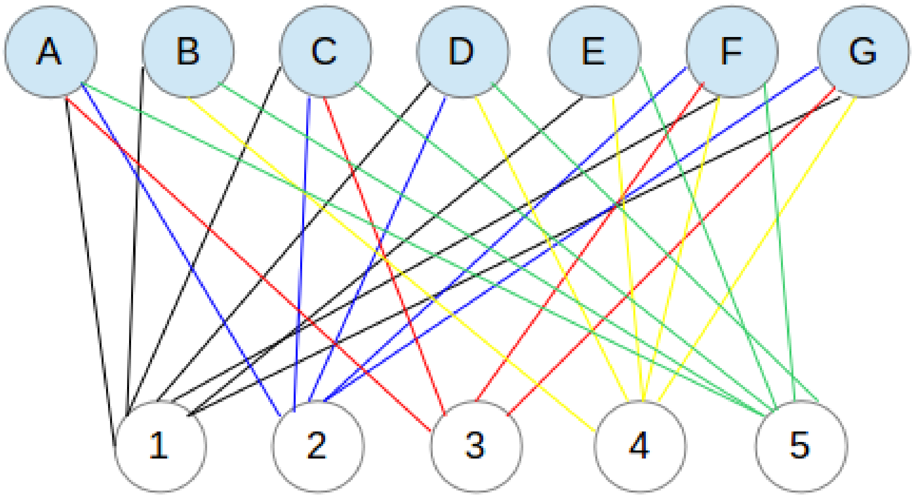


Figure 4.3: Near-clique representation of an example

To make prediction, prediction phase starts with merging clusters until it reaches enough cluster counts. Suppose that, at least four elements must be in the clusters to make the algorithm be able to predict. In the third step, this is satisfied by clusters

Table 4.6: Array that contains clusters hierarchically

	$Cluster_1$	$Cluster_2$	Distances between the clusters	Explanation
1	2	3	0.04	2,3 $\rightarrow$ 6
2	1	5	0.13	1,5 $\rightarrow$ 7
3	6	7	0.23	1,2,3,5 $\rightarrow$ 8
4	4	8	0.75	

$< 1, 2, 3, 5 >$. Then, the unrated items are determined which are $< 2, 4, 5, 7 >$. The important point that since $user$ 4 does not exist in the processed clusters, $item$ 3 is not included in the unrated item list. And then, the unrated elements are predicted if enough rated elements exist in the clusters. Assume that, at least 3 rated items must be in the clusters for prediction. According to these threshold, $items < 4, 7 >$ are predicted by computing average of the rated items. For item 4, the average of the rated $users < 1, 2, 5 >$ is 2.33. Since $item$ 4 has an unrated element, The only $user$ 3 is predicted. However, $movie$ 2 and $movie$ 5 is not predicted due to insufficient number of the rated items.

4.2.3 Stepwise Matrix Completion Algorithm

Unlike the first algorithm 4.2.2, Stepwise Matrix Completion Algorithm (STMCA) completes unrated elements one by one.

As in the first algorithm, hash table is created and clusters are processed step by step. In addition, this algorithm finds the unrated elements too. After all, this algorithm uses conditions and thresholds to achieve better results. The difference with the first algorithm is that this algorithm finds ratings for each user separately. Thus, the running time of the second algorithm is greater than the first one. In addition to the first algorithm thresholds, distance threshold is also used in the second algorithm while making predictions. In the prediction phase, rated users are compared according to the their distances which are computed in the preprocessing section. In this step, the algorithm eliminates users who are less similar.

To illustrate how this algorithm works, we display an example run. As in the first

Algorithm 4 Stepwise Matrix Completion Algorithm

Input: $dataFile$, $cThreshold$, $distanceThreshold$

output: $predictions$

```
1: function ALGORITHM2($dataFile$)
2:     for $i := 1 \leftarrow 0, repetition$ do
3:         $clusterArray \leftarrow$ PREPROCESSING($dataFile$)
4:         for $j := 1 \leftarrow 0, len(clusterArray)$ do
5:             $list \leftarrow unratedItems(clusterArray[j])$
6:             for all $keyinlist$ do
7:                 for all $userinkey$ do
8:                     if $i\ != repetitions$ and $len(list(item)) < cThreshold$ then
9:                         if $distance(betweenUsers) > distanceThreshold$ then
10:                            $item \leftarrow$ average of rated elements
11:                        end if
12:                    else if $len(ratedElements) > threshold$ then
13:                        if $distance(betweenUsers) > distanceThreshold$ then
14:                            $item \leftarrow$ average of rated elements
15:                        end if
16:                    end if
17:                end for
18:            end for
19:        end for
20:    end for
21: end function
```

algorithm example, suppose that $user < 1, 2, 3, 5 >$ satisfy the conditions. Then the algorithm determines missing values which are $item < 2, 4, 5, 7 >$. For $item$ 2, since there is not enough rated elements, any prediction is not performed. Then, $item\ 4$ is evaluated. $Item\ 4$ with $user3$ satisfies the condition that enough rated elements exist. To make prediction, the distances between $user\ 3$ and $users\ 1, 2, 5$ are compared. Assume that, we set the distance threshold to 0.5. All users satisfy the distance threshold. Thus, $Item\ 4$ with $user3$ is predicted by computing the average of the rated $users < 1, 2, 5 >$ which is 2.33.

CHAPTER 5

EXPERIMENTAL RESULTS

In this chapter, we present the experimental results of our proposed algorithms.

Since there exists many parameters for our algorithms, we present only the optimal results in SIMCA 5.2, STMCA 5.3 and Demographic Results 5.4. On the other hand, we include a general evaluation in section 5.5.

5.1 Experimental Setup

Data had been distributed into five distinct parts and all experimental results are obtained using these data sets which are called as $test1$, $test2$, $test3$, $test4$ and $test5$. Data sets contain two parts which are named as training data and test data. While computing predictions, MAE and RMSE results are calculated simultaneously. For experimental results, we determine the value of some parameters after an exhaustive period of tests. We observe that as the number of round increases, after a convergence point similar results are obtained with the smaller rounds. Moreover, time consuming increases in parallel with the algorithm call count. Also, in order to obtain better results, we tried to set the optimal cluster count. This is due to the fact that, higher and smaller cluster counts can give rise to poor results. Thus, according to the lots of the test results, in these experiments, we set iterative call count to 3, minimum cluster count to 5, linkage criteria to complete and similarity metric to cosine similarity.

All operations are performed on a computer which has INTEL i7 processor with 8 cores and 8 GB memory.

5.2 Simultaneous Matrix Completion Algorithm

Simultaneous Matrix Completion Algorithm experimental results can be seen in figure 5.1, figure 5.2, figure 5.3, figure 5.4, figure 5.5 and figure 5.6.

The figures show that the first round gives better results due to the near-cliques which are found in this step. MAE and RMSE results generally increase as the iterative call count increases. They also increase as the number of rated items increases. This is due to the fact that, when similarity decreases, it is hard to find the near-cliques in the cluster array. In the first call, we used distinct data sets. Since their sparse data locations and similarities effect the clustering phase and near-cliques, MAE and RMSE graphics have different patterns for different datasets. At the end of the process, predictions converge to the similar accuracy ratios which can be seen in figure 5.3 and figure 5.6 . Moreover, almost overall test data is predicted between $\%98,5$ and $\%99$ ratio. The running time of the algorithm is 6.5 minutes on the average. The most time-consuming parts are similarity calculations and hierarchical clustering. The prediction part is faster than the similarity and clustering. On the other hand, not only test data is predicted but also other missing indexes are predicted. The compared results are shown in the table 5.1 in detail.

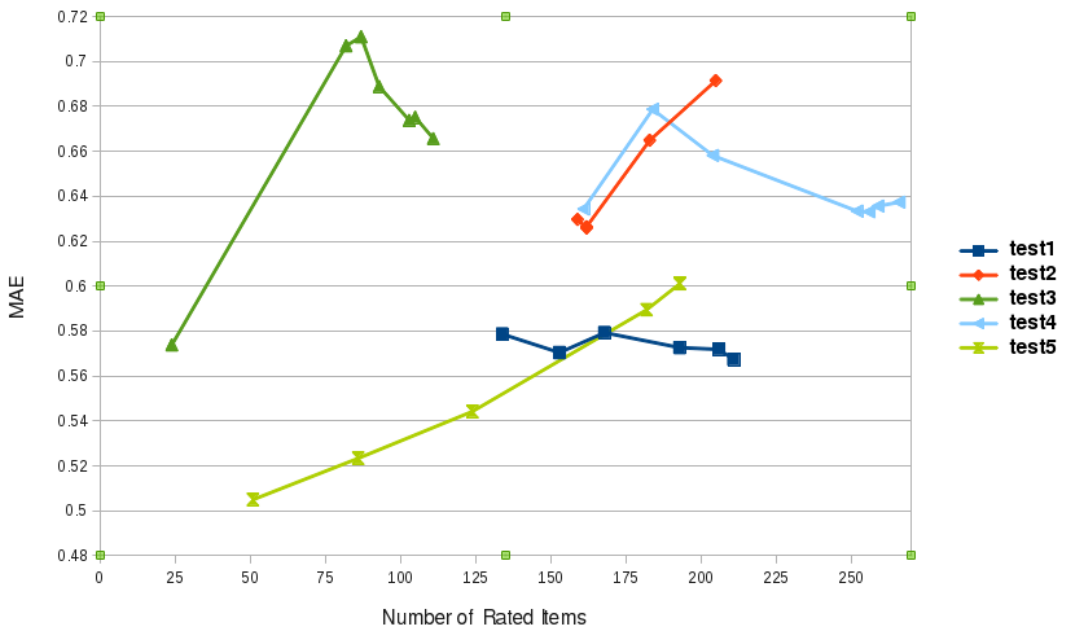


Figure 5.1: MAE results of the first step of SIMCA

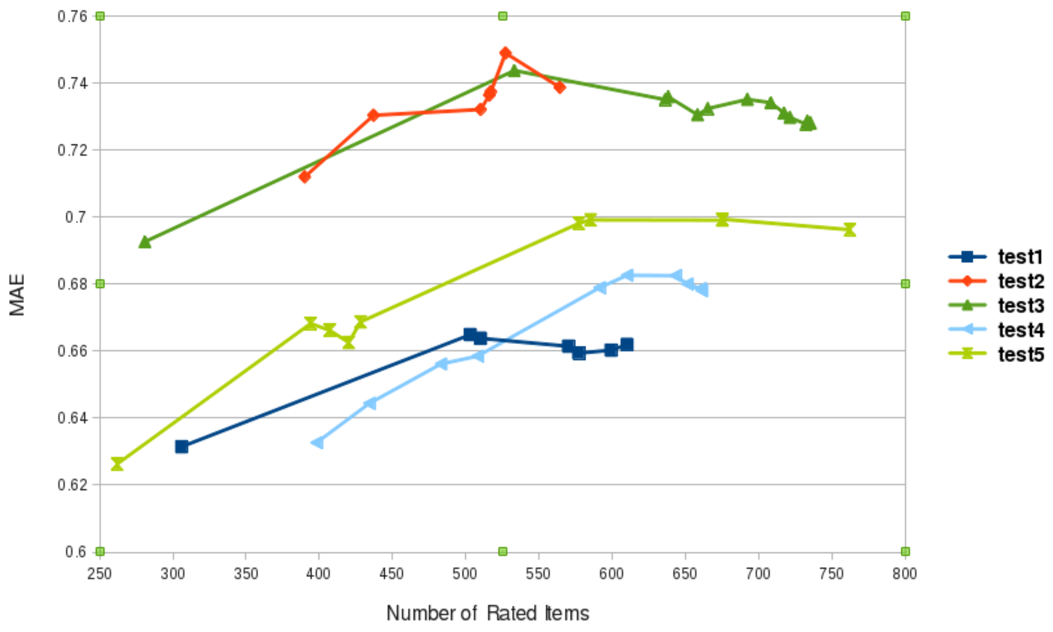


Figure 5.2: MAE results of the second step of SIMCA

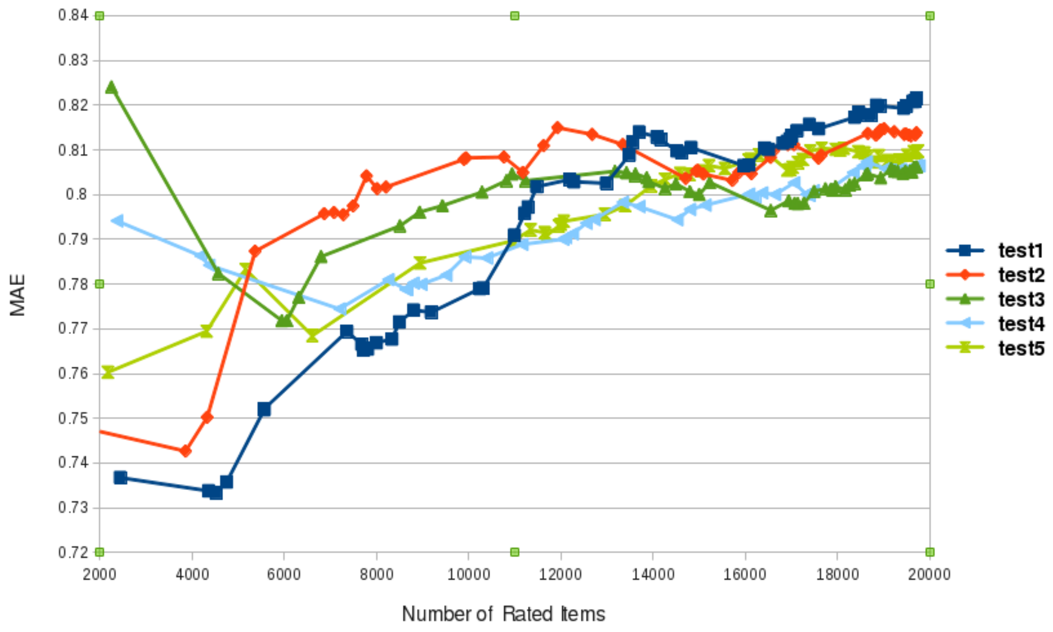


Figure 5.3: MAE results of the last step of SIMCA

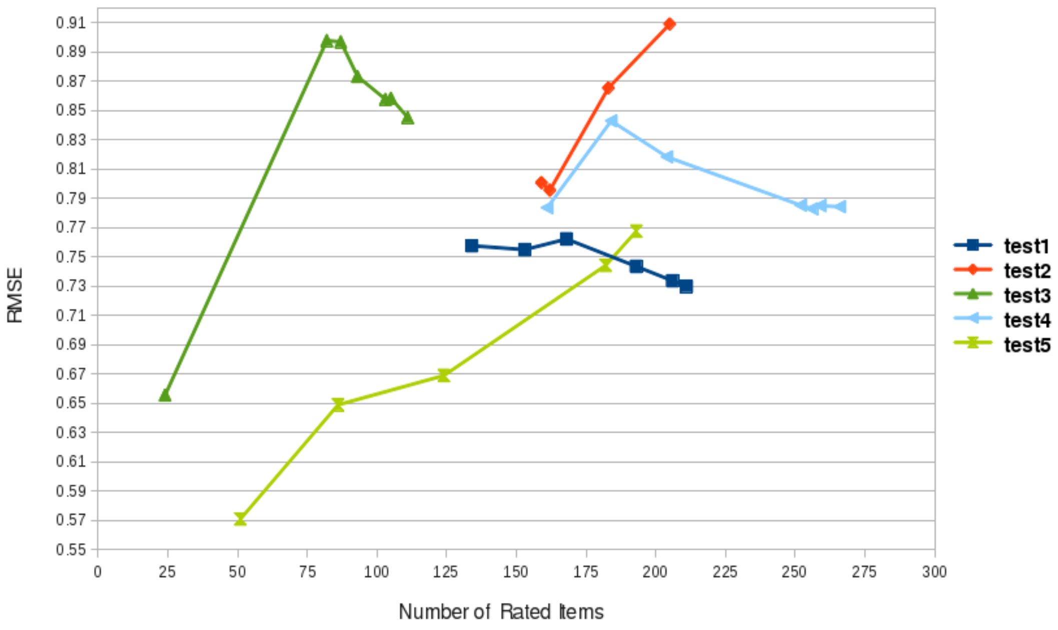


Figure 5.4: RMSE results of the first step of SIMCA

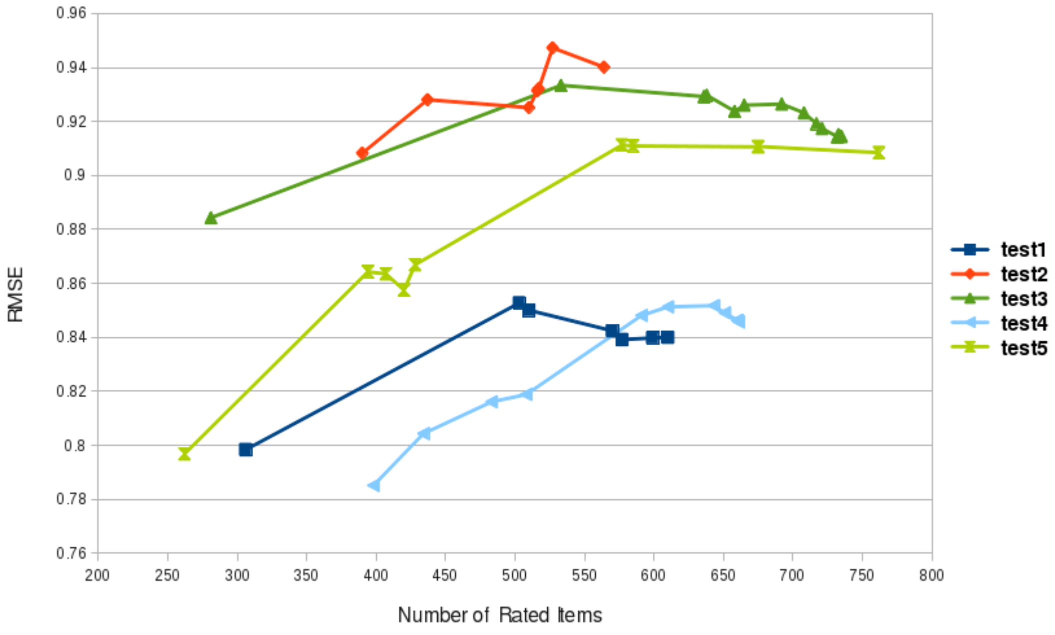


Figure 5.5: RMSE results of the second step of SIMCA

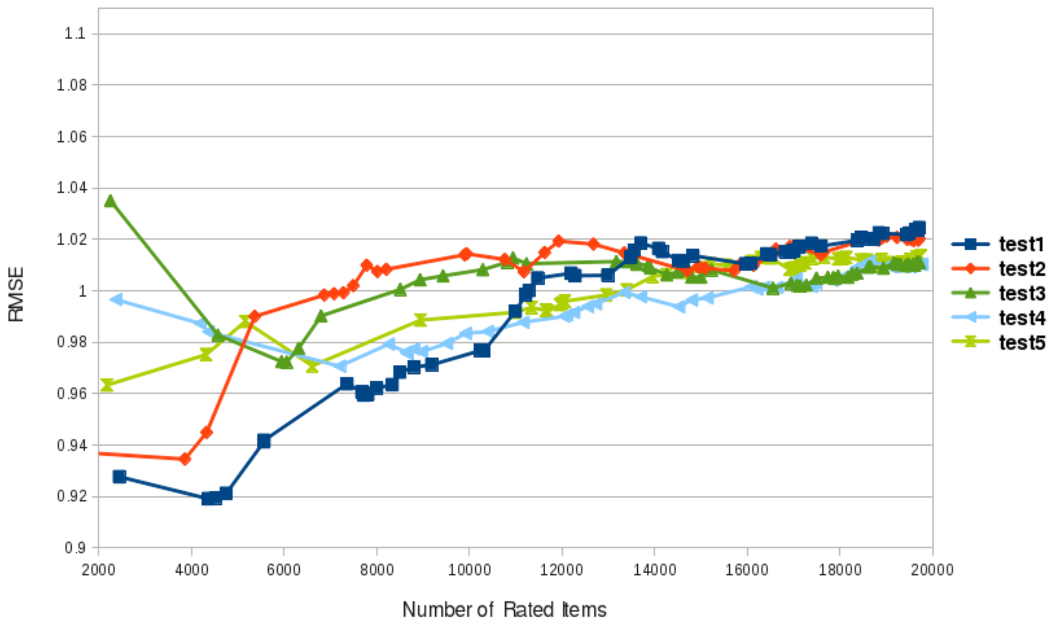


Figure 5.6: RMSE results of the last step of SIMCA

Table 5.1: Simultaneous Matrix Completion Algorithm results

	Round 1			Round 2			Round 3			Processing Time(seconds)
	MAE	RMSE	Number of Predicted Items	MAE	RMSE	Number of Predicted Items	MAE	RMSE	Number of Predicted Items	
test1	0.56	0.72	211	0.66	0.84	610	0.82	1.02	19715	401
test2	0.69	0.90	205	0.73	0.94	564	0.81	1.02	19718	392
test3	0.66	0.84	111	0.72	0.91	735	0.80	1.01	19723	386
test4	0.63	0.78	266	0.67	0.84	661	0.80	1.01	19780	361
test5	0.60	0.76	193	0.69	0.90	762	0.81	1.01	19748	377

5.3 Stepwise Matrix Completion Algorithm

STMCA experimental results can be seen in figure 5.7, figure 5.8, figure 5.9, figure 5.10, figure5.11 and figure 5.12.

The main difference between the SIMCA and the STMCA is the running time. Since the STMCA computes predictions one by one, the running time is about eight times longer than SIMCA. In the first call, $test3$ has the most remarkable results. Although number of predicted elements is few, the prediction ratio is successful for the most similar users due to near-cliques. As in the SIMCA, almost overall test data is predicted about $\%97.5$. Since STMCA uses extra threshold that check distances among users, the number of predicted items in the STMCA is less than SIMCA. Therefore, especially in the first and second round predicted elements remained limited according to the SIMCA. In third call, algorithm converges to the similar points. The detailed results are shown in table 5.2.

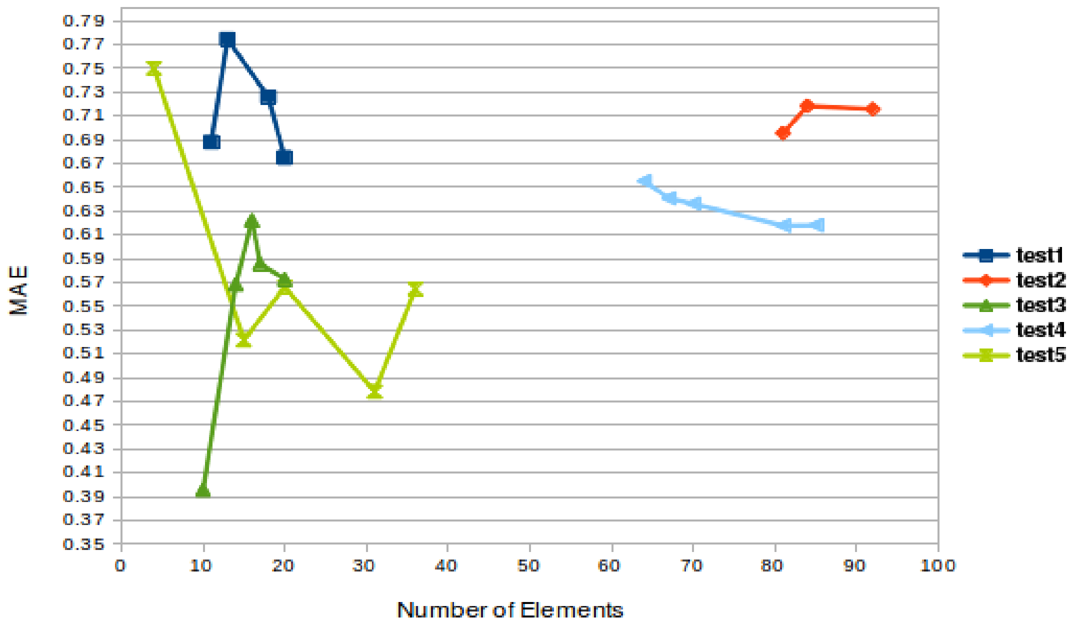


Figure 5.7: MAE results of the second step of STMCA

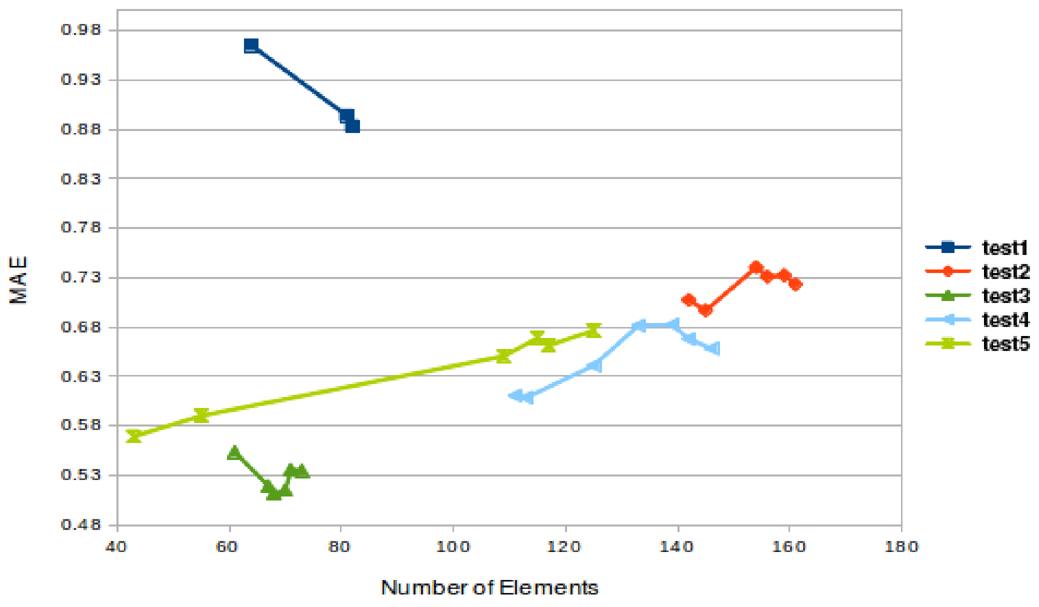


Figure 5.8: MAE results of the second step of STMCA

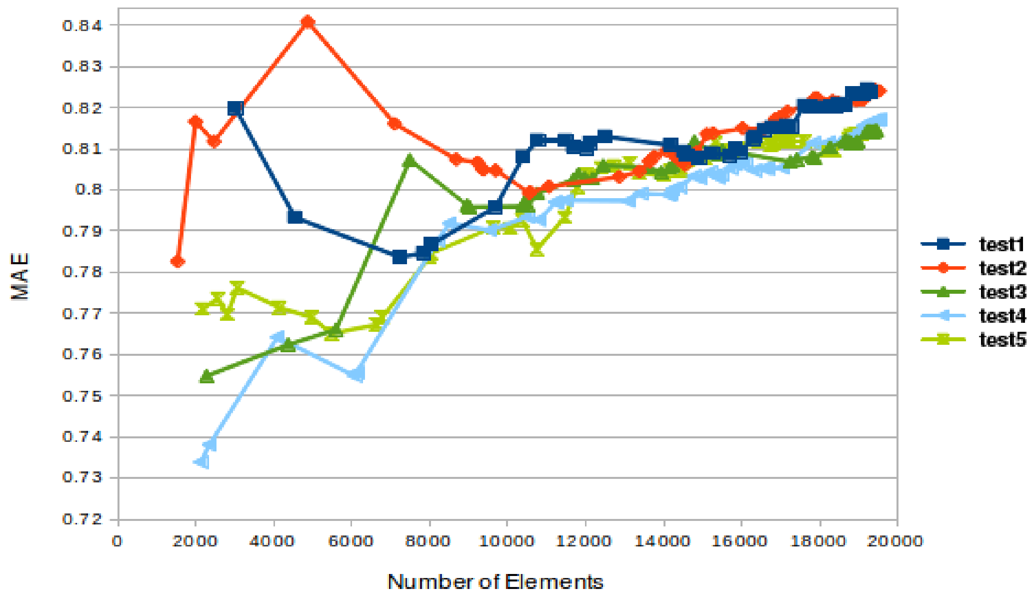


Figure 5.9: MAE results of the last step of STMCA

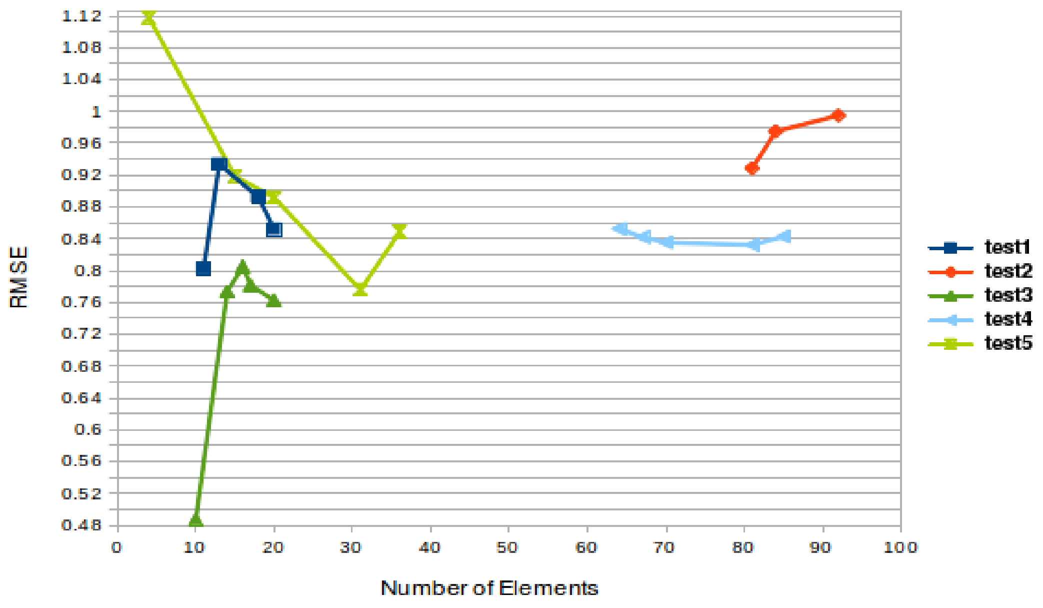


Figure 5.10: RMSE results of the first step of STMCA

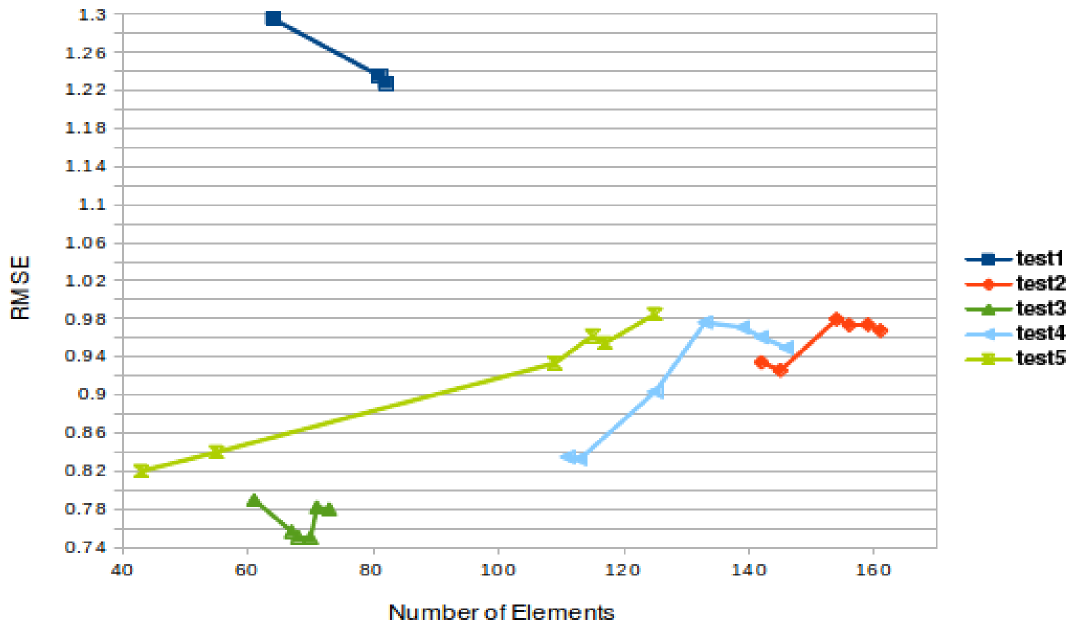


Figure 5.11: RMSE results of the second step of STMCA

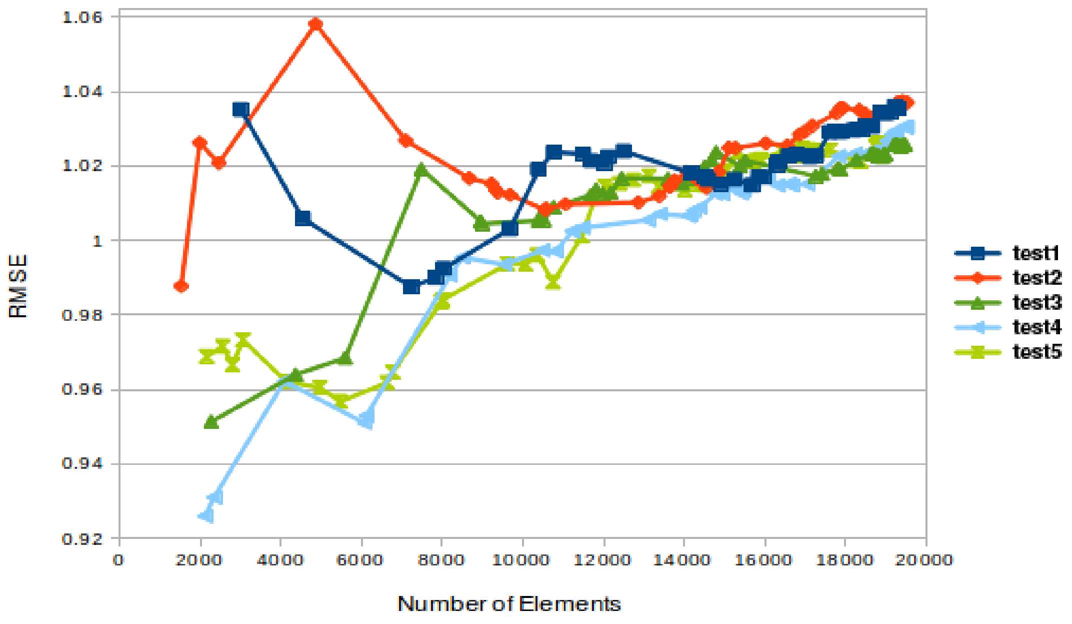


Figure 5.12: RMSE results of the last step of STMCA

Table 5.2: Stepwise Matrix Completion Algorithm results

	Round 1			Round 2			Round 3			Processing Time(seconds)
	MAE	RMSE	Number of Predicted Items	MAE	RMSE	Number of Predicted Items	MAE	RMSE	Number of Predicted Items	
test1	0.67	0.85	20	0.88	1.22	82	0.82	1.03	19313	3601
test2	0.71	0.99	92	0.72	0.96	161	0.82	1.03	19528	3510
test3	0.57	0.76	20	0.53	0.77	73	0.81	1.02	19481	3153
test4	0.62	0.84	85	0.66	0.95	146	0.82	1.03	19526	2767
test5	0.56	0.85	36	0.67	0.98	125	0.81	1.02	19272	1902

5.4 Demographic Experimental Results

Demographic features which are age, gender and occupation are performed in this study while calculating similarities among users. To compute similarities, ratings and demographic features are combined in different proportions.

5.4.1 Simultaneous Matrix Completion Algorithm

The results are shown in figure 5.13, figure 5.14, figure 5.15, figure 5.16, figure 5.17 and figure 5.18.

The experimental results using demographic features are similar with normal ones. Two results converges to the similar points. However, in the first call, the results without demographic features are slightly better than demographic. The details of results are shown in table 5.3 in detail. As in 5.2 and 5.3, test data is predicted almost all. The algorithm running time using demographic features are not differ from SIMCA without demographic fuatures.

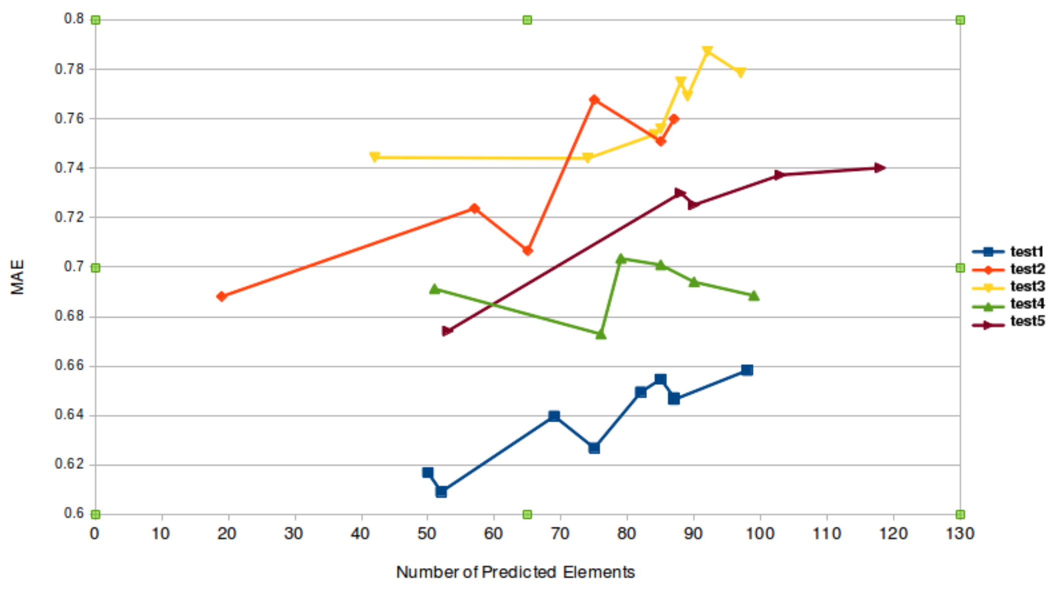


Figure 5.13: MAE results of the second step of STMCA

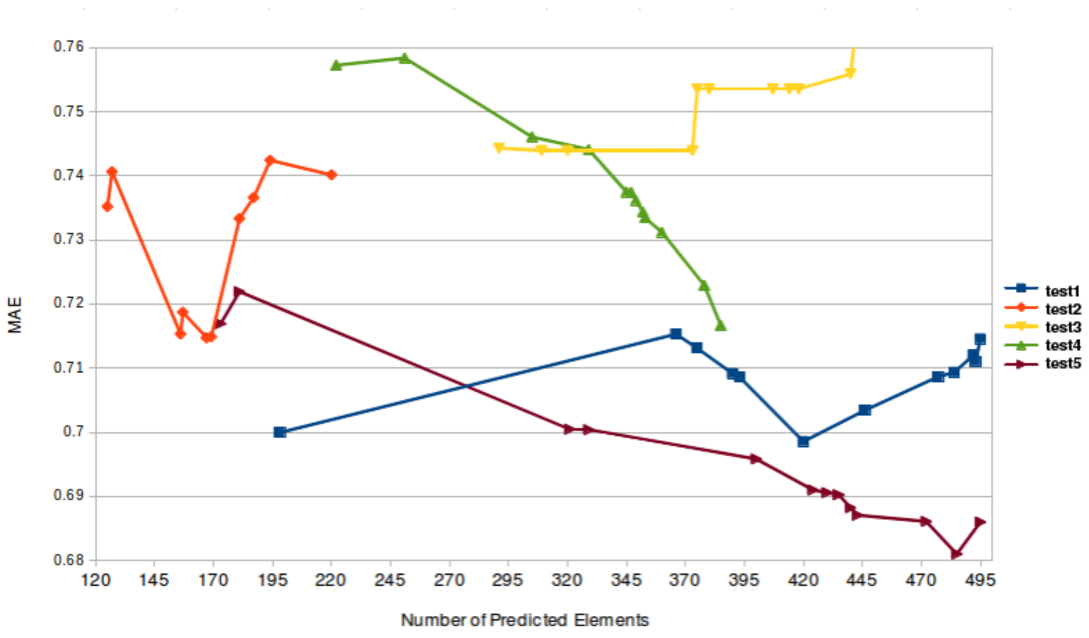


Figure 5.14: MAE results of the second step of STMCA

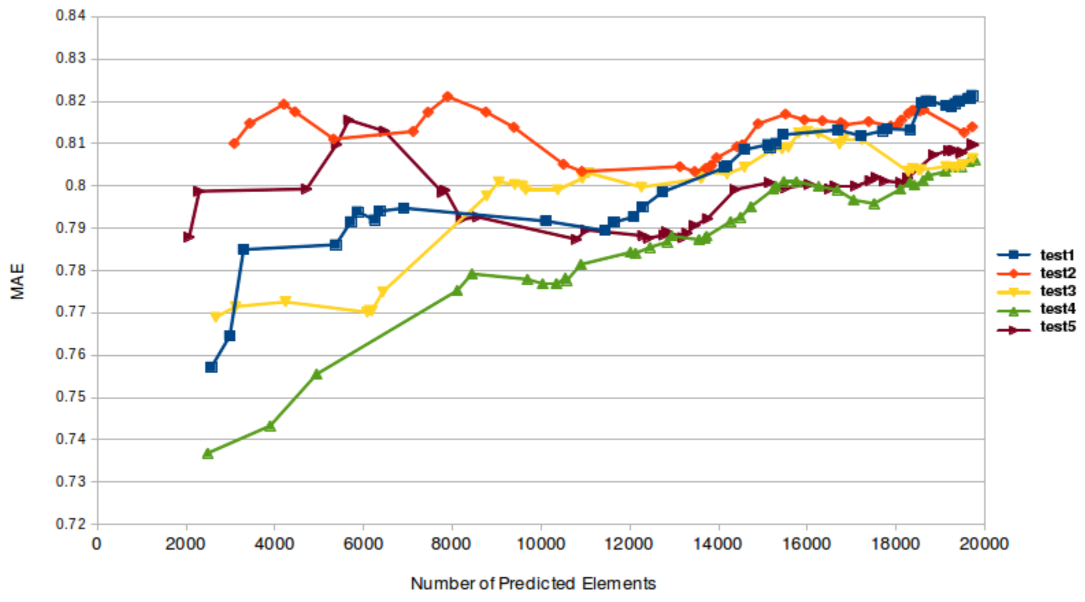


Figure 5.15: MAE results of the last step of STMCA

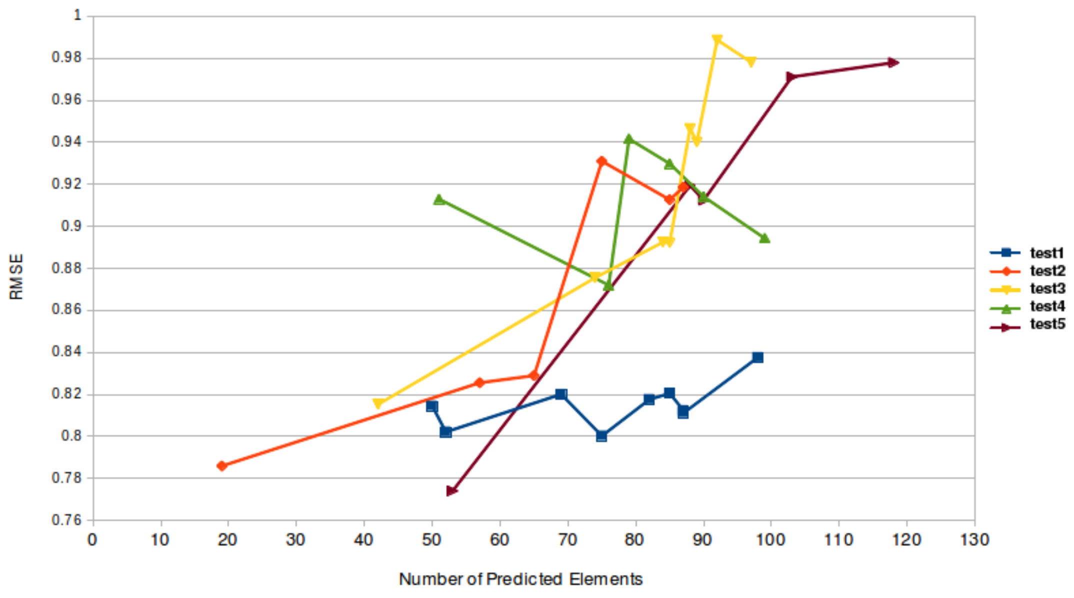


Figure 5.16: RMSE results of the first step of STMCA

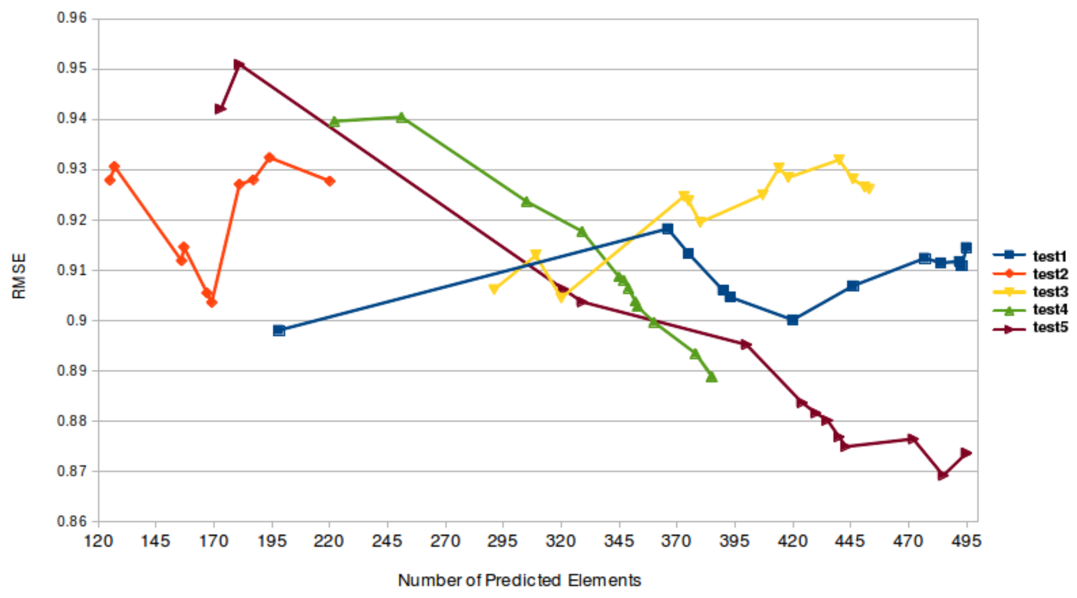


Figure 5.17: RMSE results of the second step of STMCA

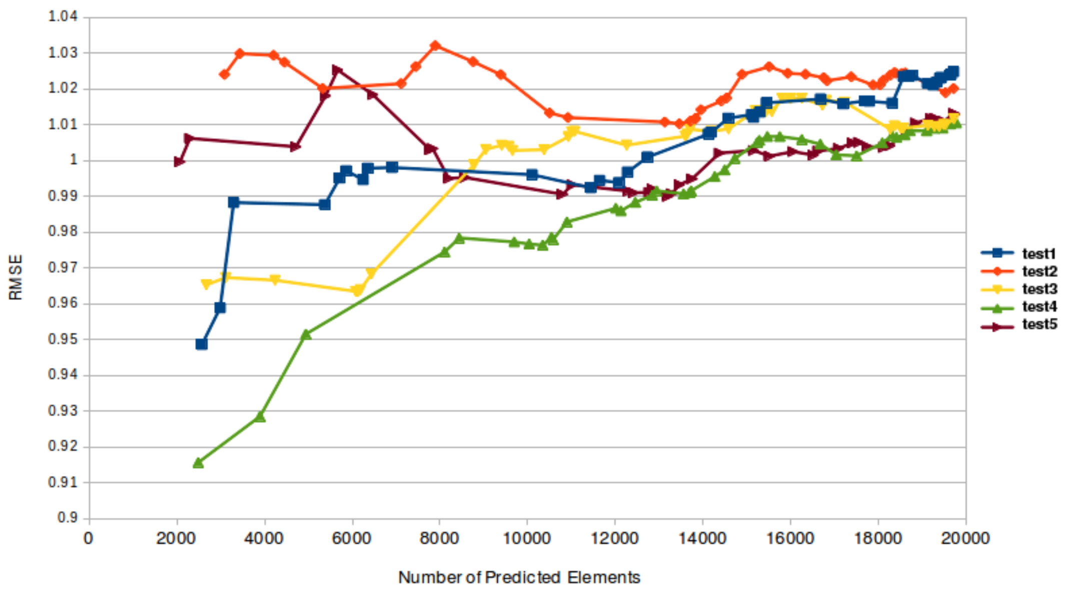


Figure 5.18: RMSE results of the last step of STMCA

5.4.2 Stepwise Matrix Completion Algorithm

The results are shown in figure 5.19, figure 5.20, figure 5.21, figure5.22, figure 5.23 and figure 5.24.

STMCA results are slightly worse than others in the initial calls. The reason for that, the near-cliques are not found with that similarity calculation. Also, the number of the predicted items and the running time of the algorithm with demographic features are slightly better than STMCA 5.3. The details of the results are shown in table 5.3.

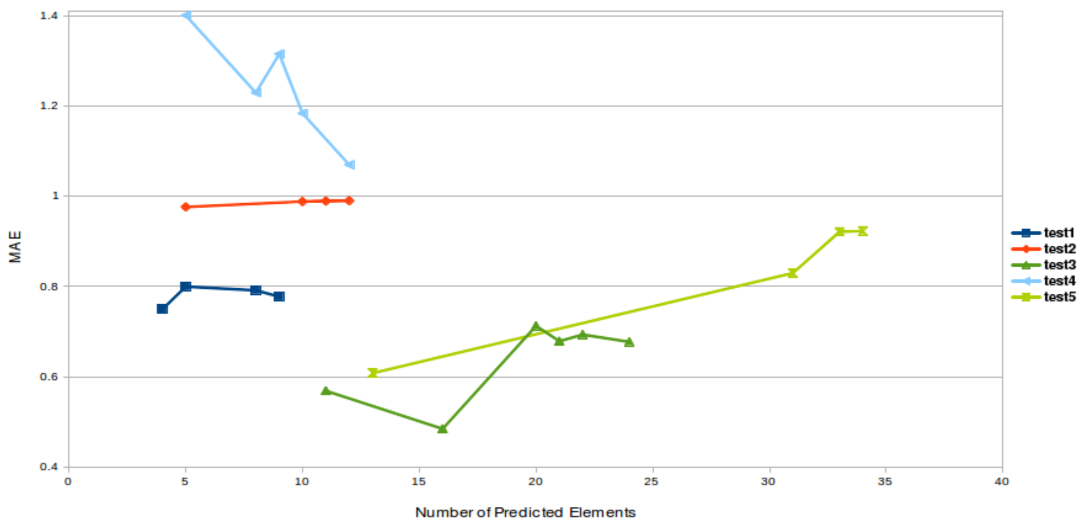


Figure 5.19: MAE results of the second step of STMCA

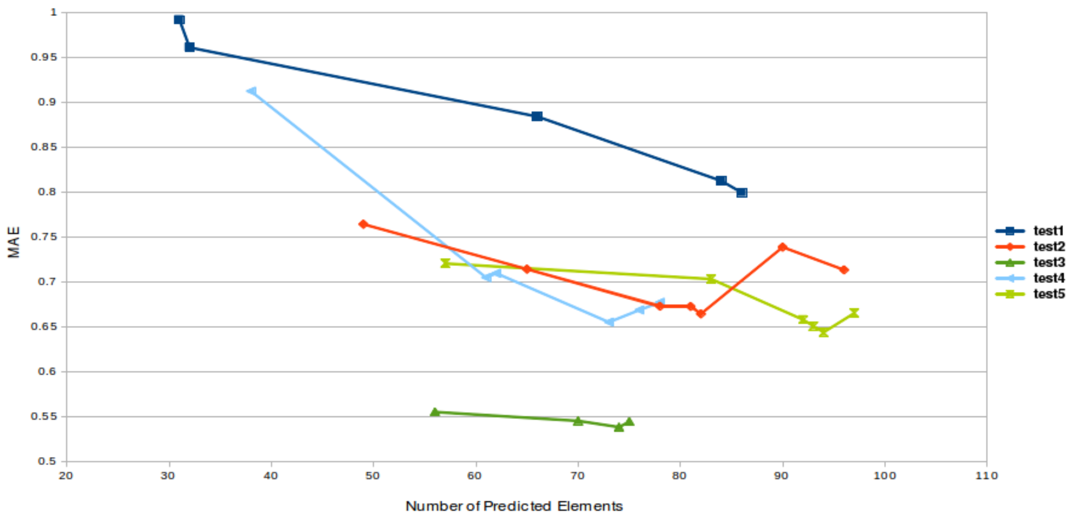


Figure 5.20: MAE results of the second step of STMCA

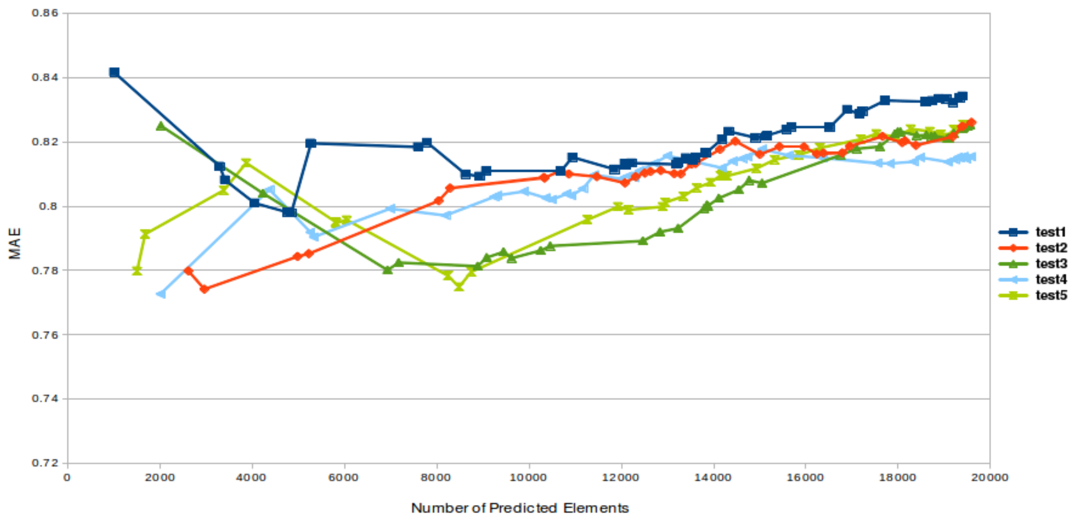


Figure 5.21: MAE results of the last step of STMCA

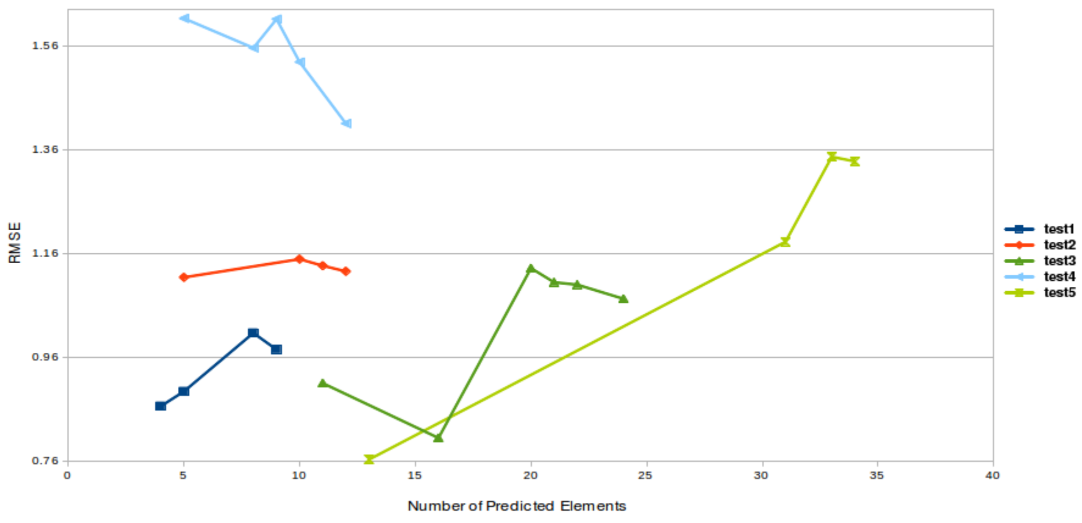


Figure 5.22: RMSE results of the first step of STMCA

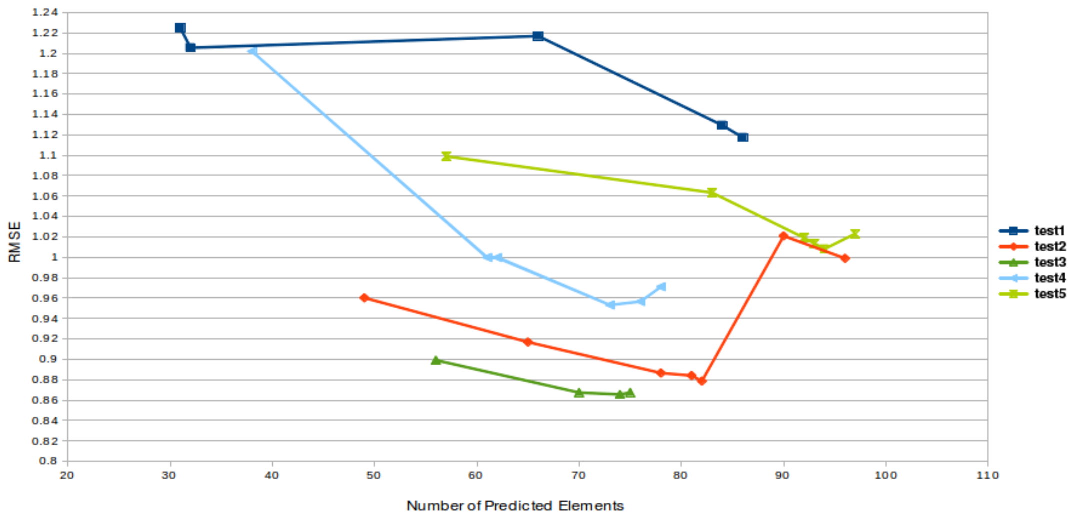


Figure 5.23: RMSE results of the second step of STMCA

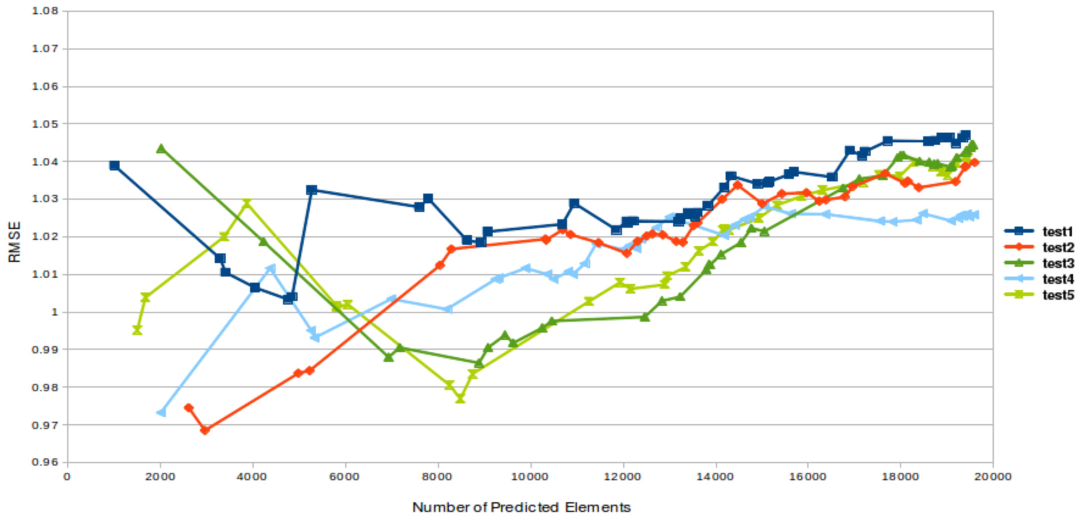


Figure 5.24: RMSE results of the last step of STMCA

Table 5.3: SIMCA and 2 results with demographic features

		Round 1			Round 2			Round 3			Processing Time
		MAE	RMSE	Number of Predicted Items	MAE	RMSE	Number of Predicted Items	MAE	RMSE	Number of Predicted Items	
SIMCA	test1	0.65	0.83	98	0.71	0.91	495	0.82	1.02	19715	304
	test2	0.76	0.92	87	0.74	0.93	220	0.81	1.02	19718	309
	test3	0.78	0.98	97	0.74	0.93	453	0.81	1.01	19723	308
	test4	0.68	0.89	99	0.72	0.89	385	0.81	1.01	19780	301
	test5	0.74	0.97	118	0.69	0.87	495	0.81	1.01	19748	377
STMCA	test1	0.77	0.97	9	0.79	1.11	86	0.83	1.04	19397	2375
	test2	0.99	1.12	12	0.71	1.0	96	0.83	1.04	19593	2080
	test3	0.67	1.07	24	0.54	0.86	75	0.82	1.04	19561	2218
	test4	1.07	1.4	12	0.68	0.97	78	0.82	1.02	19584	2374
	test5	0.92	1.33	34	0.66	1.02	97	0.82	1.04	19455	1892

5.5 General Comparison

During the experiments, different options such as similarity metrics, linkage criteria and different thresholds are used. Among of them, optimal ones are chosen. For linkage criteria, we use $single$, $average$ and $complete$ for the experiments. However, $single$ and $average$ criteria are not better than $complete$ in terms of prediction accuracy and the running time of the algorithm. Also different similarity metrics are implemented. Rather than euclidean distance and PPMCC, cosine similarity gives better results for the two algorithms.

Table 5.4: The results for linkage criteria: single and average

Average						
		test1	test2	test3	test4	test5
SIMCA	MAE	0.82	0.81	0.81	0.81	0.81
	RMSE	1.02	1.02	1.01	1.01	1.01
	The number of the predicted items	19715	19718	19723	19780	19748
	The processing time(seconds)	728	713	633	602	660
STMCA	MAE	0.82	0.82	0.81	0.81	0.81
	RMSE	1.03	1.03	1.03	1.02	1.02
	The number of the predicted items	19327	19538	19499	19537	19295
	The processing time(seconds)	4735	3491	2885	2888	3052
Single						
		test1	test2	test3	test4	test5
SIMCA	MAE	0.82	0.81	0.81	0.81	0.81
	RMSE	1.02	1.02	1.01	1.01	1.01
	The number of the predicted items	19715	19718	19723	19780	19748
	The processing time(seconds)	2232	3400	3422	3537	3426
STMCA	MAE	0.82	0.82	0.81	0.81	0.81
	RMSE	1.03	1.03	1.03	1.02	1.01
	The number of the predicted items	19345	19542	19500	19538	19291
	The processing time(seconds)	50033	47298	45862	48230	49178

To compare our algorithms, we implement Singular Value Decomposition(SVD). Firstly, data is decomposed using SVD. Using singular values, dimensionality reduction is performed. Then, distances among the users are computed using the reduced matrix and predictions are made one by one. Rather than hierarchical clustering, SVD results are not predicted hierarchically. For this reason, only the overall results are shown in the table 5.5. There are three results shown in the table which the overall data is covered in $\%90$ $\%45$, $\%32$ ratios. The results show that SVD has similar accuracy ratios but the running time of the algorithm is longer than our algorithms.

Table 5.5: The comparative results

		u1	u2	u3	u4	u5
SIMCA	MAE	0.82	0.81	0.81	0.81	0.81
	RMSE	1.02	1.02	1.01	1.01	1.01
	The number of the predicted items	19715	19718	19723	19780	19748
	The processing time(seconds)	401	392	386	361	377
SVD(90%)	MAE	0.83	0.82	0.81	0.82	0.83
	RMSE	1.04	1.03	1.01	1.04	1.05
	The number of the predicted items	19866	19840	19848	2000	19851
	The processing time(seconds)	87516	87871	86783	90006	88032
SVD(45%)	MAE	0.82	0.81	0.80	0.81	0.81
	RMSE	1.02	1.02	1.00	1.02	1.03
	The number of the predicted items	19873	19746	19847	19901	19856
	The processing time(seconds)	6812	6558	7008	6798	6746
SVD(32%)	MAE	0.82	0.81	0.80	0.81	0.82
	RMSE	1.02	1.02	1.01	1.02	1.03
	The number of the predicted items	19875	19849	19849	19900	19859
	The processing time(seconds)	1052	1097	1130	1136	1145

CHAPTER 6

CONCLUSION AND DISCUSSION

In this study, we design heuristic collaborative filtering algorithms for recommendations. Hierarchical clustering, similarity metrics and near-cliques constitute the main parts of the algorithms. We aim to find near-cliques using hierarchical clustering and similarity metrics since near-cliques denote strong connectivity among users for predictions. Also, to make prediction to the users, at least one connected item must be found in the data set for unrated item. Otherwise, without accepting this reality, predictions will not be trustworthy. To obtain better results, our algorithms works based on the parameters which are thresholds for similarity comparison and the number of the clusters that starts the prediction phase. Actually, the experimental results show that predictions, especially the first ones, give high accuracies since initial steps include near-cliques. In each step, while increasing the distances among users, accuracies are converged to the similar points for all data sets. Moreover, to compare our algorithms, we implemented an algorithm using SVD. Although the overall SVD results are similar with our algorithms, there are two advantages for our algorithms which are the running time and initial predictions. Especially, our first algorithm running time and accuracy ratio has remarkable results.

In the future, we aim to add new similarity metrics for clustering operations. Since existing similarity metrics does not cover all of the situations, a new similarity metrics may provide better pairing among users. Also, using parallel algorithms, the much faster results may be obtained.

REFERENCES

[1] Movielens dataset, GroupLens Research. `http://files.grouplens.org/datasets/movielens/ml-100k.zip`. Accessed: 2014-08-05.

[2] H. J. Ahn. A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem. *Information Sciences*, 178(1):37–51, 2008.

[3] R. M. Bell and Y. Koren. Improved neighborhood-based collaborative filtering. In *KDD Cup and Workshop at the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. sn, 2007.

[4] J. Bobadilla, A. Hernando, F. Ortega, and A. Gutiérrez. Collaborative filtering based on significances. *Information Sciences*, 185(1):1–17, 2012.

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

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

[7] X. Cai, M. Bain, A. Krzywicki, W. Wobcke, Y. S. Kim, P. Compton, and A. Mahidadia. Collaborative filtering for people to people recommendation in social networks. In *AI 2010: Advances in Artificial Intelligence*, pages 476–485. Springer, 2011.

[8] K. Choi and Y. Suh. A new similarity function for selecting neighbors for each target item in collaborative filtering. *Knowledge-Based Systems*, 37:146–153, 2013.

[9] P. du Boucher-Ryan and D. Bridge. Collaborative recommending using formal concept analysis. *Knowledge-Based Systems*, 19(5):309–315, 2006.

[10] M. Gao, Z. Wu, and F. Jiang. Userrank for item-based collaborative filtering recommendation. *Information Processing Letters*, 111(9):440–446, 2011.

[11] B. Jeong, J. Lee, and H. Cho. Improving memory-based collaborative filtering via similarity updating and prediction modulation. *Information Sciences*, 180(5):602–612, 2010.

[12] D. Kim and B.-J. Yum. Collaborative filtering based on iterative principal component analysis. *Expert Systems with Applications*, 28(4):823–830, 2005.

[13] F. J. Király, L. Theran, R. Tomioka, and T. Uno. The algebraic combinatorial approach for low-rank matrix completion. *arXiv preprint arXiv:1211.4116*, 2012.

[14] Q. Li and B. M. Kim. Clustering approach for hybrid recommender system. In *Web Intelligence, 2003. WI 2003. Proceedings. IEEE/WIC International Conference on*, pages 33–38. IEEE, 2003.

[15] G. Linden, B. Smith, and J. York. Amazon. com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80, 2003.

[16] X. Luo, H. Liu, G. Gou, Y. Xia, and Q. Zhu. A parallel matrix factorization based recommender by alternating stochastic gradient decent. *Engineering Applications of Artificial Intelligence*, 25(7):1403–1412, 2012.

[17] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King. Recommender systems with social regularization. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 287–296. ACM, 2011.

[18] P. Melville, R. J. Mooney, and R. Nagarajan. Content-boosted collaborative filtering for improved recommendations. In *AAAI/IAAI*, pages 187–192, 2002.

[19] R. Peeters. The maximum edge biclique problem is np-complete. *Discrete Applied Mathematics*, 131(3):651–654, 2003.

[20] M. K. Rafsanjani, Z. A. Varzaneh, and N. E. Chukanlo. A survey of hierarchical clustering algorithms. *The Journal of Mathematics and Computer Science, 5*, 3:229–240, 2012.

[21] F. Ricci, L. Rokach, and B. Shapira. *Introduction to recommender systems handbook*. Springer, 2011.

[22] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.

[23] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Incremental singular value decomposition algorithms for highly scalable recommender systems. In *Fifth International Conference on Computer and Information Science*, pages 27–28. Citeseer, 2002.

[24] L. Terveen and W. Hill. Beyond recommender systems: Helping people help each other. *HCI in the New Millennium*, 1:487–509, 2001.

[25] M. G. Vozalis and K. G. Margaritis. Using svd and demographic data for the enhancement of generalized collaborative filtering. *Information Sciences*, 177(15):3017–3037, 2007.

[26] G.-R. Xue, C. Lin, Q. Yang, W. Xi, H.-J. Zeng, Y. Yu, and Z. Chen. Scalable collaborative filtering using cluster-based smoothing. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 114–121. ACM, 2005.

[27] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan. Large-scale parallel collaborative filtering for the netflix prize. In *Algorithmic Aspects in Information and Management*, pages 337–348. Springer, 2008.