HAL
open science

Optimization-based User Group Management : Discovery, Analysis, Recommendation

Behrooz Omidvar Tehrani

To cite this version:

Behrooz Omidvar Tehrani. Optimization-based User Group Management : Discovery, Analysis, Recommendation. Informatique ubiquitaire. Université Grenoble Alpes, 2015. Français. NNT : 2015GREAM038 . tel-01271933

HAL Id: tel-01271933

https://theses.hal.science/tel-01271933v1

Submitted on 9 Feb 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

UNIVERSITÉ GRENOBLE ALPES

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES

Spécialité : **Informatique**

Arrêté ministériel : 7 août 2006

Présentée par

Behrooz OMIDVAR-TEHRANI

Thèse dirigée par **Sihem Amer-Yahia** et
codirigée par **Alexandre Termier**

préparée au sein du **Laboratoire Informatique de Grenoble**
dans **Ecole Doctorale Mathématiques, Sciences et Technologies de l'Information, Informatique**

Optimization-based User Group Management: Discovery, Analysis, Recommendation

Thèse soutenue publiquement le **6 Novembre 2015**,
devant le jury composé de :

Mme, Ahlame Douzal
Maître de conférence (HDR) à l'Université Grenoble Alpes, Présidente
Mr, Divesh Srivastava
Directeur de Recherche au laboratoire AT&T, Rapporteur
Mr, Bruno Crémilleux
Professeur à l'Université de Caen Normandie, Rapporteur
Mr, David Gross Amblard
Professeur à l'Université de Rennes 1, Examinateur
Mme, Elisa Fromont
Maître de conférence à Université Jean Monnet, Examinatrice
Mme, Sihem Amer-Yahia
Directrice de Recherche (DR1) au CNRS, Directrice de thèse
Mr, Alexandre Termier
Professeur à l'Université de Rennes 1, Co-Directeuer de thèse

"Data matures like wine, applications like fish."

James Governor, Founder of Redmonk.

Résumé

Les données utilisateurs sont devenue de plus en plus disponibles dans plusieurs domaines tels que les traces d'usage des smartphones et le Web social. Les données utilisateurs, sont un type particulier de données qui sont décrites par des informations socio-démographiques (ex., âge, sexe, métier, etc.) et leurs activités (ex., donner un avis sur un restaurant, voter, critiquer un film, etc.). L'analyse des données utilisateurs intéresse beaucoup les scientifiques qui travaillent sur les études de la population, le marketing en-ligne, les recommandations et l'analyse des données à grande échelle. Cependant, les outils d'analyse des données utilisateurs sont encore très limités.

Dans cette thèse, nous exploitons cette opportunité et proposons d'analyser les données utilisateurs en formant des groupes d'utilisateurs. Cela diffère de l'analyse des utilisateurs individuels et aussi des analyses statistiques sur une population entière. Un groupe utilisateur est défini par un ensemble des utilisateurs dont les membres partagent des données socio-démographiques et ont des activités en commun. L'analyse au niveau d'un groupe a pour objectif de mieux gérer les données creuses et le bruit dans les données. Dans cette thèse, nous proposons un cadre de gestion de groupes d'utilisateurs qui contient les composantes suivantes: découverte de groupes, analyse de groupes, et recommandation aux groupes.

La première composante concerne la découverte des groupes d'utilisateurs, c.-à-d., compte tenu des données utilisateurs brutes, obtenir les groupes d'utilisateurs en optimisant une ou plusieurs dimensions de qualité. Le deuxième composant (c.-à-d., l'analyse) est nécessaire pour aborder le problème de la surcharge de l'information: le résultat d'une étape découverte des groupes d'utilisateurs peut contenir des millions de groupes. C'est une tache fastidieuse pour un analyste à écumer tous les groupes trouvés. Nous proposons une approche interactive pour faciliter cette analyse. La question finale est comment utiliser les groupes trouvés. Dans cette thèse, nous étudions une application particulière qui est la recommandation aux groupes d'utilisateurs, en considérant les affinités entre les membres du groupe et son évolution dans le temps.

Toutes nos contributions sont évaluées au travers d'un grand nombre d'expérimentations à la fois pour tester la qualité et la performance (le temps de réponse).

Abstract

User data is becoming increasingly available in multiple domains ranging from phone usage traces to data on the social Web. User data is a special type of data that is described by user demographics (e.g., age, gender, occupation, etc.) and user activities (e.g., rating, voting, watching a movie, etc.) The analysis of user data is appealing to scientists who work on population studies, online marketing, recommendations, and large-scale data analytics. However, analysis tools for user data is still lacking.

In this thesis, we believe there exists a unique opportunity to analyze user data in the form of user groups. This is in contrast with individual user analysis and also statistical analysis on the whole population. A group is defined as set of users whose members have either common demographics or common activities. Group-level analysis reduces the amount of sparsity and noise in data and leads to new insights. In this thesis, we propose a user group management framework consisting of following components: user group discovery, analysis and recommendation.

The very first step in our framework is group discovery, i.e., given raw user data, obtain user groups by optimizing one or more quality dimensions. The second component (i.e., analysis) is necessary to tackle the problem of information overload: the output of a user group discovery step often contains millions of user groups. It is a tedious task for an analyst to skim over all produced groups. Thus we need analysis tools to provide valuable insights in this huge space of user groups. The final question in the framework is how to use the found groups. In this thesis, we investigate one of these applications, i.e., user group recommendation, by considering affinities between group members.

All our contributions of the proposed framework are evaluated using an extensive set of experiments both for quality and performance.

Contents

Résumé **iii**

Abstract **iv**

Contents **v**

List of Figures **ix**

List of Tables **xi**

1 Introduction **1**
1.1 Data Analysis 2
1.2 User Data Analysis 3
1.3 User Group Management 6
1.3.1 Online Advertising 7
1.3.2 Social Sciences 8
1.3.3 Social Web 9
1.4 Components of User Group Management 10
1.5 Contributions and Overview 11

2 Preliminaries **13**
2.1 User Data Model 13
2.2 Attribute Taxonomies 15
2.3 User Group 16
2.4 Lattice of User Groups 17
2.5 User Data Model in One View 18
2.6 User Datasets 19
2.6.1 Movie-Review Dataset: MovieLens 1M 20
2.6.2 Book-Review Dataset: BookCrossing 21
2.6.3 Mobile-Usage Dataset: Nokia 21
2.6.4 Dataset of Data Management Researchers: DM-Authors 22
2.6.5 Social Network Dataset: Facebook-72 25
2.7 Conclusion 26

3 User Group Discovery **27**

3.1 Motivating Examples . . . 27
3.2 Challenges and Contributions . . . 28
3.3 User and Group Data Models . . . 30
3.3.1 Group Quality Dimensions . . . 31
3.3.2 Multi-Objective Optimization Principles . . . 33
3.4 Group Discovery Problem Definition . . . 34
3.5 Group Discovery Algorithms . . . 35
3.5.1 Exhaustive Algorithm . . . 36
3.5.2 Approximation Algorithm . . . 37
3.5.3 Heuristic Algorithm . . . 38
3.6 Experiments . . . 41
3.6.1 Need for Multi-objective Optimization . . . 42
3.6.2 Effect of Application-Defined Parameters . . . 43
3.6.2.1 Minimum Group Size (σ) . . . 43
3.6.2.2 Number of Intervals and Iterations . . . 44
3.6.2.3 Number of Returned Groups (k) . . . 45
3.6.3 Comparison of Algorithms . . . 46
3.7 Conclusion . . . 49

4 User Group Analysis 51
4.1 User Group Analysis via Abstraction . . . 52
4.1.1 Motivating Example . . . 52
4.1.2 Data Model . . . 52
4.1.3 Abstraction Primitive . . . 54
4.1.4 Experiments . . . 56
4.2 Interactive User Group Analysis . . . 60
4.2.1 Motivating Examples . . . 61
4.2.2 Challenges and Contributions . . . 63
4.2.3 Group Navigation Operations . . . 64
4.2.4 Group Navigation Problem . . . 65
4.2.5 Interactive User Group Analysis (IUGA) . . . 66
4.2.6 Group Navigation Algorithm . . . 67
4.2.7 Experiments . . . 69
4.2.7.1 Interactive Analysis Validation . . . 70
4.2.7.2 Quality of Group Navigation . . . 78
4.2.8 Graphical User Interface . . . 79
4.3 Conclusion . . . 81

5 User Group Recommendation 83
5.1 Challenges and Contributions . . . 83
5.2 Data Model and Preliminaries . . . 86
5.2.1 User Affinity Models . . . 87
5.2.2 User-Item Preference Model . . . 89
5.2.3 Problem Definition and Hardness . . . 90
5.3 User Group Recommendation Algorithm . . . 92
5.4 Experiments . . . 99
5.4.1 Quality Experiment . . . 100

5.4.1.1 User Collection Phase 101
5.4.1.2 Static and Dynamic Affinities 102
5.4.1.3 Group Formation 102
5.4.1.4 Quality Assessment 103
5.4.2 Scalability Experiment 106
5.4.2.1 Varying Time Period 107
5.4.2.2 Varying k, Group Size and Number of Items 108
5.4.2.3 Similarity/Dissimilarity 109
5.4.2.4 Time Models 109
5.4.2.5 Consensus Functions 109
5.5 Conclusion 110

6 Related Work 111
6.1 User Group Discovery 111
6.2 User Group Analysis 115
6.3 User Group Recommendation 117

7 Summary and Perspectives 119
7.1 Summary 119
7.2 Perspectives 120
7.2.1 System Perspectives 120
7.2.2 Evaluation Perspectives 124
7.2.3 Applications' Perspectives 126

A Optimality and Near-Optimality Proofs 129

B NP-Hardness Proofs of GroupNavigation Problem 133

C Thesis Reviews 135

Bibliography 137

List of Figures

1.1 Example of User Group Management 8
1.2 User Group Management Framework 11

2.1 Example of User Data Model 14
2.2 Taxonomy of Applications 15
2.3 Location Taxonomy 15
2.4 Illustration of User Groups 17
2.5 Lattice of User Groups 18
2.6 Components of User Data Model 19

3.1 Example Dataset and Group-set 31
3.2 Different Rating Distributions for a Group-Set 32
3.3 Illustration of User Groups and Group-sets 34
3.4 Dominance Areas 39
3.5 Illustration of the h-MOMRI Process 40
3.6 Conflicting Objectives on MovieLens 43
3.7 Number of Groups as a Function of σ 44
3.8 Effect of the *nbintervals* parameter 45
3.9 Effect of the Number of Returned Groups (k) 46
3.10 Distribution of Solutions in Intervals 47
3.11 Comparison of α-MOMRI and h-MOMRI Algorithms 48

4.1 User Attribute Taxonomy 53
4.2 Application Taxonomy 53
4.3 *Desktop Communication* Taxonomy 54
4.4 *Web App* Taxonomy 54
4.5 Location Taxonomy 54
4.6 Item Usages for Abstraction 56
4.7 Abstraction Volume 57
4.8 User Group Space Reduction for Nokia 58
4.9 Abstraction per Level for Nokia 59
4.10 Scenario of Discovering Several Users 62
4.11 Scenario of Finding a Specific User 62
4.12 Illustrations of *diversity*() and *divCoverage*() functions 66
4.13 Illustration of GroupNavigation within IUGA 67
4.14 Number of Steps in IUGA for PC Selection 71
4.15 Scenarios Know13, KnowIn and KnowOut and NonExpertIn 72

4.16 Word Frequency Cloud Comparison between Jian Li and the Whole WebDB 2014 PC . . . 74
4.17 IUGA Comparison with Clustering Algorithm . . . 76
4.18 ATA as a Function of k and # Groups . . . 77
4.19 User Preference Results . . . 79
4.20 GUI Layer on top of IUGA . . . 80
4.21 Graphical User Interface . . . 81

5.1 Components of User Group Recommendation . . . 87
5.2 Independent Evaluation . . . 103
5.3 Comparative Evaluation . . . 105
5.4 Qualitative Evaluation of Consensus Functions . . . 106
5.5 Different Time Periods . . . 107
5.6 Average Percentage of `SA`s for Time Models and Different Groups . . . 108
5.7 Average Percentage of `SA`s by Varying Result Size, Group Size and Number of Items . . . 108
5.8 Average Percentage of `SA`s for Different Consensus Functions . . . 109

List of Tables

1.1 User Groups 6

2.1 Statistics on Datasets 20
2.2 Nokia Dataset Attributes 23

3.1 Input Sets of Rating Records 42

4.1 IUGA Actions 67
4.2 Real Datasets 70
4.3 Synthetic Dataset 70
4.4 Validation Scenarios 73
4.5 Interactive Analysis Methods 77
4.6 IUGA vs Optimal Methods 77

5.1 Absolute Preference Lists 93
5.2 Static Affinity Lists 94
5.3 $\mathcal{L}_{aff_V}$ Lists for Period p_1 94
5.4 $\mathcal{L}_{aff_V}$ Lists for Period p_2 94

Dedicated to

my wife, Shadi
my brother and my parents

Chapter 1

Introduction

The growing availability of data in different domains has been a strong motivation for *data-driven* research: science whose progress is compelled by data. For instance, research in social computing is characterized by a heavy reliance on large-scale analytics of users' activities in order to understand their needs and design services [AYLT+15]. In 2011, the amount of digitally recorded data by humans was 1.8 zettabytes. It is estimated that in 2020, it will grow to 40 zettabytes.

We *analyze* this huge data to make sense of it and discover interesting subsets and correlations. These results are useful in many real-world applications such as social studies, market prediction, online recommendation and online advertising. It is approximated that only 2% of this whole data has been analyzed to date.[1] Hence the great necessity of data-driven research to enable data analysis.

In this thesis, we propose a novel data analysis framework for user data. User data is a special type of data that is characterized by user demographics and user activities. For instance, in Twitter[2], each user has a profile containing some demographics information (e.g., age, gender, occupation, etc.) and she writes comments and follows other users, i.e., her activities. In this thesis, we believe there exists a unique opportunity to analyze user data in the form of *user groups*. This is in contrast with individual user analysis and also statistical analysis on the whole population. We consider a group of users whose members have common demographics and common activities. This reduces the amount of sparsity and noise in data and leads to new insights. In this thesis, we investigate how to discover user groups, analyze them and apply real-world scenarios to them.

In this introductory chapter, we first review some data analysis techniques (Section 1.1). Then we narrow down to user data analysis in Section 1.2. Some real-world scenarios

[1] *Parallel Techniques for Big Data, Patrick Valduriez, BDA 2013*
[2] *https://twitter.com*

are discussed in Section 1.3. We introduce different components of our framework in Section 1.4. Finally, contributions of this thesis are summarized in Section 1.5.

1.1 Data Analysis

Data analysis is crucial to find *value* in data, i.e., interpretations and insights. The interpretation of data is at least as important as the data itself. Data without analysis can be easily become a distracting black hole without any actionable insight to make better decisions.[3] For instance, a complete biomarker data for thousands of patients is not useful unless a data analysis technique finds subsets of the patient population who can benefit from a particular therapy [AYLT+15]. While many efforts have been made in recent years to tackle the *volume* challenge in large-scale data, i.e., handling a petabyte-sized data across thousands of machines, the *value* aspect of data has received much less attention to date.

Different methodologies and software suits are designed to support large-scale data analysis and leverage businesses. Many of them are general-purpose and serve different types of data and provide some common insights about them. Examples are Tableau[4] (a visualization suite), OpenRefine[5] (a data cleaning suite), Apache Spark[6], KNIME[7], RapidMiner[8], NodeXL (a network visualization suite) etc. It is desirable that a data analysis methodology satisfies the following characteristics [NJ11, BKT+13, OTAYT15]:

- **Available** for different purposes, different types of data, and work with different platforms.
- **Easy-to-Use**. A non-expert should be able to use such a methodology without any programming or querying knowledge. In a data analysis system like Mime [GMV11], many analysis tools are available, but it is not clear for the end-user which ones are relevant and which best answer specific questions. On the other hand, complex data analysis GUIs[9] (e.g., RapidMiner) require a steep learning curve.
- **Interactive**. The methodology should provide different interaction facilities with the analyst in order to catch the analyst's interest and provide more relevant results in future interactions.

[3] *http://www.nngroup.com/articles/analytics-user-experience/*
[4] *http://www.tableau.com*
[5] *http://openrefine.org*
[6] *http://spark.apache.org*
[7] *https://www.knime.org*
[8] *https://rapidminer.com*
[9] *Graphical User Interface*

- **Computationally Powerful**. The system should be able to perform different analysis tasks in a reasonable time.
- **Describable**. The methodology should be able to provide a concise description that explains the results.

One of the most famous analytical tools is OLAP.[10] It is an online analytical processing technique that enables an analyst to get a subset of data of her interest and make comparisons from different points of view. OLAP works on multi-dimensional tables and crosses dimensions to obtain different views. For example, an analyst can request OLAP to display the sold products of a special type for the month of May for a company. She can then compare the revenue with the month of September. She can then extend her analysis to see the revenue for other types of products.

Although OLAP can be a great methodology for data analysis, it has the following drawbacks: *i*. data modeling is required before any analysis can happen, *ii*. a non-expert cannot get the most out of it, *iii*. the response time is low, and *iv*. few interaction facilities between the analyst and the methodology are available.

As a summary, all current data analysis approaches, have some challenges and flaws which prevent the analyst to obtain a full-fledged set of results and interpretations.

1.2 User Data Analysis

The primary input to the data analysis process (Section 1.1) is *data*, which is produced either by users (e.g. social networks, phone usage data) or machines (e.g. storage capacities, energy consumption on machines). Data from these two sources is different in two aspects:

- **Granularity**. User-generated data (or simply user data) is less granular. For instance, a user may perform an action (e.g., commenting in a social network) every one minute, while machines may perform hundreds of actions in a fraction of a second.
- **Modularity**. There exists more modularity for humans than machines that makes the nature of user data more heterogeneous. The behavior of a machine is often easily predictable as we already know the whole lifecycle. For humans, however, this is not the case as they may behave surprisingly in different situations.

[10] *http://olap.com*

Less granularity and more modularity motivate us to think and focus on user data analysis. Different user data sources are available. We provide some examples.

- In Twitter[11], occupation and location are considered as user attributes and tweets are user activities. Also, we often process each tweet to get some other attributes which leads to a more detailed analysis, e.g., tweet topic (using topic models), sentiment analysis, etc.
- For a user having a smartphone, her demographics (e.g., age, gender, occupation) are user attributes and what she does with her phone (e.g., opening an application, listening to a music, sending a short message) makes her activities.

User data analysis brings two following benefits:

1. First it identifies *an interesting subset of data* that optimizes a quality dimension. For instance, in IMDb[12], we can detect the relation between a subset of reviews for a movies and the probability of watching that movie. So it can automatically promote and rank such reviews higher to ensure that the probability of watching the movie increases.
2. Second, it identifies *an interesting subset of users* that collectively optimizes a quality. If in IMDb, we form two groups of users, novices and experts, then it can provide different services based on their need by showing abstract information to the former and detailed information to the latter group which leads to increased satisfaction.

User data can be collected from very different sources: user profiles and their usage of web applications, online games, navigation paths and social media interactions. Usually a *cleaning* step precedes the collection phase to remove useless and redundant information [CMI+15].

Due to the *sparsity* and *impurity* of user data, we propose to analyze it based on forming user groups. Grouping in user data analysis is a very important practice which leads to interesting results which are hidden when analyzing individual users. Main motivations of grouping users are as follows.

- **Sparsity Reduction.** Often the user data is sparse, i.e., many pieces of information for different individual users are missing. Grouping is an aggregation of data and reduces the amount of sparsity.

[11] *http://www.twitter.com*
[12] *http://www.imdb.com*

- **Noise Reduction.** User data may be noisy, i.e., contains wrong information for individual users. Grouping reduces the effect of noise. Noises are less visible or become negligible when grouping users.
- **Improved Analysis.** Grouping users leads to more insights that can help each member of the group make better decisions. For instance, an insight obtained based on analyzing a group of diabetic patients can benefit each member of that group (as in Example 1.2).

The following examples illustrate how grouping can help achieve a better analysis.

Example 1.1 (Telecom Customers)**.** *Anderson, a data analyst, wants to help a telecom company develop a list of "jobs to be done" for new products and services. He has already made some interviews with customers regarding the way they use their mobile devices throughout the day. After analyzing customer stories, he found two interesting groups of users: i.* multi-usage customers *requiring 2 SIM cards for the same smartphone; ii.* customers who are mothers *requiring parental control and other relevant applications which facilitate their lifestyle as a mother. Discovery of these two groups calls for distinct analysis and possibly different products and services that the company would provide to increase satisfaction.*[13]

Example 1.2 (Diabetes Disease)**.** *Mary has insulin-dependent diabetes and her blood sugars are running high at night. But she is not able or doesn't feel motivated to understand why. In case a group of users is already formed (e.g., in PatientsLikeMe research network* [14]*) whose members are all suffering from the same disease, she could see the profiles of other people like her, and see where she falls relative to the "norm".*

The above examples motivate the need for grouping in user data analysis. In other words, user group data analysis is a shift from *Quantified-Self* to *Quantified-Us.*[15] Quantified-Self is an approach to exploit "massive" personal datasets for self-discovery. On the contrary, Quantified-Us combines different personal data and forms groups of users to provide improved discoveries. It helps identify patterns and make meaningful recommendations for the whole group. Some instances of the Quantified-Us movement are PatientsLikeMe (research network to improve lives and a real-time research platform to advance medicine), Crohnology[16] and StockTwits[17].

User groups can be formed by combining any user attributes or activities which describe a set of users. Users in a group may know each other (e.g., researchers working in the

[13] *https://hbr.org/2013/09/story-driven-data-analysis/*
[14] *https://www.patientslikeme.com/*
[15] *http://www.wired.com/2014/04/forget-the-quantified-self-we-need-to-build-the-quantified-us/*
[16] Crohnology is a patient-to-patient information sharing platform.
[17] StockTwits is a social media platform for sharing ideas between investors, traders, and entrepreneurs.

User Group	**Attributes**	**Activities**
Female students who live in Grenoble and go to Shannon pub	female (gender), Grenoble (location), student (occupation)	going to a pub
Male users in Twitter who follow Barack Obama and Joe Biden	male (gender)	following in Twitter
Set of people who suffer from diabetes (Example 1.2) and follow a special diet	diabetes (disease)	following a diet

Table 1.1: User Groups

same laboratory) but it is more likely that they are a set of strangers who have many common points (e.g., researchers from different parts of the world who work on similar topics). Table 1.1 illustrate some instances of user groups.

The notion of "user group" is similar in concept to *frequent item-sets* in Frequent Item-set Mining [AIS93], and *cohort* [18] in statistics. All these concepts refer to an aggregation of objects (e.g., users) which is describable by some common attributes.

Grouping users should follow a semantics. Not any user group is interesting. To define the interestingness of a user group or a set of user groups, we propose *quality dimensions*, e.g., diversity, coverage, conciseness, etc. A quality dimension is a function that takes one or many user groups as input and returns a score. The higher the score, the better the user group. Quality dimensions are problem-dependent, that means a user group may be very interesting for one problem and not interesting for another scenario. We discuss quality dimensions in more details in Chapter 2.

1.3 User Group Management

We call *discovering*, *analyzing* and *applying* user groups to real-world scenarios, *user group management*. We present three different domains where user group management is helpful: online advertising, social sciences and the social web.

[18] *http://cohortanalysis.com*

1.3.1 Online Advertising

Online advertising is an important advertising method for a variety of businesses because it is relatively easy to carry out, scalable and cost-efficient. One of the main steps of online advertising is *audience targeting*, i.e., finding the best audience for an online advertisement. This audience is usually the subject of a sale or promotion with the aim of increasing the overall satisfaction or introducing a new product.

For an analyst who has access to the profile of thousands of users, it is a challenging task to find the best target audience. Its hardness comes from the fact that many different user groups can be made. There should be some quality dimensions to compare different groups and obtain qualified ones.

For instance, consider Julia who works in an advertising company and is responsible for finding the best target audience for a promotion on books of *John Grisham*, the American author known for his popular thrillers. To find a target group, Julia goes to BookCrossing[19], a database of book ratings, and finds 6913 rating records for all Grisham's books. In this way, she can observe how works of Grisham have been appreciated. Although the number of rating records is manageable, the number of user groups over these ratings is much higher because many different combinations of attributes can potentially make a group.

Julia observes that 89% of users who rated Grisham's books are either young reviewers who live in Connecticut, United States, middle-age reviewers in France or old females. These three groups are interesting because they are diverse, i.e., they do not overlap because their reviewers belong to different age-categories. Here, diversity is considered as quality dimension for the set of groups. Julia finds the second group promising, as its average rating score is 4.6 out of 5. Thus it is a candidate group for audience targeting. This is the *discovery* step in user group management which led Julia to one or more candidates for her goal.

Julia later understands that the advertising company has a strict policy to prioritize young people as target. Thus she needs to find another user group which is similar to what she already found (middle-age reviewers in France) but for young reviewers. She hence needs to navigate the space of groups around the group at hand. She uses an intelligent navigation mechanism which leads her towards her goal, i.e., finding a similar user group for young reviewers. Finally she finds the following group: young reviewers in Europe who speak French. This user group is similar with while describing young users. This is the *analysis* step, the second step in the user group management framework

[19] *http://www.bookcrossing.com*

which lets Julia to navigate semantically in the space of groups to reach the group of interest.

Julia has now the target audience. Now the question is what to do with this user group. For example she can recommend a subset of Grisham's books to the target audience with a promotion. Thus the precise question becomes: which subset of Grisham's books should be recommended to this user group? Julia will then need a recommender tool to make recommendations not for individuals but for a group of users. Julia receives 3 following books to recommend: *A Time to Kill (1989)*, *Sycamore Row (2013)* and *The Litigators (2011)*. This is the third step in user group management where the analyst uses discovered groups in an *application*. In this example, we mentioned Group Recommendation as an application. The journey that Julia made to reach the recommendation, is illustrated in Figure 1.1.

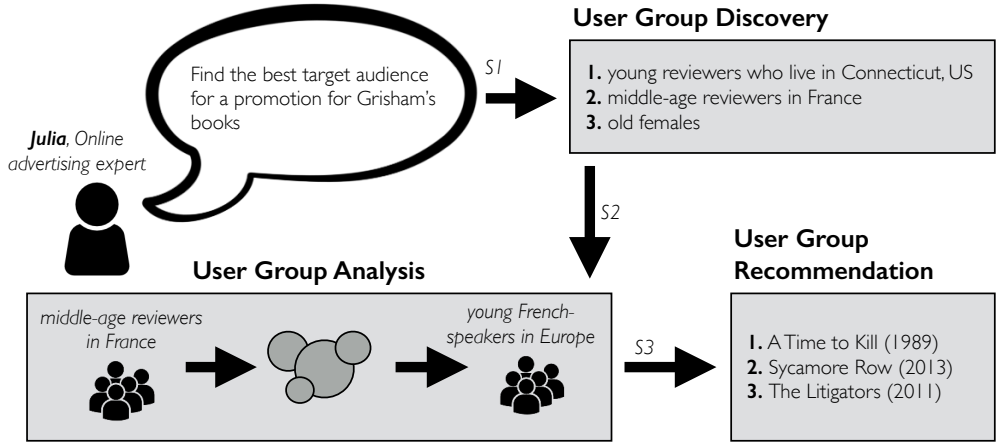


FIGURE 1.1: Example of User Group Management.

1.3.2 Social Sciences

Social science is an effort to discover relations between humans and their behaviors. Social scientists usually try to find evidences for phenomenas in daily lives in the form of hypotheses. The *status quo* of social science is to pose a hypothesis and observe behaviors and characteristics of participants of a social experiment. Some example hypothesis in social science are as follows[20]: *does violent media make people aggressive?* [BRR63], *does the demographics characteristic of a job applicant affect the likelihood of being hired?* [Pag03], etc.

[20] *Laboratory Experiments in the Social Sciences: An Introduction, Stephen Benard, Department of Sociology, Indiana University*

The social science research process suffers from being local. As recruiting participants for experiments is a timely and costly activity, they often end up recruiting less than 100 participants. This small set may contain biases. For instance, in [BB13], the hypothesis is that people perform greater within group cooperation when their groups face external threats, such as natural disasters. For this study, authors recruited 31 males and 35 females from the Cornell University community between 18 and 55 years old. User group management can strongly contribute to social science to improve its research process and bring firm explainable justifications.

We consider one instance of a social hypothesis: it is generally believed that *western* movies (e.g., *The Good, the Bad and the Ugly (1966)*) are mostly watched by the older generation. This observation is based on demographics breakdown reports on IMDb website. Anna, who is a social scientist, wants to validate this hypothesis by exploring user groups that cover most ratings for *western* movies. Here, the quality dimension is coverage. She considers all rating records for Western movies in IMDb.

Anna receives 3 following user groups: old male reviewers, young female reviewers, and middle-age reviewers from Texas, United States, with average ratings of 4.7, 3.1 and 3.9, respectively. By observing those groups, Anna finds that although the hypothesis is correct, it also depends on other demographics like gender and location. More precisely, found groups show that not all old people prefer *western* movies. She also observes that only young females give a low rating to those movies. Finally, the results show another group of *western* genre lovers, *middle-age reviewers from Texas*, which contradicts the hypothesis. This is the *discovery* step of user group management.

Anna has examined her hypothesis, but she is also interested to observe what is happening in other similar groups. Thus she needs to navigate in the space of "neighbor" user groups to what she has seen before. This would be an *analysis* step. For instance, she confirms her hypothesis regarding old generation in a group of old reviewers whose members have liked both *Western* and *Family* genre movies.

1.3.3 Social Web

The social Web is a set of social relations between users through Web [HT10]. It covers design and development standards for websites, software and online services in order to support social interactions [Por10]. Social interactions form the basis of online activities like collaborative websites whose data come from users' reviews and comments. Examples are MovieLens and IMDb (movie reviews), BookCrossing (book reviews),

Facebook[21] (general comments and reviews) and LastFM[22] (music reviews). We can also consider DBLP[23] as a collaborative website that expresses researcher activities, i.e., publications. User group management can give interesting insights and patterns which express the way users in collaborative websites behave and interact.

Consider an example in Facebook. Nicole met a person at last night's party in San Francisco, but she doesn't remember his name and has lost his contact information. She asks the party host for an access to his social network which contains some information about his friends. In the *discovery* step, she observes different user groups in the host's friends, based on a given quality dimension. In the *analysis* step, she navigates in the space of discovered groups to gain insights. The group of senior software engineers in San Francisco captures her attention as she remembers the person was a manager. Focusing on users of this group leads Nicole reach her target.[24]

1.4 Components of User Group Management

Examples and scenarios in Section 1.3 illustrate the wide usage of our *user group management* framework from online advertising to the social Web. In this thesis, we propose to formalize such a framework for user group management. An analyst needs first to discover user groups based on one or more quality dimensions. In the second step, she can navigate through the space of user groups to reach her target group(s). Finally, she uses the targets which are discovered and analyzed in the first and second steps, in an application. We already motivated an application, i.e., group recommendation. Figure 1.2 illustrates this framework.

The *first step* in the framework is to **discover** user groups. Discovering user groups is a challenging task, because the pool of candidates is very large and the quality needs to be optimized in more than one dimension to discover high quality groups.

The *second step* in the framework is to **analyze** user groups. The main challenge of the analysis step is to tackle the problem of *"Information Overload"*, i.e., the number of possible user groups is often very large which hinders their effective management.

In the *third step*, we show how to use discovered groups in action. One application that we discuss in this thesis, is group recommendation: finding the best items that a set of users will appreciate together. Our contribution in group recommendation is exploring a new dimension when computing group recommendations, that is, *temporal affinities*

[21] *www.facebook.com*
[22] *www.last.fm/*
[23] *dblp.uni-trier.de/*
[24] *This example is detailed in Section 4.2.1 (Example 4.3).*

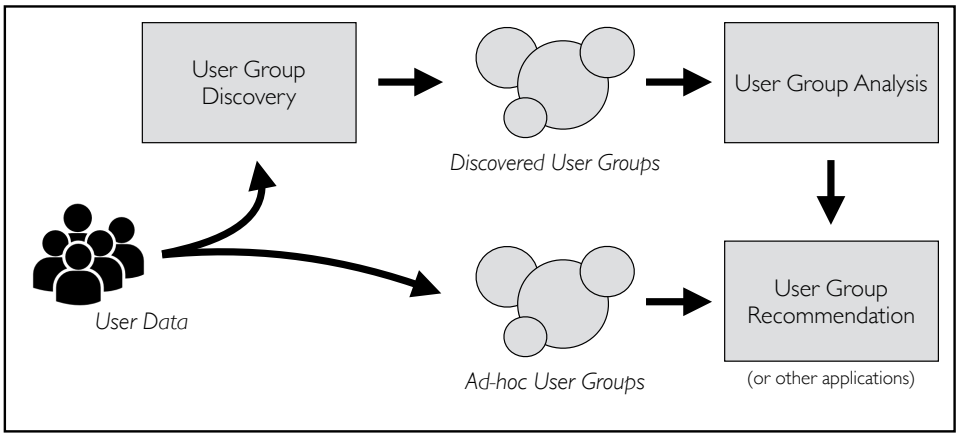


FIGURE 1.2: User Group Management Framework.

between group members. This is a challenging task, because a new formulation is needed to account for temporal affinities between group members. On the other hand, the other challenge is that the algorithm should be fast, so we need to think of online approaches which computes group recommendation efficiently.

1.5 Contributions and Overview

The exposition is structured around the different components of a user group management framework (Figure 1.2).

1. We introduce the **User Group Discovery** problem in Chapter 3. For that, we formalize the following group quality dimensions: coverage, diversity and rating distribution. We formalize the problem as a constrained multi-objective optimization problem with quality dimensions as objectives. We develop α-MOMRI, an α-approximation algorithm for user group discovery and h-MOMRI, a heuristic-based algorithm that exploits the lattice formed by user groups to speed up group discovery. In an extensive set of experiments on MovieLens and BookCrossing datasets, we analyze different solutions of α-MOMRI and h-MOMRI and show that high quality groups are returned by our approximation and very good response time is achieved by our heuristic. This work has been submitted and is under review.

2. We introduce the **User Group Analysis** problem in Chapter 4. We propose two different solutions for this problem: *abstraction* and *interactive navigation*.

   (a) We formalize an *abstraction* primitive and study experimentally the potentials in reducing the space of user groups [OTAT+13].

(b) For interactive navigation, we propose IUGA, a formalization of interactive user data analysis based on simple yet powerful *group navigation* primitives that enable an exploratory navigation of user groups [OTAYT15]. We also propose a principled experimental methodology to evaluate interactive navigation.

3. We formalize one application of user group management, i.e., **User Group Recommendation** in Chapter 5 [AORS15].

   (a) In this work, we motivate the need to account for interaction between group members when computing recommendations and propose to capture affinities in *the relative preference* of individual group members for each item. Relative preference modifies a user-item preference with the user's affinity with other group members.

   (b) Since affinities may evolve over time, we propose models to represent affinity drift over time.

   (c) We develop GRECA, an efficient algorithm that computes recommendations on-the-fly for user groups.

   (d) We run extensive experiments to examine the impact of our temporal affinity model on group recommendation quality and efficiency.

Chapter 2

Preliminaries

In this thesis, we propose a framework for user group management. User groups are discovered in user data. In this chapter, we introduce the model we define for user data and provide some preliminary definitions. We also describe the real user datasets we used in this thesis for experiments and proof-of-concepts.

Before we start describing the data model, we disambiguate three different human roles in this framework, i.e., user, analyst and end-user.

- **User** generates data. Our management framework consumes user data.
- **Analyst** works with our management framework to achieve a goal. (S)he observes and analyzes user data via the framework.
- **End-user** benefits from the analysis product delivered by an analyst by using our framework.

2.1 User Data Model

User data contains a set of users $\mathcal{U}$, a set of items $\mathcal{I}$, and a database $\mathcal{D}$ of tuples $\langle u, c, i \rangle$ where $u \in \mathcal{U}$ and $i \in \mathcal{I}$ and c is an action. A tuple $\langle u, c, i \rangle$ represents the action c (such as *authored*, *recorded*, *rated*, *purchased*, *tagged*, *voted*, etc.) performed by u on i. For instance, the tuple $\langle$*John*, *watched*, *Titanic*$\rangle$ simply means that John has watched the movie Titanic. We don't mention actions wherever they are clear from the context, hence a tuple becomes $\langle u, i \rangle$. Our data model can be seen as a bipartite graph from users and their demographics to their activities (Figure 2.1). The links in this bipartite graph are user actions.

Each user u is described with attributes drawn from a set $\mathcal{A}_u$ representing demographics information such as `gender` and `age`. We also associate a set of attributes to $\mathcal{I}$ denoted as $\mathcal{A}_i$ representing item details such as `director` (for a movie) and `author` (for a book). Finally we associate a set of attributes $\mathcal{A}_c$ to actions which are mainly the *time* and *location* of the action. We refer to the set of all attributes as $\mathcal{A} = \mathcal{A}_u \cup \mathcal{A}_I \cup \mathcal{A}_c$ and each attribute $a_i \in \mathcal{A}$ has values in $\{v_i^1 \dots v_i^j \dots\}$. The domain of values of attribute a_i is D_{a_i} with $D_{\mathcal{A}} = \cup D_{a_i}$. For example, if we use a_1 to refer to `gender`, it takes two values v_1^1 and v_1^2 representing `male` and `female` respectively.

The choice of what constitutes a user attribute or a user action depends on the application and does not affect our problems and proposed solutions. For instance, for a movie, the set of attributes is `director` and `genre(s)`. Also for a book, the set of attributes is `author` and `publisher`.

Figure 2.1 illustrates an example of the user data model by considering movies as items. The figure shows that user 1 is a female student and has watched movies American Beauty, Celtic Pride and Sanjuro. It also shows that the group of male users consists of users 2, 3 and 4, while the group of *Jurassic Park* watchers are users 3 and 4.

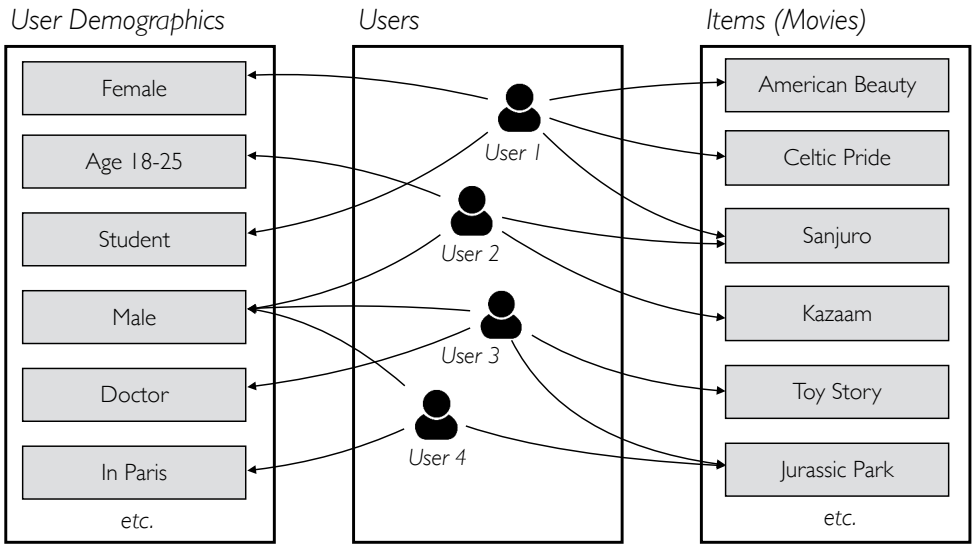


FIGURE 2.1: Example of User Data Model.

A large number of user datasets could be modeled in this manner. Examples are a dataset which contains user preferences over movies, books, musics, etc., or a dataset which contains user activities on their smartphones, e.g., using applications, playing musics, etc.

2.2 Attribute Taxonomies

Attributes in the set $\mathcal{A}$ (for users, items and actions) are organized in hand-crafted specialization/generalization taxonomies τ_A. Figures 2.2 and 2.3 show two examples of taxonomies.

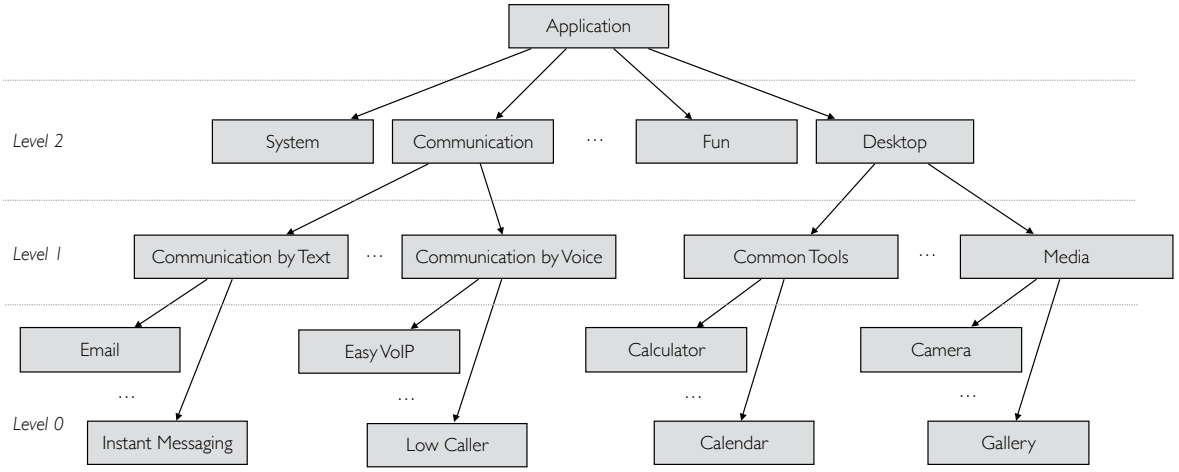


FIGURE 2.2: Taxonomy of Applications

In a taxonomy, going from level i to $i+1$ is moving to more general concepts. In Figure Figure 2.2, items in level 0 are applications installed on mobile phones. At higher levels, more general categories of applications appear. For instance, both `EasyVoIP` and `Low Caller` belong to the category `Communication by Voice` (level 1). All communication applications are themselves generalized to a more global category, i.e., `Communication` in level 2.

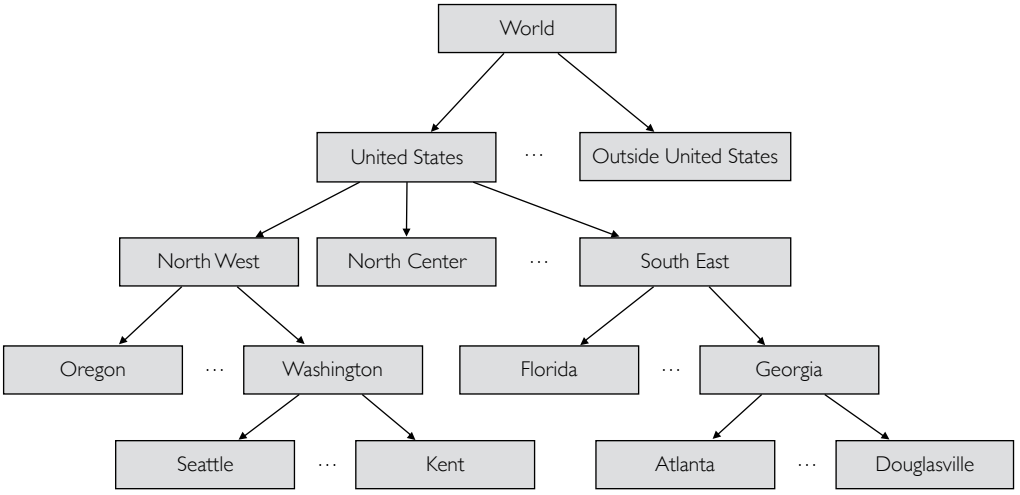


FIGURE 2.3: Location Taxonomy

Figure 2.3 illustrates a taxonomy for location, i.e., an action attribute. The taxonomy of location usually starts with street names or cities as leaves (level 0) and advances to points of interest (POIs), states, regions, countries and continents.

2.3 User Group

Based on the data model we introduced in Section 2.1, we now define the notion of user group. A group of users can be defined based on *common user attributes* or on *common user activities*. Examples are, rating records related to people who watched the movie Titanic, or tweets of young female users who have posted a tweet related to "health".

Definition 2.1 (Common-Attribute User Group)**.** A *common-attribute* user group g is a subset of $\mathcal{U}$ to which is associated a label $l_g = [P_g, I_g]$ where P_g is a conjunction of predicates on user attributes and I_g is a set of items. Each user in g must satisfy P_g and $\forall u \in g, i \in I_g, \langle u, i \rangle \in \mathcal{D}$.

Definition 2.2 (Common-Activity User Group)**.** A *common-activity* group g is a set of tuples $\langle u, i \rangle \in \mathcal{D}$ where $u \in \mathcal{U}' \subseteq \mathcal{U}$ and $i \in \mathcal{I}' \subseteq \mathcal{I}$ to which is associated a label $l_g = [P_g^u, P_g^i, \mathcal{I}']$ where P_g^u and P_g^i are conjunctions of predicates on user attributes and item attributes respectively. Each user in $\mathcal{U}'$ must satisfy P_g^u and $\forall i \in \mathcal{I}'$, i satisfies P_g^i.

Common-attribute user group (Definition 2.1) focuses on *users* in the group while common-activity user group (Definition 2.2) focuses on *activities of users* in the group. For instance a common-attribute user group $g_1 = [\langle gender, female \rangle, \langle age, young \rangle]$ describes young female users, while the common-activity user group $g_2 = [\langle gender, female \rangle, \langle movie, Titanic \rangle]$ contains all records regarding the action of watching or rating the Titanic movie whose users are all female.

In this thesis, unless otherwise stated, whenever we mention *user group* without any clarification of type, we mean the one of Definition 2.1. However, in Chapter 3 we use Definition 2.2. Also, whenever it is clear from context, we simply user the word *group* instead of user group.

Group labels express the behavior and common demographics of group members. For a group g, we use the notion $|g|$ to denote the number of members in a common-attribute group, and number of records (activities) in a common-activity group. We use $\mathcal{G}$ to refer to the set of all user groups. $\mathcal{G}$ is often very large even with a small number of attribute values and items as it is exponential in the number of values each attribute and action take (i.e., the information overload problem.)

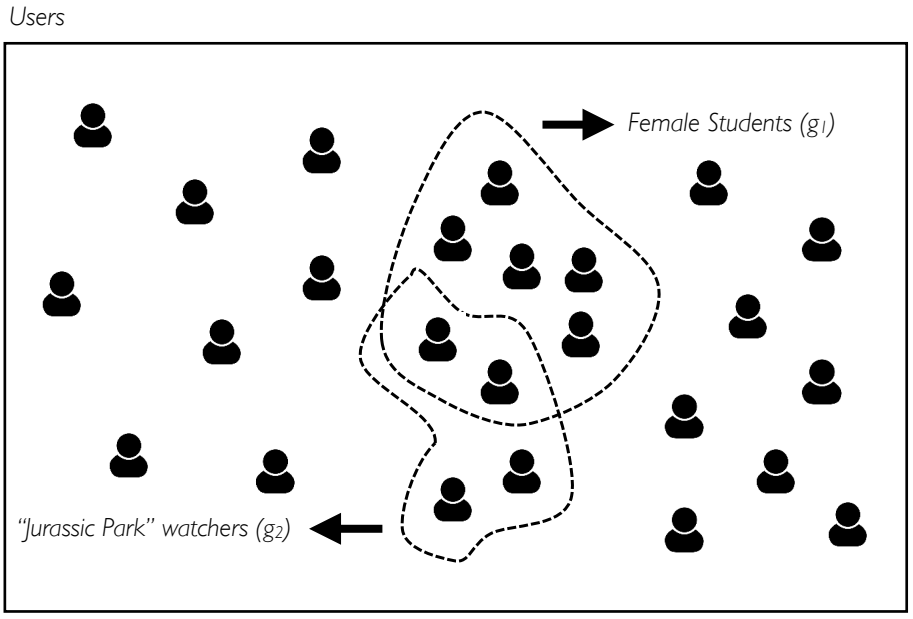


FIGURE 2.4: Illustration of User Groups.

Figure 2.4 illustrates an example of user groups. In this figure, a group g_1 contains 7 users where $l_{g_1} = [\langle \texttt{occupation} = \texttt{student}\rangle, \langle \texttt{gender} = \texttt{female}\rangle]$. When attributes are clear from the context, we usually show group labels in a more concise from, e.g., $l_{g_1} = [\texttt{student}, \texttt{female}]$. Another user group g_2 has 4 users where $l_{g_2} = [\texttt{Jurassic Park}]$. Two users are common between g_1 and g_2, i.e., they are female students who have watched the Jurassic Park movie.

2.4 Lattice of User Groups

Similarly to data cubes, the set of all possible user groups form a virtual *lattice* where nodes correspond to groups and edges correspond to parent/child and ancestor/descendant relationships. A user group g_1 is considered an ancestor of another group g_2, denoted $g_1 \supseteq g_2$, iff $\forall j$ where $\langle a_j, v_j\rangle \in g_2$, $\exists \langle a_j, v'_j\rangle \in g_1$, such that $v_j = v'_j$, or v'_j contains v_j. For example, the group of students is an ancestor of the group of young students who live in California. A partial lattice is illustrated in Figure 2.5. We consider four attributes in this figure, i.e., `gender`, `age`, `location` and `occupation` and exactly one distinct value per attribute.

Going one level up in the lattice is moving from a parent user group to its children. For instance, the user group labeled $[\texttt{student}, \texttt{male}]$ is the child of groups $[\texttt{student}]$ and $[\texttt{male}]$. A child user group has a more specific label whose members are subset of the

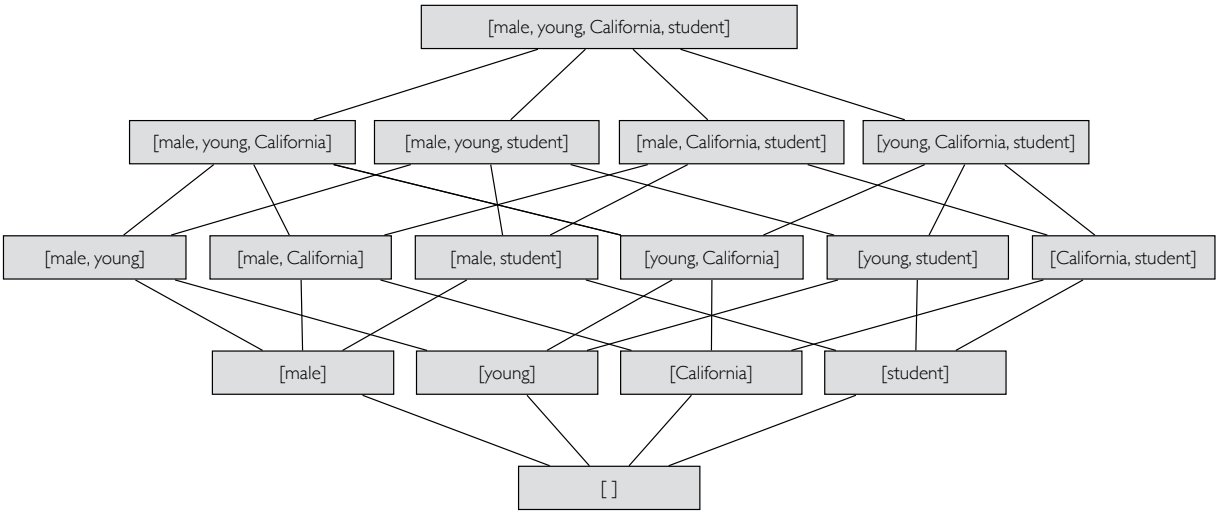


FIGURE 2.5: Partial Lattice of User Groups in MOVIELENS.

parent user group. In the above example, male students are a subset of all students and all males.

By considering all attributes, their values and all combinations, the lattice becomes extremely huge and hard to manage. To solve this problem, we often use one of the following techniques.

- **Partial Materialization.** Materialize the lattice whenever needed, i.e., generate the lattice not all at once but partially;
- **Pruning.** Prune user groups (e.g., prune groups containing fewer than a certain number of members/activities).

2.5 User Data Model in One View

Figure 2.6 illustrates all components of our user and group models in one view, in form of an example. We consider Mary, a young student. She is indeed a member of the user group labeled with [`female`, `student`]. The lattice of user groups tells us that this user group is the descendant of the group labeled [`student`] (and of course the other one labeled [`female`], which is not shown in the figure.) Also the demographics taxonomy shows in a more general concept that Mary is socially active (and not unemployed) as she is a student.

Mary has used `Google Photos` application on her phone on the 15th of August 2015 where she was at 3365 Indiana Street, San Diego based on her phone's GPS sensor. We

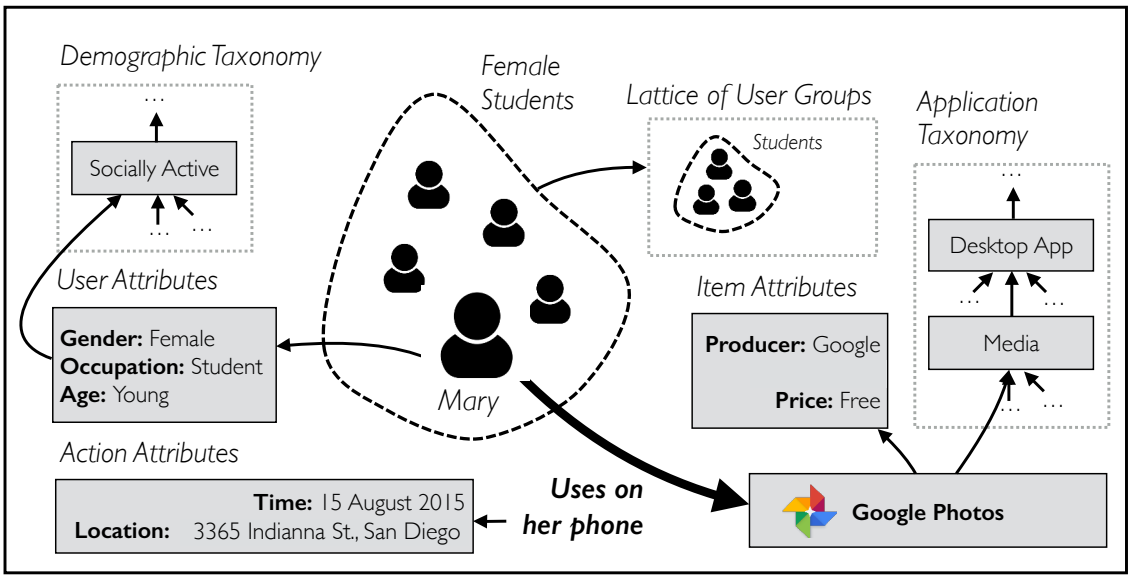


FIGURE 2.6: Components of User Data Model.

assume that a location taxonomy contains street names whose parents are POIs in that address. Based on such a location taxonomy, it becomes clear that Mary was in the Aero Club bar in California. Also based on a time taxonomy, we know that the day Mary has performed this action was a weekend. Mary is then a member of a user group labeled [`Aero Club bar`, `weekend`] whose members have once used an application in Aero Club bar in a weekend.

Mary has used `Google Photos`. This application has values for attributes `producer` and `price`. So Mary is also a member of another user group labeled with [⟨`producer` = `Google`⟩, ⟨`price` = `free`⟩] whose members use at least one application from Google which is free to download.

The `Google Photos` application can also be seen in the application taxonomy. It belongs to the category of `Media` applications. At a higher level, it belongs to the category of `Desktop applications`. Another group which Mary is a member of, has the label [`female`, `media`] whose members are females who use a media-category application on their smartphones.

2.6 User Datasets

The data model we described in Section 2.1 can be used to model many different user datasets. In this section, we introduce different datasets we use in this thesis for our experiments and proof-of-concepts. Each dataset has its own characteristics which makes

its results different from others . The datasets we have used are either available online (like MovieLens and BookCrossing) or have been collected by Web crawling (like DM-Authors and Facebook-72). Table 4.2 gives some statistics on these datasets.

	MovieLens	BookCrossing	Nokia	DM-Authors	Facebook-72
# users	6,040	278,858	38	4,907	72
# items	3,900	27,379	170	11,890	100
# attributes	8	7	80	4	2

Table 2.1: Statistics on Datasets

2.6.1 Movie-Review Dataset: MovieLens 1M

MovieLens is the dataset published by the GroupLens research group[1]. GroupLens Research continuously collects this data via the MovieLens website[2], a virtual community website that recommends movies to watch. MovieLens contains rating records $\langle u, i, s\rangle$ representing that user u has rated movie i with score s.

Different versions of the dataset have been released from 1998 to 2015 growing from 100,000 ratings (so-called MovieLens 100K) to its latest version with 20 millions of ratings (so-called MovieLens 20M). In this thesis, we use the version that contains 1,000,209 anonymous ratings (so-called MovieLens 1M) of 3,952 movies by 6,040 users. Rating records consist of a user ID (between 1 and 6040), movie ID (between 1 and 3952), a rating (based on a 5-star scale) and time. Each user provided at least 20 ratings. We assume that when user u has rated the movie i, it means u has watched i. Unless otherwise stated, in the rest of the thesis, whenever we use the term MovieLens, we mean the 1M version.

In this dataset, there exist four user attributes: `gender`, `age`, `occupation` and `zipcode`. All demographics information is provided voluntarily by users. The attribute `gender` takes two distinct values: `male` or `female`. We convert the numeric age into four categorical attribute values, namely `teen` − `ager` (under 18), `young` (18 to 35), `middle` − `age` (35 to 55) and `old` (over 55). There are 21 different occupations listed in MovieLens e.g., `student`, `artist`, `doctor`, `lawyer`, etc. Finally, we convert zip-codes to states in the USA (or "foreign", if not in USA) by using the USPS zip code lookup.[3] This produces the user attribute `location` which takes 52 distinct values.

Concerning item attributes, MovieLens only provides movie genres. The dataset contains 18 different genres. Note that a movie can have more than one genre. To enrich

[1] http://www.grouplens.org/
[2] http://movielens.org
[3] *http://zip4.usps.com*

movie attributes, we crawled IMDb[4] using the OMDb API[5] to get extra attributes, i.e., `director`, `writer` and `release year` for each movie. To join MovieLens and IMDb, we use movie titles. This mapping was not possible for all 3952 movies, because of some mismatchings. For instance some titles are in their original language (which is other than English) in one dataset and in English in the other dataset. Thus not all movies have enriched attributes, but 3650 movies.

2.6.2 Book-Review Dataset: BookCrossing

BookCrossing[6] is a dataset which is collected in a 4-week crawling in August and September 2004 from the Book-Crossing website[7], a free online book club. The Book-Crossing community was launched in 2001 with the aim to make the whole world a library. BookCrossing contains 278,858 users (anonymized but with demographic information) providing 1,149,780 ratings about 271,379 books. The number of users and items are one order of magnitude larger than MovieLens.

There are only two attributes for each user in BookCrossing: `age` and `location`. Concerning `age`, we apply the same conversion as for MovieLens (teen-ager to old). Note that in this dataset, the `age` attribute is missing for 110,776 users, which is not the case for MovieLens. Concerning `location`, we consider different geographical levels (city, state and country) as different independent attributes, hence we end up with 4 different user attributes (age plus 3 location attributes). Note that unlike MovieLens, users of BookCrossing are not located only in the United States. BookCrossing also offers information on each book (item), i.e., `writer`, `release year` and `publisher`.

2.6.3 Mobile-Usage Dataset: Nokia

Nokia dataset was made available for Nokia Mobile Data Challenge 2012[8]. It was collected from smartphones of almost 200 participants in the course of a year. The data collection campaign was conducted by the Lausanne Nokia Research Center[9].

The data collection campaign exploited sensors embedded in smartphones and other wireless devices to collect large quantities of continuous data pertaining to the behavior of individuals and social networks. In the data collection, smartphones that collect behavioral data were allocated to nearly 200 individuals from Lake Geneva region. Nokia

[4] *http://www.imdb.com*
[5] *http://www.omdbapi.com*
[6] *http://www2.informatik.uni-freiburg.de/čziegler/BX/*
[7] *http://www.bookcrossing.com*
[8] *http://research.nokia.com/page/12340*
[9] *http://research.nokia.com/locations/lausanne*

dataset contains rich data on location, social interactions, contextual attributes, media consumption, application usage and device control. Participants in the data collection campaign have given consent for the research use of their data. Table 2.2 illustrates 13 different attribute categories in this dataset.

In this thesis, we use only two sensors, *application* and *GPS* to focus on application usage: the opening of applications by users indicating what they use their smartphones for, at any time of the day. Hence a tuple $\langle u, i \rangle$ in NOKIA simply means that user u has used the application i on his/her phone. We extend tuples with two action attributes coming from GPS sensor, i.e. $\langle u, i, t, l \rangle$ which means that user u has used the application i at time t and in location l. There exists 38 users and 170 applications in this dataset.

This dataset also includes responses to a questionnaire by some users in the experiment. The questionnaire contains 17 questions. Demographics which are obtained from the questionnaires are as follows.

- Gender, age group, occupation status;
- Typical travel means (own car, bus, metro, train, bike or walk)
- Types of contacts (siblings, parents, etc.) and frequency
- Typical contact means (fixed/mobile phone, SMS, blog, etc.)
- Recent activity type (going out, visiting friends, shopping, etc.)
- Typical sharing content (photos, videos, contents, etc.) and sharing means (email, blog, social network, etc.)
- Typical online activities (e-shopping, play games, use maps, download, etc.) and frequency
- Social network of interest and frequency of usage

NOKIA is a rich dataset despite its small number of users.

2.6.4 Dataset of Data Management Researchers: DM-Authors

DM-AUTHORS contains 4,907 researchers who have at least 3 publications in one of the following top data management conferences: WWW, KDD, SIGMOD, CIKM, ICWSM, EDBT, ICDM, ICDE, RecSys, SIGIR or VLDB. We crawled authors in DBLP in October

Category	**Descriptions**	**Attributes**
Accelerometer	Contains the scan of the accelerometer sensors	square change, accelerometer samples, etc.
Application	Contains the application events	event name (closed, foreground, etc.), language, etc.
Bluetooth	Contains the bluetooth devices seen by the user	MAC prefix and address, etc.
Calendar	Contains the calendar entries	status (confirmed, tentative), event start time, event location, class (private,public), etc.
Call Log	Contains the call logs	status of short message (delivered, sent, etc.), direction (incoming, outgoing) type (voice call, short message), duration, etc.
Contacts	Contains the contact entries	anonymized name and phone number, etc.
GSM	Contains the GSM cells that the user has seen	location area code (LAC), signal strength (0 to 7), etc.
Media Play	Contains information on how user play media	artist name, track title, player state, duration, etc.
Media	Contains the media found on the device	file size, anonymized file name, etc.
Process	Contains informations on the running processes	process pull path, unix time, etc.
System	Contains general system information about the phone	battery level in %, charging state (charging, decharging, etc.), free space, free RAM space, etc.
WLAN	Contains the WLAN devices seen by the user	received signal level, active channel, etc.
GPS	Contains the GPS positions of the user	altitude, longitude, attitude, accuracy, etc.

Table 2.2: Nokia Dataset Attributes

2014 from DBLP[10] for years between 2000 and 2014. A tuple $\langle u, i \rangle$ in this dataset means that researcher u has contributed to conference/journal/keyword i.

The main advantage of this dataset over others is that each user in this dataset is known and verifiable. Thus we can check if results over this data are meaningful. For instance, in MovieLens, we know that *user 2945* is a 35 years old female engineer who lives in Nevada, United States, but we don't know who this user is. In DM-Authors instead, we mention e.g. Michalis Potamias is a young researcher who works on *probabilistic modeling* and has published a paper in CIKM. These pieces of information are easily verifiable using the author homepage and other sources (e.g., DBLP).

For each researcher, we compute the four following attributes:

Seniority. It shows the number of years since the author's first publication in DBLP. For instance, for Divesh Srivastava (AT&T Labs), this value is equal to 24, as his first publication in DBLP is in 1990. In our collected data, the lowest value for seniority is 1 and the highest 67.

To make seniority levels more granular, we discretize seniority values. Two typical ways of discretization is *equal-length* and *equal-frequency* binning [CB00]. Considering n different bins, in *equal-length* binning, the intervals of n bins have all the same size. While in *equal-frequency* binning, the number of researchers in each of the n bins, are equal. We chose $n = 5$ and discretized seniority levels using *equal-frequency* binning, as it is more data-centric than *equal-length* which is static. We denote our seniority bins with values "starting" (1 to 8 years), "junior" (9 to 12 years), "senior" (13 to 15 years), "highly senior" (16 to 21 years) and "confirmed" (22 and higher years). For instance, we label Divesh Srivastava with `confirmed`.

Number of Publications. It shows the number of publications indexed by DBLP for the researcher. Note that this counter is not limited to the above data management conferences, but it counts all publications of the researcher on DBLP. The largest number of publications in our dataset is 885. We discretize values of this attribute in the same way we did for seniority level and denote the bins with values "very few" (3 to 14 publications), "few" (15 to 28 publications), "fair" (29 to 53 publications), "high" (54 to 107 publications) and "very high" (108 publications and higher).

Publication Rate. It shows the average number of publications per year. It is calculated by dividing seniority value by number of publications before discretization. It shows how active and dynamic a researcher is. We discretize values of this attribute in the same way we did for seniority level and denote the bins with values "active" (0.18 to

[10] *http://dblp.uni-trier.de/db/*

1.47), "very active" (1.48 to 2.48), "productive" (2.49 to 3.71), "very productive" (3.72 to 6.0) and "prolific" (6.1 and higher).

Venues. It contains the set of all conferences and journals where the author published a paper at least once. For instance, the list of venues for Mumtaz Ahmad (University of Waterloo) is $\{\texttt{CIKM}, \texttt{EDBT}, \texttt{ICDE}\}$. We consider 6524 venues in our dataset.

Topics. It contains the set of topics extracted from the author's publications. First we obtained the set of author keywords using their publication titles and by removing stop-words (like *the*, *an*, etc.). We obtained 39,537 unique keywords. The most repeated word in publication titles is obviously *data* with 37,987 repetitions. Then we used LDA topic model [BNJ03] (Latent Dirichlet Allocation) to obtain 124 topics. We used DAMA-DMBOK Functional Framework[11] as a seed reference to gain 45 topics in data management. Then we specialized topics using *topics of interest* for data management conferences mentioned above. Instances of topics are data enrichment, temporal databases, query processing, etc.

Gender. We also consider author's gender as another attribute. It is a challenging task to obtain gender information for researchers because most of the time this piece of information is intentionally kept hidden. For this aim, we used an NLP resource of first names containing 54,915 names.[12] We then matched authors' first name with the resource to detect the gender. Among 4908 researchers, we obtained 3189 males, 459 females and 1259 names remained unknown.

Note that many names are common between males and females (e.g., Shadi Ibrahim is a male scientist at INRIA but Shadi Saberi is a female scientist in San Francisco Bay Area), and also many other names are not frequent and are not listed in our NLP resource (e.g., Senjuti Basu is a female scientist in Washington University, but her gender has remained unknown in our dataset due to the scarcity of the name "Senjuti".)

The number of all attribute values in DM-Authors dataset reaches 11,890 values.

2.6.5 Social Network Dataset: Facebook-72

Users in MovieLens, BookCrossing and Nokia are anonymized. Users in DM-Authors are known but not involved in the experiments. We exploit the availability of Facebook users to collect information about real users and constructed the Facebook-72 dataset. This is a great opportunity to also capture interactions between users, a piece of information which is missing in the other datasets. A tuple in Facebook-72

[11] *http://dama-dach.org/dama-dmbok-functional-framework/*
[12] *http://korrekt.org/page/Note:Sex_Distributions_in_Research*

has the same meaning as MovieLens. Thus a tuple $\langle u, i \rangle$ means that the Facebook user u has watched (or rated) the movie i.

For data collection, we developed an application using the Facebook API[13] to recruit users. Our Facebook application asks only for *public profiles* and *friend list access* permissions. Also, we anonymize the dataset by mapping Facebook IDs to a random 5-digit number.

To gather participants, we recruited 13 *seed users* (our colleagues) with our Facebook application. Seed users had to complete two tasks: *i*. rate at least 30 movies in MovieLens, and *ii*. invite between 10 and 20 of their friends to participate in the study. Seed users were not allowed to invite another seed as a friend. Friends are only asked to rate movies and not invite friends, i.e., we stopped at depth-1 of the social graph for this data collection. We recruited 72 Facebook users in the course of two months and we obtained a total of 1981 ratings. Plus, we recorded information on the way these users interact with each other in Facebook.

To prevent sparsity in the collected data, we select a subset of MovieLens movies for 72 participants to provide their preferences. The preference of each participant is asked for the two following movie sets:

- **Similar Set** which contains the top-50 movies in MovieLens in term of popularity (i.e. the number of users who rated a movie in the set).
- **Dissimilar Set** which contains top-25 popular movies, plus 25 movies with the highest variance among their ratings.

Users were instructed to provide a rating between 1 and 5 (5 being the best) for at least 30 movies listed in random order.

2.7 Conclusion

All introduced datasets have their own advantages and characteristics. MovieLens and BookCrossing are *review* datasets. Nokia is a small but rich dataset of demographics and application usage. Also DM-Authors and Facebook-72 contains real known users. In the three different components of our user group management framework, we use a subset of these datasets in different experiments based on their specific characteristics.

[13] *https://developers.facebook.com*

Chapter 3

User Group Discovery

The aim of this thesis is to propose a user group management framework as shown in Figure 1.2. The very first step in this framework is *group discovery*, i.e., given raw user data, obtain user groups by optimizing one or more quality dimensions. Parts of this work have been submitted recently and are under review.

3.1 Motivating Examples

The availability of a number of collaborative rating datasets on the social Web, such as MovieLens, a movie rating site, LastFM, a music rating site and BookCrossing, a book rating site, appeals to scientists today who design algorithms that help analysts make better decisions on complex user data analysis tasks such as crowd data sourcing (which users to ask ratings from), advertisers in determining which items to recommend to which users, and social scientists in validating hypotheses such as *young professionals are more inclined to buying self-help books*, on large datasets.

In practice, however, there does not exist analytics tools that enable the scalable, on-demand discovery of user groups. Recall that a *user group* is defined as a conjunction of attributes over rating records (Definition 2.2), such as *rich young professionals* or *teachers who live in the countryside*. Given a dataset, e.g., ratings of Woody Allen movies, we formalize the problem of discovering *high quality* user groups. Quality can be formulated as the optimization of one or more dimensions. Pattern mining algorithms can mine groups by optimizing one dimension, a.k.a interestingness measure. In Chapter 4 we use LCM algorithm [UKA04] to obtain user group. In this Chapter, however, we aim to have a deeper look at quality and formulate it as the optimization of quality dimensions dimensions such as *coverage* and *diversity*.

Optimizing coverage ensures that most input records $\langle i, u, s \rangle$ will belong to at least one group in the output. Optimizing diversity ensures that found groups are as different as possible from each other, e.g., *males and females* or *young and old*, and unveils ratings by different users. User groups with high coverage and high diversity, can help analysts make a variety of decisions such as audience targeting in advertising or hypothesis validation in social science.

Beyond coverage and diversity, another interesting dimension of group quality is its *rating distribution*. As it has been argued in previous work [DAYDY11], groups with *homogeneous* ratings may be more appealing to some applications, while groups with *polarized* ratings are preferred by others. Indeed the rating distribution in a group provides analysts with the ability to tune the quality of found groups according to specific needs. The example mentioned in Section 1.3.2 is a good case for *homogeneity*: Anna, a social scientist, wants to validate the hypothesis that western movies are mostly watched by the older generation. By reporting the average rating of 4.7 for old male reviewers, we know that most individuals in that group have high ratings. The following example shows how tuning the *rating distribution* of discovered groups leads to new discoveries when used alongside coverage and diversity.

Example 3.1. *Consider the example discussed in Section 1.3.1: Julia is responsible for finding the best target audience for a promotion on books of* John Grisham*, the American author. She has already found two promising groups:* young reviewers from Connecticut *and* middle-age community in France*. She then looks at the* variance *of ratings in those groups and finds that the former has a higher variance. This is potentially because law students in Connecticut like Grisham's books a lot (Grisham writes thrillers involving detailed trials in court), but this is not the case for all young people in that state. Since Julia is more interested in a homogenous group, she can either choose the second group or ask the system to find other groups in Connecticut.*

3.2 Challenges and Contributions

Given an input set of rating records (e.g., Sci-Fi movies from the 90's, David Lynch movies, movies starring Scarlett Johansson, etc.), our problem is that of discovering a set of user groups. Even when the number of records is not very high, the number of possible groups that could be built may be very large. Indeed, the number of groups is exponential in the number of user and item attributes' values and many groups are very small or empty. Therefore, given the ad-hoc and online nature of group discovery, our challenge is to *quickly* identify high quality user groups. We hence define desiderata

that user groups should satisfy (local desiderata) and those that must be satisfied by the set of returned groups (global desiderata).

Two qualities as local desiderata are *Describability* and *Size*.

- **Describability.** Each group should be easily understandable by the analyst. While this is difficult to satisfy through unsupervised clustering of ratings, it is easily enforced in our approach since each group must be formed by rating records of users that share at least one attribute value, which is used to describe that group.
- **Size.** Returning groups that contain too few rating records is not meaningful to the analyst. We hence need to impose a minimum size constraint on groups.

Also, we consider four qualities as global desiderata, i.e., *Coverage*, *Diversity*, *Rating Distribution* and *Number of Groups*.

- **Coverage.** Together, returned groups should cover most input rating records. While ideally we would like each input record to belong to at least one group, that is not always feasible due to other local and global desiderata associated with the set of returned groups.
- **Diversity.** Returned groups need to be different from each other in order to provide complementary information on users.
- **Rating Distribution.** Ratings in selected groups should follow a requested distribution (e.g., homogeneity).
- **Number of groups.** The number of returned groups should not be too high in order to provide the analyst with an at-a-glance understanding of the data.

A candidate solution is a group-set that verifies all above desiderata. Finding such a group-set is a hard problem because of two reasons:

1. **Huge Candidate Set**. First the pool of candidate group-sets is very large. Any possible combination of attribute value pairs can form a group, and any number of groups can form a group-set. For example, by having only 20 attribute value pairs, we end up with $1,048,575$ groups (i.e., $(2^{20}) - 1$) and over 10^{12} groups of size 5 (i.e., $\binom{1,048,575}{5}$).

2. **Need for Multi-Objective Optimization**. The second reason of hardness is that diversity, coverage and rating distribution are conflicting objectives. That means optimizing one does not necessarily lead the best values for others. Thus the need for a multi-objective optimization approach that will not compromise one objective over another. Such an approach would return *the set of all candidate group-sets* that are not dominated by any other along all objectives. In Section 3.6.1, we illustrate the conflict between our objectives.

In this chapter, we propose α-MOMRI, an α-approximation algorithm for user group discovery that considers local and global desiderata and guarantees to find group-sets that are α-far from optimal ones. Since α-MOMRI relies on an exhaustive search in the space of all groups, we propose h-MOMRI, a heuristic that exploits the lattice formed by user groups and prunes exploration in order to speed up group-set discovery. Both our algorithms admit a set of rating records of the form $\langle i, u, s\rangle$ and a constrained multi-objective optimization formulation [DRST09] and return group-sets that satisfy the formulation and are not dominated by any other group-set. In an extensive set of experiments on MovieLens and BookCrossing datasets, we analyze different solutions of α-MOMRI and h-MOMRI and show that high quality group-sets are returned by our approximation and very good response time is achieved by our heuristic.

3.3 User and Group Data Models

In this chapter, while we inherit the same global data model we already proposed for the whole framework in Section 2.1, we introduce some additions and modifications. We consider a rating tuple $r \in \mathcal{D}$ in form of $\langle u, i, s\rangle$ where $i \in \mathcal{I}$, $u \in \mathcal{U}$, and s is the integer rating that reviewer (user) u has assigned to item i. The values of s are application-dependent and do not affect our model.

We extend a rating tuple $r = \langle u, i, s\rangle$ with attributes of u and i and it becomes $\langle a_1, a_2, a_3 \ldots, s\rangle$ which concatenates the attributes for i, the attributes for u, and the numerical rating score s. Note that this is just the way we describe the data model and of course we do not make this concatenation on the data in the database.

In this chapter, we use Definition 2.2 in Section 2.1 for *common-activity user group*. We use the term *user group* to mention common-activity user group. For instance, the user group $g = [\texttt{young}, \texttt{programmer}, \texttt{NY}, \texttt{comedy}]$ contains 503 rating records (activities) in MovieLens for comedy movies whose reviewers are all young programmers in New York, United States. Figure 3.1 illustrates an example dataset with 7 rating records. Two user groups are made, g_1 for female reviewers with 4 rating records, and g_2 for

ID	Movie	Name	Gender	Age	Occup.	Rating
r_1	Toy Story	John	M	young	teacher	4
r_2	Toy Story	Jennifer	F	old	teacher	3
r_3	Toy Story	Mary	F	old	teacher	2
r_4	Titanic	Carine	F	old	other	4
r_5	Toy Story	Sara	F	young	student	3
r_6	Toy Story	Martin	M	young	student	5
r_7	Titanic	Peter	M	young	student	1

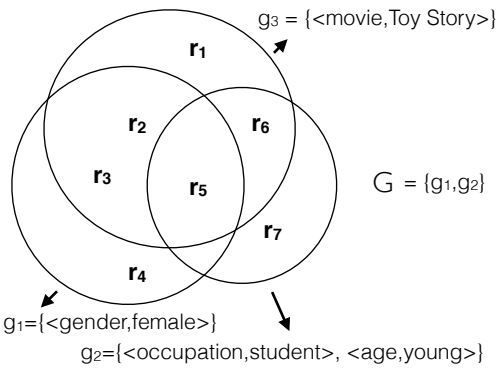


FIGURE 3.1: Example Dataset and Group-set

young students with 3 rating records. Note that there exists one record in common between the two mentioned user groups.

3.3.1 Group Quality Dimensions

We now define three quality dimensions for groups, i.e., coverage, diversity and rating distribution. We are given a set of rating records $R \subseteq \mathcal{D}$ and a group-set $G \subseteq \mathcal{G}$.

Given a rating record $r = \langle v_1, v_2 \ldots v_k, s\rangle$ where each v_i is a set of values for its corresponding attribute $a_i \in \mathcal{A}$ and a user group g with label $l_g = [a_1, a_2 \ldots a_n], n \leq k$, we say that g covers r, denoted as $r \lessdot g$, iff $\forall i \in [1, n]$, $\exists r.v_j$ such that v_j is a set of values for attribute $g.a_i$ and $g.v_j \subseteq r.v_i$. For example, the rating $\langle \texttt{female}, \texttt{WA}, \texttt{middle} - \texttt{age}, \texttt{student}, 4.0\rangle$ is covered by the group $\{\langle \texttt{gender}, \texttt{female}\rangle, \langle \texttt{location}, \texttt{WA}\rangle, \langle \texttt{age}, \texttt{middle} - \texttt{age}\rangle\}$.

Coverage is a value between 0 and 1 and measures the percentage of rating records in R contained in groups in G. Coverage guarantees the quality of completeness, i.e., how much of the input data (i.e., R) match with G.

$$coverage(G, R) = |\cup_{g \in G} (r \in R, r \lessdot g)|/|R| \tag{3.1}$$

For instance, in Figure 3.1, $coverage(G, R) = 0.8$ where $G = \{g_1, g_2\}$ and R contains rating records for the movie Toy Story.

Diversity is a value between 0 and 1 that measures how distinct groups in group-set G are from each other. Diversity penalizes (exponentially) group-sets containing overlapping groups.

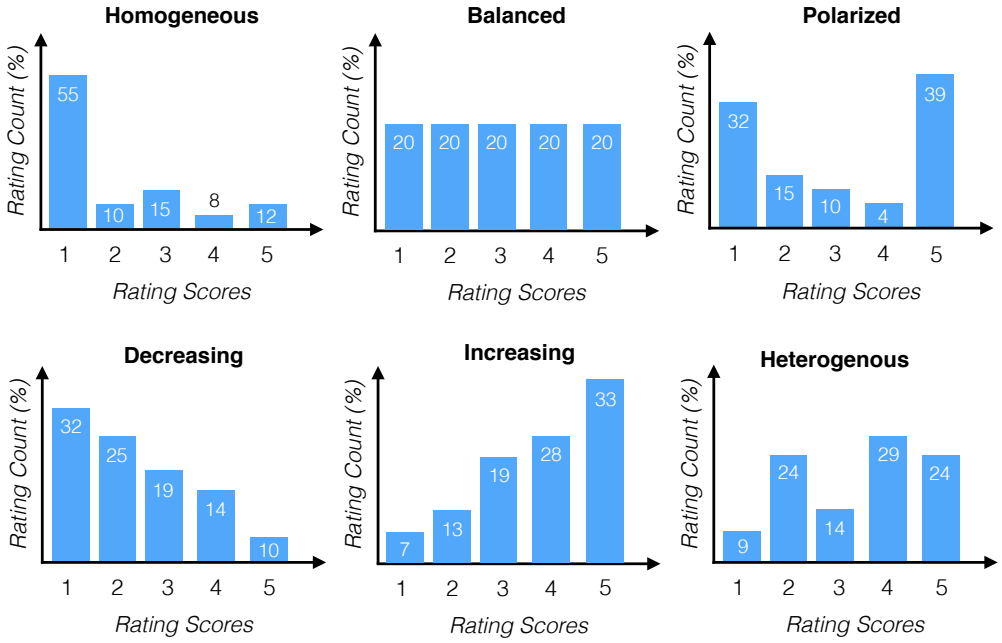


FIGURE 3.2: Different Rating Distributions for a Group-Set

$$diversity(G, R) = 1/(1 + \Sigma_{g_i, g_j \in G, i<j} | r \in R, r \lessdot g_i \wedge r \lessdot g_j |) \tag{3.2}$$

For instance, in Figure 3.1, $diversity(G, R) = 0.5$. By convention, if $|G| = 1$, we consider $g' = R$ in Equation 3.2 which leads the lowest diversity value. Diversity is useful to obtain different aspects of the input data.

Rating Distribution. A group-set G may be characterized by its rating distribution. Figure 3.2 illustrates different distributions. A rating distribution is a function $rDistb(G)$ over the set of rating scores in the rating records of groups in G. Equation 3.3 shows an example of such a function which computes the average *diameter* of ratings. Other aggregation functions could be defined.

$$rDistb(G) = avg_{g \in G}(max_{r \in g}(r.s) - min_{r' \in g}(r'.s)) \tag{3.3}$$

In Figure 3.1, $rDistb(G) = 3$. We now explain different rating distributions in Figure 3.2.

Homogeneous. A homogeneous rating distribution shows that all users in G have approximately agreed on a unique score (i.e., "1" in Figure 3.2). We use this rating distribution when we are seeking a consensus between group members and to provide

a *representative unique score* for the whole group-set. An example for this rating distribution is the movie The Godfather in IMDb, as 53.7% of ratings are for the highest score.[1]

Balanced. A balanced rating distribution shows that the preference of group members are equally distributed among scores. A user group with balanced rating distribution counts as a *neutral* group: there is no preference for any score. A neutral group can be used as a reference to see how other groups are biased towards a score.

Polarized. A polarized rating distribution shows that group members have the farthest possible preferences from each other over the set of rating records. A real example for this rating distribution is the movie Fifty Shades of Grey in IMDb, as 28.8% and 15.9% of ratings are for the lowest and highest scores, respectively.[2]

Increasing/Decreasing. We can take into consideration many other distributions depending on problem needs and specifications. For instance, *increasing* rating distribution is the one where for each score s, the number of rating records with score s is larger than or equal to the one for $s-1$. *Decreasing* rating distribution is also the inverse of the above distribution. In these two rating distributions, there exists a total order between the number of rating records in consecutive scores. A group with increasing/decreasing rating distribution potentially represents rising/falling items, i.e., items which currently have relatively low /high acceptability but may eventually emerge as prominent popular/weak items.

Based on Definition 3.3, a small value of $rDistb(G)$ leads a homogeneous group-set G and a high value leads a polarized group-set G.

3.3.2 Multi-Objective Optimization Principles

We propose to use the quality dimensions defined in Section 3.3.1 as optimization objectives. When dealing with more than one dimension to optimize, there may be many incomparable group-sets. For instance, consider the set of ratings R and group-sets G_1 and G_2 in Figure 3.3 where $coverage(G_1, R) = 0.48$ and $diversity(G_1, R) = 0.33$ and G_2 with $coverage(G_2, R) = 0.23$ and $diversity(G_2, R) = 0.5$. Each group-set has its own advantage: the former has higher coverage and the latter has higher diversity. Another group-set G_3 with $coverage(G_3, R) = 0.23$ and $diversity(G_3, R) = 0.2$ has no advantage compared to G_1, hence it can be ignored. In other words, G_3 is dominated by G_1. In this section, we borrow the terminology of multi-objective optimization and define these concepts more formally.

[1] *http://www.imdb.com/title/tt0068646/ratings?ref_=tt_ov_rt*
[2] *http://www.imdb.com/title/tt2322441/ratings?ref_=tt_ov_rt*

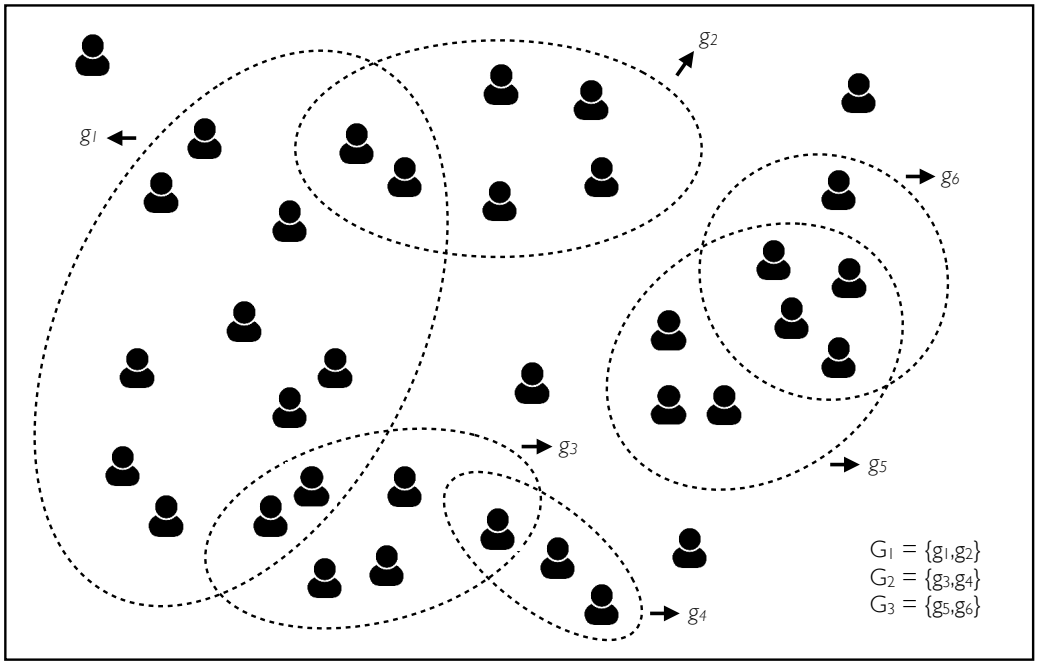


FIGURE 3.3: Illustration of User Groups and Group-sets

Definition 3.1 (Plan)**.** A plan p_i, associated to a group-set G_i for a dataset R, is a tuple $\langle |G_i|, coverage(G_i, R), diversity(G_i, R), rDistb(G_i)\rangle$.

Definition 3.2 (Sub-plan)**.** A plan p_i is the sub-plan of another plan p_j if their associated group-sets satisfy $G_i \subseteq G_j$.

Definition 3.3 (Dominance)**.** Plan p_1 dominates p_2 if p_1 has *better* or equivalent values than p_2 in every objective. The term *"better"* is equivalent to *"greater"* for *maximization* objectives (e.g., diversity, coverage and polarization), and *"lower"* for *minimization* ones (e.g., homogeneity). Furthermore, plan p_1 strictly dominates p_2 if p_1 dominates p_2 and the values of objectives for p_1 and p_2 are not equal.

Definition 3.4 (Pareto Plan)**.** Plan p is Pareto if no other plan strictly dominates p.

In the example above, plan p_2 that corresponds to G_2 dominates p_3 (for G_3) and plan p_1 (for G_1) strictly dominates p_3. Furthermore, p_1 and p_2 are Pareto plans. The set of all Pareto plans is denoted as $\mathcal{P}$.

3.4 Group Discovery Problem Definition

We define our constrained multi-objective optimization problem as follows: for a given set of rating records R and integer constants σ and k, the problem is to identify all group-sets, such that each group-set G satisfies:

- $coverage(G, R)$ is maximized;
- $diversity(G, R)$ is maximized;
- $rDistb(G)$ is optimized;
- $|G| \leq k$;
- $\forall g \in G : |g| \geq \sigma$.

The last constraint states that a group g should contain at least σ rating records, an application-defined threshold. Note that while we always maximize coverage and diversity, we may either minimize (e.g., in case of homogeneity) or maximize (e.g., in case of polarization) the rating distribution objective based on the analyst's needs. We stress the face that we optimize only one rating distribution at a time which is based on analyst request.

We state the complexity of our problem as follows.

Theorem 3.5. *The decision version of our problem is NP-Complete.*

Proof. (sketch) It is shown in [DAYDY11] that a single-objective optimization problem for user group discovery is NP-Complete by a reduction from the Exact 3-Set Cover problem (EC3). There, homogeneity is maximized and a threshold on coverage is satisfied. In our case, two new conflicting dimensions (diversity and coverage) are added. This means that the problem in [DAYDY11] is a special case of ours, hence our problem is obviously harder. □

3.5 Group Discovery Algorithms

In this section, we propose efficient algorithms for the problem we defined in Section 3.4. Multi-objective optimization is the simultaneous optimization of several objectives. The main challenge in designing an algorithm for this aim, is the multi-objective nature of the problem. A multi-objective problem can be easily solved in two following cases.

1. **Scalarization:** if it is possible to combine all objective dimensions into a single dimension and use classic single-objective optimization algorithms (e.g., Randomized Hill Climbing Exploration);
2. **Consistent Objectives:** if optimizing one dimension leads an optimized value for other dimensions.

First, it is not possible in our problem to combine all objective dimensions into a single dimension [GHK92]. We provide an intuition of the reason in the following example.

Example 3.2 (Scalarization)**.** *Let us consider the* sum *aggregation function to combine coverage and diversity values of a plan into a single score. Let* p_1 *and* p_2 *be two plans corresponding to two group-sets* G_1 *and* G_2 *in Figure 3.3 where* $coverage(G_1, R) = 0.48$*,* $diversity(G_1, R) = 0.33$*,* $coverage(G_2, R) = 0.23$ *and* $diversity(G_2, R) = 0.5$*. In this case, the score of* p_1 *is* 0.81 *and the score of* p_2 *is* 0.73*. Hence, we would prune* p_2 *while it has a higher value for coverage.*

Second, our objectives are *conflicting*, i.e., optimizing one does not necessarily lead to optimizing others. We denote a group-set that optimizes all quality dimensions at a same time, as *Zenith group-set*. Achieving the Zenith group-set is infeasible in almost all problems. For instance, a group-set may cover almost all input rating records but contains highly overlapping groups thereby hurting its diversity.

In [GHK92], a dynamic programming approach is employed to solve multi-objective optimization problems based on the *optimality principle* (POO) defined as follows:

Definition 3.6 (POO)**.** In case of maximization, if the objective value(s) of sub-plans of a plan p increases, then the objective value(s) of p cannot decrease.

As an example, for plans p_{12} and p_{13} where $G_{12} = \{g_1, g_2, g_3, g_4\}$ and $G_{13} = \{g_1, g_2, g_5, g_6\}$, we define the following sub-plans p_1, p_2 and p_3 where $G_1 = \{g_1, g_2\}$, $G_2 = \{g_3, g_4\}$ and $G_3 = \{g_5, g_6\}$. If $diversity(G_1, R) = 0.4$, $diversity(G_2, R) = 0.5$ and $diversity(G_3, R) = 0.7$, then $diversity(G_{13}, R)$ cannot be lower than $diversity(G_{12}, R)$. This example is true if the diversity objective satisfies POO (Definition 3.6). In Appendix A, we provide proofs that all our objectives (diversity, coverage and rDistb) satisfy POO. Note that we should consider one or more constraint for each objective to satisfy POO. These constraints are translated to pruning conditions in our algorithms.

We discuss 3 different algorithms for our problem: exhaustive (Section 3.5.1), approximation (Section 3.5.2) and heuristic (Section 3.5.3).

3.5.1 Exhaustive Algorithm

We adapt the algorithm in [GHK92] to our problem. It starts by calculating Pareto plans for single groups. Then it iteratively calculates plans for group-sets containing more than one group by combining single groups. At each iteration, dominated plans are discarded. The algorithm combines sub-plans to obtain new plans and exploits POO

Algorithm 1: α-approximation MOMRI (α-MOMRI)

Input: $k, \alpha > 1, R$
Output: Pareto result set $\mathcal{P}$

```
1  P ← ∅;
2  for all user groups g do
3      p_g ← construct_plan(g);
4      if p_g is not α-dominated by any other plan in P then P.add(p_g) ;
5  end
6  for n ∈ [2, k] do
7      for group-sets G of size n do
8          p_G ← construct_plan(G);
9          if p_G is not α-dominated by any other plan in P then P.add(p_G) ;
10     end
11 end
12 return P;
```

for pruning. This approach makes an exhaustive search over all combinations of user groups to find Pareto plans. This is both time and space consuming.

We propose two ways of improving the complexity of the exhaustive algorithm: *approximation-based* and *heuristic-based*. An approximation algorithm makes less enumerations with a theoretical guarantee on the quality of results. On the other hand, a heuristic can exploit the properties of the search space and prevent a brute-force execution. Sections 3.5.2 and 3.5.3 describe our two algorithms. Both algorithms are independent from the number of objectives and can be extended to consider any other objective on user groups.

3.5.2 Approximation Algorithm

Our approximation algorithm is based on the near-optimality principle (PONO) defined in [TK14]. We adapt this definition to the context of our work. For simplicity, we use $f(G)$ to denote the value of an objective function f for a group-set G.

Definition 3.7 (PONO)**.** Given an objective f and $\alpha \geq 1$, derive G' from G by replacing G_1 by G'_1 and G_2 by G'_2. Then $f(G'_1) \geq f(G_1) \times \alpha$ and $f(G'_2) \geq f(G_2) \times \alpha$ together imply $f(G') \geq f(G) \times \alpha$.

The way we defined quality dimensions in Section 3.3.1 is to satisfy PONO property. In Appendix A, we formally prove that all our objectives (coverage, diversity and rDistb) satisfy PONO. PONO overrides POO (Definition 3.6). Thus a new notion of dominance is introduced in Definition 3.8 to be in line with PONO.

Definition 3.8 (Approximated Dominance)**.** Let $\alpha \geq 1$ be the precision value, a plan p_1 α-dominates p_2 if for every objective f, $f(G_1) \geq f(G_2) \times \alpha$ where $f \in \{$*coverage*, *diversity*, *polarization*$\}$ and $f(G_1) \leq f(G_2) \times \alpha$ where f is *homogeneity*.

Definition 3.9 (Approximated Pareto Plan)**.** For a precision value α, plan p is an α-approximated Pareto plan if no other plan α-dominates p.

It is shown in [TK14] that generating less plans makes a multi-objective optimization algorithm run faster. This is because the execution time heavily depends on the number of generated plans. Thus a pruning strategy dictated by PONO is at the core of an approximation algorithm for multi-objective optimization.

We adapt the α-approximation algorithm proposed in [TK14] to the context of our problem and propose α-MOMRI (Algorithm 1). The main idea is to exploit a dynamic programming approach. The algorithm begins by constructing a plan for each single user group (lines 2 to 5). We keep all non α-dominated plans of single groups in a buffer. Then it builds group-sets of size 2 up to size k using plans in the buffer (lines 7 to 11). After each iteration, we remove α-dominated plans from the buffer. At the end, we return the buffer content. This approach creates a tree between group-sets, which we call *group-set tree*. For instance in a group-set tree, two group-sets $\{g_1, g_2\}$ and $\{g_3, g_4\}$ are children of the parent group-set $\{g_1, g_2, g_3, g_4\}$.

The crucial part of this simple algorithm is its pruning mechanism using the precision value α. In the special case of $\alpha = 1$, the algorithm operates exhaustively (as in Section 3.5.1). If $\alpha > 1$, the algorithm prunes more and hence is faster. In the latter case, a new plan is only compared with all plans that generate the same result. But a new plan is only inserted into the buffer if no other plan approximately dominates it. This means that α-MOMRI tends to insert fewer plans than the exhaustive algorithm. Figure 3.4 helps illustrate this statement using two of our objectives: diversity and coverage. The exhaustive algorithm inserts new plans if they do not fall within the dominated area, but α-MOMRI inserts new plans if they neither fall into the dominated nor into the approximately dominated area.

3.5.3 Heuristic Algorithm

A heuristic algorithm has obviously its own advantages and disadvantages. Of course a heuristic algorithm does not provide any approximation guarantee. Eventually, it returns a subset of Pareto set. Nevertheless, the fact that it generates a subset of the Pareto makes it faster.

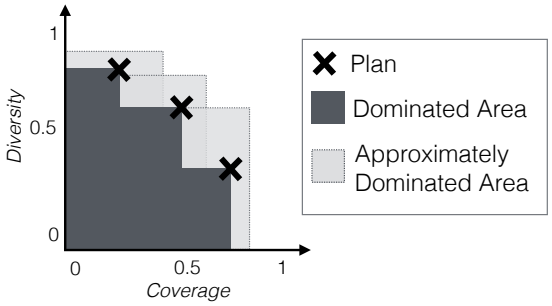


FIGURE 3.4: Dominance Areas

Algorithm 2: Heuristic MOMRI (h-MOMRI)

Input: k, α, R
Output: Result set $\mathcal{P}_i$

1 $\mathcal{P}_i \leftarrow \emptyset$
2 $\mathcal{N} \leftarrow$ Set of intervals on *diversity* values
3 **for** n *times* **do**
4 $\quad G_s \leftarrow random_groupset(k)$
5 $\quad G_s^* \leftarrow SHC(G_s)$
6 $\quad interval \leftarrow get_interval(G_s^*)$
7 $\quad \mathcal{N}[interval].add(G_s^*)$
8 **end**
9 **for** $interval \in \mathcal{N}$ **do**
10 $\quad$ Keep non-dominated plans in *interval* and add them to $\mathcal{P}_i$
11 **end**
12 $\mathcal{P}_i \leftarrow satisfy_rDistb(\mathcal{P}_i)$
13 **return** $\mathcal{P}_i$

Algorithm 2 illustrates our heuristic algorithm h-MOMRI. The algorithm starts by making n different iterations on finding optimal points to avoid local optima (lines 3 to 8). At each iteration, the algorithm begins with a random group-set of size k called G_s (line 4). Then a *Shotgun Hill Climbing* [RN03] local search approach (*SHC*) is executed (Algorithm 3) to find the group-set with optimal value starting from G_s (line 5). *SHC* maximizes coverage. Diversity is already divided into intervals $\mathcal{N}$ for each of which a buffer is associated. The resulting group-set of *SHC* is placed in the buffer whose interval matches the diversity value of the group-set (line 7). Finally, n different solutions are distributed in different interval buffers. The algorithm then iterates over interval buffers to prune dominated plans (lines 9 to 11). Based on Definition 3.3, a plan is pruned and removed from its buffer if it is dominated by other plans. Finally, for each interval, we report one unique solution that has the best value for the requested rDistb (line 12). The best value for homogeneity is the lowest, while for polarization, it is the highest. If rDistb is not specified, all plans in all buffers will be returned.

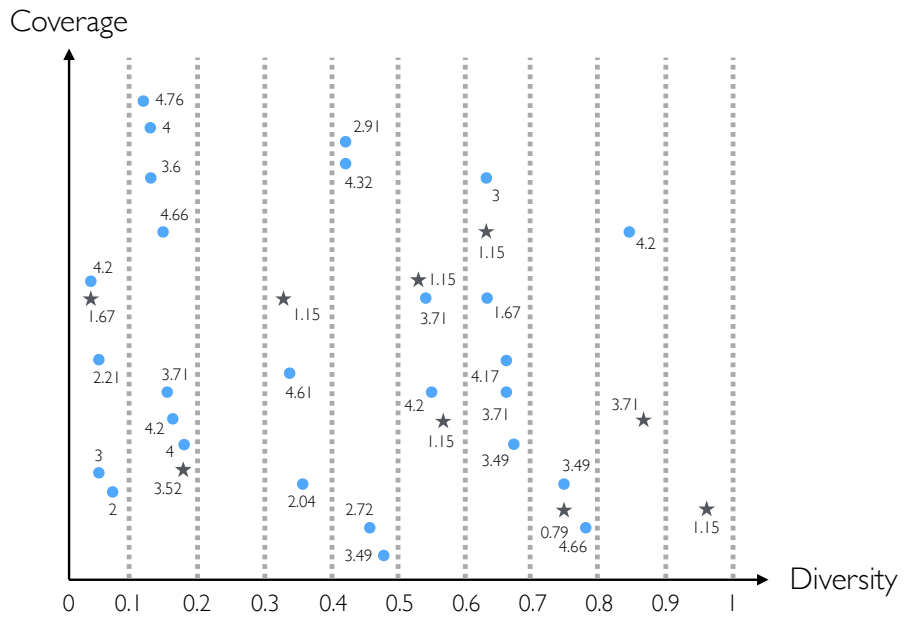


FIGURE 3.5: Illustration of the h-MOMRI Process

Figure 3.5 illustrates the process of h-MOMRI by an example. Each point in the figure is a solution. Diversity is divided in intervals of length 0.1. In each interval, different executions of *SHC* produce different maximized values for coverage. Among these points, the one having the optimized value of rDistb (minimum value in case of homogeneity) is marked with an asterisk. The set of points marked with an asterisk is the output, i.e. a subset of the whole Pareto set.

Algorithm 3: Shotgun Hill Climbing (*SHC*) Algorithm

Input: Group-set G, R
Output: Optimized group-set G^*

```
1  G* ← ∅
2  while true do
3      C ← ∅
4      for g ∈ G and each lattice-based parent g' of g do
5          G' ← G − {g} + {g}'
6          C.add(G', coverage(G', R))
7      end
8      let (G'_m, coverage(G'_m, R)) be the pair with maximum coverage
9      if coverage(G'_m, R) ≤ coverage(G, R) then
10         G* ← G
11         return G*
12     end
13     G ← G'_m
14 end
```

SHC operates on a generalization/specialization lattice of groups (as in Figure 2.5).

Navigation of this lattice in a downward fashion satisfies a monotonicity property for coverage described in the following theorem.

Theorem 3.10. *Given any two groups g and g' where g is the parent of g', the coverage of g is no smaller than the coverage of g'.*

Proof. Given any two user groups g and g' where g is the parent of g', let X denotes the description of g. For g' to be the child of g, its description should have one more attribute-value pair. Thus the description for g' should be $X \cup \{a = v\}$, where $\{a = v\}$ is the attribute-value pair which holds for all ratings in g' but not g. Thus g covers all ratings which are covered by g' plus the ratings r where $r.a \neq v$. Thus g covers as many ratings as g' covers or more. □

Note that in a bi-objective context, *SHC* can optimize each one of coverage and diversity. However, to benefit from the monotonicity property, we use *SHC* to optimize coverage. Nevertheless, if we optimize diversity using *SHC*, navigation in the generalization/specialization lattice is nothing but a random walk over the space of groups.

SHC verifies all local neighbors of a group for an improvement of coverage. If no improvement is achieved, it stops and returns the current group-set. For instance, consider the input group-set $G_s = \{g_1, g_2\}$ where $g_1 = [\texttt{male}, \texttt{student}]$ and $g_2 = [\texttt{California}, \texttt{student}]$. We obtain a coverage of 0.79 for G_s. Keeping g_2 fixed, the resulting combinations by swapping g_1 with its parents are either $g_3 = [\texttt{male}]$ or $g_4 = [\texttt{student}]$. For instance, the coverage of $G'_s = \{g_2, g_3\}$ is 0.81. As we observe an improvement, we iterate on this new group-set G'_s to improve coverage.

3.6 Experiments

We run 3 sets of experiments. The first set justifies the need for multi-objective optimization. In the second set, we vary different parameter values in order to find the most appropriate values. The last set is a comparative evaluation of α-MOMRI and h-MOMRI on the quality of retuned groups and the scalability of those algorithms.

We consider MovieLens (Section 2.6.1) and BookCrossing (Section 2.6.2) for our experiments. Ratings in MovieLens are expressed on a scale from 1 to 5 (higher values denoting higher appreciation) while in BookCrossing, it is from 1 to 10. We divide the rating scores of the latter dataset by two, to make both datasets uniform. We implement our prototype system in Java (JDK 1.8.0). All scalability experiments are conducted on a 2.4 GHz Intel Core i5 with 8 GB of memory on OS X 10.9.5 operating system.

For our experiments, we consider four different sets of input records described in Table 3.1. Each item contains at least 50 ratings. We assume that an analyst can go through all ratings manually, if they are fewer than 50 ratings.

Dataset	Item (movie or book)	Characteristic
MOVIE LENS	American Beauty (1999)	Highest # of ratings (3429)
	Celtic Pride (1996)	Lowest # of ratings (51)
	Sanjuro (1962)	Highest avg. rating score (4.6)
	Kazaam (1996)	Lowest avg. rating score (1.47)
BOOK CROSSING	Wild Animus (2004)	Highest # of ratings (2502)
	Scarlet Letter (1850)	Lowest # of ratings (51)
	Free (2002)	Lowest avg. rating score (0.36)
	Ground Zero & Beyond (2003)	Highest avg. rating score (4.0)

TABLE 3.1: Input Sets of Rating Records

For an input set of records, our algorithms return a set of group-sets. We now illustrate an example output of α-MOMRI. The same observation holds for h-MOMRI. Given a set of records R for the movie American Beauty in MOVIELENS, $k = 3$, $\sigma^3 = 10$ and the request for minimizing the rating diameter (i.e., homogeneity), one of the returned group-sets is $G_1 = \{g_1, g_2, g_3\}$ where $g_1 = [\texttt{male}]$, $g_2 = [\texttt{female}, \texttt{old}]$ and $g_3 = [\texttt{female}, \texttt{Connecticut}]$. The objective values for G_1 are as follows: $coverage(G_1, R)$=0.74, $diversity(G_1, R)$=0.25 and $rDistb(G_1, R)$=0.38. This group-set has a high coverage, as it only misses female reviewers who are neither old nor living in Connecticut. It also has a high diversity[4], as only 3 female reviewers (out of 946) for American Beauty are both old and living in Connecticut. Finally, it has also a low rDistb, i.e., all groups in G_1 are homogeneous.

Another group-set for R is $G_2 = \{g_4, g_5, g_6\}$ where g_4=$[\texttt{male}, \texttt{teen} - \texttt{ager}]$, $g_5 = [\texttt{Arizona}]$ and $g_6 = [\texttt{old}]$. The objective values for G_2 are as follows: $coverage(G_2, R)$=0.1, $diversity(G_2, R)$=0.33 and $rDistb(G_2, R)$=0.11. While G_2 has a lower coverage than G_1, it has a better score for the two other objectives. Thus G_1 and G_2 are incomparable.

3.6.1 Need for Multi-objective Optimization

We already discussed in Section 3.5 that consistency of objectives makes the multi-objective problem trivial. In this experiment, we maximize coverage and observe how values of diversity and rDistb evolve. To maximize coverage, we use Algorithm 3, i.e. a Shotgun Hill Climbing algorithm. Figure 3.6 illustrates the results for different sets of

[3] *Minimum group size*

[4] *Note that our diversity formula defined in Equation 3.2 heavily penalizes overlapped users. Thus we consider the diversity value of* 0.25 *as a high diversity as the number of overlapping users are only 3.*

input rating records: American Beauty (A), Celtic Pride (C), Sanjuro (S) and Kazaam (K). Each point illustrates the objective values for each of 20 runs. Note that this experiment illustrates the need for multi-objective optimization and is is independent from the heuristic and the approximation algorithms.

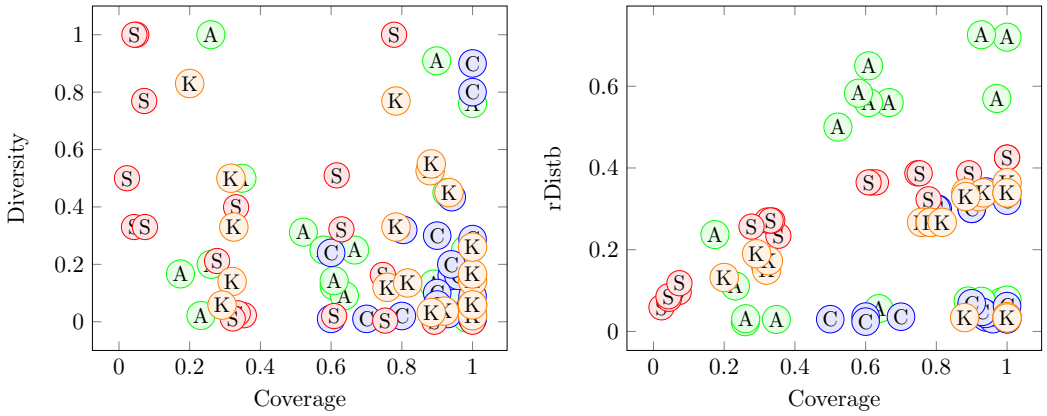


FIGURE 3.6: Conflicting Objectives on MOVIELENS

If optimizing coverage leads an optimized value for diversity, then in Figure 3.6 left, most points should fall in the top-right corner of the figure. Also in case of rDistb, in Figure 3.6 right, most points should fall in in bottom-right of the figure. We observe that in general, no correlation exists between the optimized value of coverage and other objectives. Thus each objective should be optimized independently. The same result was obtained for BOOKCROSSING.

3.6.2 Effect of Application-Defined Parameters

In this section, we examine the influence of different parameters of Algorithms 1 and 2. The parameters which are employed by h-MOMRI are the number of intervals (*nbintervals*) and the number of iterations (*nbiterations*). Also both algorithms employ two other parameters: minimum group size (σ) and maximum number of groups in a group-set (k). By default, we consider 10 intervals of diversity and 500 iterations for h-MOMRI and $\alpha = 1.5$ for α-MOMRI. For both algorithms, we consider $k = 5$ and we maximize homogeneity. We report results for different sets of input rating records in Table 3.1.

3.6.2.1 Minimum Group Size (σ)

Not all combinations of attribute values can form a group, because some combinations may not cover at least σ rating records. For instance, among rating records for the movie Toy Story, there exists only 1 record which can be described by this label: $\langle$`Male`, `Young`, `lawyer`, `CA`$\rangle$. Thus for any $\sigma > 1$, this group would not be formed. In the

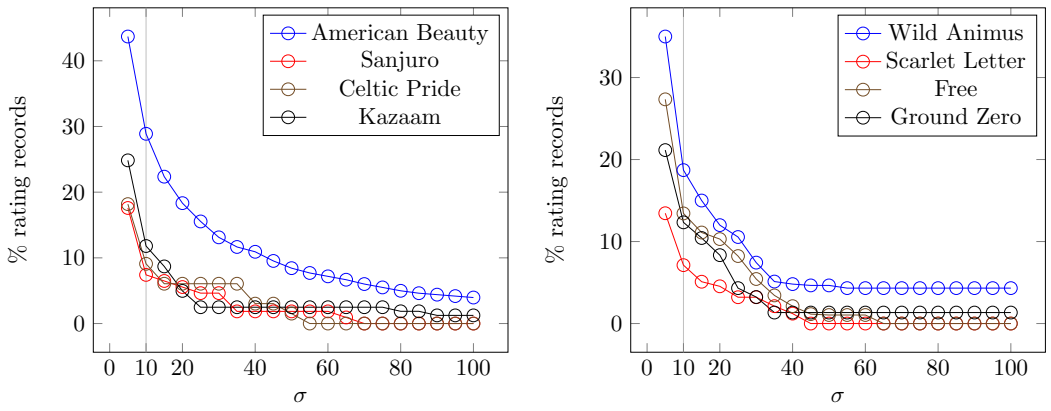


FIGURE 3.7: Number of Groups as a Function of σ for MOVIELENS (left) and BOOKCROSSING (right)

first experiment, we illustrate the evolution of the number of groups by varying σ. Figure 3.7 illustrates the results for our 4 different sets of input ratings in Table 3.1. The figure demonstrates a *long-tail* [GBGP10]: A few rating records are extremely frequent, but the majority of the dataset is composed of a large number of infrequent rating records. The long-tail transition is smoother in case of MOVIELENS as it is denser, i.e., its average number of ratings per user is 4.14 times larger than BOOKCROSSING. The long-tail reveals that choosing a fair value of σ is indeed challenging. In our experiments, we fix $\sigma = 10$ for both datasets, as this value is a border-line between the frequent ratings and the long tail.

3.6.2.2 Number of Intervals and Iterations

Next, we examine the effect of other parameters on *execution time* and *number of solutions*. When there is more than one objective to optimize, there exists potentially many optimal solutions. Because those are incomparable (Example 3.2), it becomes tedious for an analyst to deal with thousands of solutions. On the other hand, a limited subset of these solutions may miss some interesting ones. Also, the execution time is computed for fetching the ratings, analyzing the input set of ratings and constructing final solutions.

Figure 3.8 shows the effect of *nbintervals* on execution time and number of solutions. We vary *nbintervals* from 2 to 40. Obviously increasing *nbintervals* implies increasing the result precision. However, we observe that it does not influence the size of the result space or the execution time. Each set of input ratings has almost a same value for all number of intervals. The order in which the values appear is in accordance with their number of input rating records. There exists different classes of values. For items (movies and books) with less than 100 rating records, the execution time and the result

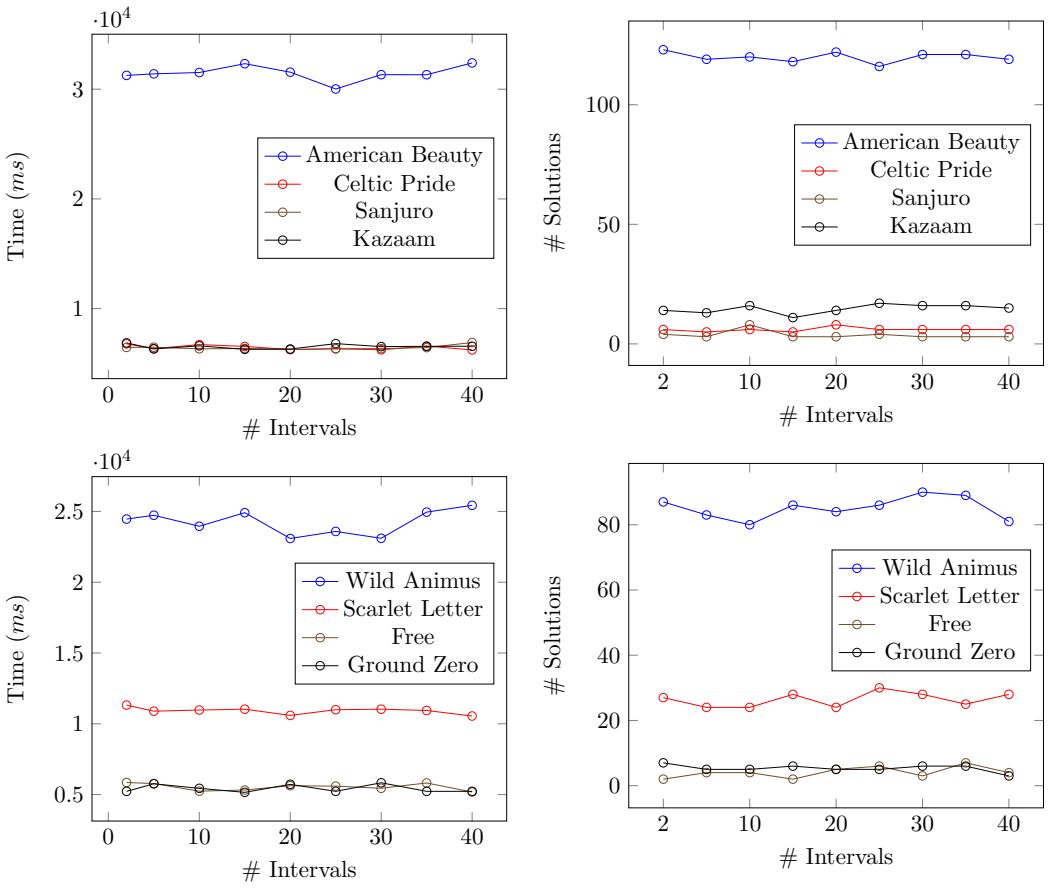


FIGURE 3.8: Effect of *nbintervals* on Execution Time (left) and Result Space Size (right) for MOVIELENS (top) and BOOKCROSSING (bottom)

space size are pretty similar. It is also the case for items with more than 1000 rating records (i.e., the movie American Beauty and the book Wild Animus).

We vary *nbiterations* from 2 to 2000 to measure its effect on execution time and number of solutions. The hypothesis is that increasing the number of iterations leads to increasing the result space size. We observe that this hypothesis is only true when there is more than 1000 input rating records. In all other cases, the increase in number of solutions is negligible. Regarding the execution time, a linear behavior is observed which was expected.

3.6.2.3 Number of Returned Groups (k)

Finally, we examine the effect of k on execution time and performance (Figure 3.9). We vary k from 2 to 10. In all sets of input rating records, increasing k leads decreasing the size of the result space. Indeed, a bigger k means having bigger group-sets and less results. Nevertheless, when there are less than 1000 input rating records, the decrease is

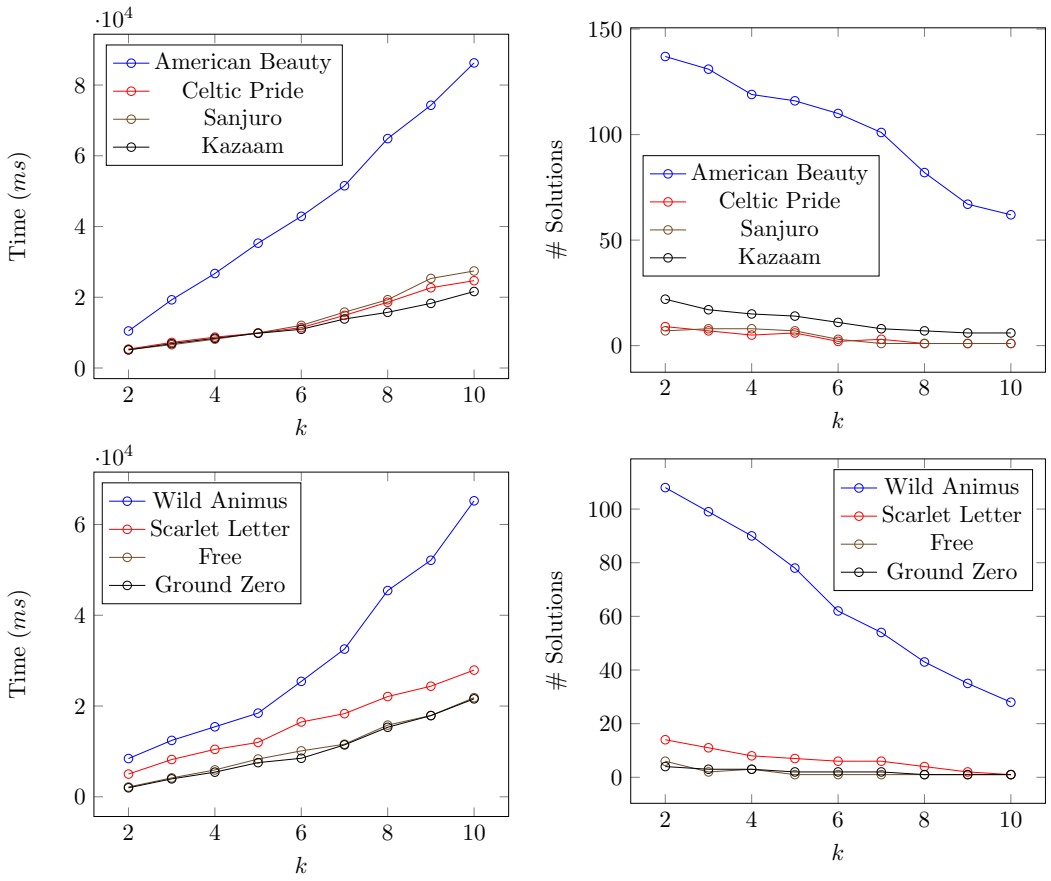


FIGURE 3.9: Effect of the Number of Returned Groups (k) on Execution Time (left) and Result Space Size (right) for MOVIELENS (top) and BOOKCROSSING (bottom)

negligible. The hypothesis is that increasing the number of iterations leads to decreasing performance.

3.6.3 Comparison of Algorithms

In this section, we compare h-MOMRI and α-MOMRI regarding their execution time and the number of solutions they produce. Our hypothesis is that h-MOMRI has a manageable solution space size compared to α-MOMRI which leads to a reduced execution time.

First we compare the quality of algorithms regarding the dominance of solutions. In multi-objective optimization, if for two algorithms X and Y, the majority of X's solutions dominate Y's, it means that X is able to produce solutions with higher quality than Y. In this experiment, we run the same comparison between α-MOMRI and h-MOMRI. For this experiment, we need to compare each pair of α-MOMRI and h-MOMRI solutions. We count the number of times each algorithm is the winner and also the number of times each algorithm is the winner in each single objective. For each algorithm and

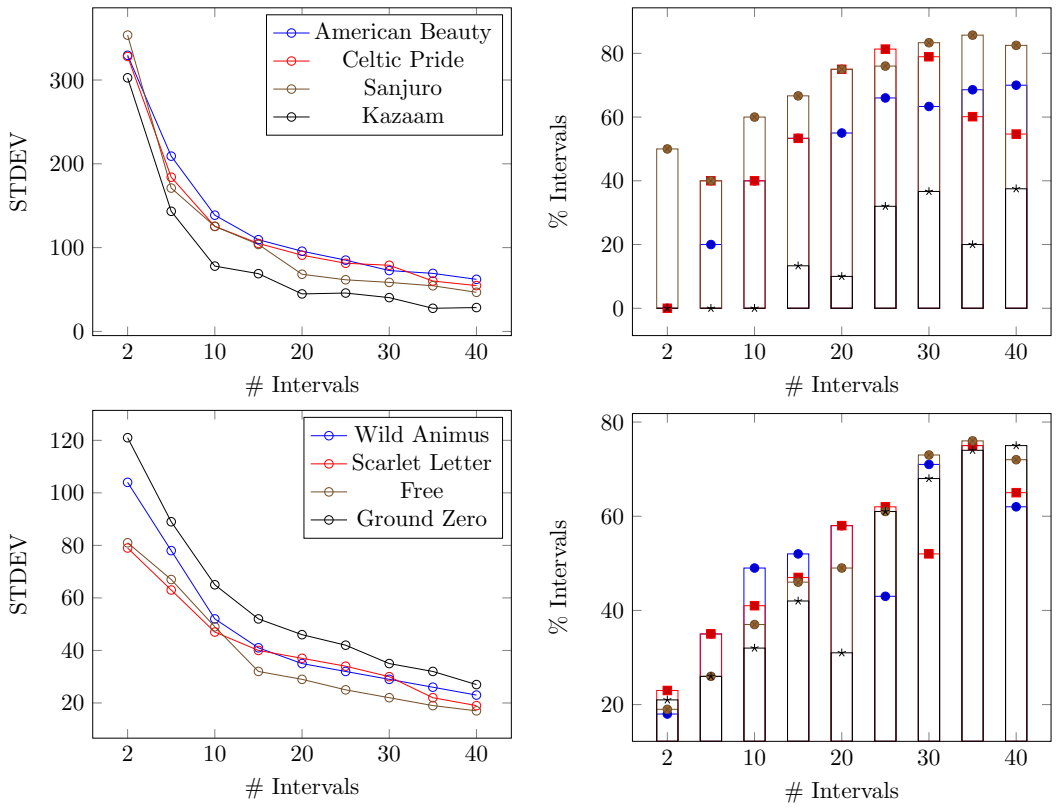


FIGURE 3.10: Distribution of Solutions in Intervals in MOVIELENS (top) and BOOKCROSSING (bottom)

for each objective, we also report the average supremacy. We consider $\alpha = 1.15$ for α-MOMRI and $nbintervals = 40$ for h-MOMRI. We denote the set of α-MOMRI solutions as $\mathcal{P}$ and the set of h-MOMRI solutions as $\mathcal{P}_i$.

We observe that for all sets of input rating records in Table 3.1, at least 62% of solutions in $\mathcal{P}_i$ are dominated by solutions in $\mathcal{P}$. This is because α-MOMRI generates the complete set of α-approximated Pareto plans, while h-MOMRI produces a subset. For instance, for the movie American Beauty, α-MOMRI produces 16 times more solutions than the heuristic algorithm. It is 14 times larger for the book Wild Animus. Evidently the solutions in $\mathcal{P}_i$ are either as good as $\mathcal{P}$'s or worse. Our results show that although α-MOMRI presents a huge set of all Pareto plans, h-MOMRI can return an acceptable representative subset where almost half of solutions are as good as the set $\mathcal{P}$.

Concerning the huge difference in the size of solution sets, potentially a fairer comparison is to consider objective values to neutralize the influence of size. We observe that for all sets of input rating records in Table 3.1, h-MOMRI can achieve a supremacy over α-MOMRI in 39.4% of cases. This is a promising result for h-MOMRI which is in-line with our findings regarding the dominance comparison. We believe that the supremacy of h-MOMRI can be increased in two ways: (i). by achieving a better balance of the

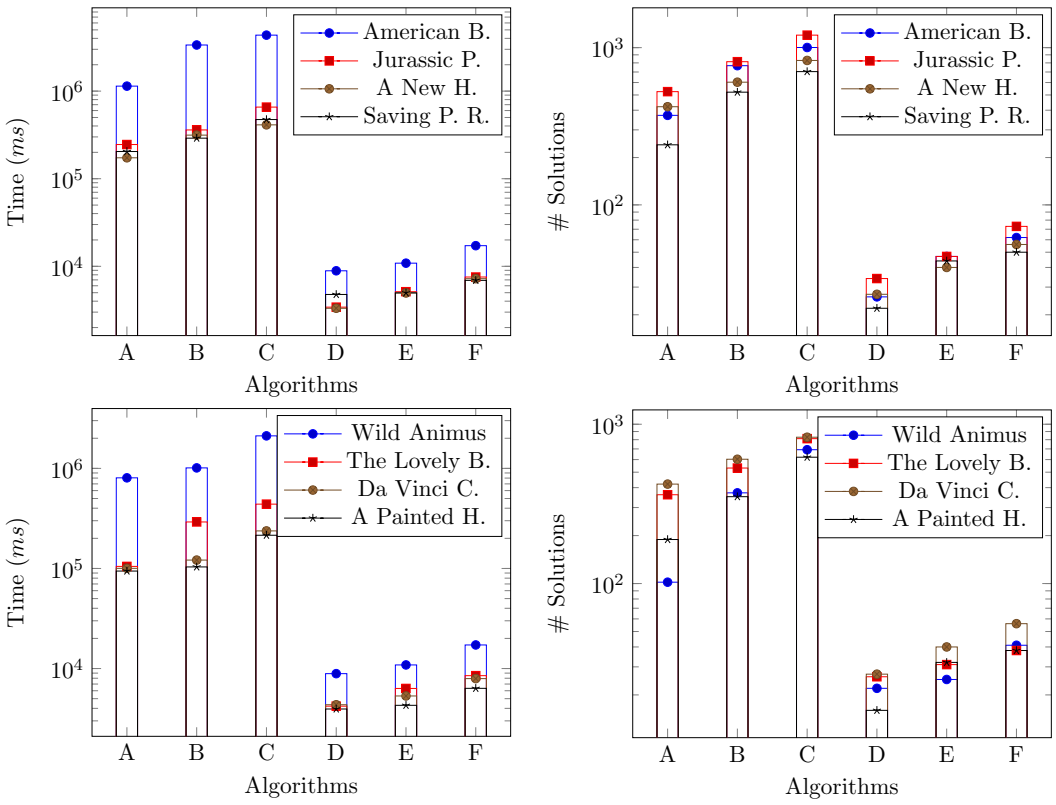


FIGURE 3.11: Comparison of α-MOMRI and h-MOMRI Algorithms in Execution Time (left) and # Solutions (right) on MOVIELENS (top) and BOOKCROSSING (bottom). α-MOMRI instances are $\alpha = 2$ (A), $\alpha = 1.5$ (B) and $\alpha = 1.15$ (C), and h-MOMRI instances are with 5 intervals (D), 10 intervals (E) and 40 (F) intervals.

solution space size in each interval, and (ii). by employing a more intelligent navigation mechanism for diversity and rDistb as we do for coverage. We discuss the former in the next piece of experiments, while the latter is future work.

In the second comparative experiment, we analyze the distribution of solutions in h-MOMRI among diversity intervals. Note that the intervals have the same width. If the solutions are equally distributed among intervals, the probability of missing Pareto plans decreases. Because in this case, there exists enough instances in each interval which makes the probability of achieving Pareto plans statistically more powerful. For this experiment, we first observe a huge amount of empty intervals for most sets of input rating records. This is mainly because for some set of input rating records, the maximum possible diversity is not 1, but lower. In this case when we discretize diversity values into fixed-width intervals, many of them remain empty. Hence in this experiment, we discretize diversity values between zero and maximum possible diversity value for the set of input rating records.

Figure 3.10 illustrates the results for different intervals and different sets of input rating records. The left chart illustrates the standard deviation for the number of solutions in

intervals. If for a set of input rating records, all intervals contain the same number of solutions, then the standard deviation is equal to zero. Also, the right chart illustrates number of intervals with no solution, i.e., empty intervals (like the interval [0.2,0.3] in Figure 3.5).

We observe a high heterogeneity when *nbintervals* < 10 for all sets of input rating records and for both datasets. This means that by considering less than 10 intervals, we will potentially miss many Pareto plans. On the other hand, increasing the number of intervals leads to increasing the number of empty intervals which has the same consequence, i.e., missing Pareto plans. We then fix *nbintervals* to 10 as it exhibits the best tradeoff between heterogeneity and emptiness. This value of *nbintervals* increases the chance of discovering more Pareto plans in h-MOMRI, but as some amount of heterogeneity still remains even for *nbintervals* > 10, we cannot consider h-MOMRI as a safe replacement for α-MOMRI.

Now we compare α-MOMRI and h-MOMRI concerning their performance and the number of solutions they produce. We consider 3 different instances for each algorithm: for α-MOMRI, we consider instances with $\alpha = 2$ (A), $\alpha = 1.5$ (B) and $\alpha = 1.15$ (C), and for h-MOMRI, we consider instances with 5 (D), 10 (E) and 40 (F) intervals. We have seen in previous experiments that all sets of input rating records with less than 1000 records exhibit a similar behavior, thus we decided to run this experiment with 4 items having the highest amount of rating records. In MovieLens, top-4 movies in number of rating records are American Beauty with 3428 records, A New Hope with 2991 records, Jurassic Park with 2672 records and finally Saving Private Ryan with 2653 records. In BookCrossing, top-4 books in number of rating records are Wild Animus with 2502 records, The Lovely Bones with 1295 records, The Da Vinci Code with 898 records and finally A Painted House with 838 records.

Figure 3.11 illustrates the results. As expected, in general the number of solutions produced by h-MOMRI is one order of magnitude smaller than α-MOMRI in both datasets. In both algorithms, the number of rating records plays an important role and increases the number of solutions.

3.7 Conclusion

In this chapter, we discussed the first component of our user group management framework, i.e., group discovery. We investigated the question of finding the best group-sets that characterize a database of rating records of the form $\langle i, u, s\rangle$, where $i \in \mathcal{I}$, $u \in \mathcal{U}$, and s is the integer rating that user u has assigned to item i. We showed that the problem

of finding high-quality group-sets is NP-Complete and proposed a constrained multi-objective formulation. Our formulation incorporates local and global group desiderata. The hardness of our problem is due to the large space of user groups and the difficulty of achieving coverage of input ratings and diversity of returned group-sets at the same time. We proposed two algorithms that find group-sets as instances of Pareto plans. The first one α-MOMRI, is an α-approximation algorithm and the second, h-MOMRI, is a heuristic-based algorithm. Our extensive experiments on MovieLens and BookCrossing datasets show that our approximation results in high quality groups and that our heuristic is very fast without compromising quality too much.

Chapter 4

User Group Analysis

We discussed in Chapter 3 how to discover user groups by optimizing one or more quality dimensions. As shown in Figure 1.2, the next step in the user group management framework is to *analyze* user groups. This step is necessary to tackle the problem of Information Overload: The output of a user group discovery step often contains millions of user groups. It is a tedious task for an analyst to skim over all produced groups. Thus we need *analysis* tools to provide insights to analysts. The different approaches to handle discovered groups, are as follows:

Compression. In this approach, we return a subset of user groups that compresses the whole set of groups in the best possible way [VVLS11]. This idea is based on *Minimal Description Length* (MDL) principle [Grü07] which leads to pruning many user groups. This is a *lossy* analysis. Although with lossy analysis, the size of the user group space becomes easily manageable, there is a high probability that one or more interesting groups are pruned. In this thesis, we propose two *lossless* analysis methods described below.

Abstraction. In this approach, instead of pruning, we summarize groups using an *abstraction* operator on a generalization/specialization taxonomy of items. This is one of our contributions for *user group analysis* which we discuss in Section 4.1. This work was published in [OTAT+13].

Navigation. Instead of reducing the size of the user group space either by compression or abstraction, another approach is to navigate in this space while optimizing one or more desiderata at each step to reach a subset of interesting user groups. This is our second contribution which we describe in Section 4.2. This work was published in [OTAYT15].

4.1 User Group Analysis via Abstraction

Nowadays, large amounts of user-generated content representing behavioral data are made available. This is particularly true for data generated by users carrying a mobile phone and moving in different geographic regions. User groups can reveal interesting information about users. For instance, a user group can be the set of members who use *Media Player* on their phone on weekends. We already introduced in Chapter 3 an approach to discover user groups. The challenge is that there can be millions of automatically discovered user groups, hindering their analysis. The possibility to organize attributes forming group labels along space and time taxonomies is a new opportunity to reduce the number of groups in the space. We define an *abstraction* analysis primitive which operates on a single user group at a time. When applied to a user group g, *abstraction* reduces the size of its label l_g and as a side-effect, the size of the user group space.

4.1.1 Motivating Example

User group abstraction exploits hand-crafted domain taxonomies. We illustrate it on the following example.

Example 4.1 (Abstraction in NOKIA)**.** *We are given a group* $g_1 = [$`female`, `age` $39-50$, `Email`, `Bluetooth`, `Contacts`, `noon`$]$ *whose members are 39-50 years old females who use Email, Bluetooth and Contacts applications on their smartphone around lunch time. The group* g_1 *could be abstracted into* $g_1^A = [$`female`, `age` $39-50$, `Desktop Communication`, `noon`$]$ *if the collective usage of the applications in the original user group covers that of a more general* `Desktop Communication` *class in the taxonomy. This abstraction makes use of a taxonomy on applications (e.g., Figure 4.2) that dictates the semantics of abstraction.*

4.1.2 Data Model

In this chapter, we introduce a specialization of the general data model (discussed in Section 2.1). We consider tuples $d \in \mathcal{D}$ of the form $\langle u, i, l, t \rangle$ where $u \in U$, $i \in I$, and l and t, a time-stamp, represent the location and time user u has used (opened, watched, rated, voted, etc.) item i. In this section, we use the *common-attribute* definition of user group (Definition 2.1).

User attributes, items, location and time, are organized in *hand-crafted* taxonomies. The values of each user attribute in $\mathcal{A}$ are organized in a taxonomy τ_A (see Figure 4.1 for

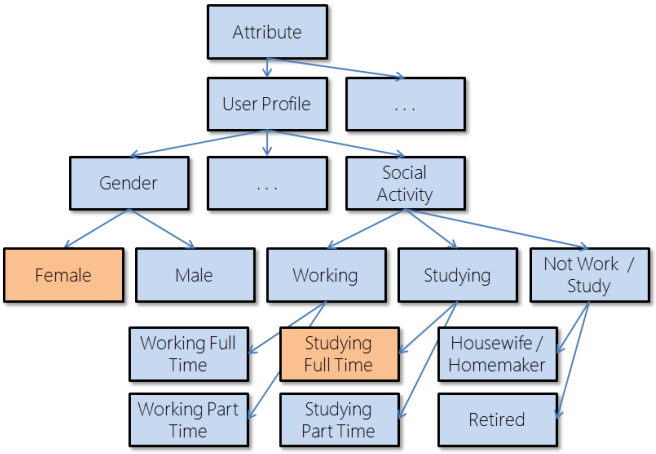


FIGURE 4.1: τ_A: User Attribute Taxonomy

instance). Similarly, items in $\mathcal{I}$ (applications in NOKIA and movies in MOVIELENS) and locations are organized into their respective taxonomies τ_I (Figures 4.2) and τ_L (Figure 4.5). The set of all taxonomies is referred to as $\mathcal{T}$. We do not aim to show all the taxonomies we built for our datasets, rather we illustrate some examples that will be used later in this section.

Figure 4.2 shows a subset of the taxonomy we built for NOKIA applications. Figure 4.3 zooms in and shows the sub-taxonomy for item `Desktop Communication`. Figure 4.5 shows a subset of the location taxonomy for MOVIELENS.

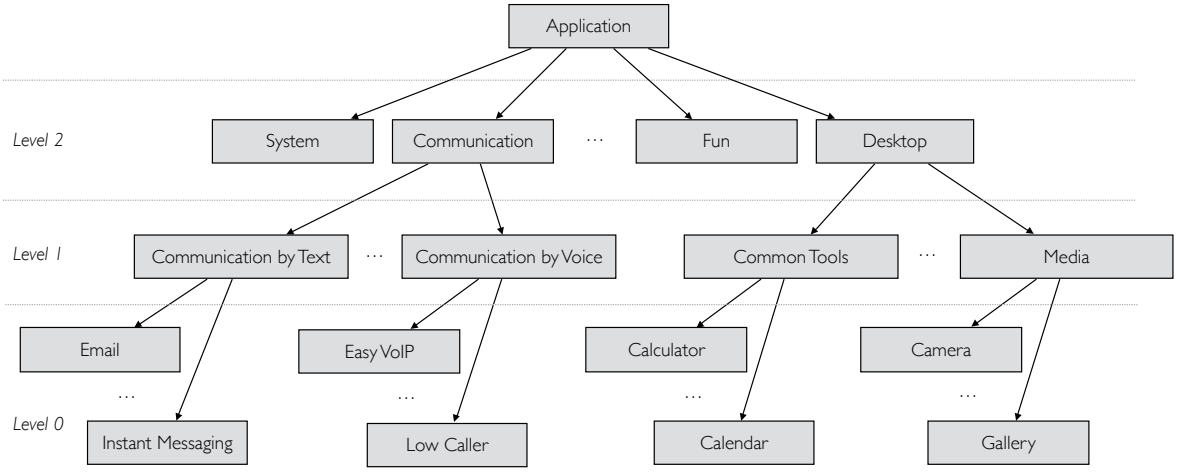


FIGURE 4.2: τ_I: Application Taxonomy

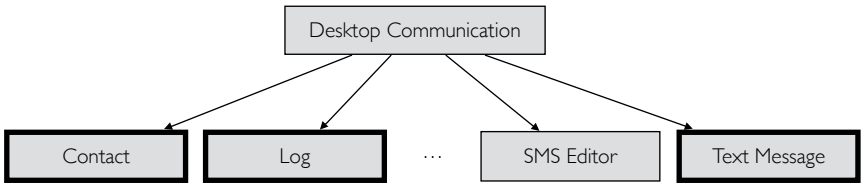


FIGURE 4.3: *Desktop Communication* Taxonomy

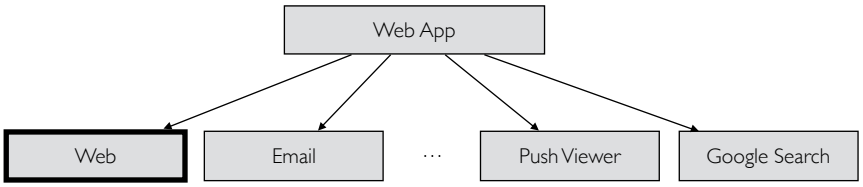


FIGURE 4.4: *Web App* Taxonomy

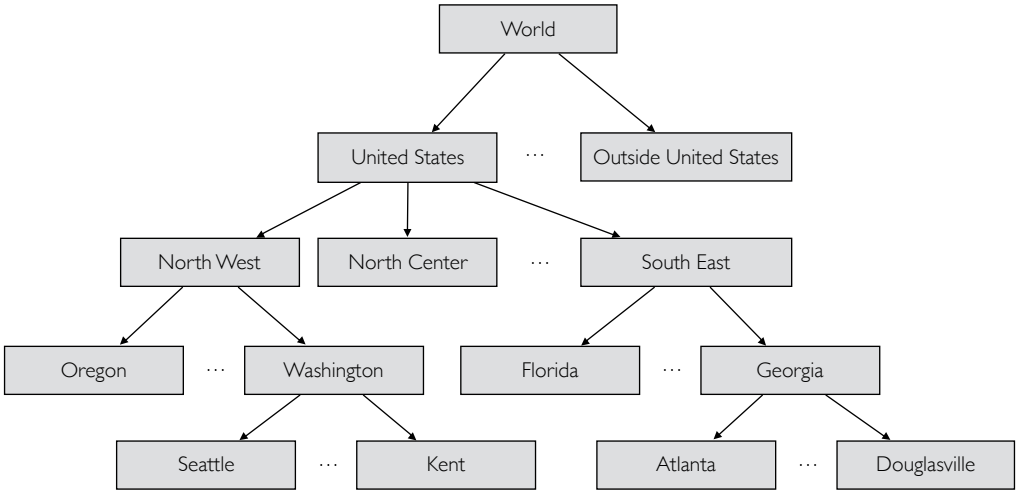


FIGURE 4.5: τ_L: Location Taxonomy

4.1.3 Abstraction Primitive

A user group discovery algorithm (e.g., our contribution in Chapter 3) may result in a very large space of user groups which is potentially exponential in the number of items. In order to enhance user group analysis, we propose to use the semantics provided in taxonomies to abstract items in a user group into their parent item in the taxonomy. The intuition behind abstraction is simple yet powerful. Our abstraction method is not merely syntactic and relies on a taxonomy-based usage measure and reflects a way of *approximating the interests of users.* This approximation could be applied to items, time of day or to location.

We define the *usage* of an item i for a set of users $V \subseteq U$, $usage(V, i) = |\{\langle u, i\rangle \in D \mid u \in V\}|$ as the number of times users in the set V used item i. Note that in NOKIA, a user may use an item more than once (e.g. browsing the web more than once on a smartphone) while in MOVIELENS, a user rates a movie only once. The usage of an item i with respect to a user group g, $usage(g, i)$ is the number of times the item i has been used by all users of g.

Definition 4.1 (Taxonomy-Based Usage)**.** Given a set of sibling items $i_1, i_2 \ldots i_n$ and their parent item $\hat{i}$ in the taxonomy τ_I, their taxonomy-based usage in a user group g, denoted $Gusage(g, \{i_1, i_2 \ldots i_n\}) = \frac{\Sigma_{i_j} usage(g, i_j)}{usage(g, \hat{i})}$, is the proportion of usage between sibling items and their parent in τ_I.

The intuition of taxonomy-based usage is that if most of the usage of a given item is that of some of its children in the taxonomy τ_I, those children could be replaced by their parent in all groups they appear in, thereby reducing the size of those groups.

Definition 4.2 (Valid User Group Abstraction)**.** Given an abstraction threshold ρ and a user group g whose label contains sibling items $i_1, i_2 \ldots i_n$ with parent $\hat{i}$ in τ_I, we say that a user group g^A is a *valid abstraction* of a user group g iff $Gusage(g, \{i_1, i_2 \ldots i_n\}) \geq \rho$ and $\forall i_j, i_j \notin g^A$ and $\hat{i} \in g^A$.

The abstraction primitive can be applied to a user group g recursively, looks for all items in g that can be abstracted resulting in a maximal abstraction of g as defined below.

Definition 4.3 (Maximal User Group Abstraction)**.** Given an abstraction threshold ρ, we say that a user group g^A is a *maximal abstraction* of a user group g iff g^A is a valid abstraction of g and $\nexists i_1, i_2 \ldots i_n \in l_{g^A}$ s.t. $Gusage(g, \{i_1, i_2 \ldots i_n\}) \geq \rho$ is satisfied.

Note that the way we define abstraction is based on the notion of *usage*. It may be difficult to apply this operator on datasets where no usage can be defined. Part of our future work is to consider other definitions of abstraction which are applicable on other types of datasets.

Let us now illustrate the definitions above on our datasets. In NOKIA, the user group $g_1 =$ [`studying full − time`, `female`, `FG_Thread`, `WLAN_Wizard`, `Calculator`, `Calendar`, `Bluetooth`, `Contacts`, `Log`, `Web`, `Text_message`, `Messaging`] has 4 members. Given an abstraction threshold of 50%, we obtain *a maximal abstraction* of g using the application taxonomy into $g_1^A =$ [`studying full − time`, `female`, `system`, `WLAN_Wizard`, `Calculator`, `Calendar`, `Desktop Communication`, `Web App`].

The pie charts A, B and C in Figure 4.6 show usages that enable a recursive abstraction of the user group g_1 into g_1^A. In Figure 4.6 A, we can see that 87.68% of usage

for item `Desktop Communication` is for its children items `Bluetooth`, `Contacts`, `Log`, `Text_Message` and `Messaging`. Pie charts B and C of Figure 4.6 contain two other usages, one showing that 50% usage of `System` items is for `FG_Thread` and another showing that 53.61% of usages of `Web App` items (parent), is for `Web` (child). Finally, in D and E of Figure 4.6, we show two examples of non-valid abstractions given a 50% threshold. We see that 9% usage of `Configuration` items, is for `WLAN_Wizard` and that 22% of usage of `Desktop Common` items, is for `Calculator` and `Calendar`. Thereby, none of those could be abstracted in group g_1.

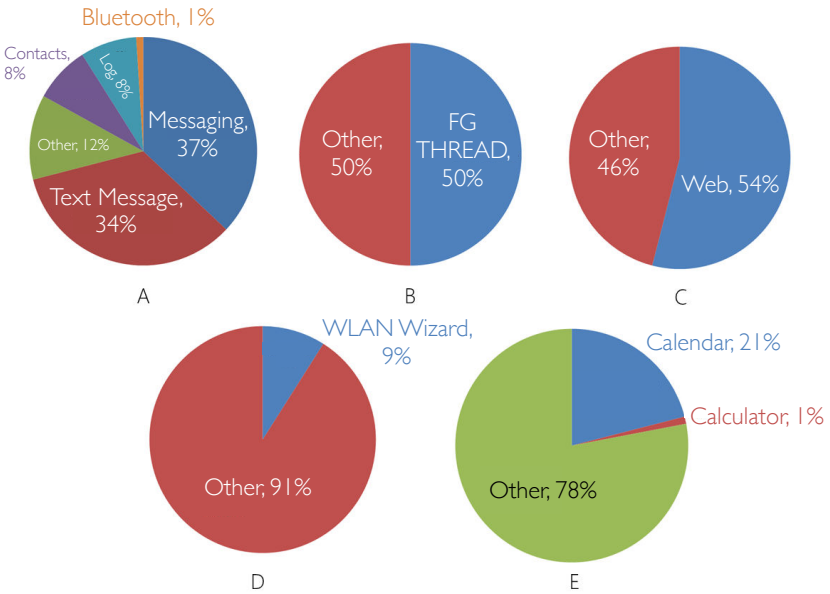


FIGURE 4.6: Item Usages for Abstraction

4.1.4 Experiments

The goal of this section is to evaluate the *abstraction* primitive on NOKIA (Section 2.6.1) and MOVIELENS (Section 2.6.3) datasets. We propose quantitative and qualitative evaluations. We discuss some interesting results for each evaluation.

To discover user groups for analysis, we can consider different group discovery tools. In Chapter 3, we propose an approach to discover groups by optimizing multi-objectives. However, for the sake of our experiments in this Chapter, we simply use LCM closed frequent itemset mining algorithm [UAUA03]. We pick LCM because it is an efficient approach to discover groups when there is no more than one objective. We tune LCM in a way to discover groups only if they have at least 7% of the dataset users. This resulted in 74,723 patterns for NOKIA and 50,299,230 patterns for MOVIELENS. Each

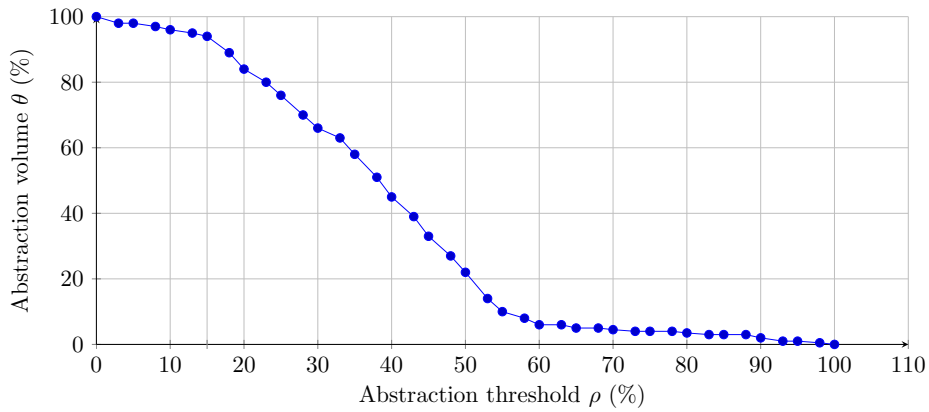


FIGURE 4.7: Abstraction Volume

pattern can be mapped to a user group where the pattern body becomes the group label and pattern support users become members of the group.

In order to evaluate the benefit of abstraction, we propose to quantify two measures, *abstraction volume* and *user group space reduction* as described below.

Abstraction Volume. As seen in Definition 4.2, the abstraction primitive only abstracts group of items if a condition is met. We want to evaluate how often this condition is met, depending on the abstraction threshold ρ chosen. We thus define an *abstraction volume* measure, which evaluates for each user group the ratio between the number of abstractions performed (given $\rho > 0$) and the maximal number of abstractions possible (case of $\rho = 0$).

Given N the number of occurred abstractions in a user group g and M the total number of classes of the taxonomy that have at least one of their child items in g, the abstraction volume of g denoted by θ is equal to (N / M * 100). We perform *abstraction volume* experiment on user groups from NOKIA. Group labels may include demographics or applications. We applied the *abstraction* method using different *abstraction* thresholds ρ varying from 0% to 100%. Figure 4.7 shows the result of this experiment. The evolution of *abstraction volume* can be categorized into three different periods by two cutting points $\rho_1 = 15\%$ and $\rho_2 = 60\%$.

Before ρ_1, we observe a very mild slope in the diagram and the *abstraction volume* decreases by only 10%. It shows that in NOKIA, low values of ρ lead to many abstractions. Choosing ρ in this range causes to have many abstractions with less attention to the usage of attributes. It will abstract nearly syntactically except for some extreme cases where usage is very low. Therefore, it is useful to select an abstraction threshold in this range only when data does not provide much usage information.

After ρ_2, we observe that the *abstraction volume* decreases drastically and it remains very close to zero. It means that in this range, the number of abstractions is very low. Thus, choosing the abstraction threshold in this range is useless for simplifying the analysis.

Between ρ_1 and ρ_2, we observe that the plot has a derivative close to -1. Thus changing ρ in these values gives a predictable reduction in the number of abstractions. We thus consider that this range of ρ threshold values is the most useful, and we will focus on it for most of the following experiments.

User Group Space Reduction. Applying abstraction on distinct user groups will sometimes result in the same abstracted group. For instance groups $g_1 = [\texttt{student}, \texttt{LA}]$ and $g_2 = [\texttt{engineer}, \texttt{AZ}]$ can both be abstracted to $g_1^A = g_2^A = [\texttt{socially active}, \texttt{west USA}]$. Hence, in addition to reducing group label size, a beneficial side effect of the abstraction primitive is to reduce the number of groups in the space. We want to evaluate the scale of this user group space reduction experimentally.

Given a minimum group size σ and an abstraction threshold ρ, the user group space reduction is equal to $1 - \frac{|\mathcal{G}^A|}{|\mathcal{G}|}$ where $\mathcal{G}^A$ is the set of all abstracted groups and $\mathcal{G}$ is the set of initial user groups. For Nokia, we generated the set of maximal abstracted user groups using 4 different values for σ i.e. 10, 25, 50 and 75% by varying ρ from 0% (i.e. syntactic abstraction) to 100%(i.e. no abstraction). The result is shown in Figure 4.8.

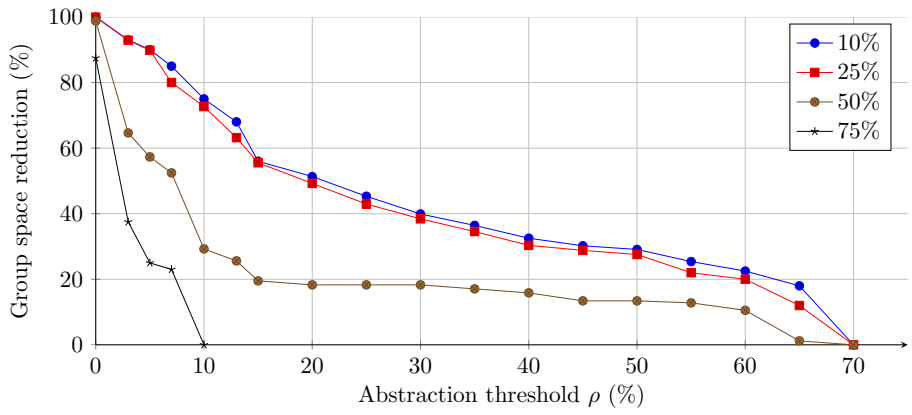


Figure 4.8: User Group Space Reduction for Nokia

As an example, using abstraction with $\sigma = 25\%$ and $\rho = 20\%$, the group space reduces to half of its initial size. The three periods mentioned in Figure 4.7 with the cutting points $\rho_1 = 15\%$ and $\rho_2 = 60\%$ are also visible in Figure 4.8, with a group space reduction between 20 and 30% in the most interesting range $[\rho_1, \rho_2]$. When fixing the abstraction threshold ρ, the lower σ, the higher the space reduction. However, for low σ values, the gain in reduction comes from lowering σ. This can be explained by the fact that

with lower σ, groups with longer labels are produced, many of them having mostly the same items (can differ by one item or two only). Thus abstraction is more likely to abstract those groups to the same user group. However the lower σ, the smaller the group size, which reduces the differences in usage values. These results show that the space reduction given by abstraction is not negligible, and can help reduce the burden on analysts.

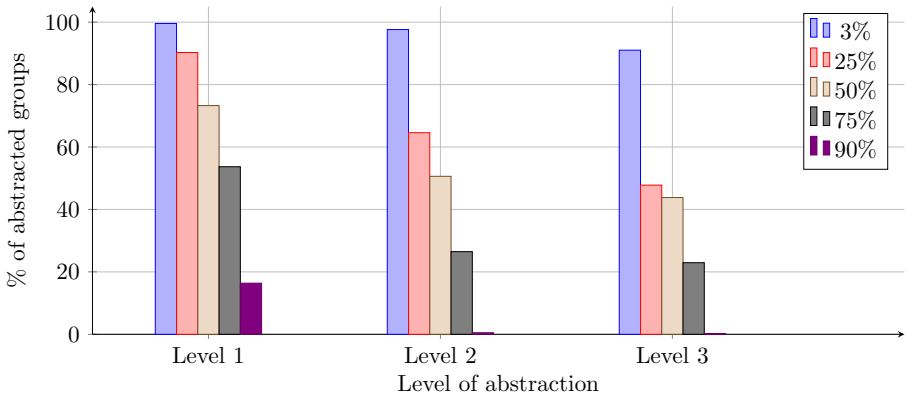


FIGURE 4.9: Abstraction per Level for NOKIA

To reach maximal abstraction, in the worst case an item may be abstracted at most 3 times, which is the depth of the NOKIA application taxonomy (as shown in Figure 4.2). It is interesting to see the influence of successive iterations of the abstraction primitive, and the distribution of abstracted items in the different levels of the application taxonomy. This result is presented in Figure 4.9. For each application of the abstraction primitive, and thus each level of the taxonomy as shown by Figure 4.2, the percentage of user groups that got abstracted to a class of the taxonomy of that level is shown. The bars correspond to different abstraction thresholds ρ, the group size threshold is set to $\sigma = 25\%$.

One can note that for too low abstraction thresholds ($\rho = 3\%$), 90% of user groups are abstracted to the top level of the taxonomy, which is the least informative: it confirms the poor interest of such low abstraction thresholds. Conversely excessively high abstraction thresholds ($\rho = 90\%$) lead to less that 20% of user groups abstracted on the lower level of the taxonomy, and nearby no higher level abstraction: this is not enough to help the analyst. On the other hand, abstraction thresholds between the bounds ρ_1 and ρ_2 lead to a reasonable percentage of user groups abstracted per level, with a decrease of more than 20% of user groups abstracted from level 1 to level 3. This indicates that the analyst will be presented with user groups containing a mixture of classes from the taxonomy, which is what is expected to help in the analysis.

4.2 Interactive User Group Analysis

In this section, we advocate an exploratory navigation of the space of groups and develop IUGA, an interactive user data analysis framework that provides analysts with the ability to incrementally discover user groups efficiently. Interactive analysis is a solution for common challenges of other user data analysis methods in the literature, as follows.

- **Pattern Mining** [AIS93] and **Subspace Clustering** [AGGR98]. The main limitation of those approaches is that, on real data, they can output millions of labeled groups inside which it is hard to know a priori which ones are of interest to the analyst.
- **Constraint-based Analysis**. A large body of work has been dedicated to providing knowledgeable analysts with the ability to specify *constraints* on groups of interest [BGMP03, BGKW02]. However, that is not adapted to exploratory scenarios where only limited knowledge is available on the dataset and the analyst does not necessarily know which subset of the data is of interest.
- **Expressing Queries on Raw Data**. SQL [CCD+13] (or any other query language) being declarative in nature, it is difficult to use it to express an exploration scenario which is iterative in nature: e.g., finding a set of groups with a SQL query then asking to find "related" groups.
- **Compression**. Other work proposed to reduce the output of a pattern mining algorithm to a *representative pattern set* of limited size (typically tens of labeled groups) [SVvL06]. However, resulting groups may be too coarse and miss groups that contain users of interest. Group granularity may be reduced with parameter relaxation but the number of resulting groups is bound to increase and quickly become hard to manage by the analyst.

Acknowledging the limitations of previous solutions, a recent line of work based on *interactive data mining* is being developed.[1] Such work is based on providing operations on the result of pattern mining in order to help the analyst navigate in the space of labeled groups and find groups and group members she is most interested in [BKT+13]. Existing work in this area is mainly a description of systems that have been designed to show the potential of interactive approaches. The approach we proposed in Section 4.1 is complementary to interactive analysis as *abstraction* can reduce the size of group space and makes it more manageable for interactive analysis.

[1] *http://poloclub.gatech.edu/idea2013/; http://poloclub.gatech.edu/idea2014/*

4.2.1 Motivating Examples

When faced with the daunting task of analyzing user data, an analyst may have different goals in mind. In this section, we focus on helping analysts find one or several users of interest by exploring relevant groups until she reaches her target users. More specifically, an analyst may want to *discover and gather several users* who may be scattered in different groups of interest. An analyst may also be interested in *finding a specific group member*, i.e., a user, for whom she remembers some but not all information. We illustrate these variants in the following two realistic examples.

Example 4.2 (Program Committee Formation)**.** *Martin is a PC Chair looking to build a program committee formed by geographically distributed male and female researchers with different seniority levels and different expertise. Figure 4.10 shows a simplified scenario for the* WebDB *2014*[2] *PC.*

As is often the case, PC chairs think of a set of potential members first. In this case, G. Fletcher, M. Theobald, S. Michel and X. Xiao are 4 initial members. Martin decides to use S. Michel and X. Xiao as seeds because they are junior and prolific (high frequency of annual publications). The action of keeping those 2 researchers is followed by an exploration which delivers 3 groups each of which containing one of S. Michel or X. Xiao (Step1). He then decides to keep the highlighted one: `prolific`, `high publications` *and publishing at* `SIGMOD` *(with which* WebDB *is associated.) The selected group contains 29 researchers out of which 4 geographically distributed (L. Popa, A. Doan, M. Benedikt, S. Amer-Yahia). In order to find more users related to that group, Martin decides to perform an action that removes the predicate* `high publi`*, because it has been investigated before.*

Step 2 explores the resulting group and outputs 3 diverse groups. Martin ignores the first group because he has already seen enough male candidates. He notices that the highlighted group contains 119 highly senior researchers who published in PVLDB (which is related to WebDB*). Therefore, he decides to get more information about that group by first adding the predicate* `data integration` *(the* WebDB *main theme in 2014) to specialize it and asking to split it into 3 groups. Step 3 shows the result of this exploitation operation. In particular, the group labeled with* `query processing`, `PVLDB` *and* `ICDE` *contains 26 senior researchers out of which 8 are of interest to the PC chairs (J. Wang, F. Bonchi, K. Chakrabarti, P. Fratenali, D. Barbosa, F. Naumann, Y. Velegrakis and X. Zhou). At this stage, Martin covered 80% of the* WebDB *PC.* □

Example 4.3 (Finding Someone with Little Information)**.** *Nicole liked a person she met at last night's party in Cole Valley, San Francisco, but she doesn't remember his name*

[2] *http://webdb2014.eecs.umich.edu*

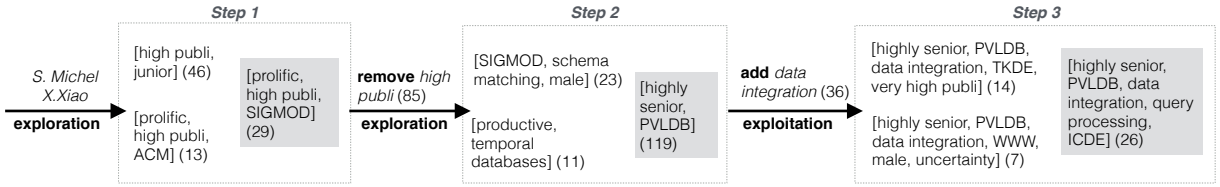


FIGURE 4.10: Discovering Several Users (WEBDB 2014 Program Committee)

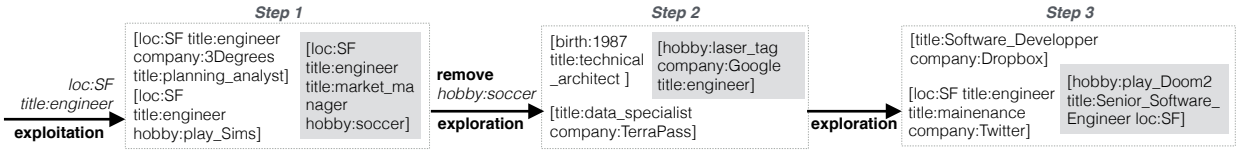


FIGURE 4.11: Finding a Specific User

and has lost his phone number. She only knows that he lived in the same neighborhood as Mike, the party host, and works as an engineer. Nicole asks Mike to have access to his social network which contains some of his friends' information: job title(s), company, location, birth year, and hobby(ies).

Mike is an avid Facebook user and has over 800 friends, most of which are computer engineers and live in San Francisco (SF). Thus no querying mechanism could lead Nicole directly to the person she is looking for. Also, advanced search tools (e.g., Facebook Graph Search[3]*) can only show similar people based on an input query. Nicole needs a tool for her navigational analysis of Mike's friends. She first uses the query* `loc:SF` *and* `title:engineer` *which returns 3 different user groups among Mike's friends that are highly related to her query (Step 1 in Figure 4.11). Nicole remembers that the person was talking about "website design", thus he shouldn't be working for* `3Degrees` *which is a renewable energy certificate provider. She also remembers that he mentioned he only likes "shooting" computer games thus he should not belong to the group labeled with* `Sims`*, a life simulation computer game. So she prefers to select the group labeled* `loc:SF`*,* `title: engineer`*,* `title:market manager` *and* `hobby: soccer` *as a seed. As Nicole doesn't remember any discussion about sports, she prefers to remove the attribute* `hobby:soccer` *to widen her navigation scope. The tool then finds 3 other groups in Step 2.*

Nicole is sure the person she met is at least 30 years old. This eliminates the first group with `birth:1987`*. Also, she herself works for* `TerraPass` *and knows the person does not work there. Also being a fan of shooting games, the group with* `hobby:laser tag` *and* `company:Google` *would be the best choice. Now, the tool returns 3 other user groups in step 3. The title "Senior Software Engineer" captures her attention as she remembers*

[3] *https://www.facebook.com/graphsearcher*

he said he was a manager. This group contains 3 users among whom Kevin Systrom, co-founder of Instagram and Mike's friend, is the one she was searching for. □

Our examples show that with simple group navigation operations, an analyst can navigate a good proportion of the space of users of interest. In this work, we formalize two such operations: *opExplore()* that finds groups *outside of the seed group*, and *opExploit()* that finds groups *inside.* The examples also show that before applying group navigation operations to a seed group, an analyst may want to transform that group using *actions* that remove or add specific users (in our example by modifying group labels).

4.2.2 Challenges and Contributions

In this section, we propose two important contributions to further advance the state of the art of interactive data mining with a focus on interactive user data analysis:

1. IUGA, a formalization of interactive user data analysis based on simple yet powerful *group navigation* primitives that enable an exploratory navigation of user groups.
2. A principled validation methodology. To the best of our knowledge, there exists no such methodology in the literature.

IUGA is *an optimization-based interactive framework where analysts are free to select any group of interest at each step and use it as a seed for further optimization.* It is based on 3 key principles:

- **P1: The analyst must be able to explore different groups and not be overwhelmed with analysis options.** The analysis process is broken into successive steps during which an analyst chooses a seed group, examines the users it contains, takes actions such as remove/add users, and continues with a group navigation operation. This principle is in line with *Enumeration* and *Insights* principles discussed in [NJ11] as *guided interaction* principles.
- **P2: Groups offered to the analyst must be of high quality.** The analysis process must help the analyst cover the space of groups of interest. We propose a "holistic" measure that finds k groups that are relevant to the seed group and are as diverse as possible.
- **P3: The train of thought of the analyst must not be lost.** Each interactive group navigation step must be fast. This principle is in line with *Responsiveness* principle discussed in [NJ11].

Devising an efficient multi-step group navigation approach is a challenge due to the large number of available groups. We hence propose to formulate group exploration and exploitation as optimization problems that find relevant and diverse groups at each step of the interaction. Both operations discover k diverse groups that have some relevance to the seed group, i.e., users in common. In the case of exploration, diversity aims to cover *as many different users as possible* outside of the seed group. For exploitation, diversity aims to *cover the seed group* while providing *distinct options* inside that group. We show that both problems are NP-complete by reductions from the Maximum Edge Subgraph Problem and the Maximum Coverage Problem respectively. We design GroupNavigation, a greedy algorithm to solve those problems.

Our last challenge is to devise a principled methodology that evaluates the need for an interactive multi-step group navigation approach. In particular, since our focus is to solve the multi-target and single-target search questions, we validated IUGA on two real use cases, namely, Program Committee (PC) formation by building a dataset from DBLP, and a single target scenario by building a synthetic dataset. Our results show that IUGA leads analysts to their target(s) in a small number of steps regardless of their starting points and their level of expertise.

4.2.3 Group Navigation Operations

We now introduce our formalization of group navigation operations and actions that form building blocks of IUGA. We first define the *exploration* operation that allows to *navigate in the group space in an outward way* starting from a set of users, it discovers groups containing new users.

Definition 4.4 (Group Exploration)**.** We define a function $gExplore(U, \mathcal{G}, \mu)$ that takes a set of users $U \subseteq \mathcal{U}$ and finds all groups in $\mathcal{G}$ that overlap with U with at least μ, a given threshold. More formally, $gExplore(U, \mathcal{G}, \mu) = \{(g, overlap(U, g)) | g \in \mathcal{G} \wedge g \neq U \wedge overlap(U, g) \geq \mu\}$ where $overlap(U, g) = \frac{|U \cap g|}{|U \cup g|}$ (i.e., Jaccard similarity coefficient).

The overlap condition provides a progressive exploration of the space, which helps the analyst build an understanding of the underlying data. Figure 4.10 illustrates several steps in IUGA used to build the WebDB 2014 PC. At the beginning, the analyst (here a PC chair) has two "seed members" in mind: S. Michel and X. Xiao. She then performs an exploration step over these two researchers, which produces (among others) three groups: a group of 46 researchers labeled as [`junior`, `high publi`], a group of 13 researchers labeled as [`prolific`, `high publi`, `ACM`], and a group of 29 researchers labeled as [`prolific`, `high publi`, `SIGMOD`]. All these groups lead to different research communities, the third one is most adapted to a database workshop.

When an interesting group is found, another important operation is *exploitation*, i.e., an operation that "drills down" into the most interesting subgroups contained in an input group.

Definition 4.5 (Group Exploitation)**.** We define a function $gExploit(U, \mathcal{G})$ that takes a set of users $U \subseteq \mathcal{U}$ and finds all groups in $\mathcal{G}$ that are contained in U. More formally, $gExploit(U, \mathcal{G}) = \{g \in \mathcal{G} | g \subseteq U\}$.

Figure 4.10 shows the result of applying *gExploit()* on the group labeled [`highly senior`, `PVLDB`, `data integration`] (Step 3). That results in 3 subgroups: one formed by 26 experts in query processing who publish in ICDE, the other by 14 prolific researchers who publish in TKDE, and the last one by 7 male researchers who work on uncertainty in databases. All 3 groups contain solely highly senior researchers who publish in PVLDB and work in the area of data integration. This example clearly illustrates that 32 out of 36 users of the selected group are covered.

4.2.4 Group Navigation Problem

The navigation of new groups relies on the two functions, *gExplore()* and *gExploit()* (Definitions 4.4 and 4.5 respectively), that are applied to an input group. In order to comply with principles **P1** and **P2** (Section 4.2.2), the number of groups returned to the analyst at each step must be limited, and output groups must exhibit diversity. Hence, we define the GroupNavigation Problem as follows: given a set of users $U \subseteq \mathcal{U}$, an overlap threshold μ, the GroupNavigation Problem returns k groups in $\mathcal{G}$, referred to as $\mathcal{G}_U$ and is expressed either as an exploration or an exploitation problem depending on an analyst's needs.

For exploration, we define $opExplore(U, \mathcal{G}, \mu, k)$ that must satisfy the following conditions:

1. $\mathcal{G}_U \subseteq gExplore(U, \mathcal{G}, \mu)$
2. $|\mathcal{G}_U| = k$
3. $diversity(\mathcal{G}_U)$ is maximized.

where $diversity(\mathcal{G}_U)$ is defined as follows: $diversity(\mathcal{G}_U) = \Sigma_{\{g_1, g_2\} \subseteq \mathcal{G}_U | g_1 \neq g_2}(1 - overlap(g_1, g_2))$.

In exploration, the aim is to start from an input set of users of interest U, and find k groups that have some relevance to U, using $gExplore(U, \mathcal{G}, \mu)$, and that have maximal diversity (as little overlap as possible with each other).

For exploitation, we define $opExploit(U, \mathcal{G}, k)$ that must satisfy the following conditions:

1. $\mathcal{G}_U \subseteq gExploit(U, \mathcal{G})$
2. $|\mathcal{G}_U| = k$
3. $divCoverage(\mathcal{G}_U)$ is maximized.

where $divCoverage(\mathcal{G}_U)$ is defined as follows:

$$divCoverage(\mathcal{G}_U) = diversity(\mathcal{G}_U) \times (|\bigcup_{g \in \mathcal{G}_U} g|/|U|) \tag{4.1}$$

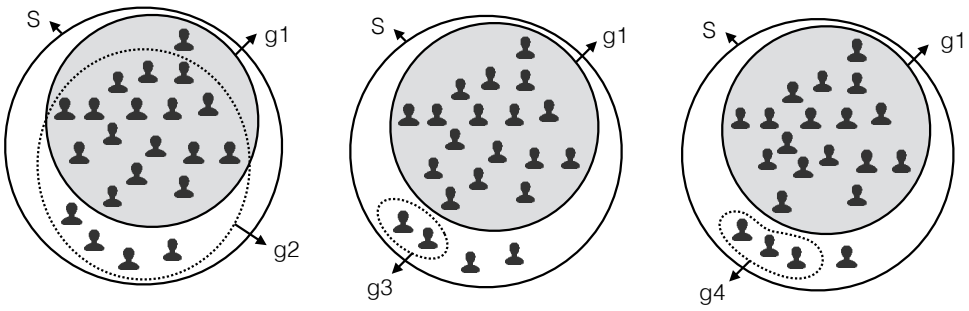


FIGURE 4.12: Illustrations of *diversity*() and *divCoverage*()

In exploitation, the aim is to find k groups that maximize coverage of the seed set U. Choosing k groups that have the highest coverage may potentially cause high overlap between those groups. Figure 4.12 left illustrates that, with $k = 2$ and two highly overlapping groups g_1 and g_2. Therefore, in the case of exploitation, we revisit the definition of diversity in a way that it prioritizes k diverse groups which cover as many users as possible in U. In [IMMM14], it is shown that there does not exist a unique optimal solution for both diversity maximization and coverage maximization. In other words, as shown in Section 3.6.1, diversity and coverage are not consistent objectives. Therefore, the diversity formula is modified by adding $(|\bigcup_{g \in G_U} g|/|U|)$ (see Equation 4.1). For example, in Figure 4.12, $diversity(\{g_1, g_3\}) = diversity(\{g_1, g_4\}) = 1$. Thus for *opExplore()*, both g_3 and g_4 can be chosen with g_1. However, for *opExploit()*, g_4 is preferred because $divCoverage(\{g_1, g_4\}) > divCoverage(\{g_1, g_3\})$.

4.2.5 Interactive User Group Analysis (IUGA)

IUGA builds on the GROUPNAVIGATION operations letting an analyst apply one of *opExplore()* or *opExploit()* on a set of users U and obtain k groups that constitute

further analysis options. Figure 4.13 illustrates that process. In order to comply with principle **P3** (Section 4.2.2), IUGA introduces a time limit parameter. Each step of IUGA solves the GroupNavigation Problem and returns the best possible k groups within a given time limit.

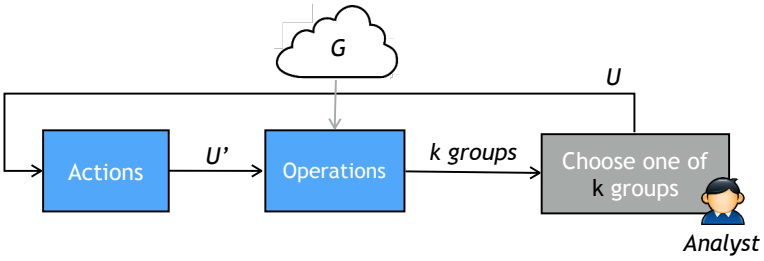


Figure 4.13: GroupNavigation within IUGA

In addition to *opExplore()* and *opExploit()*, the analyst is provided with a set of actions that could be performed on a chosen group to transform it according to his/her needs. The analyst examines the set of k groups at each step and chooses a new input group on which one of 3 actions could be performed: *actKeepUsers(U,U′)*, *actModifyLabel(U,l)*, and *actUndo()* to undo the previous step. Table 4.1 summarizes each action. The action *actKeepUsers(U,U′)* allows the analyst to mark which users to keep for the next step. *actModifyLabel(U,l)* is used to remove/add predicates or items to l_U, the label of U, resulting in new seed users.

Action	**Description**
actKeepUsers(U,U′)	keeps U' users in U
actModifyLabel(U,l)	replaces l_U with a new label l
actUndo()	back-tracks to the previous step

Table 4.1: IUGA Actions

The ability to *manipulate group membership* using actions on a input group, provides additional flexibility at each step. For example, as illustrated in Figure 4.10, in order to narrow down the set of 119 senior researchers who publish in PVLDB, the analyst adds the predicate `data integration` and obtains 36 researchers as the new input group to analyze. Also in Figure 4.11, the analyst removes the predicate `hobby:soccer` to direct the navigation towards her preferences.

4.2.6 Group Navigation Algorithm

Our GroupNavigation Problem requires to develop an efficient algorithm for dynamically finding and comparing user groups. We first discuss the complexity of our problem, then we describe our algorithm.

In Appendix B, we show that the GroupNavigation Problem is NP-complete by reductions from the Maximum Edge Subgraph Problem for *opExplore()* and from the Maximum Coverage Problem for *opExploit()*.

Our overall approach operates in two steps: an off-line process to produce initial user groups $\mathcal{G}$ and an online iterative process during which the analyst chooses a selected group for which k groups are discovered.

In the off-line process, a set of groups $\mathcal{G}$ are generated using a group discovery algorithm. It can be either multi-objective group discovery approach like the one we introduced in Chapter 3 or any single-objective group discovery approach like LCM [UKA04]. In this Chapter, we simply use LCM algorithm with a minimum support σ. Each frequent pattern corresponds to a user group, which has at least σ users. To feed LCM, we convert predicates in group labels into an item. For instance, the predicates $\langle \texttt{gender}, \texttt{male} \rangle$ and $\langle \texttt{gender}, \texttt{female} \rangle$ become two independent items. In addition, in order to speedup computing group relevance, we pre-compute an inverted index for each user group $g \in \mathcal{G}$ (as is commonly done in Web search.) Each index $\mathcal{L}^g$ stores all other groups in $\mathcal{G}$ in decreasing order of their overlap with g. Thanks to the parameter μ, we only partially materialize the indices. In the case of datasets we used in our experiment, we only materialize in average 10% of the whole index size.

Algorithm 4 summarizes a single greedy procedure that solves the GroupNavigation Problem, be it exploration or exploitation. It is called at each step of IUGA (as described in Figure 4.13). The algorithm admits as input a user group g, an operation *op* (*gExplore()* or *gExploit()*), an overlap threshold μ, k, and a time limit *tlimit*, and returns the best k groups denoted $\mathcal{G}_g$. Line 1 selects the most overlapping groups with g by simply retrieving the k highest ranking groups in $\mathcal{L}^g$. Function $getNext(\mathcal{L}^g)$ (Line 2) returns the next group g_{in} in $\mathcal{L}^g$ in sequential order. Lines 3 to 11 iterate over the inverted indices to determine if other groups should be considered to increase diversity while staying within the time limit and not violating the overlap threshold with the selected group. Since groups in $\mathcal{L}^g$ are sorted on decreasing overlap with g, the algorithm can safely stop as soon as the overlap condition is violated (or if the time limit is exceeded.)

The algorithm then looks for a candidate group $g_{out} \in \mathcal{G}_g$ to replace in order to increase diversity. The boolean function $betterDiv()$ (Line 5) checks if by replacing g_{out} by g_{in} in $\mathcal{G}_U$, the overall diversity of the new $\mathcal{G}_U$ increases. Obviously, the diversity of a group set $\mathcal{G}_k$ depends on the operation *op*.

The number of diversity improvement loops (lines 3 to 11) is $|\mathcal{L}^g|$ in the worst case. For each group $g_{in} \in \mathcal{G}_g$, we verify if the diversity score is improved by $betterDiv()$, hence $\mathcal{O}(k^2)$. The time complexity of the algorithm is then $\mathcal{O}(k^2.max_{g \in \mathcal{G}}|\mathcal{L}^g|)$.

Algorithm 4: GroupNavigation Algorithm

Input: $g \in \mathcal{G}$, op, μ, k, $tlimit$
Output: $\mathcal{G}_g$

```
1  G_g ← topk(L^g)
2  g_out ← getNext(L^g)
3  while (tlimit not exceeded ∧ overlap(g, g_out) ≥ μ) do
4      for g_in ∈ G_g do
5          if betterDiv(G_g, g_out, g_in, op) then
6              G_g ← replace(G_g, g_out, g_in)
7              break
8          end
9      end
10     g_out ← getNext(L^g)
11 end
12 return G_g
```

4.2.7 Experiments

Our experiments aim to validate the usability and efficiency of the interactive analysis and the quality of discovered groups at each step. All experiments are implemented in C on a 2.4GHz Intel Core i5 machine with an 8GB main memory, running OS X 10.9.2.

Summary of Results: In our first experiments, we observe that IUGA leads a knowledgable analyst to cover most PCs of major data management conferences in 12 steps (multi-target scenario). We also show that IUGA arrives sooner to target than its competitors (single-target scenario). Our second experiment is a user study of the quality of groups found by GroupNavigation in each step of IUGA. We find that most participants prefer IUGA to other options mainly because it helps them better understand the landscape of user groups.

Datasets. We use 2 real datasets for our experiments, i.e., DM-Authors (Section 2.6.4) and MovieLens (Section 2.6.1) plus one synthetic dataset with the same characteristics as MovieLens. DM-Authors and the synthetic dataset are used to validate the interactive analysis and MovieLens is used to validate group quality.

Table 4.2 summarizes the real datasets. It contains the number of user groups ($|\mathcal{G}|$) with at least σ users. For DM-Authors, σ is set to a very low value because smaller groups are of interest (e.g., 976 researchers are associated to `high publi` predicate, but only 28 researchers have published both in `WWW` and in `CIKM`). For MovieLens, we set σ to 7% of all users in order to obtain an adequate number of groups for our group quality validation. In both datasets, we set μ in a way that each group overlaps with 10% of groups in $\mathcal{G}$, hence pruning around 90% of inverted indices.

	MovieLens	DM-Authors
# users	6,040	4,907
# items	3,952	11,890
# attributes	4	4
# predicates	80	18
avg index size $\lvert\mathcal{L}\rvert$	485	79,127
# groups $\lvert\mathcal{G}\rvert$	4,918	790,017
σ	450	3
μ	0.1	0.01

Table 4.2: Real Datasets

Our synthetic dataset is generated by initializing a binary matrix of users and items M with *0* and then randomly adding some initial groups $\mathcal{G}_{initial}$, i.e., rectangles in M that are completely filled with *1*. For each group in $\mathcal{G}_{initial}$, we randomize its number of users (between 10 and 2000) and items (between 5 and 50 items). Then we mark $|\mathcal{G}_{target}|$ groups as target. We mine M with a minimum support threshold σ to obtain the group set $\mathcal{G}$. More details are presented in Table 4.3. All parameters are chosen in a way to mimic MovieLens with a larger number of users.

# users	10,000	*maxlenght*	50
# items	3000	*tlimit*	20 ms
$\lvert\mathcal{G}_{target}\rvert$	50	σ	10
$\lvert\mathcal{G}_{initial}\rvert$	500	μ	0.06

Table 4.3: Synthetic Dataset

4.2.7.1 Interactive Analysis Validation

We validate the effectiveness of our interactive analysis IUGA by addressing the two motivating scenarios described in Section 4.2.1. We first verify the utility of IUGA for building a program committee on DM-Authors (multi-target). Then, we describe a thorough validation of IUGA using our synthetic dataset that mimics MovieLens (single-target). In both cases, *tlimit* is set to 20ms.

Multi-Target Scenario. We study the effectiveness of IUGA with a realistic task of interactively building the PC of major conferences/workshops in data management. We first start with an experiment with many PCs then we delve into the details of WebDB 2014.

Figure 4.14 illustrates the results of interactively building the PCs of the following conferences in 2014: SIGMOD, VLDB[4], WebDB and CIKM[5]. For a given PC, we start

[4] *We only considered* Review Board *members for VLDB.*
[5] *We only considered the* knowledge management *track for CIKM.*

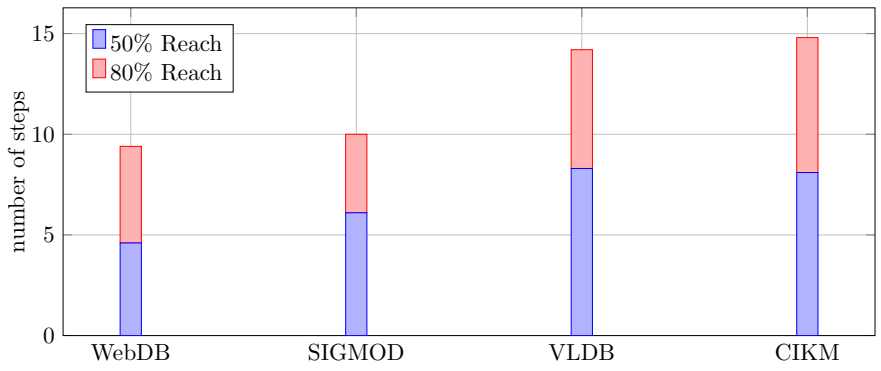


FIGURE 4.14: Number of Steps in IUGA for PC Selection

from 5% of its members and use IUGA to find the remaining ones. Target PC members should be discovered in user groups proposed in different steps of IUGA. The figure reports the number of steps to discover 50% and 80% of PC members as the average of 50 runs of IUGA for each PC.

We can observe that regardless of the analyst's expertise and the starting point of analysis, any PC selection can be done in 12.04 steps on average. CIKM's PC is the hardest to discover and WebDB's the easiest. Our conjecture is two key factors influence that: PC *size* and PC *diversity*. Indeed, the PCs of VLDB, CIKM and SIGMOD contain over 100 members while WebDB is smaller. This is why the former require a higher number of steps to cover 50% of their members (6.7, 6.5 and 5.9 steps respectively). In addition, the average pairwise Jaccard similarity[6] between PC members of CIKM is 7.35. This high diversity results in more steps to reach 80% of their PCs (8.3 and 8.1 steps respectively). SIGMOD has the least heterogeneous PC which leads to 4.8 steps to reach 80% of its PC. We also consider *disconnectedness*, i.e., the average number of PC member pairs that have no attribute in common. We observe that there exist a direct relationship between diversity and disconnectedness, i.e., CIKM conference has also the highest disconnectedness score, i.e., 5.72 versus 0.48 for WebDB for instance.

We now have a closer look to the WEBDB 2014 PC selection. Our detailed illustration will show the following facts: **F1**: how the analysis of user groups is more useful than analyzing individual users; **F2**: how our actions and operations (defined in Section 4.2.3) are *adequate* and *necessary* in interactive analysis; **F3**: which users are reachable or not, depending on their similarity to other users; **F4**: how *relevance* and *diversity* contribute to the analysis.

We characterize different scenarios based on the *analyst's expertise* and the analysis *starting points*. We assume 5 virtual scenarios summarized in Table 4.4. To measure the

[6] *Computed based on the profile of researchers.*

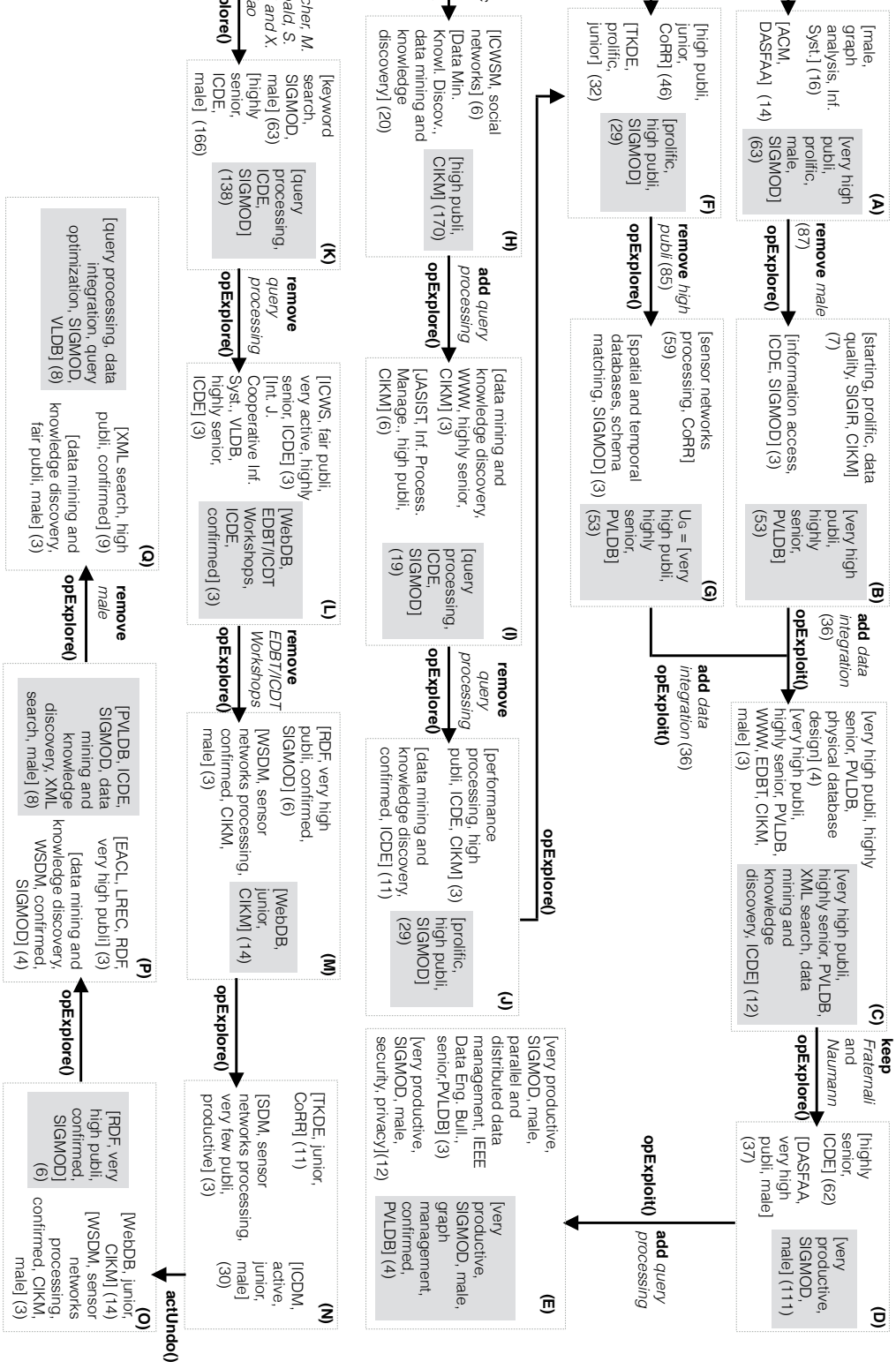


FIGURE 4.15: Scenarios KNOW13, KNOWIN and KNOWOUT and NONEXPERTIN

effect of expertise, we consider two cases where the analyst is *knowledgeable* about PC selection and the case where she is *a novice PC chair*. We also examine different starting points to build the PC: a subset of the final PC, a subset of the previous year's PC (i.e., WebDB 2013), or a set of arbitrary researchers outside the PC. Figure 4.14 shows that the average number of steps to cover 80% of the PC is 9.4. At each analysis step, $k = 3$. Figure 4.15 illustrates the results. Notation is simplified by replacing *actKeepUsers()* with **keep** and *actModifyLabel()* with **add/remove**.

Scenario	**Analyst**	**Starting Point**
KnowIn	Knowledgeable	Inside WebDB 2014
KnowOut	Knowledgeable	Outside WebDB
Know13	Knowledgeable	Inside WebDB 2013
NonExpertIn	Non-expert	Inside WebDB 2014
NonExpertOut	Non-expert	Outside WebDB

Table 4.4: Validation Scenarios

We observe that the *analyst's expertise* and the choice of *starting points* influence the process of building the PC. Overall, IUGA can reach the target in case of a knowledgable analyst, in 8 steps. We observe that a non-expert analyst is more oriented toward exploration to discover the unknown space, while a knowledgeable one uses both exploration and exploitation. We also observe that the relevance and diversity components alongside simple actions guide the analyst.

In the KnowIn scenario (Figure 4.15), a knowledgeable analyst starts with a subset of the final PC, i.e., G. Fletcher, M. Theobald, S. Michel, and X. Xiao, and selects the last two as a seed group (because they are prolific young researchers with a high number of publications.) Exploring this group results in 3 groups out of which the one labeled with `SIGMOD` (the conference that hosts WebDB) contains 4 researchers of interest (L. Popa, A. Doan, Benedikt and S. Amer-Yahia). This already shows the advantage of user group analysis (fact **F1**) where in one single step, 4 PC members are retrieved. The analyst then uses *actModifyLabel()* to replace the predicate `high publi` with `data integration` (i.e., the WebDB main theme in 2014) and decides to exploit the resulting group. In step D, the analyst keeps only P. Fraternali and F. Naumann among 12 group members using *actKeepUsers()* action. This action makes it easier to reach groups containing items like `SIGMOD` (P. Fraternali and F. Naumann have 9 and 6 SIGMOD publications respectively) and `ICDE` (e.g., F. Naumann has 14 ICDE publications). This shows the necessity of actions, confirming fact **F2**. Up to step E, the analyst is able to find 14 out of 15 PC members. The missing PC member is Jian Li. We compare Li's word cloud in Figure 4.16 containing all his publication title words, conferences and journals, with the cloud for all WebDB 2014 PC members. This shows that Li's research areas

differ significantly. This is an observation of the fact **F3** that shows the limitation of interactive analysis.

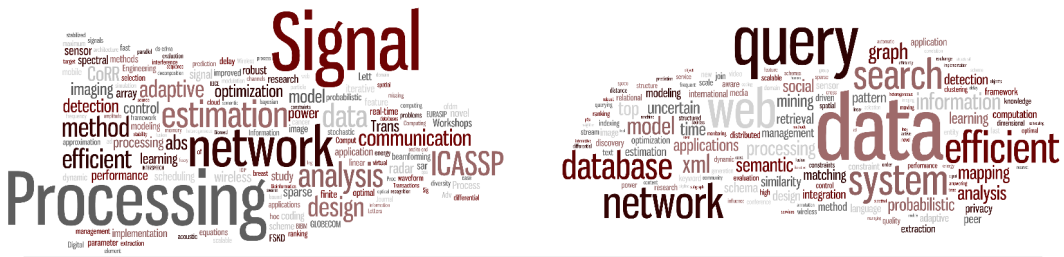


FIGURE 4.16: Word Frequency Cloud for Jian Li (left) and the Whole WEBDB 2014 PC (right)

In KNOWOUT (Figure 4.15), the knowledgeable analyst starts with J. Leskovec and A. Siebes, two researchers outside the final WEBDB PC. The *opExplore()* operation first finds k related groups that expand possible candidates. In step H, the analyst encounters the same group as in step A of scenario KNOWIN. This shows that in this case, a knowledgeable analyst only needs 2 more steps to reach relevant groups from a random departure point. Step H is also an illustration of fact **F4** and shows that all 3 returned groups are *relevant* and *diverse* leading the analyst to pick the group labeled with `SIGMOD`.

In KNOW13 (Figure 4.15), the analyst starts from a subset of the WEBDB 2013[7] PC which has 10 researchers in common with 2014. The analyst selects prolific researchers with a very high number of publications to begin with (E. Rahm, G. Li and B. Zhao), and who do not belong to the 2014 PC. The analyst applies *opExplore()* which results in 3 groups: one labeled with `SIGMOD` and two others with `Inf. Syst.` (Journal of Knowledge and Information Systems) and `DASFAA` respectively. Step A is an illustration of the fact **F4** and shows that all 3 returned groups are *relevant* and *diverse* which leads the analyst to pick the group labeled with `SIGMOD`. At this stage, the analyst applies the action *actModifyLabel()* to the group and removes the `male` predicate to achieve gender balance. The analyst then adds `data integration` to specialize the group. Although in this scenario, we started from outside of WEBDB 2014 PC members (like the KNOWOUT scenario), but as all WEBDB 2013 and 2014 PC members are related to WEBDB, we could find a path towards the 2014 PC from the 2013 PC.

In NONEXPERTIN (Figure 4.15), we consider a junior PC. The aim is to observe the effect of *analyst expertise* by comparing this scenario with KNOWIN. We will see that in the case of poor expertise, the analysis is done mostly by exploration. We also observe a tendency to manipulate group labels rather than group membership (specific researchers in groups). The analyst starts with 4 given researchers and applies *opExplore()* to

[7] *http://webdb2013.lille.inria.fr*

expand the analysis scope. The analyst finds a group of 138 researchers labeled with `query processing`, `SIGMOD` and `ICDE`, and decides to expand that group by removing `query processing`. The analyst then navigates up to step L, where she does not find any helpful group. Thus she commands an *actUndo()* action. Up to step O, she finds 12 out of 15 PC members. Compared to KnowIn, the number of useless steps (without any PC member discovery) has increased.

Finally, in NonExpertOut, we examine the case where a non-expert analyst starts with researchers outside the final PC. In this scenario, the analyst may abandon a path and start again with different groups. She may need to repeat that until a satisfying starting point is found. In our experiment, a non-expert analyst jumps 4 times to land at step K, the first step of NonExpertIn.

Single-Target Scenario. The previous experiment showed how effective our interactive analysis is in building a program committee by "gathering" members of interest along the way during the analysis. In this experiment, we focus on validating the effectiveness of IUGA in finding a single target as described in Example 4.3 in Section 4.2.1. We use the synthetic dataset that was generated to scale up MovieLens (Section 2.6.1). Our dataset is a matrix M with 3×10^7 cells, where squares with at least 10 users (i.e., minimum support σ) filled with 1, represent user groups. We propose a measure called Average Target Arrival (ATA), i.e., the average number of iterations to reach a group containing a target group starting from a non-target group. We randomly mark 50 groups as targets and compute ATA for those groups (we refer to target groups as $\mathcal{G}_{target}$). We compare IUGA with two different baselines: *unsupervised* and *interactive*. Briefly, if m_1 and m_2 are two different methods and ATA(m_1) < ATA(m_2), then m_1 is considered faster. Note that the concept of ATA differs significantly from finding the shortest path. For the latter, we assume the starting and target points are known, while this is not the case in the interactive analysis.

Algorithm 5: Experimental ATA Protocol *ATAalg*

Input: $\mathcal{G}$, $\mathcal{G}_{target}$, k, μ, g, *len*, *method*, *maxlen*
Output: length of navigation path

```
1 if g ∈ G_target then return len
2 if len > maxlen then return -1 // lost path
3 G_k ← choose(opExplore(g, G, μ, k, method), opExploit(g, G, k, method))
4 foreach g ∈ G_k do
5 |  ATAalg(G, G_target, k, μ, g, len + 1, method, maxlen)
6 end
```

Algorithm 5 illustrates how ATA is computed. We designed 200 different sessions each of which has a different synthetic dataset and is repeated 100 times for each method. Hence, we compute 20,000 ATA values for each one of the interactive analysis methods

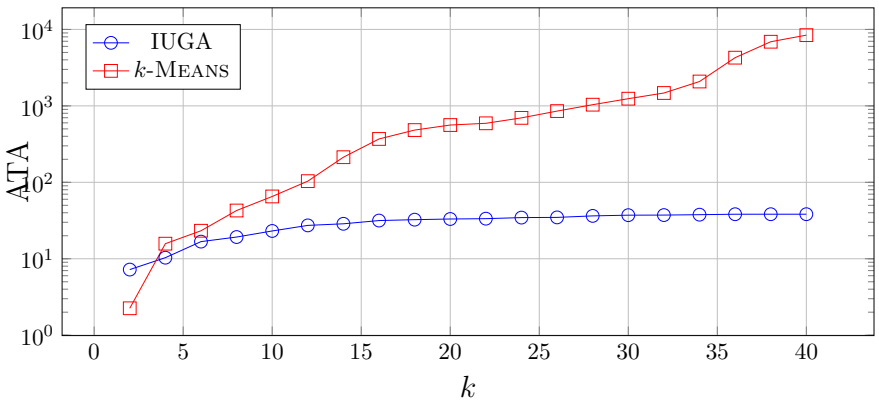


FIGURE 4.17: IUGA Comparison with Clustering Algorithm

we defined. For a random group g_{rnd}, k groups are returned using *method*, and a random choice between *opExplore()* and *opExploit()* (the algorithm starts always with an *opExplore()*). *Each of* the k groups becomes the new seed. This depth-first recursive call terminates either when one group in $\mathcal{G}_{target}$ is found or when a path of length 50 has been built (*maxlength* in Table 4.3). These recursive calls form paths inside the group space. A path is called *valid* if its last group belongs to $\mathcal{G}_{target}$. The ATA is computed as the average of valid path lengths for each method.

The first piece of experiments compares IUGA to a variant of k-MEANS with *Jaccard* as the distance measure. At each step, both IUGA and k-MEANS return k groups while respecting *timelimit*. Any number of iterations is allowed for k-MEANS within *timelimit*. We then report ATA for both methods. For k-MEANS, we randomly add/remove attributes at each step i so that a new set of k clusters is obtained in step $i+1$. Presence or absence of an attribute changes the clusters' membership, as the *Jaccard* distance between users varies. For instance, adding a specific value of `age` reduces the distance between two users having the same age.

Figure 4.17 illustrates ATAs for IUGA and k-MEANS in log scale. We vary k from 2 to 40 and observe how ATA for both algorithms evolves. While k-MEANS performs better for very small values of k, IUGA outperforms it by two orders of magnitude for higher values of k. When k is very small, clusters are huge. Thus most of the time, there exists a cluster that contains all users of a target group. For larger values of k, more clusters with smaller size are generated and more steps are needed to finally reach the target. We can conclude that the superiority of IUGA over unsupervised methods comes from the use of diversity at each step in order to cover as many users as possible.

We now compare IUGA with some interactive analysis baselines: DIVRAND, RANDOM, EXHAUSTIVE and ILP. At each step, DIVRAND randomly generates as many sets of k groups as possible within *tlimit* and returns the one with the highest diversity. RANDOM

navigates randomly in the space of groups and does not respect *tlimit*. EXHAUSTIVE generates all possible k among n groups in $\mathcal{G}$, i.e., $C\binom{n}{r}$ and chooses the one with the highest diversity. ILP returns k groups with maximal diversity using an integer linear programming formulation (using CHOCO 3.0 solver[8]). Table 4.5 summarizes the methods.

	No *tlimit*	**Within** *tlimit*
Sub-optimal Solution	RANDOM	IUGA DIVRAND
Optimal Solution	EXHAUSTIVE ILP	

TABLE 4.5: Interactive Analysis Methods

Table 4.6 illustrates ATA and execution times for IUGA and optimal methods. Since both EXHAUSTIVE and ILP generate optimal paths, their ATA are very close. However, their execution times are different. This experiment shows that IUGA is faster than EXHAUSTIVE and ILP (3.49 minutes faster than ILP) while maintaining a comparable ATA.

	EXHAUSTIVE	ILP	IUGA
ATA	9.90	9.91	10.13
Time (sec)	862.47	213.12	3.35

TABLE 4.6: IUGA vs Optimal Methods

Figure 4.18 illustrates ATA for all heuristic-based methods: IUGA, DIVRAND and RANDOM by varying k from 2 to 40, and varying # groups from 50,000 to 1,000,000. Optimal methods (EXHAUSTIVE and ILP) do not terminate for this experiment. In general, we observe that IUGA has much lower ATA for $k \leq 16$ and $k \geq 30$. This simply shows that considering relevance and diversity at each step reduces ATA by an average of 15.91 steps.

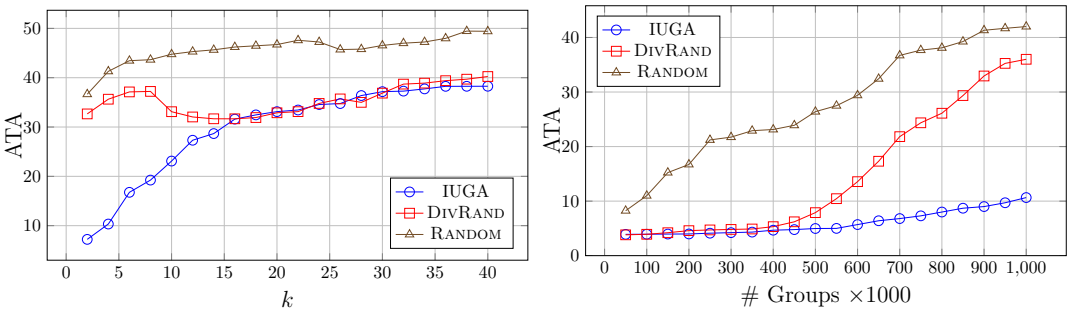


FIGURE 4.18: ATA as a Function of k and # Groups

[8] *http://choco-solver.org/?q=Choco3*

For $k \in [16, 30]$, DivRand and IUGA have close results. This shows that although the *relevance* component, i.e., the difference between DivRand and IUGA, is shown to be very useful in general, it is less effective for large values of k. In [BKT+13, WL12], it is shown that in a context with too many options and no hint for further navigation, *long* jumps are preferred to *short* jumps. In our case, *relevance* tends to favor *short* directed jumps in the space of groups while DivRand does not. This is why when few options are available, IUGA performs better and DivRand performs as well as IUGA for larger values of k. In another research[9], it has been shown that people faced with numerous choices, whether good or bad, find it difficult to stay focused on a task. Choosing a small value of k is hence better both for performance and effectiveness.

We observe that increasing the number of groups has a huge effect on DivRand. When the number of groups increases, the target groups are more likely to be diverse. Thus, precision (ratio of valid paths over all navigated paths) decreases for all methods, while thanks to relevance, the decrease is negligible for IUGA.

4.2.7.2 Quality of Group Navigation

We focus on a single step of IUGA and evaluate the quality of k obtained groups at each iteration of GroupNavigation. We ask participants in a user study to compare the top 5 groups obtained by GroupNavigation with some competitive methods. For this experiment, we setup a questionnaire which was answered anonymously by 35 students.

For this experiment, we use MovieLens (Section 2.6.1) because students are familiar with the content of this datasets. The evaluation consists of a *set evaluation* where results of competitive methods are evaluated together, and an *individual evaluation* where each set of top 5 groups is evaluated separately.

Summary of Results. We observe that participants prefer GroupNavigation because they believe it helps them to understand better the user group under study and also the relation between users and their activities.

In the set evaluation, we compare the top 5 groups obtained by GroupNavigation with others returned by 4 baselines: *Largest Groups*, *Most Overlapping*, *Least Overlapping* and *Most Concise* (groups which have the shortest description). Those baselines were designed using interestingness measures used in pattern evaluation [GH06]. Also, students were instructed to select a justification for their preferred method: *"it helps better understand who does what"*, *"it helps discover new users"* or *"it helps discover new group labels"*.

[9] *Too many choices (good or bad) can be mentally exhausting: http://phys.org/news127404469.html*

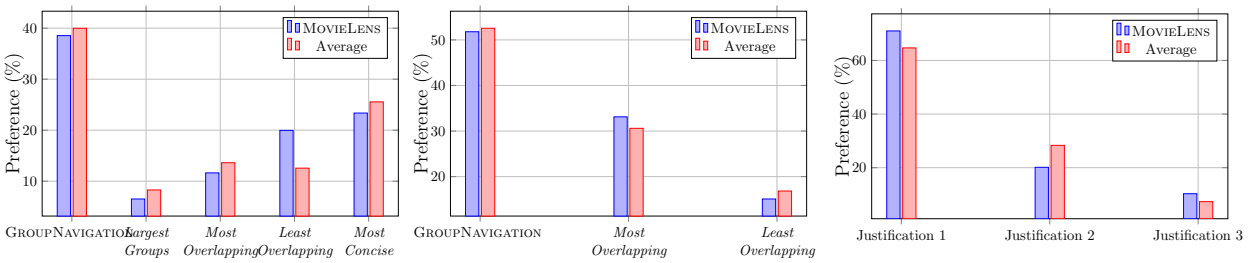


FIGURE 4.19: User Preference Results for Set Evaluation (left) and Individual Evaluation (middle) and Justifications (right)

The left chart in Figure 4.19 illustrates the average percentages of responses for each analysis option. In this part, participants have mostly preferred the results of GroupNavigation followed by *Most Concise* groups. Also, they have mostly (52.75%) justified their responses as *it helps better understand who does what*. The choice of *Most Concise* groups reveals that people often understand better groups with shorter descriptions.

In the individual evaluation, we compare each of top 5 groups of GroupNavigation with *Most Overlapping* and *Least Overlapping* groups, i.e. two extremes. The participant also chooses one of the following justifications: *Justification 1*: *understand the selected group*, *Justification 2*: *discover new items/users*, or *Justification 3*: *understand the whole data*.

The middle chart in Figure 4.19 illustrates the average percentages of responses for each group. Results show that on average, participants have preferred groups of GroupNavigation to other groups. Another observation is that participants have preferred larger groups to smaller ones.

The right chart in Figure 4.19 shows the aggregated justifications in the individual evaluation. Whenever our solution was selected, *Justification 1* was chosen on average by 56% of participants, followed by *Justification 2* (34.22%). In general, 63% of participants mentioned that their preferred group helps better understand the selected group (*Justification 1*), followed by 28% who believe the preferred group helps discover new users (*Justification 2*).

4.2.8 Graphical User Interface

In this section, we introduce the graphical user interface we designed on top of IUGA. This interface is built in collaboration with Miratul Mufida in form of a Master's internship. The interface is built using D3 Java Script library.[10]

[10] *http://d3js.org*

To build this user interface, we had a look at existing approaches in the literature for data visualization [HBO10]. Each visualization tool is appropriate for a set of data analysis tasks. Then we need to identify the characteristics of each tool as well as the characteristics of user groups to verify which one matches better the notion of user group. We use MovieLens dataset to explain the functionality of our user interface.

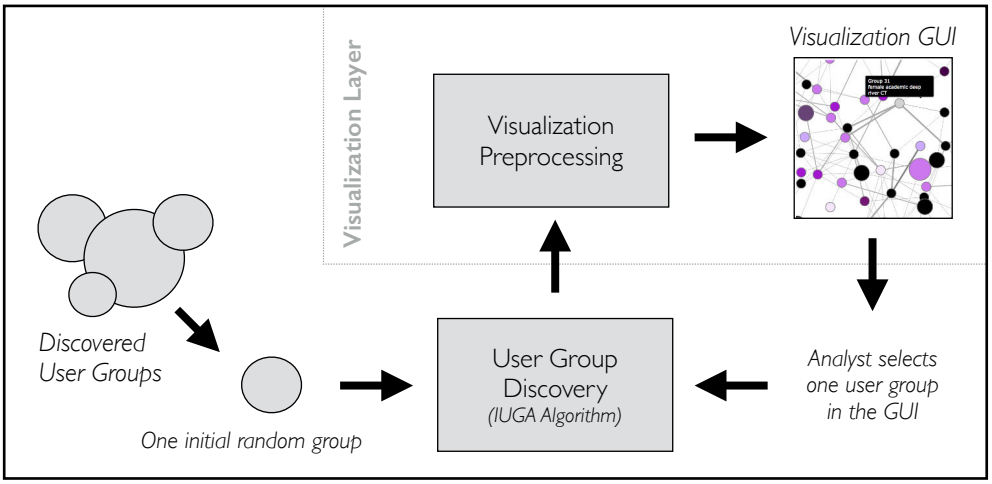


Figure 4.20: GUI Layer on top of IUGA

A group label with maximum amount of information may contain the following attributes: `gender`, `occupation`, `age`, `city` and `state`. We also map a unique integer value to each user group as its identifier. Groups interact with each other by sharing (or not) common users or attributes. The graphical layer above IUGA, should have different visual variables (color, size, etc.) [Ber83] to visualize above attributes. We experimentally verified different mappings between attributes and visual variables and we finally reached the following conventions.

- **Shape.** Groups are shown by circles.
- **Tooltip.** Moving the mouse over each group shows its label.
- **Color Intensity** represents `age`, where groups with younger members gets lighter color and groups with older members gets darker. We don't use different colors but different intensities. The reason is that the human mind is not capable of memorizing different colors mapped to items. But intensity can be easily translated to age levels in human mind.
- **Size** represents the number of members in a group. Human eye is not good at detecting small changes in size. Thus group sizes are discretized with *fixed-length* to 3 different categories, small, medium and big.

- **Solid line** between a pair of groups shows that there exists at least one user in common between these two groups. The thickness of this line represents the amount of common users. The amount of common users is also discretized with fixed-length to 3 categories: few, medium, huge.
- **Dashed line** between a pair of groups shows that there exists at least on attribute in common between their labels. The thickness of this line represents the amount of common attributes. This visual variable is also discretized with fixed-length to 3 categories.
- **Layout** defines how the set of k groups are positioned in the 2D space of the screen. There exists different graph layouts in the literature, e.g., divergent forces, multiple foci, force-directed tree, force-directed symbols, etc. For our GUI, we chose force-directed layout: to place nodes and edges in more or less equal distance and with as few crossing edges as possible. This layout assigns forces among the set of nodes and edges, based on the volume of their common users and attributes and uses these forces to simulate the motion nodes and edges [DBETT94].

Figure 4.20 illustrates how a GUI interacts with IUGA. Layers (GUI and IUGA) communicates with each other via the group identifier: an identifier is given to GUI for visualization and the choice of the analyst in GUI will also be reported as an identifier to IUGA to generate the result of the next step. Figure 4.21 illustrates one screenshot of our GUI.

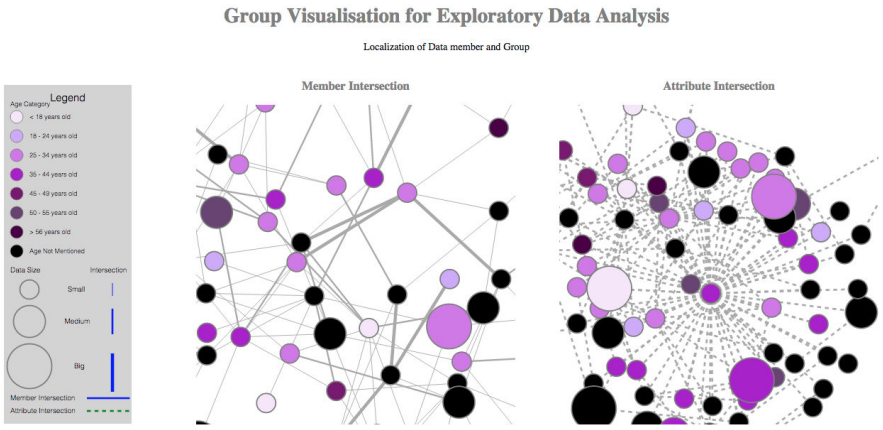


FIGURE 4.21: Graphical User Interface

4.3 Conclusion

In this chapter, we introduced two methods for user group analysis, i.e., abstraction and interactive analysis.

In Section 4.1, we proposed the abstraction analysis primitive to summarize the space of user groups. Our evaluation on two real datasets showed that abstraction reduces the size of the user group space and produces more readable groups.

In Section 4.2, we introduced IUGA, the first interactive user data analysis framework that is based on a simple and intuitive optimization formulation: the GroupNavigation Problem that finds the k most diverse and relevant user groups to an input group. IUGA relies on two group navigation operations: exploration and exploitation. We proved the hardness of our problem and devised greedy algorithms to help analysts navigate in the space of groups and reach one or several target users. Our extensive experiments on real and synthetic datasets showed the utility of relevance and diversity in group navigation and in finding users of interest in different scenarios.

Chapter 5

User Group Recommendation

We presented in Chapter 3 how to discover user group. We then discussed in Chapter 4 how to effectively analyze the space of discovered groups. Those two components constitute the first steps of our framework for user group management. When the analysis step is done, it is expected that the analyst has found one or more user groups of interest. The final question is *how to use these user groups*. As shown in Figure 1.2, the third and final step in our framework is to use groups in action.

User groups can be used in many different applications: population studies, online marketing or recommendations. In this thesis, we investigate one of these applications, i.e., *user group recommendation*. In the literature, there has been lots of efforts for individual user recommendation. Recently a new track proposed to find recommendations that interest a group of users together. In this chapter, we introduce an approach to group recommendation using temporal interactions between users. Parts of this chapter were published in [AORS15].

5.1 Challenges and Contributions

Group recommendation refers to finding the best items that a set of users will appreciate together. It is an active research area as exemplified by numerous publications [AYRC+09, BCC+09, JS07, OCKR01a, RTAY+14]. The main focus of existing work in group recommendation is the design of appropriate consensus functions that aggregate individual group members' preferences to reflect the overall group's preference for each item. A variety of consensus functions have been used ranging from majority voting to least misery [AYRC+09]. We are interested in exploring *how affinity between group members* and *its evolution over time* affect group recommendations. To the best

of our knowledge, our work is the first to study *affinity and its evolution over time* in combination with existing group consensus functions.

The premise of this work relies on a simple conjecture that is, a user appreciates recommendations differently in the company of different people and at different times. When with girlfriends, a female user may want to watch a romantic movie that she may not want to watch with men. When with her parents, she may prefer to go to a nice Italian restaurant while she would prefer a burger joint with her kids. In addition, her appreciation of an item with the same group of people may change over time depending on how their connection and shared interests evolve. In other terms, *the affinity of a user with other group members* should be captured in how that user appreciates an item.

Previous studies on single user recommendation have shown that *contextual dimensions* such as a user's mood and company or time and place, may affect her preferences [ASST05, AT05]. Indeed, according to behavioral research studies [BJP91, KY89, LO79], consumers use different decision-making strategies and favor different brands and products depending on their context. Such observations can be incorporated in different ways into single user recommendations. In [ASST05], a multidimensional recommendation model is developed to account for contextual information into a user's recommendation, one of which could be her affinity with other users. However, to the best of our knowledge, multidimensional recommendation has not been applied to group recommendation. In group recommendation, we conjecture that each user will have a *relative preference for an item* depending on her affinity with other group members.

Formalizing the semantics of relative preference raises four following challenges:

Challenge 1: How to account for user affinities in the definition of relative preference. It is challenging to integrate the evolution of affinities between users over time. For example, interns at a research lab may subscribe to a Facebook group during their internship. When the internship period is over, the group becomes an alumni of the research lab and affinities between its members will likely change. Therefore, if events such as workshops or conferences are to be recommended to the alumni group in the future, affinities between its members should be accounted for, in order to decide which subgroup would be interested in which event.

While numerous recent studies have shown the importance of accounting for time in recommender systems [DL05, KDK11, Kor10, XCH$^+$10], they have focused on user-item preferences and single-user recommendations. In this work, we propose two dynamic models to capture temporal affinities: a *discrete* model where time is discretized over a set of time periods and affinities computed for each sub-period, and a *continuous* model where time is represented as an exponential function that positively or negatively affects

affinity over time. Both models have a static component that denotes how close two users are in a time-independent fashion and a dynamic component that captures the drift that the affinity of a user-pair exhibits compared to the overall user population. Finally, while the discrete model is an approximation of the continuous one, they are both used to capture increasing and decreasing affinities.

Challenge 2: How to combine relative preference with popular group recommendation consensus functions [JS07]. Clearly, combining user-item preferences of group members independently of each other to produce group recommendations is not enough to capture the impact of affinities on those recommendations. In other terms, applying the well-known group consensus functions such as aggregated voting, average preferences or least misery on individual group members' preferences, does not capture a scenario where the same user appreciates the same item differently in different groups. Therefore, we propose a two-step approach to address the second challenge. First, we modify individual user-item preferences on-the-fly to account for affinities and then we apply a group consensus function over the modified preferences. This approach has the benefit of dissociating recommendation computation from affinity computation and therefore being able to use relative preferences with any group recommendation consensus semantics.

Challenge 3: How to assess the quality of group recommendations. To address this challenge, we build a Facebook application and generate movie recommendations using MovieLens dataset[1]. We leverage friendship and common page-likes to compute affinities and run an extensive set of experiments varying *group size*, *cohesiveness* (rating similarity between group members) and *affinity* between group members.

Challenge 4: How to efficiently compute affinity-aware recommendations on-the-fly for ad-hoc groups. To address this challenge, we develop `GRECA`, an algorithm that non-trivially adapts the family of threshold algorithms [FLN01], to account for affinities between user pairs that evolve over time. `GRECA` leverages index structures that are extremely efficient with updates for maintaining time-variant affinities, and are used to efficiently produce the top-k recommended itemset for a group. In fact, as affinity between users evolves over time, `GRECA` does not need to recalculate any of the previously calculated affinities and just augments the index to account for the latest affinities. In addition to being instance optimal, the key novelty of `GRECA` is the use of a new buffer condition for termination, which constitutes a clear departure from traditional top-k style algorithms [FLN01]. This condition simply implies that just by examining the items in the buffer, `GRECA` can terminate with the guarantee to have found the correct top-k itemset.

[1] *http://movielens.umn.edu/*

In summary, in this chapter, we discuss following contributions.

- We motivate the need to account for user affinities between group members when computing recommendations and propose to capture affinities in *the relative preference* of individual group members for each item. Relative preference modifies a user-item preference with the user's affinity with other group members.
- Since affinities may evolve over time, we propose two models, discrete and continuous, to represent affinity drift of two users over time. This dynamic component is combined with a static component, that captures how close two users are in a time-independent fashion, in order to form temporal affinities.
- We extend group recommendation semantics, i.e., average preferences, least misery and pairwise disagreement, to include temporal affinities and design `GRECA`, an efficient algorithm that computes recommendations on-the-fly for ad-hoc groups. `GRECA` uses a new early termination condition to efficiently produce the top-k itemset for a group.
- We run extensive experiments using Facebook-72 and MovieLens datasets and examine the impact of our temporal affinity model on group recommendation quality and efficiency.

5.2 Data Model and Preliminaries

The underlying scenario that will be used to illustrate our model is a social network of individuals who have some intrinsic characteristics (e.g., birthplace, gender and age) and who express interest for items via likes and votes as in Facebook and Twitter. At any given point in time, we are interested in recommending content items (e.g., movies, books, conferences) to an ad-hoc group.

In this chapter, while we inherit the same global data model we already proposed for the whole framework in Section 2.1, we introduce some additions and modifications. In this chapter, $G \subseteq \mathcal{U}$ is an ad-hoc user group. We consider time as a set of consecutive timestamps that form periods. Each period p is a time interval of the form $[s, f]$ where s is its starting timestamp and f its ending timestamp. Figure 5.1 illustrates all components of the data model for user group recommendation. Dark boxes are the contributions of this work.

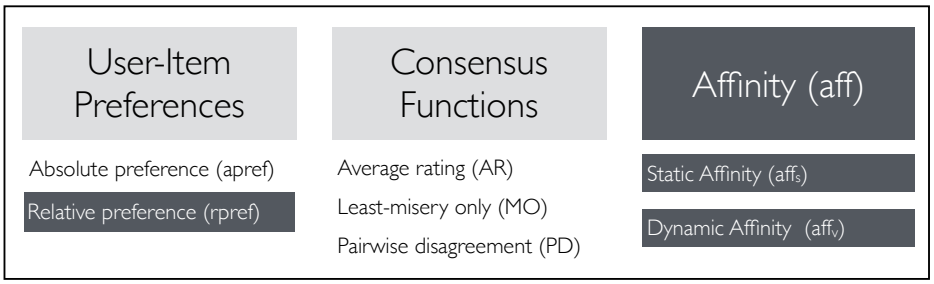


FIGURE 5.1: Components of User Group Recommendation

5.2.1 User Affinity Models

Affinity describes the *bonding* between a pair of users u and u', denoted as $\mathit{aff}(u, u')$. It could be as simple as explicit friendship or users in the same age group or more sophisticated such as users who like similar movies, have visited similar places and have friends who live in different parts of the world. For simplicity, we assume that affinity between a user pair is symmetric, i.e., $\mathit{aff}(u, u') = \mathit{aff}(u', u)$. More importantly, $\mathit{aff}(u, u')$ is dynamic and changes over time. We therefore compute affinity $\mathit{aff}(u, u', p)$ for a time period $p = [s, f]$. This dynamic affinity captures changes over time by combining its *static* and *dynamic* components defined below.

Static Affinity. ($\mathit{aff}_S(u, u')$) This is a time-agnostic affinity component and is used to capture how close two users are in a time-independent fashion. Stable factors such as birthplace, age, and education naturally contribute to this component. However, depending on the application, other dimensions could be accounted for. For example, Facebook friendship being stable, we use it to model static affinity in our experiments.

Dynamic Affinity. ($\mathit{aff}_V(u, u', p)$) This is a time-variant component that captures affinity between two users u and u' during period p by considering how close they are during that period. For example, shared political interests, common likes, and shared interests for world events, vary over time and could contribute to formulating this component.

Intuitively, the objective is to capture the *aggregated drift* that the affinity of a user pair exhibits for every time period from the beginning of time s_0 to the end of the current period $p = [s, f]$, compared to the overall user population.

More specifically, time starts at the beginning of time s_0 and is segmented into subsequent time periods $p_0, \ldots, p_{now}$ of varying lengths. Given two time periods $p_i = [s_i, f_i]$ and $p_j = [s_j, f_j]$, $p_i \leq p_j$ is used to denote that p_i precedes p_j, i.e., $s_i \leq s_j$ and $f_i \leq f_j$. Determining the right granularity of a time period depends on the application at hand

and the frequency of user actions and is orthogonal to our model. For example, in a social network such as Facebook, and when affinities are computed using shared posts, granularity may vary from hours to days depending on the time of year. On Twitter, granularity is finer and may vary from minutes to hours since post frequency is higher. Not all time periods are necessarily of the same length.

Given a time period $p = [s, f]$, for every time period p' that is included in the interval starting at the beginning of time s_0 and ending at f, the end of p, the periodic affinity drift is calculated as a difference between the periodic affinity $\mathit{aff}^P(u, u', p')$ between users u and u' and the average periodic affinity $\mathit{Avgaff}^P(p')$ of the whole user population. These drifts are aggregated over all time periods included in the interval $[s_0, f]$ and normalized to generate $\mathit{aff}_V(u, u', p)$. Formally,

$$\mathit{aff}_V(u, u', p) = \frac{\Sigma_{p' \leq p}(\mathit{aff}^P(u, u', p') - \mathit{Avgaff}^P(p'))}{\Delta} \tag{5.1}$$

The exact formulation of Δ depends on how time is modeled (discrete or continuous) and is described. Interestingly, $\mathit{aff}_V(u, u', p)$ could be either positive or negative and depends on how the affinity of (u, u') evolves compared to the overall population.

The exact formulation of $\mathit{aff}^P(u, u', p')$ depends on the application. In our Facebook experiment in Section 5.4.1.2, we use common page likes between u and u' during period p'.

Finally, $\mathit{Avgaff}^P(p')$ is defined as follows:

$$\mathit{Avgaff}^P(p') = \frac{2 \times \Sigma_{(u,u') \in \mathcal{U}, u \neq u'} \mathit{aff}^P(u, u', p')}{|\mathcal{U}|^2 - |\mathcal{U}|} \tag{5.2}$$

We now describe our dynamic affinity models that use $\mathit{aff}_S(u, u')$ and $\mathit{aff}_V(u, u')$ as building blocks. The first model relies on discretized time periods to capture aff_V whereas the second represents time in a continuous fashion.

Discrete Dynamic Affinity Model. In this model, the aff_S and aff_V affinity components are aggregated using a linear function over a set of discretized time periods. Therefore, Δ, the denominator in Equation 5.1, is simply the number of time periods between s_0, the beginning of time, and e, the end of p. This simple aggregation also allows us to design efficient algorithms.

$$\mathit{aff}^D(u, u', p) = \mathit{aff}_S(u, u') + \mathit{aff}_V(u, u', p) \tag{5.3}$$

Continuous Dynamic Affinity Model. For this model, time is considered in a continuous fashion. In this case, the denominator in Equation 5.1, $\Delta = f\text{-}s_0$ is the length of time between the the beginning of time s_0 and f, the end of p. As a natural representation to capture continuous time, we consider an exponential function, which is also supported in prior work [Kor10]. Formally,

$$\mathit{aff}^C(u, u', p) = \mathit{aff}_S(u, u') \times e^{\lambda(f - s_0)} \tag{5.4}$$

Here λ is the rate of growth/decay of affinity and could simply be replaced by $\mathit{aff}_V(u, u', p)$ in Equation 5.1 to represent the cumulative effect of affinity drift over time.

Consequently, the discrete time model could be viewed as an approximation of the continuous one where time is discretized into sub-periods and each user pair's affinity is normalized over the number of periods (Equation 5.1). Alternatively, the continuous model treats time as a single interval $[f\text{-}s_0]$ and captures an exponential growth, respectively, decay, affinity model when $\mathit{aff}_V(u, u', p)$ is positive, respectively, negative.

In both $\mathit{aff}^D(u, u', p)$ and $\mathit{aff}^C(u, u', p)$ affinity drift could be negative or positive thereby capturing situations in practice where affinity between two users may increase or decrease over time. We believe that the ability to capture this varying rate of change is important in practice in particular for social networks where different users exhibit different interests over time.

5.2.2 User-Item Preference Model

We now show how affinities are accounted for in computing the preference of a user for an item in a group. We first describe how affinity is incorporated into user-item preferences without accounting for time then we show how to modify the formulation to compute time-aware user-item preferences.

Time-Agnostic User-Item Preference. Given a group g, the preference of a user $u \in g$ for an item $i \in \mathcal{I}$ is denoted $\mathit{pref}(u, i, g)$ and depends on two components:

- **Absolute preference.** ($\mathit{apref}(u, i)$) This describes how much u likes item i akin to the predicted rating of u for i. Existing single-user recommendation algorithms, such as collaborative filtering, could be used to compute $\mathit{apref}(u, i)$.
- **Relative preference.** ($\mathit{rpref}(u, i, g)$) This component captures that *a user likes an item i if close members in the group g also like i* and similarly that *a user dislikes an item i if close members in the group g dislike i*. Affinity between

group members is used to capture how close they are. More formally, $rpref(u, i, g)$ combines the affinity of a user u with other members $u' \in g$, denoted $aff(u, u')$, with the preference of u' for item i, denoted $apref(u', i)$.

$$rpref(u, i, g) = \Sigma_{\forall u' \neq u \in g} aff(u, u') \times apref(u', i) \tag{5.5}$$

The overall affinity-aware user-item preference is a simple combination of these two factors: $pref(u, i) = apref(u, i) + rpref(u, i, g)$.

Time-Aware User-Item Preference. We now modify the definition of relative preference to capture temporal affinities:

$$rpref(u, i, g, p) = \Sigma_{\forall u' \neq u \in g} aff(u, u', p) \times apref(u', i). \tag{5.6}$$

Therefore, a user u's overall preference for item i during time period p can be simply formulated as:

$$pref(u, i, g, p) = apref(u, i) + rpref(u, i, g, p) \tag{5.7}$$

5.2.3 Problem Definition and Hardness

Members of a group may not always have the same preferences for items and a *consensus function* needs to aggregate user-item preferences into a single group's preference for an item. Intuitively, there are two main aspects in a consensus function [OCKR01a]. First, the preference of a group for an item needs to *reflect the degree to which the item is preferred by all group members.* The more group members prefer an item, the higher its group preference. Second, the group preference needs to *capture the level at which members disagree or agree with each other.* All other conditions being equal, an item that draws high agreement should have a higher score than an item with a lower overall group agreement. We call the first aspect *group preference* and the second aspect *group disagreement.* We revisit the definitions that were introduced in [AYRC+09] to include a time component.

Group Preference. The preference of an item i by a group g during a time period p, denoted $gpref(\mathcal{G}, i, p)$, is an aggregation over the preferences of each group member for that item. We consider two commonly used aggregation strategies:

- **Average Preference.** $\frac{1}{|g|}\sum_{u\in g}(pref(u,i,g,p))$
- **Least-Misery Preference.** $min_{u\in g}(pref(u,i,g,p))$

Alternative aggregations (e.g. Most-Happiness, i.e., taking the maximum over all individual preferences) are also possible.

Group Disagreement. The disagreement of a group g over an item i during a time period p, denoted $dis(\mathcal{G},i,p)$, reflects the degree of consensus in the user-item preferences for i among group members over time. We revisit the two most common disagreement computation methods:

- **Average Pair-wise Disagreements.** $dis(g,i,p) = \frac{2}{|g|(|g|-1)}\sum(|pref(u,i,g,p) - pref(v,i,g,p)|)$, where $u \neq v$ and $u,v \in g$;
- **Disagreement Variance.** $dis(g,i,p) = \frac{1}{|g|}\sum_{u\in g}(pref(u,i,g,p) - mean(i,g,p))^2$, where $mean(g,i,p)$ is the mean of all the individual preferences for item i over time.

The average pair-wise disagreement function computes the average of pair-wise differences in preferences for the item among group members, while the variance disagreement function computes the mathematical variance of the preferences for the item among group members. Intuitively, the closer the preferences for i between users u and v, the lower their disagreement for i.

Time-Aware Group Consensus. We combine group preference and group disagreement in a time-aware consensus function, denoted $\mathcal{F}(g,i,p)$. The function combines group preference and disagreement for an item i and a group g into a single group consensus score using the following formula: $\mathcal{F}(g,i,p) = w_1 \times gpref(g,i,p) + w_2 \times (1 - dis(g,i,p))$ where $w_1 + w_2 = 1.0$ and each specifies the relative importance of preference and disagreement in the overall group consensus.

Note that the formulation of group consensus incorporates temporal affinities by aggregating the relative user-item preferences of its members with disagreement. The proposed formulation is orthogonal to how affinities are modeled and incorporated into individual relative preferences. This way accounting for temporal affinities in group recommendation is orthogonal to the consensus function used to aggregate group members.

Given a group g, a time-aware consensus function $\mathcal{F}$ and an integer k, the objective is to recommend to g the k best itemset $\mathcal{I}_G$ that accounts for its members' affinities during a period p, such that:

- $|\mathcal{I}_g| = k$
- $\forall i \in \mathcal{I}_g, u \in g$, i is not individually recommended to u
- $\nexists j \in \mathcal{I}$, s.t. $\mathcal{F}(g, j, p) > \mathcal{F}(g, i, p)$, where $j \notin \mathcal{I}_g, i \in \mathcal{I}_g$, i.e., there does not exist any other item j in $\mathcal{I}$ whose consensus score is higher than any item in i in $\mathcal{I}_g$.

5.3 User Group Recommendation Algorithm

In this section, we discuss how to efficiently compute k affinity-aware recommendations for ad-hoc groups, meaning for groups that are not known beforehand. Recall that given a group g, the goal, stated in Section 5.2, is to find the k best items to recommend to g according to a consensus function $\mathcal{F}$.

We propose an instance optimal algorithm to compute top-k items for a given group under different group consensus functions. The overall intuition of this algorithm is appropriately adapted from the family of Fagin-style top-k algorithms [FLN01]. Those algorithms, such as, Threshold algorithms `TA` or No Random Access Algorithm `NRA`, rely on a function that aggregates multiple score components into a single score for each item. Those algorithms are used in Web search to compute the score of each item (a document in that case) as a combination of its component scores (its scores for each keyword in the search query). They aim to find the k items that rank the highest (the ones with the highest aggregated scores) in as little time as possible. They take sorted item lists that correspond to each component and scan them using *sequential* and *random accesses* (`SA`s and `RA`s), and the computation can be terminated without scanning the input lists fully, using stopping conditions based on score bounds (thresholds). Early stopping is possible when the ranking function is *monotone* [FLN01].

Theorem 5.1. *The temporal affinity-aware consensus function $\mathcal{F}$ is monotonic w.r.t. absolute preference lists and user-affinity lists for the dynamic user-affinity model, and pair-wise disagreement lists.*

Proof. (sketch): It is shown in a prior work [AYRC+09], that all three group consensus functions without considering time-agnostic affinity (average preference, least misery and pair-wise disagreement) are monotone. If all group members, except a user u, rate items i_1 and i_2 the same, i_1 will have at least the same group preference as i_2 if u rates i_1 no less than i_2. This holds for both the average and least-misery. For pair-wise disagreement, we showed that our group disagreement functions (pair-wise and variance) can be transformed into aggregations of individual pair-wise disagreements and become monotone.

u_1	u_2	u_3
i_1 5	i_1 5	i_3 2
i_2 1	i_2 1	i_1 2
i_3 1	i_3 0.5	i_2 1

TABLE 5.1: Absolute Preference Lists $\mathcal{PL}_u$ of u_1, u_2, u_3

Monotonicity remains true with the introduction of affinities and time. For an item i, if both users like i highly, higher affinity between them only improves i's overall preference. On the contrary, for an item j, if they like j as highly as they do i, lower affinity between them only decreases j's overall preference. Introduction of time in the affinity model only makes the affinity calculation time-dependent by changing the temporal granularity at which it is computed; however, the relationship between dynamic affinity and the group consensus of an item does not change. □

As a result, we can design an instance optimal algorithm with the early stopping.

We now describe the data structures necessary to run Fagin-style top-k processing algorithms via an example that will also be used to illustrate our algorithm, `GRECA`.

Imagine a group g formed with three users u_1, u_2, u_3. Given an itemset $\mathcal{I} = \{i_1, i_2, i_3\}$, our objective is to identify the best item ($k = 1$) to recommend to the group at time period p (for example, January 2014). Also, assume that the system has information about group members u_1, u_2, u_3 for one year, i.e., January 2013 to January 2014. The user-item preference lists of those group members are provided in Table 5.1. Each list contains items preferred by each user sorted in decreasing order of preference.

The item preference of a member u of a group g is a combination $pref(u, i, g, p) = apref(u, i) + rpref(u, i, g, p)$, where $rpref$ is the relative preference that accounts for the temporal affinity of u with other group members.

Affinity between users consists of two components, static affinity aff_S and dynamic affinity aff_V . The detailed interpretations of these affinities and how they are calculated are given in Section 5.2. Out of the two aforementioned affinities, the latter is time-aware, where time is considered in a continuous fashion or over a discrete set of time periods (for example, two equal periods p_1 and p_2 each of six months in our case). For simplicity, we consider the discrete model in this example. The aff_S affinity involves all user-pairs thereby creating $3 \times (3 - 1)/2$ (i.e. $n(n - 1)/2$ in general) entries. For every time period p', similarly, there is a periodic affinity list of the same size. Notice that each affinity list aff_V or aff_S with $n(n - 1)/2$ entries could further be partitioned into a set of $n-1$ lists, where the i-th list stands for user u_i with $n-i$ entries. For example, we

u_1	u_2
u_1u_2 1	u_2u_3 0.3
u_1u_3 0.2	

TABLE 5.2: Static Affinity Lists $\mathcal{L}_{aff_S}$

u_1	u_2
u_1u_2 0.8	u_2u_3 0.2
u_1u_3 0.1	

TABLE 5.3: $\mathcal{L}_{aff_V}$ Lists for Period p_1

u_1	u_2
u_1u_2 0.7	u_2u_3 0.1
u_1u_3 0.1	

TABLE 5.4: $\mathcal{L}_{aff_V}$ Lists for Period p_2

can have a $\mathcal{L}_{aff_S}(u_1)$ that stores u_1's static affinity with u_2 and with u_3, one for $\mathcal{L}_{aff_S}(u_2)$ with u_2's affinity with u_3 only (storing u_1's affinity here again is redundant), and no static affinity list needs to be created for user u_3. This partitioning allows us to design efficient algorithms, as we describe later in Section 5.3. Table 5.2 contains aff_S affinity lists of all users sorted in decreasing order and Tables 5.3 and 5.4 contain aff_V affinity of users in periods p_1 and p_2 respectively. Note that the temporal affinity of users u_1 and u_2 has decreased between periods p_1 and p_2.

In the above example, temporal affinity between user pairs u and u' is modeled in a discrete manner as $aff^D(u, u', p)$. To facilitate efficient computation, it is easy to see that the different absolute preference lists and time-variant affinity lists are to be precomputed. Even for a small group such as the one in the example with 3 users, there are 3 absolute preference lists. Furthermore, the all-pair user affinities for a given time period p' are to be stored as well, either as a single list with $n(n-1)/2$ entries, or decomposed over a set of $n-1$ lists, where the i-th list represent user u_i's affinities with $n-i$ other users. Since the period affinities are independent of each other, we must precompute such lists for every time period. For the example case, this requires either creating 2 periodic affinity lists to capture aff_V affinity and one static affinity list to capture aff_S. Each of these lists have $n(n-1)/2$ entries (as a single list or splitted in $n-1$ lists, as described in the example). The size of each list is quadratic in the number of users, but the number of such lists ($\mathcal{T}$) is a function of how time is discretized into periods. Even for a small group such as ours, many lists are to be used in the computation. Notice that all these user-affinity lists are required to compute the *complete score* of any item, because, the relative preference $rpref(u, i, g, p)$ for every item requires accessing *all* $\mathcal{T} \times n(n-1)/2$ entries. An algorithm such as `TA` must read all those entries to compute the *complete score* of an item and will hence incur a large number of `RA`s.

We argue that all these accesses are not always necessary. For instance, based on preferences in Tables 5.1 to 5.4, we consider scanning item i_1 in $\mathcal{PL}_{u_1}$. If we were following `TA` method, to compute the complete score of this item, 21 `RA`s are needed, i.e., one `RA` for each $apref(u, i_1)$ component and 6 `RA`s for each $rpref(u, i_1, p)$ component where $u \in \{u_1, u_2, u_3\}$. Note that to compute the score of a single item i_1, we have accessed *all* entries in $aff_S(u_1)$, $aff_V(u_1, p_1)$ and $aff_V(u_1, p_2)$ lists. For instance, entries in the list

$aff_S(u_1)$ is the static affinity scores between (u_1, u_2) and (u_1, u_3) where we have accessed both.

Instead, our instance optimal algorithm `GRECA` makes *only sequential accesses, i.e.,* `SA`*s* like `NRA` and *potentially avoids consuming all these* $\mathcal{T} \times n(n-1)/2$ *entries* to determine the top-k itemset. Following previous example, if for instance $aff_S(u_1, u_3)$ (in Table 5.2) is not yet scanned, we avoid making an `RA` to get this value, but based on `NRA` principle, we use the score under the cursor in the list of Table 5.2 (i.e., initially $aff_S(u_1, u_2)$) to compute a partial score for i_1.

`GRECA` returns the top-k itemset which contains the best set of k-items, although the rank among the returned itemset may not be fully distinguishable (i.e. giving rise to a partial order). This is rather reasonable, because k is usually small, and the group is potentially interested in all of the k-items.

For ease of exposition, we describe `GRECA` using the simplest group consensus function *Average Preference* considering time-aware affinity. The other group consensus functions mimic its behavior.

Without loss of generality, for a given group with n users, `GRECA` uses n user-item preference lists, where each list $\mathcal{PL}_u$ for user u has m items that are sorted in decreasing user-item preference. Each $\mathcal{PL}$ can be obtained with any single user recommendation strategy (in our experiments in Section 5.4, we use collaborative filtering). In addition, `GRECA` uses $n-1$ static affinity lists, and another $n-1$ dynamic periodic affinity lists for each time period.

The algorithm runs in a round-robin fashion over the aforementioned lists by making only `SA`s. It reads an entry $e = (i, r)$, where i is the item-id and r is the user u's absolute preference score for i, or an entry $e' = (u', r')$, where r' is the pair-wise affinity of (u, u'). Affinity between a pair of users is either static or periodic (i.e. dynamic), and the computation does not distinguish between these two kinds. The algorithm invokes the following 3 different subroutines to determine whether to continue further or to safely terminate and return the top-k itemset during its execution:

- **(a) Compute Upper-Bound of an Item** `ComputeUB(i)`. UB_i computes the highest score that an item i can have in g based on so far accesses.
- **(b) Compute Lower-Bound of an Item** `ComputeLB(i)`. LB_i computes the lowest score that an item i can have in g based on so far accesses.
- **(c) Compute Global Threshold** `ComputeTh(`$\{E\}$`)`. Input to this function is the current set $\{E\}$ of entries read from all the lists. The output is simply a numeric score that captures the highest score that an *unseen* item can have for group g.

Subroutines can be invoked after reading one entry from each type of list (preference list, static affinity list or dynamic affinity list) to make sure all types of lists are visited before or after reading the j-th entry from all lists.

The first two subroutines return the latest bounds of an item i. Then, those updated bounds are pushed into an *item buffer* that is maintained throughout the execution of the algorithm. We describe our proposed buffer management strategy later on. Naturally, these two subroutines are to be invoked for all encountered items so far.

Illustration of the Subroutines: The upper-bound score of an item i is simply the highest score it can have on current accesses. It is computed by combining the actual encountered values for some of the entries and then assigning the current cursor readings to the rest. Consider our three-user groups and assume that `ComputeUB(`i_3`)` is invoked after the cursor reads the second entry at $\mathcal{PL}_{u_2}$. At that point, $apref(u_3, i_3) = 2$ is encountered, but for the other two users, these are to be approximated based on the current cursor readings. For example, the highest score of $apref(u_1, i_3) = 1$, $apref(u_2, i_3) = 1$. Similarly, static and dynamic periodic affinities of users u_1, u_2 and u_2, u_3 are encountered, but those of u_1, u_3 are to be guessed based on the latest cursor reading from the respective lists. This gives rise to `ComputeUB(`i_3`)` $= \Sigma_{\forall i \in \{1,2,3\}}$`UB`$[apref(u_i, i_3)] +$ `UB`$[rpref(u_i, i_3, g, p)] = 13.02$ (by ignoring normalization and final averaging).

The computation of the lower-bound of an item i is similar except that it replaces the unseen entries of the function with the lowest possible score. For example, instead of assigning $apref(u_1, i_3) = 1$, $apref(u_2, i_3) = 1$, it will consider those values to be 0 (assuming that the smallest absolute preference for an item could be 0). The same will happen in affinity calculation; as an example, it substitutes $aff_S(u1, u_3) = 0$, instead of 0.8 in the upper-bound computation case. When invoked using item i_1, `ComputeLB(`i_1`)` returns a value of 14.2 (ignoring normalization and final averaging).

Computation of `ComputeTh(`$\{E\}$`)` is rather simple. It simply incorporates each of the entries in $\{E\}$ in the function and returns a numeric score.

Buffer Management Strategy: Once the upper-bound and lower-bound scores of each item are computed, they are pushed into a buffer $\mathcal{B}$ and are sorted in decreasing order of lower-bound score. The buffer is implemented as a heap data structure which allows efficient updates since it requires to maintain sorted lists of potential results and, in some cases, item lower-bounds and upper-bounds need to be updated (for example, when the item is encountered again in one of the lists).

Stopping Condition: The algorithm has both global threshold computation and buffer management strategies. We now show that *the buffer management* itself is sufficient to

govern early stopping. More importantly, unlike traditional threshold algorithms, `GRECA` cannot terminate only based on the threshold condition in the cases, where the buffer contains more than k items.

- **Using the Global Threshold:** If the current global threshold is not larger than the lower-bound score of the k-th item in the buffer, `GRECA` will not find any item later on whose score is larger than the current threshold. On the other hand, if the current threshold is no larger than the lower-bound of the k-th item in the buffer, any unseen item can never be in the top-k itemset. This implies that a subset of the items in the current buffer is the actual top-k itemset. If the buffer contains k items only, then `GRECA` can safely terminate and return those items in the buffer as the answer. However, in general, when the buffer has more than k-items, to precisely determine the actual top-k itemset, it needs to apply the buffer management strategy that we describe now.

- **Using the Buffer:** A key novelty of `GRECA` is in using only the buffer condition for termination. This condition simply implies that just by looking into the items in the buffer, `GRECA` can terminate, as well as declare the partially ordered correct top-k itemset. The buffer stopping condition works as follows: the buffer contains k'-items ($k' > k$) such that the lower-bound of the k-th item score is no smaller than the upper-bound score of each of the remaining $k' - k$ items. In that case, those remaining $k' - k$ items could be safely pruned. Interestingly, satisfying this condition implies satisfying the threshold condition as well, as Theorem 5.2 states. The remaining k items are returned as answers.

- **Global Threshold and Buffer Management:** Global threshold can simply determine that the current buffer contains a subset of items which are the actual top-k itemset. In a general case, where the buffer has more than k items, `GRECA` applies the buffer stopping conditions to determine that subset. It is still possible that the buffer condition for stopping is not met. In that case, `GRECA` resumes computation until the buffer condition is satisfied or all lists are exhaustively scanned.

Theorem 5.2. *Satisfying the buffer condition for termination implies that the global threshold condition for termination is met.*

Proof. (sketch): At a given snapshot during the execution of `GRECA`, the score returned by `ComputeTh(`$\{E\}$`)` is strictly not greater than the upper-bound score of any item that is already seen and in the buffer, i.e., `ComputeUB(`i`)` $\geq$ `ComputeTh(`$\{E\}$`)`. Therefore, if the buffer condition is satisfied (meaning that the lower bound of the k-th item score in the buffer is not smaller than the upper-bound score of the remaining $k' - k$ items), this

automatically implies, that the lower bound of the k-th item score is not smaller than the current global threshold. Hence the proof. □

For our running example , this returns i_1 as the top-1 item to the group.

The pseudocode of `GRECA` is presented in Algorithm 6. In addition to the group g and k, it takes the preference and affinity lists of g as inputs as well as the consensus function $\mathcal{F}$. Lines $9-14$ either add a new item into the buffer $\mathcal{B}$ and compute its lower-bound and upper-bound scores, or update the latest lower-bound and upper-bound score of an existing item and reorganize the buffer. Line 16 computes the global threshold condition using the function `ComputeTh`(); lines $17-19$ checks if the threshold stopping condition is satisfied. Otherwise, the control goes on to line 21 on wards and `CheckBuffer(`$\mathcal{B}$`)` checks whether the stopping condition is met using the buffer. The computation continues unless one of these conditions are satisfied, or all lists are exhaustively scanned. Of course, in the latter case, there is no save-up. However, as our experimental results exhibits, `GRECA` achieves speed-up, compared to its naive counterpart.

Algorithm 6: Group Recommendation with Temporal Affinities (`GRECA`)

Require: Group g, k, consensus function $\mathcal{F}$;
1: Retrieve user preference lists $\mathcal{PL}_u$ for each user u in group g;
2: Retrieve pair-wise affinities for aff_S , aff_V for each period;
3: $Sc_r = \{r_u\}$, the last user preference from $\mathcal{PL}_u$, $\forall u \in g$
4: $Sc_{aff_S} = \{aff_{S\,u,v}\}$, the last pair-wise aff_S affinity values read $\forall u, v \in g$
5: $Sc_{aff_V} = \{aff_{V\,u,v}\}$, the last pair-wise periodic affinity values read for each time period p', $\forall u, v \in g$
6: Cursor $cur =$ `getNext`() round-robin accesses to $\mathcal{PL}_u$, aff_S and aff_V lists
7: **while** (cur <> NULL) **do**
8: Get entry e at cur
9: **if** !(`in` $\mathcal{B}$(`topKHeap`, e)) **then**
10: `ComputeUB`(`e`, $\mathcal{F}$)
11: `ComputeLB`(`e`, $\mathcal{F}$)
12: Add e in $\mathcal{B}$
13: **else**
14: Update `ComputeUB`(`e`, $\mathcal{F}$) and `ComputeLB`(`e`, $\mathcal{F}$)
15: **end if**
16: $Sc_{th} =$ `ComputeTh`($\{$`E`$\}, \mathcal{F}$) considering all current cursor positions
17: **if** $Sc_{th} \leq \mathcal{B}.k_{th}LB \& |\mathcal{B}| = k$ **then**
18: **return** topKList($\mathcal{B}, \|$)
19: Exit;
20: **else**
21: **if** `CheckBuffer(`$\mathcal{B}$`)` is satisfied **then**
22: **return** topKList($\mathcal{B}, \|$)
23: Exit;
24: **else**
25: $cur =$ `getNext`()
26: **end if**
27: **end if**
28: **end while**
29: **return** topKList($\mathcal{B}, \|$)

Lemma 5.3. `GRECA` *returns correct top-k itemset.*

Proof. Notice that `GRECA` returns from the buffer those k-items whose lower-bound scores are the highest and larger than the upper-bound score of any remaining item. As Theorem 5.2 proves that this also implies that the global threshold at that point cannot be larger than the lower-bound score of the k-th item in the buffer. Notice that the threshold captures the highest score that any unseen item can have. Due to the monotonicity property of the consensus function, global threshold decreases gradually, implying that the highest score of any item gets only smaller, as more entries are scanned from the lists. Therefore, when `GRECA` terminates and outputs the itemset with the highest top-k lower-bound scores, this implies that any other items that are discarded or unseen cannot have higher score than the returned itemset. Hence the proof. However, since the complete score of many of the items may not be computed upon termination, the output may give rise to a partial order among the top-k items. □

Lemma 5.4. `GRECA` *is instance optimal.*

Proof. (sketch): In [FLN01], authors prove that `NRA` is instance optimal with optimality ratio m and no deterministic algorithm can perform any better. `GRECA` mimics the cursor movement of traditional `NRA`, however, it has a different stopping condition. Theorems 5.2 and 5.3 prove that our stopping condition implies both the threshold stopping condition and result correctness, therefore, the instance optimality of `GRECA` holds. □

5.4 Experiments

We evaluate our group recommendation method from two major angles: effectiveness and efficiency. We conduct an extensive user study on Facebook (based on Facebook-72 dataset) to demonstrate that group recommendation with the consideration of temporal affinity is superior to solely relying on aggregating individual preferences (Section 5.4.1). We also run comprehensive experiments to show that `GRECA` achieves scalable performance when computing temporal affinity-aware recommendations for ad-hoc groups (Section 5.4.2).

We implemented our prototype system using Java (JDK 1.8.0). All scalability experiments are conducted on an 2.4 GHz Intel Core i5 with 8 GB of memory on OS X 10.9.5 operating system. We use the MovieLens 1M ratings dataset for our evaluation.

In our experiments, we use collaborative filtering [AT05] to generate individual user preferences where user similarity is computed with cosine similarity over $vec(u)$, i.e., the ratings of u for each movie.

$$\cos(\vec{u}, \vec{u'}) = \frac{\vec{u} \times \vec{u'}}{\|\vec{u}\|^2 \times \|\vec{u'}\|^2} \tag{5.8}$$

5.4.1 Quality Experiment

We exploit the availability of Facebook users for our user study which gives us the opportunity to obtain preferences of real users and leverage the social graph for affinities. Our aim is to compare our temporal affinity-aware group recommendation with naive methods without consideration of time or affinity. Our group recommendations are produced and compared using the following consensus functions (as discussed in Section 5.2).

- **Average Preference (AP)**, which computes the group preference for an item as the average of individual group members' preferences for that item.
- **Least-Misery Only (MO)**, which computes the group preference for an item as the minimum among individual group members' preferences for that item.
- **Pair-wise Disagreement (PD)**, which computes the group preference for an item as the combination of its average and its pair-wise disagreement between individual group members' preferences.

For each of these functions, we incorporate time-aware affinity to compute the relative user preference to an item at a given time (see Section 5.2 for an exact definition of relative preference.)

We developed an application using the Facebook API[2] and recruited 72 Facebook users overall to rate movies from the MovieLens 1M dataset. We obtained 1981 ratings. This makes our Facebook-72 dataset. Our Facebook application asks only for *public profiles* and *friend list access* permissions. Also, we anonymize the dataset by mapping Facebook IDs to a random 5-digit number. The study is conducted in two phases: *User Collection Phase* (Section 5.4.1.1) and *Quality Assessment* (Section 5.4.1.4).

Summary of Results: In summary, we observe that including temporal affinity in group recommendation significantly improves user satisfaction. The amount of satisfaction is variable and is dependent on how groups are formed. In particular, dissimilar

[2] https://developers.facebook.com

user groups as well as those with low-affinity users whose preference significantly evolves over time, are most satisfied. We found that prior work has indeed shown [OCKR01a] that reaching consensus among such group members is indeed difficult. In addition, we found that **PD**, in general, is the method of choice and works best for dissimilar and high affinity groups. This observation is also in line with one of prior results in [RAYC+10] where authors show that including disagreement in group consensus generates higher quality recommendations. We also observe that incorporating time models produces better results for high affinity groups suggesting that groups with high affinity are most sensitive to temporal affinities. Finally, the continuous time model is preferred by large groups of dissimilar members. That could be explained because it better captures variability for groups whose members are more sensitive to differences between them. The discrete model on the other hand, is a good approximation of the continuous one in the case of high affinity and high similarity groups.

We now delve into the details of the experiments.

5.4.1.1 User Collection Phase

In this phase, the goal is to recruit users and collect their data. Later, collected users are used to form different groups and perform judgments on group recommendations. For this aim, we start with 13 seed users (denoted S). Users in S have to complete two tasks: *i*. rate at least 30 movies in MovieLens, and *ii*. invite between 10 and 20 of their friends to participate in the study. The set of friends of a seed user $s \in S$ is denoted $friends(s)$. Note that we consider $\cup_{s \in S} friends(s) \cap S = \emptyset$. Friends are only asked to rate movies and not invite friends, i.e., we stop at the depth 1 of the social graph for this study.

We select a subset of MovieLens movies for participants to provide their preferences. We consider two factors in selecting those movies: *familiarity* and *diversity*. On one hand, we want to present users with a set of movies that they do know about and therefore can provide ratings for. On the other hand, we want to maximize our chances of capturing different tastes among movie-goers. Towards those two goals, we select two sets of movies. The first set is called the *popular set*, which contains the top-50 movies in MovieLens in term of popularity (i.e. the number of users who rated a movie in the set). The second set is called *diversity set*, which contains the 25 movies with the highest variance among their ratings and that are ranked in the top-200 in terms of popularity. Each participant rates movies in one of two pre-computed sets: the **Similar Set** which consists entirely of movies within the *popular set* and the **Dissimilar Set** which consists of the top-25 movies from the *popular set* and the 25 movies from the *diversity set*.

Users are instructed to provide a rating between 1 and 5 (5 being the best) for at least 30 movies listed in random order, according to their preferences.

5.4.1.2 Static and Dynamic Affinities

In addition to ratings, we store anonymized lists of *friends* and *page-likes* for each user. Since Facebook friendship is relatively stable over time, we use it to compute static affinity: $aff_S(u, u') = |friends(u) \cap friends(u')|$. We normalize all static affinity values in a group by the maximum pair-wise value in the group to obtain a number between 0 and 1.

Page likes are dynamic and are used to compute the time-varying component of affinity. To calculate dynamic affinity for each user, we store all pages (s)he has ever liked in Facebook and for each page, we record the timestamp of when the user liked it and the page category (music, movie, etc.). There exist 197 different page categories in Facebook. For privacy reasons, we do not record the name of the liked pages. Thus the periodic affinity between two users u and u' in time-period p is calculated as: $aff^P(u, u', p) = |page_likes(u, p) \cap page_likes(u', p)|$ where $page_likes(u, p)$ is the set of page categories whose pages are liked by u in time-period p. Then we calculate $aff_V(u, u', p)$ using Equation 5.1. We also normalize dynamic affinity values to be between 0 and 1. We consider 6 different two-month consecutive periods (Section 5.4.2.1). Note that the average standard deviation over number of common page-likes for all user pairs during 6 periods is 0.42.

5.4.1.3 Group Formation

We consider three main factors in forming user groups, i.e., *group size*, *group cohesiveness* and *affinity strength*. Size and cohesiveness (i.e. how similar are group members in their movie tastes) are akin to prior work [RAYC+10].

We hypothesize that varying group sizes will influence reaching consensus among the members and therefore to which degree members are satisfied with the group recommendation. We choose two group sizes, 3 and 6, representing *small* and *large* groups, respectively.

Similarly, we assume that group cohesiveness is also a significant factor in their satisfaction with group recommendation. As a result, we form two kinds of groups: *similar* and *dissimilar*. A similar group is formed by selecting users who *i*. have watched **Similar** movies and *ii*. have the maximum summation of pair-wise similarities (between group

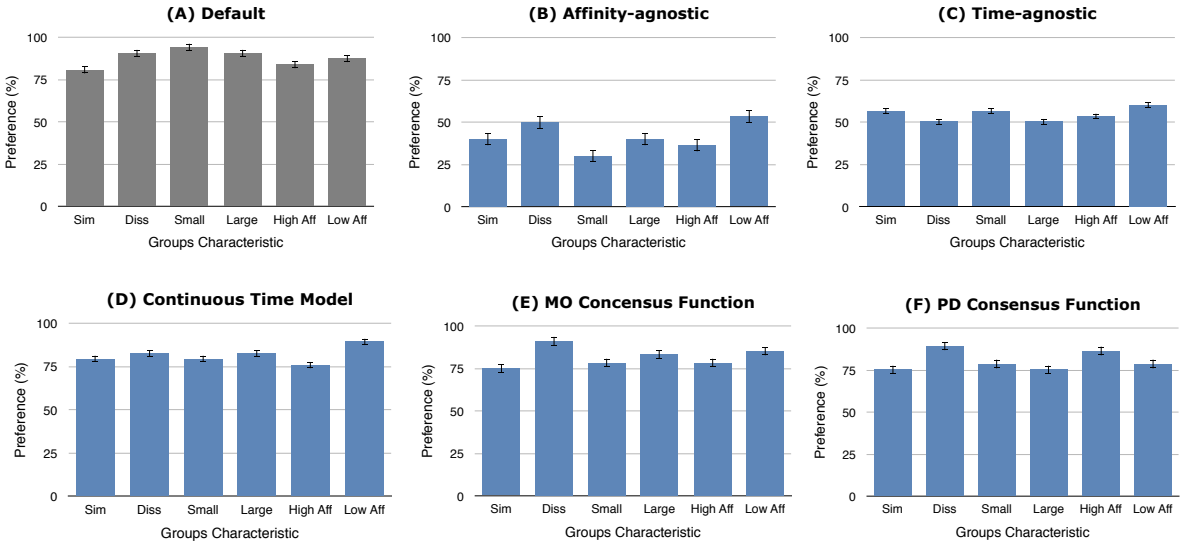


FIGURE 5.2: Independent Evaluation

members based on their provided ratings) among all groups of the same size. A dissimilar group is formed by selecting users who *i*. have completed the **Dissimilar** movie set and *ii*. have the minimum summation of pair-wise similarities among all groups of the same size.

Finally, we consider groups with *low* and *high* affinity between members. We set affinity to be high if each pair-wise affinity in a group is equal to 0.4 or higher.

5.4.1.4 Quality Assessment

In the second phase of the study, users are instructed to decide which of the recommended movies they are satisfied with in a group. We form 8 groups out of Facebook users by considering different combinations of *group size*, *group cohesiveness* and *affinity strength*. Each user evaluates movies in two phases: *Independent* and *Comparative*.

Independent Evaluation: In the independent evaluation, a user, who is a member of a group, observes a single recommendation list at each time and is asked to say how satisfied she is with watching those movies with other group members using a scale between 0 and 5 (5 being the best). Figure 5.2 illustrates the results of this evaluation phase. The score is reported as a percentage, i.e., a result with an average score of 5 gets 100%. Four parameters play a role in generating different recommendation lists in Figure 5.2, i.e., affinity awareness, time model (discrete vs. continuous), temporal awareness and consensus function. Figure 5.2. *A* illustrates results with default values,

i.e., affinity-aware, discrete, time-aware and **AP** consensus function. In all other figures, only one parameter value changes, i.e., affinity-agnostic in B, time-agnostic in C, continuous time model in D, **MO** function in E and finally **PD** function in F. That parameter is mentioned in the title of each chart in Figure 5.2.

We observe that in general, participants give a score of at least 80% to A, which is the default case with discrete temporal affinity. Participants in dissimilar groups have scored A with 90.66% preference while it is 10% lower for similar groups. This could be interpreted as: averaging individual ratings and using a discrete time model works well for groups formed by users who like different movies. Interestingly, the same result holds for low affinity and high affinity groups. This potentially shows that our model is robust to time-varying tastes. On the other hand, the low preference of high affinity groups show that members of those groups benefit from another consensus function, i.e., **PD** (chart F).

Lists without affinity (chart B) and time awareness (chart C) have at most 55% and 60% overall preference respectively. This margin of 20% difference in preference with the temporal affinity case (chart A) shows explicitly the importance of affinity and temporal affinity in group recommendation. In B, worst results are obtained for small (30.08%), high affinity (36.66%) and similar groups (40%) where we observe a decrease in satisfaction. This potentially shows those are the groups that would best benefit from using affinity in computing their recommendations. In C the worst results are for dissimilar and large groups (both 50.19%). One explanation is that dissimilar large groups, i.e., those who differ in their movie tastes among many members, prefer temporal recommendations, i.e., movies that are generated by taking into account their friendship and page-like differences over time.

Groups with different tastes (dissimilar, large and low affinity) prefer the continuous time model (chart D). This is potentially because of a higher precision in capturing time. Considering the time as a whole from the beginning of time is needed to deliver recommendations that satisfy all members of those heterogeneous groups. In case of **MO** (chart E), we observe a superior satisfaction for dissimilar and low affinity groups as *increase in uncertainty in large groups* leads members to like **MO** better.

Comparative Evaluation: In the comparative evaluation, users are asked to compare two lists L_1 and L_2 at a time and pick the list they prefer. Following the closed world assumption, when a list is not chosen by a user, it means that it is not preferred. A user has to choose one and only one of the proposed lists. Figure 5.3 illustrates the preferences of L_1 over L_2.

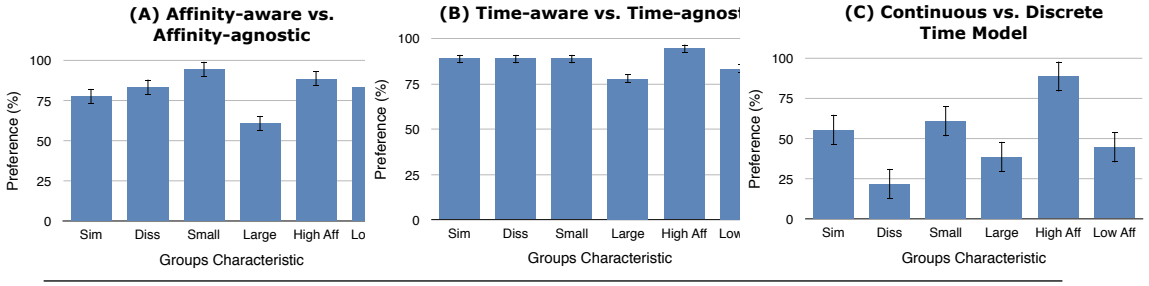


FIGURE 5.3: Comparative Evaluation

First, a user is asked to compare affinity-aware (L_1) vs. affinity-agnostic (L_2) recommendations. In A, we observe that in general, in 75% of the cases, affinity-aware recommendations are preferred. They are mostly appreciated by small groups followed by high affinity groups. Larger groups have less preference for affinity-aware results. A large group potentially leads to higher variability of preference and weaker affinity among its members, thus naturally prohibiting an early agreement.

In the second comparative study, we examine the effect of temporal affinity by comparing time-aware (L_1) vs. time-agnostic (L_2) recommendations. In B, we observe that in most groups, temporal recommendations are preferred in over 80% of the cases. This leaves no doubt that participants like better results obtained based on time. It also shows that high affinity groups prefer not only affinity-based results, but also its temporal version. Small groups have also exhibited a high preference. This is because in groups with fewer members or groups whose participants deeply know each other, the effect of time manifests itself more strongly. Finally, high preference for large groups show that the temporal dimension of affinity is a useful component for such groups to obtain higher quality results, because group members potentially observe that their common affinity history plays a role in recommendations.

We now examine which of the discrete or the continuous temporal affinity models is better and in which case. In C, we observe that in general, the discrete time model is preferred for groups with strong connections between members (high affinity and high similarity). In the case of dissimilar and large groups, it is the continuous model that is preferred. The continuous nature of the latter is certainly better to capture variability for groups whose members are more sensitive to differences between them while the discrete one is a good approximation of the continuous model in the case of high affinity and high similarity groups.

Finally, we compare different group consensus functions. This time, we compare 3 different lists together which are results of **AP**, **MO** and **PD** consensus functions. We are interested to discover which function delivers more satisfactory results when we account for temporal affinities. Figure 5.4 illustrates this comparison. In short, while

the choice of which consensus function to apply heavily depends on group characteristics, there exists a general preference for **PD** especially in the case of loosely connected groups (low affinity and dissimilar groups). That could be explained by the fact that **PD** favors items that minimize disagreement between group members which is more appropriate for dissimilar group members.

In summary, it is shown that **AP** is highly preferred in small and high affinity groups. Whenever **AP** has a high preference, **PD** is also highly preferred. **MO** provides higher quality results for larger groups (this is consistent with findings reported in [RAYC+10]) and for groups with loose connections.

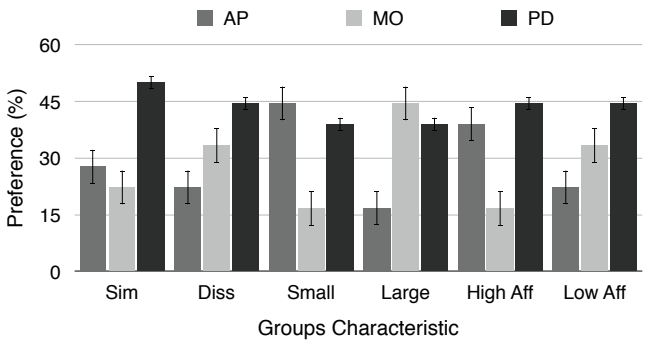


FIGURE 5.4: Qualitative Evaluation of Consensus Functions

5.4.2 Scalability Experiment

Experiment Settings: Unless otherwise stated, we form 20 different random groups by selecting a subset of users who participated in our quality experiment. The default settings of the rest of the parameters are, group size = 6, $k = 10$, number of items = 3900, consensus function = **AP**. Unless otherwise stated, affinity is computed using the discrete time model. For each scalability experiment, we compute the average percentage of `SA`s needed by `GRECA` in different settings. The percentage of `SA`s represents the computational cost that `GRECA` incurs, compared to a naive algorithm which entirely scans all lists. A smaller percentage exhibits higher scalability.

We conduct experiments by varying time periods, result size (k), group size, number of input items, similarity and dissimilarity among items and users in the group, and considering the discrete and continuous affinity models. Our results illustrate the scalability of `GRECA` with different group consensus functions. We only present a subset of these results. The omitted results are similar to the ones presented. All results are presented with standard error bars, wherever applicable.

Summary of Results: First and foremost, we observe that GRECA is highly scalable with varying k, group size, number of items and enables a significant saveup in the number of accesses (almost always, more than 75% accesses are avoided) with early termination. Then, we observe that the pruning ability is highest for similar user groups. We observe that the score distribution of top-k itemsets for such groups is different from the rest of items, therefore, the stopping condition in the buffer is satisfied early. Third, we observe that GRECA is effective across all group consensus functions. In fact, for some of the complex group consensus functions that consider user disagreement, GRECA incurs the smallest percentage of accesses ensuring the highest saveup in computation cost. Fourth, GRECA scales linearly with an increasing number of periods. Finally, we observe that GRECA is effective both for discrete and continuous models.

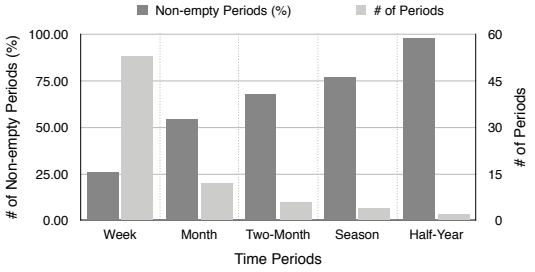


FIGURE 5.5: Different Time Periods

5.4.2.1 Varying Time Period

We explore discretizing time into periods of different lengths: week, month, two-month, season and half-year. Since dynamic affinity relies on user page-likes in Facebook and liking a page is not a frequent action, many time segments were empty after discretization (Figure 5.5). The length of a time period should be chosen in such a way that each period contains enough data to compute affinities. Figure 5.5 shows that two-month periods achieve a good balance between the percentage of non-emptiness (65%) and the number of periods (6). We hence pick a two-month discretization for the rest of our experiments.

Figure 5.6 left illustrates the average number of accesses in each period. As expected, this figure shows a linear behavior in general, as going to subsequent periods increases the number of lists. An exception happens in period 5 where its average #SA is very close to its next period. By looking more carefully at the underlying data distribution, we noticed that the number of common page-likes between user pairs in period 5 is very low. Therefore, scanning this period does not help to update bounds in order to have early termination.

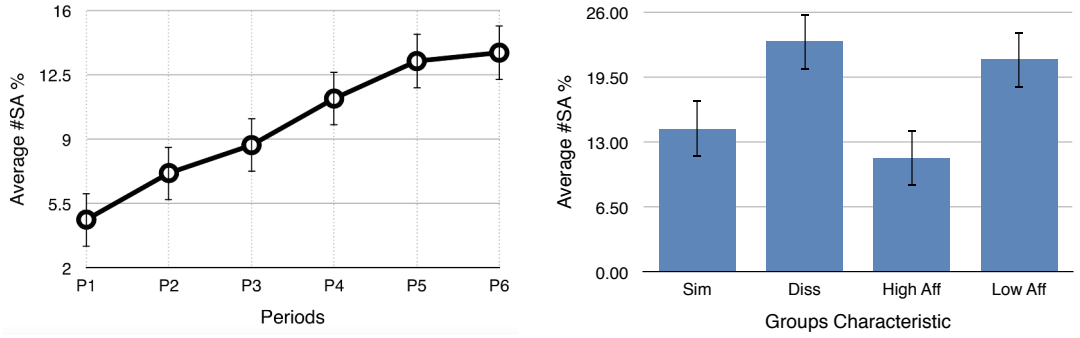


FIGURE 5.6: Average Percentage of SAs for Different Periods in Discrete Time Model (left) and Average Percentage of SAs for Similar, Dissimilar, High Affinity and Low Affinity Groups (right)

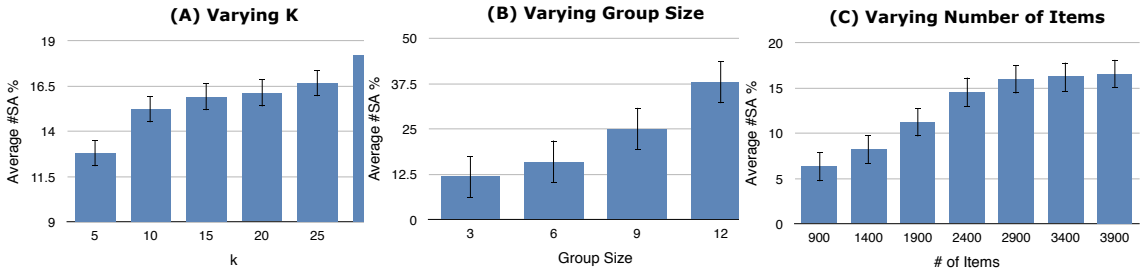


FIGURE 5.7: Average Percentage of SAs by Varying Result Size, Group Size and Number of Items

5.4.2.2 Varying k, Group Size and Number of Items

In Figure 5.7, we illustrate the scalability of GRECA by varying result size, group size and number of items. In A, we vary k from 5 to 30 and run GRECA with the **AP** consensus function for 20 different groups with 6 members. We observe that GRECA scales linearly with varying k. The algorithm always produces a saveup of 81% or higher.

In B, we examine the effect of different group sizes on performance. The results clearly demonstrate that GRECA scales well with varying group sizes. The average saveup is greater than 77%.

In C, we vary the number of available items for group recommendation from 900 to 3900. The results demonstrate that the number of accesses does not necessarily increase with that. This observation is unsurprising as the number of accesses depends on the score distribution of the item preferences and user affinities. GRECA saves more than 83% accesses even in the worst case.

5.4.2.3 Similarity/Dissimilarity

We examine the effect of similarity on GRECA in two ways: first, we compare the number of accesses between groups with similar and dissimilar ratings; then, we compare groups with high and low affinities. Figure 5.6 right contains the result. The results demonstrate that the effectiveness is higher for similar groups in both cases (item based similarity and high affinity).

5.4.2.4 Time Models

We examine the effect of continuous and discrete time models on GRECA. The average number of SAs for the continuous model is 16.32% and 16.6% for the discrete one. This means that in both cases, we obtain a saveup greater than 83%. The number of accesses for both methods are very similar with a slight superiority for the discrete model.

5.4.2.5 Consensus Functions

In this last performance study, we compare different consensus functions. Figure 5.8 contains the results. We introduce two different versions of **PD** based on [RAYC+10] by varying the weights used in the linear combination of rating aggregation (w_1) and disagreement (w_2) s.t. $w_1 + w_2 = 1$. In **PD V1**, we consider $w_1 = 0.8$ and in **PD V2**, $w_1 = 0.2$.

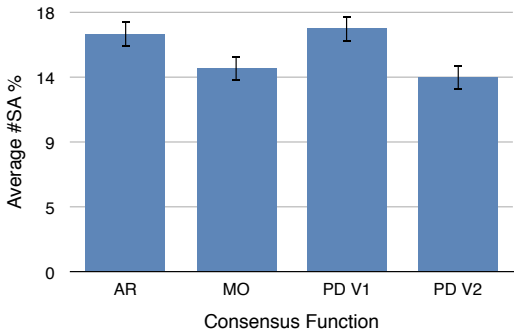


FIGURE 5.8: Average Percentage of SAs for Different Consensus Functions

All results clearly demonstrate that GRECA achieves significant saveups for those consensus functions. They also show that **PD V2** outperforms **PD V1**. During our post-analysis, we observed that a higher weight on disagreement allows faster stopping, because the items have smaller scores. **MO** is the next best performer achieving as high as 83% in accesses' saveups.

5.5 Conclusion

In this chapter, we discussed one application of user group management, i.e., group recommendation. We examined affinity-aware group recommendation over time and developed `GRECA`, an efficient algorithm with unique features that distinguish it from state-of-the-art recommendation algorithms. Our proposed semantics is compatible with popular group consensus functions. Our extensive experiments with Facebook-72 and MovieLens datasets assess the high quality of temporal affinity-aware recommendations for groups with different characteristics.

Chapter 6

Related Work

In chapters 3, 4 and 5, we explained components of our user group management framework (as in Figure 1.2). To the best of our knowledge, all our contributions in their respective domains have formalized and solved new problems which have never been addressed in the literature. However, each contribution does relate to a number of others in its concept and functionality. In this chapter, we review works related to user group discovery, analysis and recommendation.

6.1 User Group Discovery

In Chapter 3, we formalize the problem of discovering user groups for collaborative rating datasets. Recent studies [1] have shown an interest in reporting statistics about pre-defined groups, as opposed to our work where we look to discover high-quality user groups on the fly. Relevant methods for group discovery are clustering, community detection, frequent item-set mining and team formation.

Clustering. The notion of "cluster" in clustering algorithms is the same as user group. The principle of clustering is to discover groups based on *density*. Classical clustering algorithms like k-Means and k-Means++ [AV07] have three main drawbacks:

1. They do not provide an interpretation for each cluster;
2. The number of clusters should be given as input to the algorithm;
3. They discover non-overlapping clusters which is far from reality.

However there are different improvements for clustering baselines in the literature.

[1] *http://blog.testmunk.com/how-teens-really-use-apps/*

- Subspace Clustering [AGGR98] improves the state of the art and finds clusters efficiently in high-dimensional datasets and provide meaningful interpretations for each cluster. The idea is a bottom-up approach where the algorithm begins with single dimensions and try to merge them until it reaches dense clusters. Density is a user-given threshold and is defined by the number of points (in our case, users) in a cluster.

- Conceptual Clustering [Fis02] generates descriptions for each cluster based on a description language. The most known implementation is COWEB [Fis87]: it returns a classification tree where users are organized in nodes and a probabilistic description is associated to each node.

- Constrained Clustering [WCR+01] takes as input some constraints in form of *"set of users A should (not) be in a same cluster with set of user B"* and finds clusters while respecting the constraints. The approach in [PMC+14] considers *soft constraints* and minimizes the amount of constraint violation. For both conceptual and constrained clustering, the analyst needs to have an appriori knowledge about the description and query languages as well as the data structure.

- Clustering with Overlaps is an approach to improve k-MEANS and produce overlapping clusters. In [BJ03, PLL99], each cluster can have overlaps with at most two others and requires to define an overlapping threshold. Recently, OKM approach [Cle07] is proposed to approximate the optimized overlapping between clusters. Unlike k-MEANS, OKM may associate a user to more than one cluster. Not that in user group discovery, we are interested in exact membership and labels for each user group.

Community Detection. Community detection is the problem of dividing a *network* into communities, such that nodes within the same community tend to be connected by links, while those within different communities tend not to be connected by links. A network is a set of points (vertices) joined in pairs by lines (edges). Many networks are heterogeneous, consisting not of an undifferentiated mass of vertices, but of distinct groups. Our user group discovery problem is not exactly matched to community detection, because we do not consider any explicit relation (or interaction) between users in our data model. However, as we discussed in Chapter 5, we can consider any kind of interaction between users and therefore apply community detection approaches to find communities (groups) in the user data.

A community detection approach chooses one single *objective function* to capture the intuition of community as a set of nodes with better internal connectivity than external

connectivity. Community detection algorithms are usually a tradeoff between computational cost and accuracy. There exists 45 years of work in this domain, starting from bisecting data [KL70] (partitioning into two disjoint clusters) in 1970 to content-based and online community detection in recent years [QAH12, QAH13]. The revolutionary work in this domain was the introduction of *modularity* objective function by Girvan and Newman [NG04], i.e., the ratio of the number of edges between the nodes and the expected number of such edges under the null-model.

Frequent Pattern Mining [AIS93, DTL+15]. Frequent pattern mining and specifically frequent item-set mining (FIM) is another method to discover user groups and provide descriptions in form of item-sets. Other interesting approaches in Frequent Pattern Mining are redescription mining [Par05] and (constrained) pattern-set mining [BKS11, DRZ07] which bear similarities with user group discovery.

The aim of FIM algorithms is to find interesting *patterns* from datasets, such as correlations between items. A pattern is considered as a recurring structure in an enumerable, discrete domain. The recurrence is determined by a density threshold as in clustering algorithms, called support.

The input to FIM algorithms is a binary transaction database M where rows are users and columns are items. $M_{ui} = 1$ if user u does an action on item i, otherwise it is zero. To employ FIM, we need to convert our user data in form of a binary transaction database. Hence, each attribute-value pair becomes an item. For instance, one item is $\langle gender, male \rangle$ and the other is $\langle gender, female \rangle$. If user u is female and $\langle gender, female \rangle$ is the item i, thus $M_{ui} = 1$.

In [SRPC11], Aetheris approach is proposed for the problem of mining *skyline* item-sets. Skyline operator is first introduced in [BKS01] which returns results a in multi-dimensional space where each result is not worse than any other one. The methodology is the same as multi-objective optimization, while the former is in the context of *query processing* and the latter in *query optimization*. Aetheris exploits the *skylineability* property for efficient computation of skyline item-sets: a subset of objectives which preserve the dominance relations in the original set of objectives.

Team Formation. Team formation [KAZ12] and jury selection [CSTC12] are new emerging topics in crowdsourcing. The main idea is to find a team (group) of experts to collectively complete a project. The focus is therefore on putting individual workers (users) together to optimize a quality objective, i.e., minimizing the overall cost while maintaining an acceptable overall expertise. The output of such algorithms is one single optimized group.

Team formation requires the definition of *quality* and *cost* values for each worker. This is a subjective and challenging task that assumes full knowledge of researchers' profiles which is not always the case in realistic scenarios..

Multi-objective Optimization: Our approach for user group discovery is multi-objective optimization. There exists different approaches to solve a problem with a multi-objective nature. We already discussed that Scalarization does not work in our case (Section 3.5). Another popular method is the ϵ-constraints method [PY00] where we optimize one objective and consider others as constraints. The approach in [DAYDY11] can be seen as a relaxed ϵ-constraints version of our problem. Another approach is Multi-level optimization [MPV97] which needs a meaningful hierarchy between objectives. In our case, all objectives are independent and conflicting with each other, hence using this mechanism is not feasible.

Positioning. Providing a meaningful description for groups is a necessity in a data analysis process, as already discussed in Section 1.1 (i.e., *describability* principle). This principle exists in Frequent Pattern Mining and Subspace Clustering approaches. This is why we simply use an FIM algorithm called LCM to discover user groups in Chapter 4. Our group discovery contribution in Chapter 3 is distinct from existing works in following directions.

- We adapt the approach in [TK14] and propose a multi-objective optimization algorithm for user group discovery. The same idea is also proposed in [SRPC11]. However, an exhaustive approach to multi-objective optimization is very time consuming. Thus we propose an approximation algorithm, α-MOMRI, which is faster than exhaustive and provides bounds on the quality of results.

- A problem of clustering, community detection, and Frequent Pattern Mining approaches is that they usually return millions of groups and it becomes tedious for the analyst to review resulting groups (i.e., information overload). Beyond our contributions in Chapter 4 for managing the huge space of user groups, we propose a heuristic algorithm, h-MOMRI for group discovery, which outputs a limited number of high-quality representative groups.

- In [SRPC11, TK14], *generic* multi-objective optimization algorithms are proposed. In our work, we investigate on the special case of collaborative rating datasets and the specific objectives which best describe groups. We provide formal definitions of our objectives (coverage, diversity and rating distribution) and exploit their semantics to improve the efficiency of our algorithms.

- The focus in our work is to produce several user groups (akin to teams) that together optimize several objectives, while in team formation, the idea is to obtain a single optimized group.

Our work bears some similarities to the MRI (Meaningful Rating Interpretation) approach [DAYDY11] in its goal. The addressed problem in MRI is to go beyond a single numerical rating score for describing a set of ratings. The approach in [DAYDY11] characterizes user groups which contribute to this score. For instance, an output could be the group of female teenagers who have voted with a high score for Titanic (the average is 5.9), while the group of middle-age males does not provide a high score for this movie (the average is 3.4). The MRI approach is to return k user groups which maximizes the homogeneity and respect a user-given threshold on coverage. Our work is a generalization of MRI, where it takes a broader look at the problem and discovers groups which are not only necessarily homogenized, but also covers the most the set of input ratings and are most diverse. Our algorithm is threshold-free (in contrast to MRI which has a threshold on coverage) and optimizes all conflicting objectives simultaneously.

6.2 User Group Analysis

Analyzing the large space of user groups is a daunting task, for which there does not exist yet a satisfying solution. In chapter 4, we propose two different solutions for this problem, i.e., abstraction and interactive analysis. To the best of our knowledge, no previous work is designed to be data-driven and a lot of tasks is left to the analyst. Notwithstanding, there are some works related to ours.

Interactive Analysis. Interactive analysis approaches [AYLT+15, BMH12, BKT+13, vL14] focus on learning the subjective measure in the mind of the analyst to guide the analysis. For example, OneClick [BKT+13] is a personalized interactive navigation approach that learns an interestingness function based on groups that were *liked* or *unliked* by the analyst in previous steps. In IUGA, we adopt an approach based on exploration or exploitation operations and let the analyst choose which operation to apply at each step. However, the ability to *personalize* the navigation as in OneClick is an interesting direction for future work.

In [BMH12], an interactive mechanism is proposed which functions in 3 steps; the algorithm first returns randomly k groups g_1, g_2, $\ldots g_k$; then the analyst provides binary feedback f_1, f_2, $\ldots f_k$ to indicate interest on each user group. Finally the algorithm iteratively updates its scoring function using a mechanism inspired from Query by Example so that the randomly selected groups align better with analyst's interest. This

work is interesting because the analyst can interact with the group space and gradually arrives to the set of groups she is interested in. In our future work, we plan to explore an adaptation of the feedback-based scoring to our framework.

In [XSMH06, BKS11], an approach is proposed to learn the model of prior knowledge of the analyst, based on her analysis actions. In [XSMH06], the analyst has to order her analysis preferences which puts more burden on the analyst. These methods are complementary to ours and could be leveraged to develop a navigation-aware analysis framework.

Making Interactive Mining Easy (MIME) [GMV11]. B. Goethals et al. propose an interactive pattern exploration framework called MIME where the analyst is able to explore and refine the discovered patterns on the fly. In MIME, the analyst becomes an essential part of the mining algorithm as she has to select the items to include in the pattern for further analysis. Also, there is no data-driven navigation as in our case. The analyst is left alone to make an educated choice.

Constraint-Based Mining. Approaches in [BGMP03, BGKW02, UBLC12] can be seen as a group analysis mechanisms where the analyst can iteratively tune the constraints and thresholds to generate additional groups to analyze. Two challenges regarding constraint-based approaches are as follows:

- Making constraints is not an easy task and requires a knowledge of the dataset;
- Formulating constraints is also a hard task and usually requires a knowledge of a querying language, as opposed to our data-driven work where in IUGA, relevant and diverse options are suggested to the analyst.

Our interactive navigation approach does not need any constraint to be formulated. At each step of the interaction, IUGA itself adds up constraints to the data to tune future options based on analyst's preferences.

Pruning. As we discussed earlier in Chapter 4, both our analysis approaches are lossless methods. Thus we have no interest to prune user groups in our approaches.

A number of interestingness measures are discussed in [MPsM96, GH06] to rank groups and facilitate pruning. We use relevance as an interesting measure in our interactive analysis work, but it is not pruning but constructing decreasing ordered list to reach better performance. Regarding abstraction, our *taxonomy-based usage*, is also an interestingness measure. The difference is that we calculate our measure for items in a user group and not necessarily for a whole group, as we do not prune a user group, but abstract parts of it.

The method used in [MGB09] is a top-down approach over ontologies and is the most similar work to our *abstraction* method. It provides a pruning technique to filter out discovered results, hence a *lossy* analysis.

Diversity. Diversity is the optimization objective for our interactive analysis as it gives alternative options for analysis. Diversity is a widely studied subject that finds its roots in Web search with a goal similar to ours. In [CG98], the concept of diversity in text retrieval and summarization is introduced to balance document relevance and novelty. Most diversity approaches fall into two categories:

- **Content-based** [CG98, JSH04, AK11]. Search results are diversified in a way to satisfy the application context.
- **Intent-based** [CJL+11]. Search results are diversified in a way to satisfy the analyst intent. An intent-based diversified search result will hopefully satisfy the information need of analysts who may have different intents.

The executed algorithm in each step of the interactive analysis process, i.e., GroupNavigation, is based on a greedy approach and exploits the content-based diversification.

Visualization. There exists many visualization mechanisms for user data analysis which are table-based, matrix-based, graph-based, grid-based, etc. The graphical user interface we introduced in Section 4.2.8, is a graph-based interface where groups are nodes and edges are made based on the relevance between groups. To the best of our knowledge, there is no previous work that investigates users in user groups as the main building block of visualization.

Few efforts in the literature combine visualization and interactive analysis [sLIC08, LA00]. Few examples are data mining suites like RapidMiner and Knime. These approaches develop a toolbox to manipulate and visualize groups according to preferences specified by the analyst at each step. These methods do not provide semantics for exploration or exploitation operations nor do they rely on an optimization framework to cover the group space.

6.3 User Group Recommendation

Online recommendation is a well studied problem since mid-1990s [SMC87, HSRF95]. Efforts in both industry and academia lead new approaches for recommender systems. The aim of a recommender system, as described in [AT05, Kon04], is to predict user's rating score for items she has not rated before, and return top-k items with highest

predicted rating scores. Typically, there exists two different approaches for online recommendation in the literature:

- **Item-based**. This approach is based on *item similarity* and leverages score for items which are similar to the user's previously highly rated items.
- **Collaborative filtering**. This approach is based on *user similarity* and leverages users in the network who have common interest with the user. We mentioned in Section 5.4 that we use collaborative filtering to obtain individual recommendations.

Group recommendation has been designed for various domains such as news pages [PDCCA05], tourism [GSO11], music [CBH02] and TV programs [YZHG06]. A group may be formed at any time by a random set of users with different interests, a number of persons who explicitly choose to be part of a group, by optimizing one or more quality objectives (like the output of our user group discovery algorithm) or by computing similarities between users with respect to some similarity functions and then clustering similar users together [NSNK12, AYRC+09].

There are two dominant strategies for group recommendations [BF10, AYRC+09]: *virtual user* and *recommendation aggregation*. The former creates a pseudo-user representing the group and then makes recommendations to that pseudo-user, while the second strategy computes a recommendation list for each group member and then combines them to produce a group's list. For the latter, a widely adopted approach is to apply an aggregation function to obtain a *consensus group preference* for a candidate item. While we adopt recommendation aggregation strategy, to the best of our knowledge, none of the existing functions account for the influence between group members.

In [Kor10], the notion of temporal dynamics in group recommendation is introduced: matrix factorization is used to model user biases, item biases, and user preferences over time. The assumption is that users' taste varies over time: for instance, a user may give a higher score to comedy movies than drama in July and give inverse scores in August. Hence time dimension is an important factor to consider in group recommendation. To the best of our knowledge, no previous work considers temporal dimension of user affinities in group recommendation.

Chapter 7

Summary and Perspectives

In this thesis, we introduced a framework for user group management which consists of discovering, analyzing and recommending to user groups. Section 7.1 concludes the thesis and Section 7.2 discusses some perspectives and future works.

7.1 Summary

In Chapter 1, we first discussed the importance of analyzing user groups instead of individuals to gain new insights and reduce the noise and sparsity in data. Examples in Section 1.3 illustrated different use-cases where such a framework is beneficial. We then mentioned three components which we consider in our framework: discovery, analysis and recommendation.

In *group discovery* (Chapter 3), we developed an approach to discover user groups by optimizing one or more quality dimensions. In Section 3.2 we mentioned that group discovery is a challenging task because of huge space of candidate set and conflicting objectives. Also we proved in Theorem 3.5 that our problem for group discovery is NP-hard. Then in Section 3.5 we proposed α-MOMRI, an approximation algorithm, and h-MOMRI a heuristic algorithm for group discovery. Our extensive set of experiments in Section 3.6 on MovieLens and BookCrossing datasets show that our approximation results in high quality groups and that our heuristic is very fast without compromising quality too much.

In *group analysis* (Chapter 4), we introduced two different approaches to analyze the large space of user groups, abstraction and interactive analysis. Abstraction is a primitive that reduces the size of user group space and helps to understand better the space. Our evaluations on two real datasets showed that abstraction reduces considerably the

size of the user group space. For interactive analysis, we introduced IUGA based on a simple and intuitive optimization formulation: find the k most diverse and relevant user groups to an input group. We proved the hardness of group analysis problem in Theorems B.1 and B.2. We proposed a greedy algorithm (Algorithm 4) for GROUP-NAVIGATION to help analysts navigate in the space of groups and reach one or several target users. The set of extensive experiments (Section 4.2.7) on both real and synthetic datasets showed the utility of relevance and diversity in group navigation and in finding users of interest in different navigation scenarios.

Finally in Chapter 5, we discussed one usage of user group discovery and analysis, i.e. *recommendation to user groups*. We introduced an approach for group recommendation that accounts for temporal affinities between group members. In Section 5.1, we mentioned why recommending to a group of users by considering temporal affinities is a challenging task. We also proved in Theorem 5.1 the hardness of our problem. Then in Section 5.3 we proposed GRECA, an efficient algorithm for temporal affinity-based group recommendation. Our extensive experiments with real Facebook users and MOVIELENS datasets assessed the high quality of temporal affinity-aware recommendations for groups with different characteristics.

7.2 Perspectives

We envision our future work in three different directions: *system*, *evaluation* and *applications*. In the *system* perspective (Section 7.2.1), we discuss general considerations and improvements in case of building a real system based on our proposed user group management framework. In the *evaluation* perspective (Section 7.2.2), we mention missing pieces in evaluating a user-group-based methodology. Finally in the *applications* perspective (Section 7.2.3), we elaborate applications of user group management other than recommendation to illustrate its broad applicability in the domain of the Social Web and other domains.

7.2.1 System Perspectives

We believe that three main concerns of building a user group management system are *quality*, *performance* and *integration*. We discuss each of the following concerns as follows.

Quality. A user group management system should return high quality results for any given input dataset. A very first step to insure quality, is to verify the quality of the raw

user data. Although user group analysis aims to reduce noise and sparsity in data (as discussed in Section 1.2), however based on GIGO[1] (Garbage-In Garbage-Out) principle, the analysis results of a nonsensical data is often nonsensical too. Thus a *cleansing* and *preparation* step is required before any group discovery or analysis occur. In [AIK+14] a preparation framework is proposed for the data on the social Web making it readily available for further processing. In [KIJ+15], a fast and scalable big data cleansing framework is proposed. Also the CHRONOS framework [LWT+12] provides a precise and evolution-tolerant history of users for recommendation. In our future work, we aim to plug such a component to our framework before discovering user groups.

Another aspect of quality in our system for a real application, is to appropriately tune algorithms' parameters for all three components, i.e., discovery, analysis and recommendation. We already provided in this thesis some advises regarding the way to choose parameter values, e.g., the value of k for IUGA (Section 4.2.7.1), the minimum size of user groups for α-MOMRI and h-MOMRI (Section 3.6.2.1), etc. The parameters' values of different algorithms heavily depend on the application and the input dataset. On the other hand, we have made many objective choices based on the statistical results we obtained in our datasets. However, different applications may interest in other quality dimensions which we have not investigated. In future, we plan to investigate other objectives partially listed below [GH06].

- **Conciseness**. A user group or a group-set is concise if it provides interesting information in the most succinct form possible. For instance, between groups [`Hitchcock movies`] and [`suspense`, `thriller`], the former is more concise as it provides the same message but in a more aphoristic form.
- **Reliability**. A user group or a group-set is reliable if it holds in most cases. The concept is similar to *coverage* and it is applicable to rules.
- **Peculiarity**. A user group or group-set is peculiar, if it is far from other groups which the analyst has already seen. The concept is similar to *diversity*, but it is the diversity between a group and all other groups already seen.
- **Novelty**. The collective set of previously seen groups provide a knowledge to the analyst regarding the user data. A user group or group-set is novel, if it is controversial to the actual knowledge of the analyst. It also bears some similarity with diversity and helps to observe different aspects of the data.

[1] *http://www.worldwidewords.org/qa/qa-gar1.htm*

In our future work, we aim to increase the number of objectives and adapt AETHERIS framework [SRPC11] by exploiting *skylineability* property to maintain efficiency in high-dimensional space of objectives.

Another aspect of quality is in using a feedback mechanism. In interactive analysis, we aim to integrate a more intelligent feedback mechanism that takes into consideration the history of analyst's choice. In [SNJ12], an analyst logging model is proposed to adapt sampling data to what the analyst desires to observe. Also in [BMH12] an algorithm is proposed to iteratively update its scoring function based on the analyst's feedback using a mechanism inspired from Query by Example so that the randomly selected user groups to present, align better with analyst's interest.

Performance. Sub-seconds execution is a necessity for a user group management system. We have always considered this concern in our contributions. In group discovery, we propose h-MOMRI, an efficient heuristic algorithm which is pretty fast in price of quality. In interactive analysis, we consider sub-seconds execution as a principle and consider a time limit parameter in our greedy algorithm. In group recommendation, we propose an instance optimal algorithm which makes many save-up in accesses. However, there exists room for performance improvement in our future work.

In group discovery, achieving a high performance in h-MOMRI is highly dependent of monotonicity property of coverage. However, monotonicity is a generic property which holds for many different functions. In future, we plan to discover more specific properties of our quality dimensions which potentially lead more pruning. For instance, for *diversity*, we can exploit set properties. We already know that a set of disjoint groups (i.e., they have no user in common) has the highest diversity and there exists an inverse relation between the size of the overlap and the diversity (see Equation 3.2). We can benefit from these properties by making pre-computed list of overlaps between group pairs to make the diversity improvement process more guided.

In abstraction as a group analysis method, the costly process is the preparation of hand-crafted taxonomies, which is application- and dataset-dependent. In future, we aim to use online ontologies like DBpedia[2]. The idea is to automatically find a subset of a generalization/specialization online ontology resource and use it as the taxonomy for abstraction. This reduces the burden on the system designer to manually craft taxonomies. Potentially a post-processing step is needed over an online ontology, e.g., to remove cycles by selecting one single parent for each child in the ontology.

In interactive navigation as a group analysis method, sub-second performance is in its core because there is no interactivity if the underlying algorithms are not fast. This

[2] *http://wiki.dbpedia.org*

is stressed out as a main principle for interactive analysis in [OTAYT15, NJ11]. To improve the performance of our interactive navigation approach, we aim two different solutions, *offline processing* and *distributed processing*. In our current contribution in [OTAYT15], we have investigate the former, but the latter is a part of future work.

- **Offline Processing**. Our GroupNavigation algorithm maximizes diversity among the most relevant groups to an input group. We consider relevance as the Jaccard similarity between a group pair. In an offline process, we compute for each user group, the amount of relevance with other groups. Then for each group, we store an inverted list which contains other groups in decreasing order of their relevance value (See Section 4.2.6). This offline computation makes GroupNavigation efficient.

  Two challenges regarding offline computation of inverted lists are as follows: (i). precomputed lists may consume lots of memory; (ii). what if in the middle of the interactive process, the analyst requests to change relevance measure to another function, e.g. Cosine instead of Jaccard. We already addressed the first challenge by considering a threshold for relevance value. If the relevance value between a group pair is lower than the threshold, it will not be materialized. The second challenge is a part of our future work where we plan to either pre-compute multiple inverted lists for each group or design an approach to efficiently reorder top-k elements in inverted lists based on a new requested measure.

- **Distributed Processing**. In [KJT+14], an architecture of distributed system for interactive analysis is proposed. Another effort of distributed processing in this area is the development of jLCM[3], a distributed implementation of LCM, a tool for efficient discovery of user groups. The most significant concern while thinking of a distributed architecture is to cluster data in a way that there exists least amount of shared information between clusters. It is a challenging task in our problem, as for computing inverted lists and maximizing diversity, we may potentially need to compare any pair of groups together. One idea could be to cluster user groups using a distance measure on their attributes and map each group cluster to a data cluster. This idea is based on an observation that there exists groups which are very far from each other, e.g., young females in France and old males in United States. More specifically, we can consider community detection techniques to obtain group communities with maximum average similarity inside each community and minimum average similarity outside (i.e., modularity) [NG04].

[3] *http://slide-lig.github.io/jlcm/*

In group recommendation, performance is a function of two following elements: *stopping condition* and *number of lists*. Although some efforts is done to make top-k algorithms stop faster [APV07], but they are mainly designed for TA algorithms and a system designer with GRECA i.e., an NRA-like algorithm does not have any control on the former. In worst case the GRECA algorithm may scan the entire lists. But for the latter, having less lists leads less accesses. In [RAYC+10], a partial materialization strategy is developed which identifies which subset of lists to materialize in order to maximize space reduction and minimize processing time. In future, we aim to investigate different aggregation functions or partial metallization techniques to merge similar lists together or choose representatives to gain some save-up.

Integration. Having three efficient components in user group management framework, the integration of these together should be in a way to maintain their efficiency. The framework should integrate components in a way that they remain independent, but at the same time they are connected together strongly. Independence of each component is important, so we insure that any other approach can replace any of the components.

To have a continuous flow between components, there should be a widely-accepted standard for data exchange. Usually in user data analysis, a time-consuming step is to convert user data or analytical results from one format to another to be readable for another algorithm. The way we store user groups and other complementary information should be common and known for all three components. On the other hand, our components should be flexible enough to more than one commonly known standards. Parts of our future work is to come up with such a standard for our framework. In group recommendation, the input is not only user groups but also the temporal affinities between group members. Thus the standard should be designed in a way to cover the exchange of complementary information for a user group such as their temporal affinities.

7.2.2 Evaluation Perspectives

The extensive set of performance and quality experiments are already presented for our contributions (Sections 3.6, 4.1.4, 4.2.7 and 5.4). However, in our future work, we plan to improve our experiments in following directions.

User Study. A user study often consists of two parts. In the first part, the results of the approach is shown to participants and they are asked how informative and useful they find the results (independent user study). In the second part, the results are shown in comparison with results of other approaches and participants are asked which approach they prefer (comparative user study). But the main question which remains unanswered in most of the user studies is the following: *is an analyst able to use and benefit from*

the approach in practice? Parts of our future work is to refine our user studies using a larger population on an online framework like AMT[4] or Crowd4U[5], and considering the aforementioned question.

In group discovery, we aim to use questions with *statistically known answers* (SNA), i.e. statistical facts like *the old generation like Western genre movies more than the young generation.* These SNAs can be obtained using polls in related forums or demographic break-down reports in IMDb. We envision the following context for our user study: we consider an SNA S about an input set of ratings R. We execute our α-MOMRI and h-MOMRI algorithms for R. Then we ask a *yes/no* question from the participant about the correctness of S by showing the results of our algorithms to the participant. The participant decides based on the algorithm results. A correct answer could potentially mean that the algorithm results are useful for analysis in practice.

In group analysis, we plan to pose different scenarios for participants and see how much they are successful in fulfilling a defined task in each scenario. Success can be measured by different variables, e.g., session time in minutes, number of steps, etc. We already illustrated one such scenario in Section 4.2.7, i.e., PC selection. In the future, we plan to conduct a large scale user study by considering following complimentary scenarios.

- Given a researcher u, find the best researcher u' that can best collaborate with u.
- Given a researcher u, find k different keywords for u in k different venues.
- Given a conference c, find k themes (keywords) which differ entirely with c's theme.
- Given a researcher u, find the evolution of u's work in periods of two years.

Concerning group recommendation, recently, in [OCKR01b], a user study is reported where MovieLens users formed groups by inviting each other. A recommender system predicts ratings using *least misery* aggregation function. User satisfaction was measured by following set of criteria: "how easy the process of creating groups was", "how easy adding members to a group is" and "how useful group recommendations were". The study resulted that group members prefer group recommendations than the individual recommendations. We aim to adapt the same criteria proposed in this work and improve our user study by considering each participant in different groups, like *family* groups, *friend* groups and *unknown* groups. This study will help us better understand the relation between the behavior of participants and the group formation parameters (Section 5.4.1.3).

[4] *Amazon Mechanical Turk: https://www.mturk.com/*
[5] *https://crowd4u.org/*

Performance Study. Group discovery and group recommendation are *unsupervised* algorithms and their performance are easily measured by reporting their time and space consumption. But in case of interactive navigation as a group analysis method, being a *semi-supervised* algorithm makes its performance measurement challenging. It is mainly because the analyst is in the execution loop and human factors (e.g., analyst expertise) certainly influence the overall performance. This is a challenge that has been remained unsolved in interactive data analysis community.

In this thesis, we proposed ATA measure. The idea is to remove the analyst from the execution loop to neutralize human factors and replace her with all possible choices that an analyst can make. ATA value of two different interactive analysis methods can be compared together to see which method can achieve a target faster. The only problem associated to ATA measure is that it considers there exists some target groups in advance in the group space. This does not exactly capture the reality. Although these targets are not given to the algorithm as input, but this assumption in nature makes a bias for measuring performance. In future, we plan to improve our ATA measure in a way that it becomes target-independent.

7.2.3 Applications' Perspectives

User group management has many different applications in different domains. In Chapter 5, we discussed one of its applications, i.e., user group recommendation. In Chapter 1, we illustrated different scenarios where user group analysis instead of analyzing individual users is beneficial. Other applications are also discussed in the literature (e.g., [LDG+15]). In this section, we briefly introduce two other applications for user group management, i.e., *workforce organization in knowledge-intensive crowdsourcing* and *interactive search engine*.

Workforce Organization in Knowledge-Intensive Crowdsourcing. Crowdsourcing is the process of getting work done online by a crowd of people. Knowledge-intensive crowdsourcing is the collaborative creation of knowledge content (e.g., Wikipedia articles) through crowdsourcing. Crowd workers, each having a certain degree of expertise, collaborate to fulfill together a task. In [RLT+15], an optimized task-assignment strategy is proposed for knowledge-intensive crowdsourcing. The main motivation of exploiting user group management in knowledge-intensive crowdsourcing is illustrated in the following example.

Example 7.1 (French Sub-titles for Titanic)**.** *We are given the task of providing French sub-titles for the movie Titanic. This task needs following skills: (i). English voice-to-text transformation, (ii). English editing, and (iii). English-to-French translation. A*

task-assignment algorithm like the one in [RLT+15] associates following workers with this task: John, Mary and Peter. These three workers are experts in skills (i) *to* (iii) *and they form a user group* g_1*. Once they plan to start the task, Peter notifies that he is not available. In case a worker is not available, to be able to get the task done, an alternative user group should be formed. In our example, we consider an alternative group* g_2 *with members John, Mary and Sara. This group is the most similar to* g_1 *where Sara replaces Peter. Sara is available, but she has only 80% of expertise in the skill* (iii)*. The group* g_2 *has many users in common with* g_1*, with more availability while less quality (in the context of skills).*

User group management can be used to discover alternative groups in knowledge-intensive crowdsourcing. The aim is to find best alternative worker groups for a given task. The set of best alternative worker groups can form a path starting from the most qualified group to the least qualified group.

Interactive Search Engine. A search engine is not usable if the analyst does not how to formulate her request or if the request in her mind is not yet precise. An interactive analysis mechanism guides the analyst in a step-by-step process towards her goal. User group management is beneficial to first discover user groups on a user data which records navigated webpages as user activities. The analyst can navigate through these user groups to achieve a target. The main motivation of exploiting user group management as a search engine is illustrated in the following example.

Example 7.2 (Finding a Lost Music)**.** *Amanda wants to find the webpage dedicated to a French music which is a rehearsal of an older Russian music "Those were the days" by Dorogoy Dlinnoyu. She doesn't find the French version neither by Youtube recommendations, nor Shazam*[6] *music detection nor Google relevant search results for the Russian name and the music title. The only way is to go through an interactive process to explore and exploit relevant groups of music pages around the original Russian music to gradually reach the target music, i.e., "Le Temps des Fleurs" by Iolanda Cristina Gigliotti (Dalida).*

Recently an interactive search engine is proposed by Santosh Gangwani[7] based on Yahoo search results. The main idea of this search engine is to *explore beyond search*. For an input set of keywords, it groups Yahoo search results and the analyst can then delve into each group. Though, there is only one level of interactivity: the analyst can delve into a group, come back and delve to another group. Group descriptions in this tool are not necessarily meaningful. Plus they are not diverse as they have overlaps

[6] *http://www.shazam.com*
[7] *http://www.searchgui.com*

(e.g., “citations” and “publications” appear as two distinct groups while searching for a researcher’s name.) Nevertheless, the engine is very fast and returns results on-the-fly.

As a summary, one future application for our framework could be to design a search engine based on user navigation activities.

Appendix A

Optimality and Near-Optimality Proofs

In this appendix, we prove that all our objectives (diversity, coverage and rDistb) discussed in Chapter 3 satisfy optimality (POO) and near-optimality (PONO) principles. In all of the following theorems, we consider two group-sets G and G' and two sub group-sets G_1, G_2 for G and G'_1, G'_2 for G' such that:

- $G_1 \cup G_2 = G$, $G_1 \cap G_2 = \emptyset$;
- $G'_1 \cup G'_2 = G'$, $G'_1 \cap G'_2 = \emptyset$;
- $|G| = |G'|$, $|G_1| = |G'_1|$ and $|G_2| = |G'_2|$;
- User groups in $\{G, G', G_1, G_2, G'_1, G'_2\}$ are distinct, i.e., each group cannot appear more than once in a group-set;
- $\forall g_1 \in G_1 \wedge g_2 \in G_2, g_1 \nsubseteq g_2 \wedge g_2 \nsubseteq g_1$;
- $\Sigma_{g_1 \in G_1, g_2 \in G_2} |g_1 \cap g_2| \leq \Sigma_{g'_1 \in G'_1, g'_2 \in G'_2} |g'_1 \cap g'_2|$.

Theorem A.1. *Based on Equation 3.2, diversity satisfies POO.*

Proof. Given a group-set G and a set of rating records R, diversity is defined by this formula: $diversity(G, R) = 1/(1 + \Sigma_{g,g' \in G} |r \in R, r \lessdot g \wedge r \lessdot g'|)$. In the formula, the part "$\Sigma_{g,g' \in G} |r \in R, r \lessdot g \wedge r \lessdot g'|$" computes the amount of overlap and we will use the notation ov_G to denote this part. Thus $diversity(G, R) = 1/(1 + ov_G)$. Obviously whenever ov_G increases, $diversity(G, R)$ decreases. Thus we transform the POO implication to $ov_{G_1} \leq ov_{G'_1} \wedge ov_{G_2} \leq ov_{G'_2} \rightarrow ov_G \leq ov_{G'}$. It is obvious that larger overlaps in G'_1 and

G'_2 compared to G_1 and G_2 lead a larger overlap in G' compared to G. It is true only if there is no overlap between sub group-sets. □

Theorem A.2. *Based on Equation 3.1, coverage satisfies POO.*

Proof. Recall that coverage is defined by this formula: $coverage(G, R) = |\cup_{g \in G} (r \in R, r \lessdot g)|/|R|$. We prove this theorem based on following facts:

- Based on Theorem 3.10, $coverage(G, R)$ is a monotone function;
- $coverage(G_1 \cup G_2, R) = coverage(G, R)$;

Then the left part of the POO implication $coverage(G_1, R) \geq coverage(G'_1, R) \wedge coverage(G_2, R) \geq coverage(G'_2, R) \rightarrow coverage(G, R) \geq coverage(G', R)$ can be transformed to $coverage(G_1 \cup G_2, R) \geq coverage(G'_1 \cup G'_2, R)$ and then $coverage(G, R) \geq coverage(G', R)$, i.e., the right part of the implication, hence, the proof. □

Theorem A.3. *Based on Equation 3.3, rDistb satisfies POO.*

Proof. Recall that the rating distribution objective is defined by this formula: $rDistb(G) = avg_{g \in G}(max_{r \in g}(r.s) - min_{r' \in g}(r'.s))$. We consider *homogeneity* in this proof. The proof can be simply extended for other rating distributions. For simplicity, we convert the formulation to the following form: $rDistb(G) = avg_{g \in G}(diameter(g))$.

Step 1. The left part of the POO implication $rDistb(G_1, R) \leq rDistb(G'_1, R) \wedge rDistb(G_2, R) \leq rDistb(G'_2, R)$ is then equal to $avg_{g \in G_1}(diameter(g)) \leq avg_{g \in G'_1}(diameter(g)) \wedge avg_{g \in G_2}(diameter(g)) \leq avg_{g \in G'_2}(diameter(g))$. It can be transformed to $(avg_{g \in G_1}(diameter(g)) \times |G_1|) \leq (avg_{g \in G'_1}(diameter(g)) \times |G_1|) \wedge (avg_{g \in G_2}(diameter(g)) \times |G_2|) \leq (avg_{g \in G'_2}(diameter(g)) \times |G_2|)$ (i.e., multiplying a constraint to both parts of inequalities).

Step 2. As *summation* is a monotone function, we merge two parts of the conjunction to obtain the following: $(avg_{g \in G_1}(diameter(g)) \times |G_1|) + (avg_{g \in G_2}(diameter(g)) \times |G_2|) \leq (avg_{g \in G'_1}(diameter(g)) \times |G_1|) + (avg_{g \in G'_2}(diameter(g)) \times |G_2|)$ and then $((avg_{g \in G_1}(diameter(g)) \times |G_1|)/|G|) + ((avg_{g \in G_2}(diameter(g)) \times |G_2|)/|G|) \leq ((avg_{g \in G'_1}(diameter(g)) \times |G_1|)/|G|) + ((avg_{g \in G'_2}(diameter(g)) \times |G_2|)/|G|)$ (i.e., dividing the whole inequality by $|G|$).

Step 3. Recall $|G| = |G'|$ then $((avg_{g \in G_1}(diameter(g)) \times |G_1|)/|G|) + ((avg_{g \in G_2}(diameter(g)) \times |G_2|)/|G|) \leq ((avg_{g \in G'_1}(diameter(g)) \times |G_1|)/|G'|) + ((avg_{g \in G'_2}(diameter(g)) \times |G_2|)/|G'|)$. Based on the definition of average function, the expression is equal to

$avg_{g \in G}$ $(diameter(g) \leq avg_{g \in G'}(diameter(g))$, i..e, the right part of the formula, hence the proof. □

Theorem A.4. *Based on Equation 3.2, diversity satisfies PONO.*

Proof. We reuse the notation we introduced in the proof of Theorem A.1 and define $diversity(G, R) = 1/(1 + ov_G)$. The PONO implication for diversity based on ov_G is $ov_{G_1} < (ov_{G'_1} \times \alpha) \wedge ov_{G_2} < (ov_{G'_2} \times \alpha) \rightarrow ov_G < ov_{G'} \times \alpha$. As *summation* is a monotone function, then we can transform the left part of the implication to $ov_{G_1} + ov_{G_2} < (ov_{G'_1} \times \alpha) + (ov_{G'_2} \times \alpha) \Rightarrow ov_{G_1} + ov_{G_2} < \alpha \times (ov_{G'_1} + ov_{G'_2}) \Rightarrow diversity(G_1, R) + diversity(G_2, R) \geq \alpha(diversity(G'_1, R) + diversity(G'_2, R)) \Rightarrow diversity(G, R) \geq diversity(G', R) \times \alpha$, hence the proof. □

Theorem A.5. *Based on Equation 3.1, coverage satisfies PONO.*

Proof. Recall that coverage is defined by this formula: $coverage(G, R) = | \cup_{g \in G} (r \in R, r \lessdot g)|/|R|$. We follow the same facts we discussed in the proof of Theorem A.2. The PONO principle for coverage is $coverage(G_1, R) \geq coverage(G'_1, R) \times \alpha \wedge coverage(G_2, R) \geq coverage(G'_2) \times \alpha \rightarrow coverage(G) \geq coverage(G') \times \alpha$. The left part of the PONO implication can be transformed to $coverage(G_1 \cup G_2, R) \geq coverage(G'_1 \cup G'_2, R) \times \alpha$ and then $coverage(G, R) \geq coverage(G', R) \times \alpha$, i.e., the right part of the implication, hence, the proof. □

Theorem A.6. *Based on Equation 3.3, rDistb satisfies PONO.*

Proof. Recall that the rating distribution objective is defined by this formula: $rDistb(G) = avg_{g \in G}(max_{r \in g}(r.s) - min_{r' \in g}(r'.s))$. We consider *homogeneity* in this proof. The proof can be simply extended for other rating distributions. For simplicity, we convert the formulation to the following form: $rDistb(G) = avg_{g \in G}(diameter(g))$.

Step 1. The left part of the PONO implication $rDistb(G_1, R) \leq (rDistb(G'_1, R) \times \alpha) \wedge rDistb(G_2, R) \leq (rDistb(G'_2, R) \times \alpha)$ is then equal to $avg_{g \in G_1}(diameter(g)) \leq (avg_{g \in G'_1}(diameter(g)) \times \alpha) \wedge avg_{g \in G_2}(diameter(g)) \leq (avg_{g \in G'_2}(diameter(g)) \times \alpha)$. It can be transformed to $(avg_{g \in G_1}(diameter(g)) \times |G_1|) \leq (avg_{g \in G'_1}(diameter(g)) \times |G_1| \times \alpha) \wedge (avg_{g \in G_2}(diameter(g)) \times |G_2|) \leq (avg_{g \in G'_2}(diameter(g)) \times |G_2| \times \alpha)$.

Step 2. As *summation* is a monotone function, we merge two parts of the conjunction to obtain the following: $(avg_{g \in G_1}(diameter(g)) \times |G_1|) + (avg_{g \in G_2}(diameter(g)) \times |G_2|) \leq (avg_{g \in G'_1}(diameter(g)) \times |G_1| \times \alpha) + (avg_{g \in G'_2}(diameter(g)) \times |G_2| \times \alpha)$ and then $((avg_{g \in G_1}(diameter(g)) \times |G_1|)\ /|G|) + ((avg_{g \in G_2}\ (diameter(g)) \times |G_2|)\ /|G|) \leq (\alpha \times ((avg_{g \in G'_1}\ (diameter(g)) \times |G_1|)\ /|G|) + ((avg_{g \in G'_2}\ (diameter(g)) \times |G_2|))\ /|G|)$.

Step 3. Recall $|G| = |G'|$ then $((avg_{g \in G_1}\ (diameter(g)) \times |G_1|)\ /|G|) + ((avg_{g \in G_2}\ (diameter(g)) \times |G_2|)\ /|G|) \leq (\alpha \times ((avg_{g \in G'_1}\ (diameter(g)) \times |G_1|) + (avg_{g \in G'_2}(diameter(g)) \times |G_2|))\ /|G'|)$. Based on the definition of average function, the expression is equal to $avg_{g \in G}\ (diameter(g) \leq \alpha \times avg_{g \in G'}\ (diameter(g))$, i..e, the right part of the formula, hence the proof. □

Appendix B

NP-Hardness Proofs of GroupNavigation Problem

We consider an infinite time limit in our proofs since that does not affect the complexity of our problem.

Theorem B.1. *The exploration version of the* GroupNavigation *Problem is NP-complete.*

Proof. The decision version of the problem is as follows: For a given group g, a set of groups $\mathcal{G}$ and a positive integer k, an overlap threshold μ, is there a subset of groups $\mathcal{G}' \subseteq gExplore(g, \mathcal{G}, \mu)$ such that (i) $g' \in \mathcal{G}' \wedge g' \neq g \wedge overlap(g, g') \geq \mu$ and (ii) $\Sigma_{(g_1,g_2)\in\mathcal{G}'|g_1\neq g_2}(1 - overlap(g_1, g_2))$ is maximized. A verifier v which returns true if both conditions (i) and (ii) are satisfied runs in polynomial time in the length of its input.

To verify NP-completeness, we reduce the Maximum Edge Subgraph (Mes) [FKP01] (also known as *Dense k-subgraph*) to the decision version of our problem. The problem of Mes is defined as follows. Given an instance I consisting of a graph $G = (V, E)$, a weight function $w : E \to \mathbb{N}$, and a positive integer k, find a subset $V' \subseteq V$, $|V'| = k$ such that the total weight of the edges induced by V', i.e., $\Sigma_{(v_i,v_j)} w(v_i, v_j)$ (where $(v_i, v_j) \in V' \times V'$) is maximized. This is an NP-complete problem [FKP01] (originally reduced from the *Clique* problem).

Given I, we create an instance J of our problem as follows. J consists of a graph $G = (V, E)$ where the set of vertices $V = gExplore(g, \mathcal{G}, \mu)$ are groups that satisfy (i). Every pair of groups $(g_1, g_2) \in V \times V$ is also connected with a labeled edge *i.e.* $w(g_1, g_2) = 1 - overlap(g_1, g_2)$. The subset $V' \subseteq V$ ($|V'| = k$) is then a subset of groups

where the sum of the weights between each pair of groups in V' is maximized *i.e.*, $|E(V')| = \frac{k\times(k-1)}{2}$. The set V' is the most diverse subset of $\mathcal{G}$ that satisfies the overlap condition ($\forall g' \in \mathcal{G}, overlap(g, g') \geq \mu$). Therefore a set V' is a solution in instance I of MES *iff* it is a solution in instance J of our problem. Hence, the exploration problem is NP-complete. □

Theorem B.2. *The exploitation version of the* GROUPNAVIGATION *Problem is NP-complete.*

Proof. Similarly to the exploration version, a verifier v for exploitation runs in polynomial time in the length of its input. To verify NP-completeness, we reduce the MAXIMUM COVERAGE PROBLEM [Joh73] to the decision version of our problem. The problem of MAXIMUM COVERAGE PROBLEM (MCP) is defined as follows. Given an instance I consisting of m sets $S = \{ S_1 \dots S_m\}$ where $S_i \in S_M$ (S_M being a reference set), and a positive integer k, find a subset $S' \subseteq S$, such that $|S'| = k$ and the number of covered elements in S_M, i.e., $| \cup_{S_i \in S'} S_i|/|S_M|$ is maximized. This is an NP-complete problem [Joh73]. Given I, we can create an instance J of our problem which consists of m sets $S = gExploit(g, \mathcal{G}, \mu)$ and a reference group, i.e., $S_M = g_{in}$. In *opExploit()*, we are interested to have k groups $S' \subseteq S$ that cover maximum number of users in S_M, i.e., $| \cup_{S_i \in S'} S_i|/|S_M|$ is maximized. Therefore a set S' is a solution in instance I of MCP *iff* it is a solution in instance J of *opExploit()*. The exploitation version of the GROUPNAVIGATION Problem is hence NP-complete. □

Appendix C

Thesis Reviews

AT&T Labs-Research
Shannon Laboratory

Room 4C202B
1 AT&T Way
Bedminster, NJ 07921
908-901-2077 (phone)
divesh@research.att.com

October 5, 2015

Assessment of the Doctoral Dissertation "Optimization-based User Group Management: Discovery, Analysis, Recommendation" submitted by Mr. Behrooz Omidvar-Tehrani

Mr. Behrooz Omidvar-Tehrani's dissertation is based on the premise that **user data** (characterized by a combination of user demographics, such as age, gender, race, etc., and user activities, such as rating a movie, posting on a blog, etc.) are now widely available in many domains (such as the social web), and analysis of user data is important for many applications (such as recommendations) in these domains, but existing analysis tools (such as OLAP, visualization tools, etc.) are inadequate for the analyses that need to be supported on user data. The thesis proposes to develop techniques to support such user data analyses, by focusing attention on **user groups** (characterized as a set of users whose members have common demographics or common activities) instead of on individual users. The rationale is that doing group-level analysis will reduce the amount of sparsity and variability in the data, and produce more robust, insightful analytical results. The dissertation is well-structured and easy to read, and the proposed solutions considerably enhance the state of the art in the field of **user data management**. I will now comment on each chapter separately, summarize the contributions and provide my recommendation at the end of my report.

Chapter 1 motivates (i) the need for novel techniques and tools to support user data analysis in the age of data-driven research, and (ii) the potential benefits of focusing attention on user groups. In support of the first point, desirable characteristics of data analysis methodologies are presented (such as interactive, easy to use, etc.), and user data are distinguished from machine data along the dimensions of granularity and modularity. The second point is motivated using illustrative examples in three different domains: online advertising, social sciences and the social web, and the three components of user group management (user group discovery, user group analysis, and the use of identified groups) are presented. The chapter concludes with a summary of the thesis contributions.

Chapter 2 presents key concepts needed to understand the technical contributions of the thesis, including (i) a definition of the model for user data, and (ii) a description of the real user data sets used for illustrative examples and experiments in the thesis. The model for user data is formalized as a database of triples <u, c, i>, representing an action c

(such as rated, tagged, etc.) performed by user u on item i, with attributes associated with users, items and actions. User groups are defined as subsets of users based on common user attributes or common user activities, and the set of all possible user groups is organized as a lattice. Here, the candidate shows his ability to formalize important concepts, and also explain them intuitively using illustrative examples. Then, the five real user data sets used in the thesis (movie reviews from MovieLens, book reviews from BookCrossing, mobile usage from Nokia, data management researchers from DBLP, and a social network from Facebook) are described in detail. The chapter concludes with a summary of the characteristics that distinguish the different user data sets from each other.

The next three chapters present the innovative technical contributions of the dissertation: user group **discovery**, user group **analysis**, and user group **recommendation**, which is an illustrative use of the identified user groups.

Chapter 3 focuses on user group discovery, that is, the problem of obtaining high quality user groups from raw user data by optimizing one or more quality dimensions (such as coverage, diversity, etc.). Desirable qualities are separated into local qualities (satisfied by individual user groups) and global qualities (collectively satisfied by the set of user groups), and the group discovery problem is formally presented. The key challenge is to **quickly** identify high quality user groups, and the argument is made that this is hard for two reasons: the large space of user groups, and the difficulty of achieving coverage and diversity at the same time. More formally, the decision version of the group discovery problem is shown to be NP-complete. This chapter then presents two efficient algorithms for the group discovery problem. The first algorithm is an **approximation** algorithm α-MOMRI, which is based on the Principle of Near-Optimality, exploits a dynamic programming approach to efficiently maintain all non α-dominated plans, and provide a theoretical guarantee on the quality of the results. The second algorithm is a **heuristic** algorithm h-MOMRI, which quickly returns a subset of the Pareto set, but does not provide any approximation guarantee. The algorithms are empirically compared on the quality of the returned groups and the scalability of the algorithms using the MovieLens and BookCrossing data sets. The key results are that (i) α-MOMRI results in high quality groups, and (ii) h-MOMRI is very fast (since it produces an order of magnitude fewer solutions than α-MOMRI) without compromising quality too much (almost half the solutions of h-MOMRI are as good as those returned by α-MOMRI).

Chapter 4 considers the problem of user group analysis, which is needed to tackle the problem of information overload in the output of the user group discovery step (which can contain millions of user groups). Two **lossless** analysis methods, abstraction and navigation, are proposed and evaluated, since it is argued that lossy analysis methods may prune one or more interesting user groups with high likelihood. The first method proposed in this chapter, **abstraction**, summarizes user groups based on generalization/specialization taxonomies on user attributes, items, etc. An abstraction analysis primitive is proposed, which relies on a taxonomy-based usage measure, operates on a single user group at a time and reduces the size of its label. The proposed abstraction primitive is empirically evaluated on the Nokia and MovieLens data sets, based on the measures of abstraction volume and user group space reduction. The results show that abstraction reduces the size of the user group space and produces more understandable groups. The second method proposed in this chapter, **navigation**,

is an interactive user group analysis (IUGA) framework that navigates through the space of user groups to reach a subset of interesting groups. IUGA is based on the GroupNavigation problem that finds the *k* most diverse and relevant user groups to an input group, and relies on two group navigation primitives:group exploration and group exploitation. The GroupNavigation problem is shown to be NP-complete for both primitives, and greedy algorithms are presented to help analysts reach target users. An experimental evaluation aims to validate the usability and efficiency of the interactive analysis, and the quality of discovered groups in each step, using the DBLP and MovieLens data sets. IUGA is empirically shown to be better than its competitors. The two proposed methods are complementary, and abstraction can reduce the size of the user group space, making it more manageable for subsequent interactive analysis.

Chapter 5 investigates the use of the user groups, specifically for the problem of user group recommendation, which is one of many possible uses of the user groups that the user group analysis step has found to be of interest. Group recommendation differs from individual user recommendation in that the objective is to find the best items that a set of users will appreciate together. The candidate focuses on the **novel** problem of exploring how (i) affinity between group members and (ii) its evolution over time affect group recommendations. This work is based on the premise that each user will have a relative preference for an iterm depending on her temporal affinity with other group members. To address the key challenges for formalizing the semantics of relative preference, an efficient algorithm GRECA, which adapts the family of Threshold Algorithms to account for affinities between user pairs that evolve over time, is developed. A key novelty of GRECA is its ability to terminate with the correct top-k itemset based only on examining the items in the buffer. An experimental study is conducted using the Facebook and MovieLens data sets to evaluate effectiveness and efficiency. The results demonstrate the high quality of temporal affinity-aware recommendations for groups with different characteristics, and the scalable performance achieved by GRECA for ad hoc groups.

Chapter 6 is dedicated to prior work related to the three technical contributions investigated in Chapters 3-5. It is quite broad, and is evidence of the candidate's good understanding of related areas. The related work on user group discovery explores methods for clustering, community detection, frequent pattern mining and team formation. The related work on user group analysis explores the literature on interactive analysis, constraint-based mining and visualization mechanisms. The related work on user group recommendation describes both individual user recommendation techniques and prior group recommendation techniques. The candidate contrasts the solutions presented in the dissertation with related work to convincingly argue for their novelty.

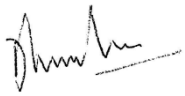
Chapter 7 summarizes the work and discusses future directions. The future work section is well organized along the perspectives of system, evaluation and applications. The **system** perspective discusses the main concerns of quality (including raw data quality), performance (including offline processing and distributed processing) and integration of the different proposed components, in building a user group management system. The **evaluation** perspective discusses the need for large scale user studies and improving the ATA measure for a more realistic performance study. The **applications** perspective identifies two additional applications beyond user group recommendation that could benefit from user group management: workforce organization in knowledge-intensive crowdsourcing and an interactive search engine.

To summarize, Behrooz Omidvar-Tehrani's dissertation is a novel and timely contribution in the field of user data management that will gain more importance as the Data Science community moves beyond Quantified-Self and embraces the Quantified-Us movement. The candidate has made a considerable effort in understanding all aspects of user group data management, ranging from user group discovery and user group analysis to uses of the identified user groups. The consistent use of multiple real data sets in empirically evaluating the proposed techniques, and the obvious effort in providing a number of illustrative examples to ensure that the proposed techniques can be easily understood, are laudable. While a real prototype system incorporating the various components is lacking, this is largely compensated by the extensive experimental studies that the candidate has performed on a variety of real data sets, demonstrating both efficiency and effectiveness of the proposed techniques.

For all those reasons and given his publication record, I am fully in favor of the candidate defending his dissertation.

If you have any questions, please feel free to contact me.

Yours sincerely,



Divesh Srivastava
Executive Director
Database Management Research
AT&T Labs-Research

Université de Caen Normandie
GREYC - CNRS UMR 6072

Groupe de REcherche en Informatique, Image, Automatique et Instrumentation de Caen
Sciences 3 - Campus Côte de Nacre
CS 14032 F-14032 Caen cedex 5 France
Téléphone : 02 31 56 73 32

Rapport sur le mémoire de thèse

Optimization-based User Group Management : Discovery, Analysis, Recommendation

présenté par Behrooz Omidvar-Tehrani
pour obtenir le diplôme de Docteur de l'Université Grenoble Alpes
spécialité « informatique »

Dans le contexte émergent de la science des données, si l'aspect passage à l'échelle de certains traitements classiques est de mieux en mieux maîtrisé, de très nombreux problèmes restent ouverts comme ceux de l'hétérogénéité des données et le traitement de données structurées (encore parfois appelées données complexes). C'est dans ce contexte que se situe le travail de Behrooz Omidvar-Tehrani qui porte sur l'étude de données liées aux activités d'utilisateurs, c'est-à-dire de données qui rassemblent des informations socio-démographiques des utilisateurs et leurs activités. Compte-tenu du développement des usages du numérique dans les activités humaines, il s'agit d'un champ de recherche particulièrement actif et compétitif.

Ce travail a l'originalité d'associer plusieurs domaines (fouille de données, optimisation multi-critère, approche interactive avec l'utilisateur, recommandation). Il s'agit d'une direction de recherche particulièrement fructueuse, qui permet de concevoir des méthodes de fouille qui ne retournent pas en bloc un nombre prohibitif de motifs et d'enrichir les méthodes de recommandation de la connaissance encapsulée dans les données liées aux activités d'utilisateurs. Le fil conducteur de la thèse est la conception d'un cadre de gestion de groupes d'utilisateurs qui associe la découverte de ces groupes, leur analyse et une application (la recommandation fondée sur les groupes d'utilisateurs). Ces trois thèmes sont successivement étudiés même si les résultats qui sont présentés ici et associés à chacun de ces thèmes ne sont pas toujours liés entre eux. Comme nous allons le voir, cette thèse contient plusieurs contributions importantes et de qualité sur chacun de ces thèmes.

La thèse, composée de 7 chapitres, est rédigée en anglais. Elle est bien écrite et est agréable à lire. Son ossature générale est plus proche de celle d'un article de recherche que celle usuellement rencontrée dans une monographie (l'état de l'art est résumé au chapitre 6). Cette structuration permet de mieux faire ressortir le positionnement de l'auteur par rapport à l'état de l'art mais certains choix effectués par l'auteur au cours des différents chapitres seraient plus fortement justifiés si ce positionnement était donné plus tôt. Les chapitres 3 à 5 sont dédiés aux contributions de l'auteur, ils sont judicieusement complétés par deux annexes donnant les preuves d'importants résultats formels obtenus par l'auteur.

Le chapitre 1 survole les grands sujets abordés dans cette thèse, à savoir l'exploration de données et les caractéristiques des données utilisateur ainsi que leur exploitation. Puis, les contributions de la thèse sont brièvement résumées.

Le chapitre 2 commence par introduire les notations et les principales notions qui sont utilisées par la suite. Il présente ensuite 5 jeux de données qui forment le support des expérimentations menées dans ce travail. Ces jeux ont des origines et des buts différents (certains sont publics, d'autres ont été obtenus en collectant les données sur le web ; certains correspondent à des évaluations d'articles de loisirs, d'autres à des comportements d'utilisateurs de téléphone ou de réseaux sociaux). J'ai apprécié le soin apporté à traiter des données variées et les résultats expérimentaux présentés dans la suite du travail tirent profit de la diversité de ces jeux de données.

Les chapitres 3 à 5 décrivent les contributions réalisées sur la découverte de groupes d'utilisateurs, leur analyse et leur recommandation. Le chapitre 3 est consacré à la méthode de découverte d'ensembles de groupes optimisant simultanément leur qualité suivant plusieurs dimensions. Une originalité de cette méthode est de s'intéresser aux groupes, non pas de façon individuelle, mais suivant des ensembles de groupes. Les 3 principales dimensions étudiées (couverture, diversité, scores) sont ainsi définies sur un ensemble de groupes. Ce positionnement rend le problème plus difficile car la combinatoire sur un ensemble de groupes est bien plus élevée que sur les groupes considérés individuellement. Cette démarche (considérer des ensembles de groupes) se situe dans la direction assez récente des travaux sur les "pattern sets" (ou ensembles de motifs) en fouille de données pour lesquels l'intérêt d'un motif dépend des autres motifs de l'ensemble. Assez curieusement, j'ai trouvé que cette originalité n'est pas particulièrement mise en avant dans la rédaction.

D'un point de vue méthodologique, l'auteur montre qu'il n'est pas possible de ramener ce problème d'optimisation multi-objectifs à des combinaisons simples des dimensions et il choisit de s'orienter sur la recherche d'optimums de Pareto. Comme une recherche exhaustive n'est pas possible compte-tenu de la combinatoire, il propose deux algorithmes, l'un fondé sur une approximation (α-MOMRI), l'autre sur une heuristique (h-MOMRI). α-MOMRI est un algorithme par niveau exploitant les principes de la programmation dynamique et dont l'astuce d'élagage repose sur des propriétés d'optimalité qui sont propres aux dimensions étudiées. La formulation de h-MOMRI lui permet d'exploiter la propriété de monotonicité de la couverture lorsqu'on cherche à la maximiser. Fondé sur une heuristique, h-MOMRI ne donne pas de garantie sur l'approximation du résultat, contrairement à α-MOMRI. Les résultats expérimentaux, menés ici sur deux jeux de données, montrent que dans la pratique h-MOMRI produit une bonne approximation des meilleurs ensembles de groupes avec un nombre limité de résultats (ce qui simplifie le travail de l'analyste) tout en ayant un temps de calcul beaucoup plus court (gain d'environ deux ordres de magnitude). Cependant, h-MOMRI nécessite de fixer des paramètres et le choix de leurs valeurs fait l'objet d'une expérimentation. Ce chapitre témoigne de la part de B. Omidvar-Tehrani d'une grande connaissance et d'un savoir-faire sur les méthodes d'optimisation, auxquels s'associe une créativité afin de mettre en avant des propriétés d'élagage issus du problème étudié. Nous avons ici une solide contribution de nature théorique et algorithmique.

Le chapitre 4 présente deux méthodes d'exploration et d'analyse interactive de groupes d'utilisateurs. La première repose sur un principe d'abstraction selon des taxonomies : lorsqu'un groupe possède une proportion suffisante (selon un seuil donné par l'utilisateur) des valeurs fils d'un élément de la taxonomie, ces valeurs sont remplacées par celle de leur parent. Les expérimentations montrent que ce principe simple (ce qui n'est pas une critique !) permet de réduire assez fortement le nombre de groupes tout en effectuant des généralisations appropriées des descriptions. Il n'est pas dit comment les taxonomies sont construites ni si cette opération est coûteuse (la construction semi-automatique de taxonomies est cependant citée comme perspective). La seconde méthode est une approche originale de navigation dans les groupes d'utilisateurs. Celle-ci formalise des opérateurs de navigation sur les groupes dont le plus important est celui qui suggère de nouveaux groupes d'utilisateurs partageant une (forte)

description avec le groupe sur lequel l'analyste est positionné. À chaque étape, pour permettre l'analyse de l'utilisateur, au plus k groupes sont proposés. La stratégie d'exploration, fondée sur une heuristique, prend soin de veiller à la diversité de ces groupes. Cet aspect est important car il limite le problème de redondance entre groupes. L'auteur combine cette mesure de diversité avec celle de couverture et on retrouve ainsi les dimensions étudiées au chapitre précédent. L'ensemble a été intégré dans une interface graphique et l'apport de la méthode est appuyé par des expérimentations, notamment un cas d'étude sur la constitution d'un comité de programme de conférences.

J'ai apprécié le principe de mettre l'analyste dans le processus de découverte de groupes ainsi que la formalisation des opérateurs de navigation. La coopération entre systèmes de fouille et utilisateurs est présentée comme inhérente au processus de découverte de connaissances mais, dans la pratique, elle est souvent réduite à des interactions ponctuelles entre deux exécutions d'un algorithme automatique. Ce constat ne fait que renforcer l'importance de la contribution de B. Omidvar-Tehrani au domaine de la fouille de données interactive. Du point de vue opératoire, la méthode part de groupes d'utilisateurs qui sont des motifs fermés produits hors ligne par l'algorithme LCM de Uno et al. La qualité de ces groupes est simplement liée à leurs fréquences et il me semble qu'une perspective intéressante serait de chercher à tirer profit du bénéfice de la qualité d'un ensemble de groupes qui est assurée par les méthodes du chapitre 3. Ainsi, peut-on définir des opérateurs de navigation à partir des ensembles de groupes Pareto-optimaux ? Par exemple, si G_1 et G_2 sont deux ensembles de groupes Pareto-optimaux avec $G_1 = \{g_1, g_2\}$ et $G_2 = \{g_1, g_3, g_4\}$ et si l'analyste s'intéresse à g_1, proposer pour la navigation soit $\{g_2\}$ soit $\{g_3, g_4\}$ assure une bonne diversité et couverture (de part la construction de G_1 et G_2) tout en suggérant deux "chemins" dont les intérêts dépendent l'un de l'autre puisque G_1 ne domine pas G_2 et vice versa. Est-ce qu'il existe des liens entre ces "chemins" et les groupes que retournerait l'opérateur *opExplore* ? Cependant, le coût de calcul des groupes Pareto optimaux est nettement plus élevé que celui de LCM.

Le chapitre 5 porte sur l'usage de groupes d'utilisateurs pour la recommandation. Cette thématique formant un champ de recherche particulièrement large, l'auteur prend soin de situer ses contributions dans ce paysage. Ses contributions possèdent une double originalité. D'une part, les relations entre utilisateurs d'un même groupe sont prises en compte et modélisées par une fonction appelée *affinité*. Pour un même utilisateur, la recommandation dépend ainsi du groupe dans lequel il est plongé. D'autre part, les relations changeant au cours du temps, deux modèles d'évolution temporelle sont proposés pour une recommandation plus fine. Ce chapitre effectue une modélisation fine des éléments intervenants dans la recommandation (affinité, préférences, agrégation de préférences). Cette modélisation constitue un socle solide au principe de recommandation proposé ici qui consiste à recommander à un utilisateur les k meilleurs éléments (e.g., produits) suivant son groupe. L'auteur décrit alors l'algorithme `GRECA` pour calculer à la volée la recommandation d'un utilisateur. La conception et la réalisation de cet algorithme nécessite une analyse fine des fonctions utilisées (pour obtenir des élagages liés à la propriété de monotonicité) ainsi qu'une manipulation rigoureuse des structures de données. L'auteur montre clairement sa capacité à concevoir et à développer un algorithme sophistiqué. Le chapitre se termine par l'évaluation de `GRECA` suivant la qualité des réponses produites et le passage à l'échelle. Cette évaluation montre l'intérêt de prendre en compte l'évolution temporelle et la capacité opérationnelle de `GRECA`. Dans ces expériences, les groupes ne sont pas découverts (et donc les méthodes des chapitres précédents ne sont pas utilisées) mais ils ont été conçus pour mettre en œuvre le protocole expérimental.

Le chapitre 6 est consacré à l'état de l'art, mais il positionne aussi les contributions de la thèse par rapport à celui-ci. La présentation de l'état de l'art suit celle du mémoire : découverte de groupes d'utilisateurs, analyse de ces groupes et recommandation. Ce chapitre montre la grande culture possédée par B. Omidvar-Tehrani (les domaines traités sont nombreux)

ainsi que sa capacité à situer ses contributions par rapport à ces domaines. Ce chapitre est synthétique, ce qui facilite sa lecture, toutefois, certains points pourraient être creusés. Par exemple, le clustering est vu sous son angle "classique", mais il existe aujourd'hui des méthodes de clustering conceptuels sous contraintes et avec chevauchements (où dans certains cas un cluster est un fermé). Est-ce que ces méthodes pourraient donner lieu à la découverte de groupes d'utilisateurs ? Le passage sur l'analyse interactive pourrait être complété par la découverte interactive de motifs (voir par exemple "Interactive Data Exploration using Pattern Mining", 2014, de M. van Leeuwen).

Le mémoire se termine par une brève conclusion et plusieurs perspectives de recherche. Ces dernières se déclinent suivant 3 axes (méthodes de découverte de groupes d'utilisateurs, évaluation, autres applications) et sont particulièrement étayées.

Pour résumer, ce mémoire témoigne d'une très bonne capacité à concevoir des méthodes et des solutions solides en s'appuyant sur une formalisation rigoureuse. B. Omidvar-Tehrani a mené un travail conséquent et a apporté d'importantes contributions de nature variée (méthodologique, implémentation d'algorithmes, étude de cas). Son travail, à la croisée de la fouille de donnée, de l'optimisation et de la recommendation, devrait donner lieu à des poursuites fructueuses dans ces domaines. Il est aussi reconnu par des publications dans des conférences de grande qualité (CIKM 2015, EDBT 2015).

Pour toutes ces raisons, je donne un avis très favorable à la soutenance de thèse de Behrooz Omidvar-Tehrani en vue de l'obtention du grade de docteur de l'Université Grenoble Alpes.

Caen, le 12 octobre 2015,

Bruno Crémilleux
Professeur des Universités
Université de Caen Normandie

Bibliography

[AGGR98] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. *Automatic subspace clustering of high dimensional data for data mining applications*, volume 27. ACM, 1998.

[AIK+14] Sihem Amer-Yahia, Noha Ibrahim, Christiane Kamdem Kengne, Federico Ulliana, and Marie-Christine Rousset. SOCLE: towards a framework for data preparation in social applications. *Ingénierie des Systèmes d'Information*, 19(3):49–72, 2014.

[AIS93] Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. In *SIGMOD*, pages 207–216, 1993.

[AK11] Albert Angel and Nick Koudas. Efficient diversity-aware search. In *SIGMOD*, pages 781–792. ACM, 2011.

[AORS15] Sihem Amer-Yahia, Behrooz Omidvar-Tehrani, Senjuti Basu Roy, and Nafiseh Shabib. Group recommendation with temporal affinities. In *Proceedings of the 18th International Conference on Extending Database Technology, EDBT 2015, Brussels, Belgium, March 23-27, 2015.*, pages 421–432, 2015.

[APV07] Reza Akbarinia, Esther Pacitti, and Patrick Valduriez. Best position algorithms for top-k queries. In *Proceedings of the 33rd International Conference on Very Large Data Bases, University of Vienna, Austria, September 23-27, 2007*, pages 495–506, 2007.

[ASST05] Gediminas Adomavicius, Ramesh Sankaranarayanan, Shahana Sen, and Alexander Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Trans. Inf. Syst.*, January 2005.

[AT05] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *TKDE*, 17(6):734–749, June 2005.

[AV07] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[AYLT+15] Sihem Amer-Yahia, Vincent Leroy, Alexandre Termier, Martin Kirchgessner, and Behrooz Omidvar-Tehrani. Interactive Data-Driven Research: the place where databases and data mining research meet. Research Report RR-LIG-049, LIG, Grenoble, France, 2015.

[AYRC+09] Sihem Amer-Yahia, Senjuti Basu Roy, Ashish Chawla, Gautam Das, and Cong Yu. Group recommendation: Semantics and efficiency. *PVLDB*, 2(1):754–765, 2009.

[BB13] Pat Barclay and Stephen Benard. Who cries wolf, and when: Manipulation of perceived threats to preserve rank in cooperative groups. *PloS one*, 8(9):e73863, 2013.

[BCC+09] Ludovico Boratto, Salvatore Carta, Alessandro Chessa, Maurizio Agelli, and Maria Laura Clemente. Group recommendation with automatic identification of users communities. In *Web Intelligence/IAT Workshops*, pages 547–550, 2009.

[Ber83] Jacques Bertin. Semiology of graphics: diagrams, networks, maps. *ESRI Press*, 1983.

[BF10] Shlomo Berkovsky and Jill Freyne. Group-based recipe recommendations: analysis of data aggregation strategies. In *RecSys*, pages 111–118, 2010.

[BGKW02] Cristian Bucila, Johannes Gehrke, Daniel Kifer, and Walker M. White. Dualminer: a dual-pruning algorithm for itemsets with constraints. In *Knowledge Discovery and Data Mining*, pages 42–51, 2002.

[BGMP03] Francesco Bonchi, Fosca Giannotti, Alessio Mazzanti, and Dino Pedreschi. Exante: Anticipated data reduction in constrained pattern mining. In *PKDD*, pages 59–70, 2003.

[BJ03] Patrice Bertrand and Melvin F Janowitz. The k-weak hierarchical representations: an extension of the indexed closed weak hierarchies. *Discrete Applied Mathematics*, 127(2):199–220, 2003.

[BJP91] James R Bettman, Eric J Johnson, and John W Payne. Consumer decision making. *Handbook of consumer behavior*, 44(2):50–84, 1991.

[BKS01] S Borzsony, Donald Kossmann, and Konrad Stocker. The skyline operator. In *Data Engineering, 2001. Proceedings. 17th International Conference on*, pages 421–430. IEEE, 2001.

[BKS11] Tijl De Bie, Kleanthis-Nikolaos Kontonasios, and Eirini Spyropoulou. A framework for mining interesting pattern sets. *Sigkdd Explorations*, 12:92–100, 2011.

[BKT+13] M. Boley, B. Kang, P. Tokmakov, M. Mampaey, and S. Wrobel. One click mining: Interactive local pattern discovery through implicit preference and performance learning. *IDEAS (ACM SIGKDD Workshop)*, 2013.

[BMH12] M. Bhuiyan, S. Mukhopadhyay, and M. Al Hasan. Interactive pattern mining on hidden data: a sampling-based solution. In *CIKM*, pages 95–104, 2012.

[BNJ03] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[BRR63] Albert Bandura, Dorothea Ross, and Sheila A Ross. Imitation of film-mediated aggressive models. *The Journal of Abnormal and Social Psychology*, 66(1):3, 1963.

[CB00] Ellis J Clarke and Bruce A Barton. Entropy and mdl discretization of continuous variables for bayesian belief networks. *International Journal of Intelligent Systems*, 15(1):61–92, 2000.

[CBH02] Andrew Crossen, Jay Budzik, and Kristian J. Hammond. Flytrap: intelligent group music recommendation. In *IUI*, pages 184–185, 2002.

[CCD+13] Ugur Cetintemel, Mitch Cherniack, Justin DeBrabant, Yanlei Diao, Kyriaki Dimitriadou, Alexander Kalinin, Olga Papaemmanouil, and Stanley B Zdonik. Query steering for interactive data exploration. In *CIDR*, 2013.

[CG98] J. G. Carbonell and J. Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Research and Development in Information Retrieval*, pages 335–336, 1998.

[CJL+11] Olivier Chapelle, Shihao Ji, Ciya Liao, Emre Velipasaoglu, Larry Lai, and Su-Lin Wu. Intent-based diversification of web search results: metrics and algorithms. *Information Retrieval*, 14(6):572–592, 2011.

[Cle07] Guillaume Cleuziou. A generalization of k-means for overlapping clustering. *Rapport technique*, page 54, 2007.

[CMI+15] Xu Chu, John Morcos, Ihab F. Ilyas, Mourad Ouzzani, Paolo Papotti, Nan Tang, and Yin Ye. KATARA: A data cleaning system powered by knowledge bases and crowdsourcing. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, pages 1247–1261, 2015.

[CSTC12] Caleb Chen Cao, Jieying She, Yongxin Tong, and Lei Chen. Whom to ask?: jury selection for decision making tasks on micro-blog services. *VLDB*, 2012.

[DAYDY11] Mahashweta Das, Sihem Amer-Yahia, Gautam Das, and Cong Yu. Mri: Meaningful interpretations of collaborative ratings. *VLDB*, 2011.

[DBETT94] Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G Tollis. Algorithms for drawing graphs: an annotated bibliography. *Computational Geometry*, 4(5):235–282, 1994.

[DL05] Yi Ding and Xue Li. Time weight collaborative filtering. In *CIKM*, pages 485–492. ACM, 2005.

[DRST09] Pierre-François Dutot, Krzysztof Rzadca, Erik Saule, and Denis Trystram. *Multi-objective scheduling*, chapter 9. Chapman and Hall/CRC Press, 2009.

[DRZ07] Luc De Raedt and Albrecht Zimmermann. Constraint-based pattern set mining. SIAM, 2007.

[DTL+15] Trong Dinh Thac Do, Alexandre Termier, Anne Laurent, Benjamin Négrevergne, Behrooz Omidvar-Tehrani, and Sihem Amer-Yahia. PGLCM: efficient parallel mining of closed frequent gradual itemsets. *Knowl. Inf. Syst.*, 43(3):497–527, 2015.

[Fis87] Douglas H Fisher. Improving inference through conceptual clustering. In *AAAI*, volume 87, pages 461–465, 1987.

[Fis02] Douglas Fisher. Data mining tasks and methods: Clustering: conceptual clustering. In *Handbook of data mining and knowledge discovery*, pages 388–396. Oxford University Press, Inc., 2002.

[FKP01] Uriel Feige, Guy Kortsarz, and David Peleg. The dense k-subgraph problem. *Algorithmica*, 29(3):410–421, 2001.

[FLN01] Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal aggregation algorithms for middleware. In Peter Buneman, editor, *PODS*. ACM, 2001.

[GBGP10] Sharad Goel, Andrei Broder, Evgeniy Gabrilovich, and Bo Pang. Anatomy of the long tail: ordinary people with extraordinary tastes. In *WSDM*, 2010.

[GH06] Liqiang Geng and Howard J Hamilton. Interestingness measures for data mining: A survey. *ACM Computing Surveys (CSUR)*, 38(3):9, 2006.

[GHK92] Sumit Ganguly, Waqar Hasan, and Ravi Krishnamurthy. *Query optimization for parallel execution*, volume 21. ACM, 1992.

[GMV11] Bart Goethals, Sandy Moens, and Jilles Vreeken. Mime: a framework for interactive visual pattern mining. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 757–760. ACM, 2011.

[Grü07] Peter D Grünwald. *The minimum description length principle*. MIT press, 2007.

[GSO11] Inma Garcia, Laura Sebastia, and Eva Onaindia. On the design of individual and group recommender systems for tourism. *Expert Syst. Appl.*, 38(6):7683–7692, 2011.

[HBO10] Jeffrey Heer, Michael Bostock, and Vadim Ogievetsky. A tour through the visualization zoo. *Commun. Acm*, 53(6):59–67, 2010.

[HSRF95] Will Hill, Larry Stead, Mark Rosenstein, and George Furnas. Recommending and evaluating choices in a virtual community of use. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 194–201. ACM Press/Addison-Wesley Publishing Co., 1995.

[HT10] Harry Halpin and Mischa Tuffield. A standards-based, open and privacy-aware social web. *W3C Social Web Incubator Group Report 6th December*, 2010.

[IMMM14] Piotr Indyk, Sepideh Mahabadi, Mohammad Mahdian, and Vahab S Mirrokni. Composable core-sets for diversity and coverage maximization. In *ACM SIGMOD SIGART*, pages 100–108. ACM, 2014.

[Joh73] David S Johnson. Approximation algorithms for combinatorial problems. In *Proceedings of the fifth annual ACM symposium on Theory of computing*, pages 38–49. ACM, 1973.

[JS07] Anthony Jameson and Barry Smyth. Recommendation to groups. In *The adaptive web*, pages 596–627. Springer, 2007.

[JSH04] Anoop Jain, Parag Sarda, and Jayant R. Haritsa. Providing diversity in k-nearest neighbor query results. In *PAKDD*, pages 404–413, 2004.

[KAZ12] Mehdi Kargar, Aijun An, and Morteza Zihayat. Efficient bi-objective team formation in social networks. In *Machine Learning and Knowledge Discovery in Databases*. Springer, 2012.

[KDK11] Noam Koenigstein, Gideon Dror, and Yehuda Koren. Yahoo! music recommendations: modeling music ratings with temporal dynamics and item taxonomy. In *RecSys*, pages 165–172, 2011.

[KIJ+15] Zuhair Khayyat, Ihab F Ilyas, Alekh Jindal, Samuel Madden, Mourad Ouzzani, Paolo Papotti, Jorge-Arnulfo Quiané-Ruiz, Nan Tang, and Si Yin. Bigdansing: A system for big data cleansing. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 1215–1230. ACM, 2015.

[KJT+14] Narendra Kamat, Prasanth Jayachandran, Karthik Tunga, et al. Distributed and interactive cube exploration. In *Data Engineering (ICDE), 2014 IEEE 30th International Conference on*, pages 472–483. IEEE, 2014.

[KL70] Brian W Kernighan and Shen Lin. An efficient heuristic procedure for partitioning graphs. *Bell system technical journal*, 49(2):291–307, 1970.

[Kon04] Joseph A Konstan. Introduction to recommender systems: Algorithms and evaluation. *ACM Transactions on Information Systems (TOIS)*, 22(1):1–4, 2004.

[Kor10] Yehuda Koren. Collaborative filtering with temporal dynamics. *Commun. ACM*, 53(4):89–97, 2010.

[KY89] N. M. Klein and M. Yadav. Context effects on effort and accuracy in choice: An inquiry into adaptive decision making. *Journal of Consumer Research*, 16:410–420, 1989.

[LA00] Anton Leuski and James Allan. Strategy-based interactive cluster visualization for information retrieval. *International Journal on Digital Libraries*, 3:170–184, 2000.

[LDG+15] Pei Li, Xin Luna Dong, Songtao Guo, Andrea Maurino, and Divesh Srivastava. Robust group linkage. In *Proceedings of the 24th International*

Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015, pages 647–657, 2015.

[LO79] Denis A. Lussier and Richard W. Olshavsky. Task complexity and contingent processing in brand choice. *Journal of Consumer Research*, 6:154–165, 1979.

[LWT+12] Pei Li, Haidong Wang, Christina Tziviskou, Xin Luna Dong, Xiaoguang Liu, Andrea Maurino, and Divesh Srivastava. Chronos: Facilitating history discovery by linking temporal records. *PVLDB*, 5(12):2006–2009, 2012.

[MGB09] Claudia Marinica, Fabrice Guillet, and Henri Briand. Post-processing of discovered association rules using ontologies. *CoRR*, abs/0910.0349, 2009.

[MPsM96] Christopher J. Matheus, Gregory Piatetsky-shapiro, and Dwight Mcneill. Selecting and reporting what is interesting: The kefir application to healthcare data, 1996.

[MPV97] Athanasios Migdalas, Panos M Pardalos, and Peter Värbrand. *Multilevel optimization: algorithms and applications*, volume 20. Springer Science & Business Media, 1997.

[NG04] Mark EJ Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.

[NJ11] Arnab Nandi and HV Jagadish. Guided interaction: Rethinking the query-result paradigm. *Proceedings of the VLDB Endowment*, 4(12):1466–1469, 2011.

[NSNK12] Eirini Ntoutsi, Kostas Stefanidis, Kjetil Nørvåg, and Hans-Peter Kriegel. Fast group recommendations by applying user clustering. In *ER*, pages 126–140, 2012.

[OCKR01a] Mark O'Connor, Dan Cosley, Joseph A. Konstan, and John Riedl. Polylens: a recommender system for groups of users. In *ECSCW*, 2001.

[OCKR01b] Mark O'connor, Dan Cosley, Joseph A Konstan, and John Riedl. Polylens: a recommender system for groups of users. In *ECSCW 2001*, pages 199–218. Springer, 2001.

[OTAT+13] Behrooz Omidvar-Tehrani, Sihem Amer-Yahia, Alexandre Termier, Aurélie Bertaux, Éric Gaussier, and Marie-Christine Rousset. Towards a framework for semantic exploration of frequent patterns. In *Proceedings of the 3rd International Workshop on Information Management for Mobile Applications, Riva del Garda, Italy, August 26, 2013.*, pages 7–14, 2013.

[OTAYT15] Behrooz Omidvar-Tehrani, Sihem Amer-Yahia, and Alexandre Termier. Interactive user group analysis. In *International Conference on Information and Knowledge Management, CIKM'15, Melbourne, Victoria, Australia, October 19 - 23, 2015*, 2015.

[Pag03] Devah Pager. The mark of a criminal record1. *American journal of sociology*, 108(5):937–975, 2003.

[Par05] Laxmi Parida. Redescription mining: Structure theory and algorithms. In *In Proc. AAAI'05*, pages 837–844, 2005.

[PDCCA05] Sebastiano Pizzutilo, Berardina De Carolis, Giovanni Cozzolongo, and Francesco Ambruoso. Group modeling in a public space: Methods, techniques, experiences. In *AIC*, AIC'05. WSEAS, 2005.

[PLL99] José Manuel Pena, Jose Antonio Lozano, and Pedro Larranaga. An empirical comparison of four initialization methods for the k-means algorithm. *Pattern recognition letters*, 20(10):1027–1040, 1999.

[PMC+14] Mojgan Pourrajabi, Davoud Moulavi, Ricardo JGB Campello, Arthur Zimek, Jörg Sander, and Randy Goebel. Model selection for semi-supervised clustering. In *EDBT*, pages 331–342, 2014.

[Por10] Joshua Porter. *Designing for the social web*. Peachpit Press, 2010.

[PY00] Christos H. Papadimitriou and Mihalis Yannakakis. On the approximability of trade-offs and optimal access of web sources. In *FOCS*, 2000.

[QAH12] Guo-Jun Qi, Charu C Aggarwal, and Thomas Huang. Community detection with edge content in social media networks. In *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*, pages 534–545. IEEE, 2012.

[QAH13] Guo-Jun Qi, Charu C Aggarwal, and Thomas S Huang. Online community detection in social sensing. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 617–626. ACM, 2013.

[RAYC+10] Senjuti Basu Roy, Sihem Amer-Yahia, Ashish Chawla, Gautam Das, and Cong Yu. Space efficiency in group recommendation. *VLDB J.*, 19(6):877–900, 2010.

[RLT+15] Senjuti Basu Roy, Ioanna Lykourentzou, Saravanan Thirumuruganathan, Sihem Amer-Yahia, and Gautam Das. Task assignment optimization in knowledge-intensive crowdsourcing. *VLDB J.*, 24(4):467–491, 2015.

[RN03] Stuart J Russell and Peter Norvig. Probabilistic reasoning. *Artificial intelligence: a modern approach*, 2003.

[RTAY+14] Senjuti Basu Roy, Saravanan Thirumuruganathan, Sihem Amer-Yahia, Gautam Das, and Cong Yu. Exploiting group recommendation functions for flexible preferences. In *ICDE Conference*, 2014.

[sLIC08] Carson Kai sang Leung, Pourang P. Irani, and Christopher L. Carmichael. WiFIsViz: Effective Visualization of Frequent Itemsets. In *ICDM*, 2008.

[SMC87] David C Schmittlein, Donald G Morrison, and Richard Colombo. Counting your customers: Who-are they and what will they do next? *Management science*, 33(1):1–24, 1987.

[SNJ12] Manish Singh, Arnab Nandi, and H. V. Jagadish. Skimmer: rapid scrolling of relational query results. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2012, Scottsdale, AZ, USA, May 20-24, 2012*, pages 181–192, 2012.

[SRPC11] Arnaud Soulet, Chedy Raïssi, Marc Plantevit, and Bruno Cremilleux. Mining dominant patterns in the sky. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 655–664. IEEE, 2011.

[SVvL06] Arno Siebes, Jilles Vreeken, and Matthijs van Leeuwen. Item sets that compress. In *SDM*, volume 6, pages 393–404. SIAM, 2006.

[TK14] Immanuel Trummer and Christoph Koch. Approximation schemes for many-objective query optimization. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. ACM, 2014.

[UAUA03] Takeaki Uno, Tatsuya Asai, Yuzo Uchida, and Hiroki Arimura. Lcm: An efficient algorithm for enumerating frequent closed item sets. In *In Proceedings of Workshop on Frequent itemset Mining Implementations FIMI'03*, 2003.

[UBLC12] Willy Ugarte, Patrice Boizumault, Samir Loudni, and Bruno Crémilleux. Soft threshold constraints for pattern mining. In *Discovery Science - 15th International Conference, DS 2012, Lyon, France, October 29-31, 2012. Proceedings*, pages 313–327, 2012.

[UKA04] Takeaki Uno, Masashi Kiyomi, and Hiroki Arimura. Lcm ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets. In *FIMI*, 2004.

[vL14] Matthijs van Leeuwen. Interactive data exploration using pattern mining. In *Interactive Knowledge Discovery and Data Mining in Biomedical Informatics*, pages 169–182. Springer, 2014.

[VVLS11] Jilles Vreeken, Matthijs Van Leeuwen, and Arno Siebes. Krimp: mining itemsets that compress. *Data Mining and Knowledge Discovery*, 23(1):169–214, 2011.

[WCR+01] Kiri Wagstaff, Claire Cardie, Seth Rogers, Stefan Schrödl, et al. Constrained k-means clustering with background knowledge. In *ICML*, volume 1, pages 577–584, 2001.

[WL12] Robert West and Jure Leskovec. Automatic versus human navigation in information networks. In *Proceedings of the Sixth International Conference on Weblogs and Social Media, Dublin, Ireland, June 4-7, 2012*, 2012.

[XCH+10] Liang Xiong, Xi Chen, Tzu-Kuo Huang, Jeff G. Schneider, and Jaime G. Carbonell. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *SDM*, pages 211–222. SIAM, 2010.

[XSMH06] Dong Xin, Xuehua Shen, Qiaozhu Mei, and Jiawei Han. Discovering interesting patterns through user's interactive feedback. In *Knowledge Discovery and Data Mining*, pages 773–778, 2006.

[YZHG06] Zhiwen Yu, Xingshe Zhou, Yanbin Hao, and Jianhua Gu. TV Program Recommendation for Multiple Viewers Based on user Profile Merging. *User Modeling and User-adapted Interaction*, 16:63–82, 2006.