DECISION-GUIDED RECOMMENDERS WITH COMPOSITE ALTERNATIVES

by

Khalid Alodhaibi

A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
in Partial Fulfillment of
The Requirements for the Degree
of
Doctor of Philosophy
Information Technology

Committee:

Dr. Alexander Brodsky, Dissertation Director

Dr. George Mihaila, Committee Member

Dr. Larry Kerschberg, Committee Member

Dr. Jessica Lin, Committee Member

Dr. David Schum, Committee Member

Dr. Daniel Menascé, Senior Associate Dean

Lloyd J. Griffiths, Dean, The Volgenau School of Engineering

Date: 7/11/2011

Summer Semester 2011
George Mason University, Fairfax, VA

Decision-Guided Recommenders With Composite Alternatives

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

By

Khalid Alodhaibi
Master of Business Administration
Santa Clara Univeristy, 2006
Bachelor of Science
King Saud University, 1998

Director: Alexander Brodsky, Associate Professor
Department of Computer Science

Summer Semester 2011
George Mason University
Fairfax, VA

Dedication

I dedicate this dissertation to my wonderful wife Reema. She took care of us while doing her PhD and that is impressive.

Acknowledgments

I would like to express my deepest gratitude to my advisor and mentor, Prof. Alex Brodsky for his endless support and patience. My research would not exist if not for his continuous guidance and supervision. He believed in me and led me along the road to reach my goal; I am deeply grateful.

I am also very grateful to Dr. George Mihaila from Google Inc. who served as a committee member, and provided continues feedback and help. Dr. Mihaila was willing to meet with me every week over the past three years, and guide me and my research to be where I am now. He was very patient and accommodating when I drove over 14 times to New York and spent a day or two every time to either implement an idea or finish writing a paper together.

I am also very grateful to my committee members, Dr. David Schum, Dr. Larry Kerschberg, and Dr. Jessica Lin for serving on my committee and for generously giving me their time and advice every time I needed them.

I would like also to thank my son Ibrahim (Abo 3ug) and my daughter Alya Boo for allowing me to be away several days a month so I can work on my research and catch up with reading. But I knew they were having great time with my wonderful wife Reema.

I offer special thanks to King Faisal Specialist hospital and research Center, which granted me the PhD scholarship, and to my country, which funded my education. Specifically, Mr. Hamad Aldaig, who continuously encouraged me to continue my education and journey in life. Also I would like to thank Dr. Abdullah Aldalaan, and Mr. Abdulaziz Albahkaly for helping me with my scholarship work.

I also wish to thank my brothers and sisters for their continuous support and encouragements. They all have expressed words of wisdom and great inspiration.

My special appreciation goes to my best friend and colleague Khalid Albarrak who has been one of the greatest resources to rely on. Khalid and I started the PhD journey together and I can not count how many days and nights we complained to each other about work and school challenges. I deeply wish him the best of luck in everything.

A big thank you goes to Susan Lee and Dan Rhoads from my management team at IBM for helping and supporting me over the past 7 years while doing my MBA and PhD work. They were following up on my progress all the way and showed me a true team spirit.

Table of Contents

List of Tables

List of Figures

Abstract

DECISION-GUIDED RECOMMENDERS WITH COMPOSITE ALTERNATIVES
Khalid Alodhaibi, PhD
George Mason University, 2011
Dissertation Director: Alexander Brodsky

Recommender systems aim to support users in their decision-making process while interacting with large information spaces and recommend items of interest to users based on preferences they have expressed, either explicitly or implicitly. Recommender systems are increasingly used with product and service selection over the Internet. Although technology has made it easy to search for and interact with most information types, the volume surge in data presented is overwhelming and hard to filter. While state-of-the-art recommender systems focus on atomic products or services, this research focuses on developing a framework, models and algorithms for recommending composite services and products based on decision optimization. Composite services are characterized by a set of sub-services, which, in turn, can be composite or atomic and make the recommendation space very large (or infinite, for continuous case). Complex recommendation models involving composite alternatives, such as product configurations and service packages, have not been addressed. The proposed framework contains models that allow for fast and easy user preference elicitation that can be captured in a utility function, and provides algorithms for diversifying a recommendation set. Such recommendations will be dynamically defined using database views extended with decision optimization based on mathematical programming.

A key challenge addressed in this research is combining the flexibility of diversity ranking functionality with the capabilities of information processing to learn and capture users' preferences through an iterative learning process.

The proposed framework presents a method for utility function elicitation, which is based on iteratively refining a set of axes in the n-dimensional utility space. User preferences are initially learned using regression analysis or Collaborative Filtering (CF) techniques. At every step, the user is asked to rank a set of recommendations, each being optimal for one of the current axes. Based on the user feedback, utility axes are adaptively adjusted based on a confidence degree. Consequently, the utility function is constructed.

In addition, the framework proposes a new approach to diversify a set of recommendations, which is based on constructing and using an m-dimensional diversity feature space, which is separate from the utility space used for utility elicitation. Furthermore, the framework presents a diversity algorithm to address the Maximum Diversity Problem (MDP) of recommendations, which is randomized and based on iterative relaxation of selections by the Greedy algorithm with an exponential probability distribution.

Finally, the proposed framework is validated with several experimental studies using publically available datasets, such as MovieLens and Yahoo, to measure the efficacy and efficiency against state-of-the art algorithms and techniques. The results show that the proposed algorithm is highly efficient computationally, and consistently outperforms competing algorithms and systems in terms of precision, recall, diversity measures, and MAE (mean absolute error). In addition, the proposed framework converges to the optimal or near-optimal solutions in under 100 ms using a machine with Intel Core 2 Duo CPU 2.53GHz and 3GB RAM. The proposed collaborative filtering technique achieved a precision of 90% on average.

Chapter 1: Introduction

While interacting with large information spaces, Recommender Systems aim to support users in their decision-making process and recommend items of interest to users based on preferences they have expressed, either explicitly or implicitly. Recommender systems are increasingly used with product and service selection over the Internet. While state-of-the-art recommender systems focus on atomic products or services, this research focuses on developing a framework, models and algorithms for recommending composite services and products based on decision optimization, and eliciting user preferences within the context of recommender systems. The proposed framework combines the flexibility of diversity ranking functionality with the information processing capabilities to learn and capture the preferences of the user through an iterative learning process. The framework proposes and informs the user of what is available in terms of composite recommendations with minimum interaction from the user side. In sections 1.1, 1.2 and 1.3, I explain the motivation and the research challenge, state the problem, and provide a summary of research contributions.

1.1 Motivation

One of the main challenges we face is information overload. Although technology has made it easy to search for, and interact with most information types, the ever-expanding volume and increasing complexity of information on the Web has made recommender systems essential tools for users in a variety of information seeking or e-commerce activities. This trend of information overload is increasing and mandates efficient ways of learning user preferences. Recommender systems help overcome this problem by exposing users to the most relevant items. Naive information retrieval (IR) approaches (e.g., [3,19,119]) provide mechanisms for effective fuzzy ranking, but are less appropriate to use when intelligence is needed in the recommendation process [35,69], especially when recommending complex products and services. Traditional Database Management Systems have shown efficient handling of data with solid integrity, but lack the ranking capability needed for answering Information Retrieval (IR) queries. In addition, DBMS are also not capable of providing diversity functionalities needed for recommender systems.

Recommender technology is the central piece of the information-seeking puzzle. Major e-commerce sites such as Amazon and Yahoo use recommendation technology in many ways. Many new systems are on their way (e.g. Microsoft Bing) and entrepreneurs are competing to find the right approach to use this technology effectively. Related research went so far as to ask the question “Will recommenders kill search?”, and analyzed emerging topics regarding recommender systems as a whole and specifically their role in the industry [47].

In an article published in CNN Money, entitled “The race to create a ’smart’ Google”, Fortune magazine writer Jeffrey M. O’Brien, writes: “The Web, they say, is leaving the era of search and entering one of discovery. What’s the difference? Search is what you do when you’re looking for something. Discovery is when something wonderful that you didn’t know existed, or didn’t know how to ask for, finds you.”

1.2 Research Challenge

Recommender systems (e.g., [1,37,49,80,90]) are expected to guide the user to find the most suitable products and services. Recommender systems suggest items of interest to users based on preferences they have expressed, either explicitly or implicitly.

The need for decision guidance over information retrieval becomes more obvious when interacting with the Internet where information space is large. Although technology has made it easy to search for and interact with most information types, the volume surge in data presented is overwhelming and hard to filter. The trend of information overload is increasing with the recent shift toward customized product bundles and service compositions. Composite services are characterized by a set of sub-services, which, in turn, can be composite or atomic and make the recommendation space very large (exponential in number of components, or infinite, for continuous case), and implicitly defined.

In addition, the new surge of current mobile computing devices such as PDAs and WAP-enabled mobile phones with screen size that could be 200 times smaller than that of a PC, is introducing pressure on recommender systems to move beyond the accuracy measure. This

technological trend requires that we carefully choose which recommendations to return in a single search [101]. If the few returned recommendations are similar to each other, then it is unlikely that the user will be satisfied [26].

Thus, key market drivers are information overload, product bundles and service compositions, and display size limitations, which are competing with each other. From one end, information overload is mainly influenced by the growing usage of the Internet, and service compositions. On the other end, the popularity of Web-enabled mobile devices is introducing space size limitations.

To address these competing drivers, recommender systems need to carefully " *reduce*" the large information space into a small number of recommendation that are highly relevant to users and meet the display size limitations. The reduction of information space can be addressed by incorporating optimization techniques to choose the best item(s) from some set of available alternatives. To perform optimization, recommender systems must be able to elicit the utility function that captures the users' preferences. Given the accurately elicited utility function, optimization enables recommender systems to identify and present users with a small number of recommendations that are optimal or near-optimal with regard to the utility function.

The elicited utility is only an estimate of users' preferences, because users may not have explained their preferences accurately, or the recommender system could not capture the preferences precisely. Therefore, it is important that the small number of recommendations returned be not only near-optimal, but also sufficiently diverse. The reason for diversity is to account for uncertainty in elicited utility and allow the user sufficient freedom of choice.

For this reason, many practical systems are interactive, allowing the user to scroll through a set of choices to make a decision [73].

There has been extensive research in the area of recommender systems (e.g., [1,22,68]). Popular surveys of recommendation techniques classify recommender systems as either content-based, collaborative, or hybrid systems. Content-based systems often employ classifier techniques to recommend an item to a user based upon a description of the item and a profile of the user's interests. In contrast, collaborative recommenders use the preferences of "similar" users, rather than the characteristics of an item, to make suggestions to the current user. More recently, utility-based, knowledge-based, and demographic techniques have been introduced for reasoning about recommendations. Most existing recommender systems are dealing with several challenges and techniques, such as multi-criteria ranking (e.g., [11,17,42,50,72,91,101]), intrusiveness of feedback (e.g., [1,2,34]), diversity of recommendations (e.g., [17,41,63,63,73,81,101,109]), and utility function elicitation (e.g., [10,15,28,43,79,93,94,105,108,110]). Many commercial and research prototype systems employing one or more of these techniques have been proposed to recommend flights, movies, restaurants, and news articles (e.g., [1,22,61]). However, these systems recommend only atomic products or services. Composite services are characterized by a set of sub-services, which, in turn, can be composite or atomic and make the recommendation space very large (exponential in number of components, or infinite, for continuous case), and implicitly defined. Most existing recommender systems provide users with a list of atomic items as a recommendation, e.g., a book or DVD. However, several domains can benefit from a system capable of recommending packages of items, in the form of bundled

sets, such as travel packages or electronic systems configuration (e.g., stereo entertainment, personal computer).

Addressing challenges outlined above in the context of complex recommendation models involving composite alternatives such as product configurations and service packages adds another dimension of complexity and have not been addressed. While there are few research recommender systems (e.g., [91,122]) that provide product bundling, the selection of atomic products in a bundle is done manually by the user as in [91]. From the system perspective, it is still an "atomic" offering. Work presented in [122] introduced a "shopbot" to search online for the lowest price of a bundle of products. Similarly, there are a few commercial recommender systems such as Expedia and Travelocity that provide partial product bundling, but again, the composition needs to be performed manually by the user, where the system is acting as a filtering agent to limit available options as the user moves from one atomic service to another. I further details research challenges and related work in Chapter 2.

1.3 Problem Statement and Summary of Contributions

As discussed, the existing recommender systems support only atomic products and services. The objective of this research is to develop *a unified framework for composite recommendations* based on decision optimization. The framework contains models that allow for fast and easy user *preference elicitation*, which is captured as a utility function, and provides an efficient *algorithm to diversify a set of recommendations*. Such recommendations

are dynamically defined using database views extended with decision optimization based on mathematical programming. In addition, this research conducts comprehensive *validation studies* to measure the effectiveness and efficiency of the proposed approach. A key problem to address in this research is to combine the flexibility of diversity ranking functionality with the information processing capabilities to learn and capture the preferences of the user through an iterative learning process. More specifically, the key contributions of this research are:

1. **Framework for recommending composite products and services:** I developed a framework based on the CARD approach [5], extended with a two-phase iterative process of 1) utility function elicitation and 2) diversification of a recommendation set. While there are several recommender systems that exist (e.g., [1,22,104]), their scope is atomic products and services. The proposed "**Decision-Guided Recommender with Composite Alternatives (DG-RCA)**" framework supports composite product and service definitions, and recommendations are based on dynamically learned utility function and decision optimization. Composite services are characterized by a set of subservices, which, in turn, can be composite or atomic. This leads to a very large recommendation space (exponential in the number of components). As a result, the existing ranking methodologies would not scale when the number of attributes is very large. The proposed framework involves fast and easy interaction with the user to (a) dynamically elicit the weighted utility function, and then (b) produce a diverse set of recommendation that contains an optimal recommendation in terms of

the estimated utility function. Finally, there are several hybrid recommender systems exist where two or more recommendation techniques are combined to produce better recommendation results. DG-RCA learns the utility of the user with a collaborative filtering technique then cascades the result set to be refined with a knowledge-based technique based on diversity space, and finally presents the user with a diverse set of composite products and services. According to popular hybrid recommender surveys [13,22,40], and to the best of my knowledge, there is no hybrid recommender system that combines Collaborative Filtering and knowledge-based recommendation techniques in this sequence. DG-RCA is the first recommender system that utilizes a hybridization approach based on a cascaded collaborative filtering and knowledge-based recommendation techniques.

2. **Utility function elicitation and learning:** I developed methods for utility function elicitation and learning, including cases when no prior knowledge of user preference is given. The learning process continues as feedback is extracted from the user. Methods are based on a low-dimensional utility space (as opposed to the $high$-dimensional recommendation/diversity space). Because we are working in composite products and services space, the number of attributes is very large due to the multitude of composite services. Consequently, the recommendation/diversity space is too $high$-dimensional to learn the recommendation utility accurately and needs an exponentially large learning set and time for accurate learning.

   The method is based on iteratively refining a set of axes in the n-dimensional utility

space, starting from the utility space standard axes when no historical learning is utilized. At every step, the user is asked to rank a set of recommendations, each being optimal for one of the current axes. Based on the user feedback, the method refines the set of axes that become closer to each other. Consequently, the utility function is constructed.

To add breadth to the research, the utility function elicitation is enhanced to be adaptive with a regression analysis technique, to predict the preferences of the current user. The preference learning is based on an historical multi-criteria rating submitted by the same user on similar products or services. Using a consistent family of criteria $m_1, \cdots, m_n$, each criterion is represented by a rating given by the user as a real value m_i in the range of $[m_i*, m_i^*]$ where m_i* and m_i^* are the worst and the best level of the i-th criterion respectively.

In addition, utility elicitation is expanded to include learning from other users with similar preferences, where historical total rating data is available for users. A new technique is introduced for "collaborative filtering" learning of the current user. It estimates similarity among users based on a confidence measure indicating the degree to which we rely on these users' ratings when predicting ratings for the current user. The technique is light yet efficient, and is based on a threshold for the number of co-rated items. It enforces a positive correlation threshold between any two users.

Works of Nielsen et al, Suryadi et al, Russell et al, and Chajewska et al [73,96,99 ,101] elicit the utility function from a database of observed behavioral patterns, while

Chajewska [111] focuses on eliciting the utility function from a database of already elicited utility functions. All of these approaches are targeted to produce a utility function from a database. While few recommender systems provide for estimating and refining the preferences of the user [68,106,116,120], works such as Pazzani [81] have exemplified the need for such techniques. However, none of these works, to the best of my knowledge, work on recommendations for composite product and services, which makes the recommendation space very large.

3. **Methods for diversifying a recommendation set:** I developed methods for diversity using the separation of utility space from recommendation/diversity space. Returning the user a set of recommendations based $only$ on the accuracy measure of the utility function learned could result in a set of recommendations where items are often similar to each other and do not have sufficient diversity. Working with composite space has the challenge of the selecting from larger space, but also gives an opportunity to diversify using higher-dimension recommendation/diversity space. One of the key contributions of this research is the "novel" approach for separating utility space from recommendation/diversity space, and basing the diversification of the returned set on the recommendation/diversity space. The result set contains a recommendation that optimizes the learned weighted utility function. In addition, I present a randomized algorithm that provides a competitive solution with respect to finding a diverse set from candidate recommendations. The algorithm is lightweight yet efficient. The idea of the algorithm is to iteratively relax the selection by the

Greedy algorithm [16,101] with an exponential probability distribution. This relaxation allows the algorithm to identify a better solution with high probability. Finally, the randomized diversity algorithm is enhanced to be adaptive based on learning. To the best of my knowledge, this research is the first attempt to incorporate learning when diversifying a recommendation set.

The work presented in [16,52,81] details several algorithms for selecting diverse recommendation alternatives based on the similarity of individual attributes. The work done by Linden, et al [64], also suggests a diverse ranking algorithm. These methods choose a set of alternatives based on a distance measure calculated for each of the multiple criteria. Zhang and Hurley [73] used a similar approach with respect to calculating diversity, but their similarity measure of recommendations was based on a set rather than individual recommendations. However, all referenced work above provides diversity by compromising similarity in a way that optimizes the similarity-diversity trade-off, namely uses the same space for similarity and diversity. In addition, learning has never been leveraged to impact the scope of a candidate set when diversifying.

4. **DG-RCA research prototype:** I developed the relevant components of DG-RCA to model highly-complex service compositions for the travel domain, which includes accommodations, rental vehicles, and air transportation. In addition, components of DG-RCA were developed to demonstrate the applicability of the DG-RCA framework for the movie domain as described by Yahoo! movies and MovieLens datasets.

Some prototype travel domain systems exist (e.g., ATA , ITR [91],SPIRE [51]); however, all of these prototypes recommend atomic travel services, except for ITR, which works with composite products. However, the composition is performed manually by the user. The relevant components of the proposed prototype provides alternative solutions of packages for users and decision makers based on the different models and methods defined.

5. **Extensive validation studies:** I conducted extensive validation studies to prove the effectiveness and efficiency of different components of DG-RCA framework. With the approval from Human Subjects Review Board (HSRB) at George Mason University, I conducted a user study of 30 users to measure quality and convergence of the DG-RCA framework. Case studies are limited by how many users are interviewed and do not scale. In addition, I used both synthetically generated datasets as well as data extracted from a publicly available travel site. Finally, I used publicly available datasets by Yahoo! Movies, and MovieLens. For example, Yahoo movies dataset has multi- criteria ranking of 34,800 ratings for 1,716 users. MovieLens dataset (movielens.umn.edu) has 10 million ratings for 10681 movies by 71567 users. Both datasets were split into training data and test data.

   I examined the efficacy of the DG-RCA framework by developing specific performance measurements of the algorithms proposed, and compared with state-of-the art available algorithms [e.g., 17,73,101]. Validation studies conducted show that the DG-RCA framework significantly outperforms competing algorithms and systems in

terms of precision, recall, and MAE (mean absolute error) [1,104].

1.4 Thesis Statement

Decision-Guided Recommender can be developed to support composite alternatives, which will:

- *Automate bundling of products and services selection in an optimal way.*
- *Outperform existing recommender systems in terms of quality of composite recommendations.*
- *Be feasible in terms of algorithms efficiency and scalability.*

The approach undertaken in this research involved extensions of the work done by Brodsky, et al, to develop a recommender framework that supports composite product and service definitions. The following chapters provide more details about areas I worked on.

This dissertation is organized as follows:

- Chapter 2: Related work
- Chapter 3: Composite Alternatives Framework (DG-RCA framework). In this chapter, I cover the main components of DG-RCA and how they interact with each other.
- Chapter 4: Iterative Utility Elicitation for Diversified Composite Recommendations. In this chapter, I describe the utility function elicitation method when no historical

data is available (using the utility space standard axes), and the diversity method, which uses the same space used by utility function elicitation.

- Chapter 5: A Randomized Algorithm for Maximizing the Diversity of Recommendations. In this chapter, I describe the refined diversity approach and algorithm.

- Chapter 6: An Adaptive Utility Learning Method for Composite Recommendations. In this chapter, I present a method for learning the user preference based on multi criteria ranking.

- Chapter 7: A Confidence-Based Recommender with Adaptive Diversity. In this chapter, a collaborative filtering technique is introduced to learn the preference of the user when total rating historical data is used. In addition, the diversity method is refined to be adaptive and incorporate learning.

- Chapter 8: Conclusions and Future Work.

Chapter 2: Related work

2.1 Introduction

A recommender system can be viewed as a complex service, which is composed of inter-related components, such as users, items (products or services), utility function, and the returned recommendation set. The problem can be formulated as follows [1]:

- Let C be the set of all users and let S be the set of all possible items that can be recommended.
- Let u be a utility function that measures the usefulness of item s to user c.
- For each user $c \in C$, we want to choose such item $s \in S$ that maximizes the user utility u.

Understanding the preference of the user and knowing what is available in terms of products and services are two crucial elements for recommender systems to succeed. Another challenge is how the recommender leverages the utility better to enhance the matching process.

Recommender systems have been extensively studied since the mid-1990s. Recent popular surveys (e.g., [1,22,68]) classify the current generation of recommendation in many

ways. Figure 2.1 shows an example of how many ways recommender systems can be classified. Of interest to this research is classification based on approach, and classification that is based on recommendation techniques. Starting with classification based on approach; there are several categories such as content-based, collaborative, and knowledge based recommendation approaches.

Content-based systems (e.g., [1,81,87]) often employ classifier techniques that rely on information retrieval and information filtering to recommend an item to a user based upon a description of the item and a profile of the user's interests. The user will be recommended items similar to the ones the user preferred in the past based on item profiles. More formally, the utility $u(c, s)$ of item s for user c is estimated based on the utilities $u(c, s_i)$ assigned by user c to items $s_i \in S$ that are "similar" to item s [1]. An example would be searching a flight using Expedia.com, where we provide keyword(s) and the recommender system matches the keyword with flights in its repository.

In contrast, Collaborative recommenders (e.g., [1,14,56,58,71,112]) use the preferences of "similar" users, rather than the characteristics of an item, to make suggestions to the current user. The user will be recommended items that people with similar tastes and preferences liked in the past. Try to predict the utility of items for a particular user based on the items previously rated by similar users (stereotypes). A collaborative-based system uses aggregation in some ways. In the simplest case, it uses simple average, or uses the weighted sum, where closer users are weighted more.

The hybrid approach (e.g., [4,62,67,100,107]) combines methods from collaborative and content-based approaches. More recently, utility-based, knowledge-based, and demographic systems have each suggested different techniques for providing recommendations. Systems employing one or more of these techniques have been proposed to recommend flights, movies, restaurants, and news articles (e.g., [1,22,61]). In sections 2.2 - 2.5 I provide more details about each.

Another way to classify recommender systems is based on techniques, either heuristic-based (e.g., [112]) or model-based (e.g., [58]). The main difference between model-based techniques and heuristic-based techniques is how to calculate the utility (rating) predictions. In the heuristic-based approach, calculation of predicted utility (rating) is based on some ad hoc heuristic rules, whereas in the model-based approach, calculation is based on a model learned from the underlying data using statistical learning techniques [1].

The research community has identified several areas where recommender systems can be enhanced to produce better outcome that meets and exceeds users expectations (e.g., [1, 5, 17, 18, 58]):

- Improving the understanding of users and items: In most Todays recommender systems, there is a basic understanding and information usage in user and item profiles; most recommendations is based on secluded matching between one or two features in either the user or item profiles. Sophisticated analysis and artificial intelligence methodologies are becoming a necessity. An example would be keeping a track of users transactional histories to enable the learning process. The profile of user i can

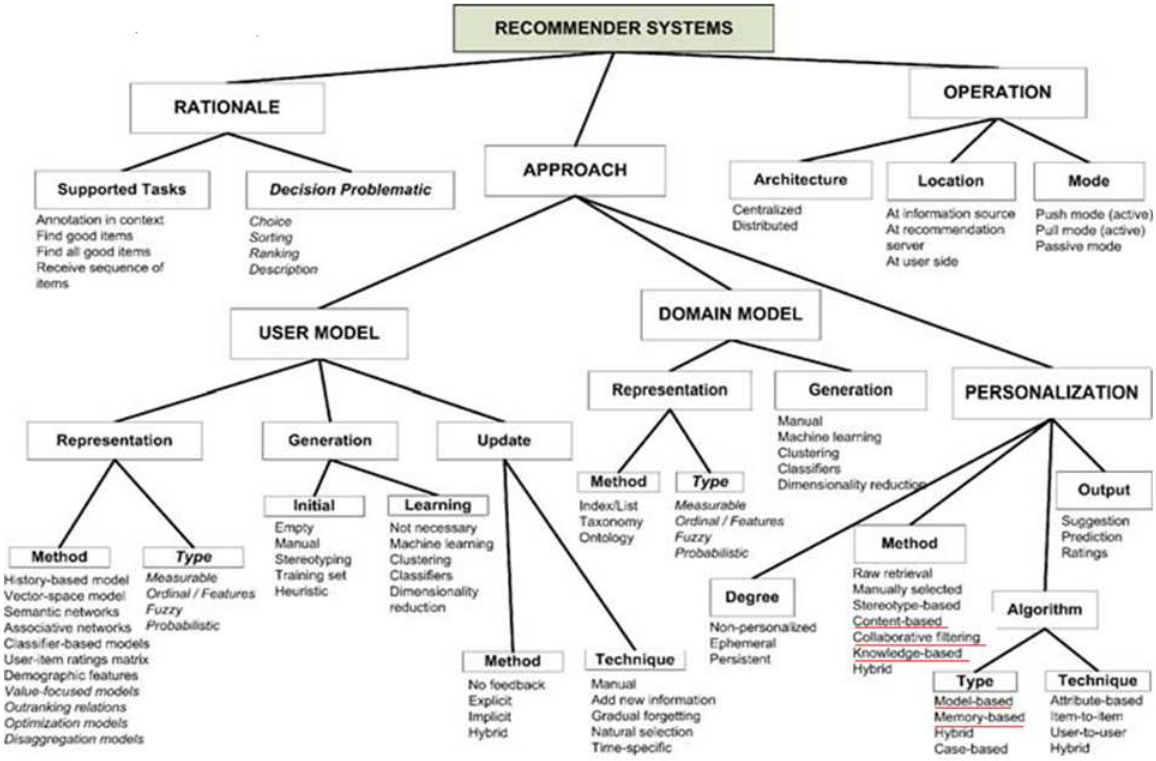

Figure 2.1: Framework for the analysis and classification of recommender systems. [68]

be defined as a vector of p features, i.e., $C_i = (a_{1i}, \cdots, a_{ip})$, Also, let the profile of item j be defined as a vector of r features, i.e. $S_j = (b_{j1}, \cdots, b_{jr})$. Features can mean different concepts in different applications, such as numbers, categories, rules, sequences, etc.

- Incorporating the contextual information into the recommendation process: Most existing recommender systems are limited to two dimensional space where only USER

x ITEM space is considered. There might be some situations where more than two-dimensional User x Item space is appropriate and needed. An example would be in the travel recommender system s domain some considerations of time season is a must where a package during summer time may not be appropriate to recommend at New Year season.

- Supporting multi-criteria ratings, in some applications, such as restaurant recommenders, there is more than the price that would attract users to enjoy a restaurant, in this case restaurant ratings should be based on all or most criteria that users would pay attention to before they proceed with their choices, it would be important to consider most or all of the following in the rating process : food, decor, and service

- Providing more flexible and less intrusive types of recommendations: Many Recommender systems (RSs) are intrusive where they require explicit and significant feedback from the user. Feedback will continue to be a primary factor in the recommender system concept, however, new generation of recommender systems should look for new ways to extract information from users implicitly. An example would be how long the user spend reading a specific document to infer how much the user liked this document and give it higher rating without explicitly asking the user. How to determine an optimal number of ratings the system should ask from a new user remain a challenge, for example, MovieLens.org first asks the user to rate a predefined number of movies (e.g., 20). In the MovieLens.org case, the cost of rating each movie is c and the cost of rating n movies is c*n). Then, the intrusiveness problem

can be formulated as an optimization problem; each additional rating supplied by the user increases the accuracy of recommendations.

- Providing a balance between diversity and optimality: Most recommender systems only limit the scope of recommendations to how similar the result set is to the query submitted by the user without incorporating enough diversity in the result set. Providing diversity to the user will demonstrate different dimensions of possible choices that users can choose from. Basing our result only on similarity might reduce the users ability to make the smartest possible choice.

2.2 Content-based Recommender

The user will be recommended items similar to the ones the user preferred in the past; based on item profile. Mainly based on information retrieval [3,19,119] and information filtering research [35,69], an example would be searching a document in Google website, where we provide a keyword and Google recommender system will match it with documents in its repository (the web). In content-based recommendation methods, the utility $u(c, s)$ of item s for user c is estimated based on the utilities $u(c, s_i)$ assigned by user c to items $s_i \in S$ that are "similar" to item s [1].

There are some Limitations with content-based approach such as [1]:

- Limited by the features of the objects: parsed automatically by a computer or assigned manually; so the easier the parsing of object features the more powerful the content-based recommender system becomes.

- Two items represented by the same set of features are indistinguishable, and example is two papers are published on the web and both contain same keywords, then a search for those keywords will result in Google returning both documents without (to a degree) knowing which paper is better in term of quality.

- Overspecialization: due to the fact that content-based recommender system try to match what we liked in the past with features of existing items, it tends to only return similar objects to the category of items we liked in the past, this result in less diverse list of recommendation; however we can introduce some randomness as some papers stated but this also introduce another challenge that is if you are only limited to return the user a limited number of recommendation (usually small) then this will impact the quality of our result set if some of those recommendations are selected in a random way that we are not sure of the degree of users‘ likeness.

- New User Problem : The user has to rate a sufficient number of items before the system can really understand the users preferences.(start-up problem)

2.3 Collaborative Filtering

As one of the most promising and successful approaches to building recommender systems, collaborative filtering (CF) leverages the known preferences of certain users to make recommendations or predictions of the unknown preferences for other users. The user will be recommended items that people with similar tastes and preferences liked in the past.

Try to predict the utility of items for a particular user based on the items previously rated by other users (stereotypes). Collaborative based systems use aggregation in some ways: In the simplest case uses simple average, or uses the weighted sum, where closer users are weighted more. Early generation collaborative filtering systems, such as GroupLens [61], use the user rating data to calculate the similarity or weight between users or items and make predictions or recommendations according to those calculated similarity values. CF techniques have less impact of overspecialization, where it can recommend any item, even the ones that are dissimilar to those seen in the past, however there are some limitations [1]:

- New User Problem: as described in content-based systems.
- New Item Problem: Until the new item is rated by a substantial number of users, the recommender system would not be able to recommend it.
- Sparsity: the success of the collaborative recommender system depends on the availability of a critical mass of users, so if only a small number of users used the system and those users share a taste that is different than the norm then the outcome of the collaborative-based system may not be predictable nor useful; to overcome this issue we can employ some “demographic filtering” until a certain number of users use the system to lessen the effect of sparsity.
- Gray Sheep: Gray sheep refers to those users with opinions that are not consistently agree or disagree with any group of people and thus do not benefit from collaborative filtering [104].

Collaborative filtering approaches are surveyed in [1,14,62,104,112]. Collaborative filtering can be memory-based or model-based. With memory-based methods, the similarity correlation can be based on items or users [1,104], there are several limitations for the memory-based technique since the similarity values are based on common items. As a result, memory-based techniques could become unreliable when data are sparse and the common items are therefore few [112]. Model-based approach is becoming more popular and more research is focusing on it [1], Model-based methods first learn a a model and then use it for predicting the suitability of an item to a user. To alleviate the data sparsity problem, many approaches have been proposed such as e.g., latent semantic model [45] Singular Value Decomposition [55,105].

As noted in [1,6,7,8,17], research in recommender systems is moving toward the quality of the recommended set from the accuracy of the prediction. Pair-wise preference between items for users, e.g., EigenRank [65], probabilistic latent preference analysis [66] and Bayesian probabilistic ranking [89], however, all these approaches can suffer significantly from expensive computations. In our previous work [7], we have demonstrated how using EigenSolver to produce 5 diverse recommendations could take 225 seconds, while it took only 1 second for other techniques. A solution that requires external Solvers, does not scale because it requires solving a binary combinatorial problem with a binary variable per each recommendation. The computational overhead of this approach becomes more pronounced in the case of composite recommendations where the number of candidate recommendations is large.

2.4 Knowledge-based systems

Knowledge based methods are gaining popularity in recently, where the reliance on user or item profiles is less, and true intelligence is added to the recommendation process. Models used for the purpose of recommendation are based on inferring what the customer might be interested in. As stated by Burke [22] “The knowledge used by a knowledge-based recommender can also take many forms. Google uses information about the links between web pages to infer popularity and authoritative value”. Knowledge-based systems are distinguished where no portfolio effect exist. This significantly reduces the drawbacks of new user and item profile. There are three types of knowledge that are involved in such a system [1]:

- Catalog knowledge: Knowledge about the objects being recommended and their features. For example, a restaurant recommender system should know that “Thai” cuisine is a kind of “Asian” cuisine.

- Functional knowledge: The system must be able to map between the users needs and the object that might satisfy those needs. For example, the system must know that a need for a romantic dinner spot could be met by a restaurant that is “quiet with an ocean view”.

- User knowledge: To provide good recommendations, the system must have some knowledge about the user. This might take the form of general demographic information or specific information about the need for which a recommendation is sought.

2.5 Hybrid approaches

Hybrid recommender systems combine two or more recommendation techniques to maximize better performance by using one technique and minimize the drawbacks of any other individual one. There are many different ways to leverage hybridization [13], most commonly, collaborative filtering is combined with some other technique to avoid the ramp-up problem exists with CF techniques [1,22]. Most recent, hybrid techniques are also leveraging Context-aware in areas of social networking [4,100,107], e.g. Facebook.com , furthermore, a new class of recommenders are introduced (reciprocal) where people are both the subjects and objects, i.e., online dating [67]. Work in [123] presented a hybrid filtering method and a case-based reasoning framework for enhancing the effectiveness of Web search. Below I list some common methods combination:

- **Weighted**: A weighted hybrid recommender is one in which two or more score of each recommended item is computed. Each score is a result of recommendation technique in the system. For example, the simplest form is a linear combination of recommendation scores [22]. It is important to point out that each score is calculated separately by the recommendation technique. Depending on the hybridization method, the final score is determined. The P-Tango system uses a simple average hybrid method using collaborative and content-based recommenders [30], however

it gradually adjusts the weighting of predictions as the learning about the user improves. Other weighted hybrid techniques do not use numeric scores, rather a voting mechanism is utilized, an example is the hybridization of collaborative, content-based and demographic methods presented in [82].

The benefit of a weighted hybrid is that the strength of each method contributes to the overall capabilities of the recommendation process and adjustments of weights can follow accordingly. However, it is hard to assume that scores of different methods used are reliable across the space of possible items. An example is that CF technique will continue to suffer in terms of accuracy score for those items with a small number of raters [22].

- **Switching**: A switching hybrid utilizes a confidence measure to determine the switching in item-level. It uses some criterion to switch between recommendation techniques. For example, it calculates the predication and the confidence level of an item, if the confidence falls below a threshold, then the hybrid recommender switches to different recommendation method to determine the suitability of that item.

  The DailyLearner recommender uses a content/collaborative hybrid where a content-based method is attempted first. Depending on a confidence level, if it is low then a collaborative method is used. This switching in The DailyLearner recommender is expected to suffer from the ramp-up problem, which exists in both the collaborative and the content-based systems [22].

  Hybrid systems that use Switching technique, need to account for switching criteria.

This introduces additional complexity into the recommendation process. However, having the option of switching in such hybrid systems is an advantage where the strengths and weaknesses of combined methods can be leveraged and avoided respectively.

- **Mixed**: Some Hybrid systems use a Mixed technique with respect to determining what recommendation to include in the presented result set. The hybrid system initially produces a large number of recommendations simultaneously. For example, some of those recommendations were produced using CF technique while others were determined using content-based technique, finally the hybrid recommender will mix some of each and present a result set to the user. Usually, a ranking of items or selection of a single best recommendation is needed, at which point some kind of combination technique must be employed [22].

  Smyth and Cotter [102] uses this approach to assemble a recommended program called "The PTV system" for suggesting television viewing. Based on the description of TV shows, it uses content-based techniques and collaborative information about the preferences of other users. Recommendations from the two techniques are combined together in the final suggested list. The mixed hybrid avoids the "new item" start-up problem due to the use of the content-based component, but it continues to suffer from "new user" start-up issue exists in CF and content-based systems. Other implementations of the mixed hybrid, ProfBuilder [118] and PickAFlick [23,24], present multiple recommendation sources side-by-side.

- **Feature Combination**: Another way to build a hybrid system is to consider the information of specific technique as simply additional feature data associated with each example and use another technique over this augmented dataset.

  For example, the inductive rule learner Ripper was applied to the task of recommending movies using both user ratings and content features. As a result, Basu et al [16] report on experiments shows they achieved significant improvements in precision over a purely collaborative approach. However, the improvement is precision did not hold when authors considered all available content features.

  The feature combination hybrid lets the system consider collaborative data without relying on it exclusively, so it reduces the sensitivity of the system to the number of users who have rated an item. Conversely, it lets the system have information about the inherent similarity of items that are otherwise opaque to a collaborative system.

- **Cascade**: The key in using the Cascade approach is the sequence in a staged process. In this technique, one recommendation technique is employed first to produce a list of candidates and then a second technique is applied on this produced list of candidates to determine the final list of recommendations. Burke [22] presented the restaurant recommender EntreeC, which is a cascaded knowledge-based and collaborative recommender. EntreeC starts off with knowledge of restaurants to make recommendations based on the users stated interests. The resulting recommendation list are of equal preference at this point in time, and then a collaborative technique is employed to determine the top five restaurants to present the user with.

Due to the notion of staged process in cascading techniques, it allows the system to only apply the second technique on smaller number of candidates that are believed to be of high quality based on the first technique. This is specifically important when we deal with composite recommendation space, where the number of recommendations is significantly large, in addition to the number of attributes representing each recommendation. We avoid the second technique on items that are sufficiently poorly rated by the first technique. Because we are only applying the cascade's second step on a smaller set of items that scored higher with respect to utility of first technique, it is more efficient than other hybridization techniques such as, weighted hybrid that examines all items, which could result in a severe scalability impact in the case of composite products and services. In addition, the cascade is accommodating by its nature where all items made it to the list will have equal chances of rating predication by the second technique. However, it is important to point out that the first technique needs to be chosen carefully as its ratings can only be refined, not overturned.

2.6 Diversity of recommendation set

Diversity has been studied in many science areas such as social, physical and management (e.g., [41,59,63,76,109]). One of the main enablers of diversity is incorporating multi-criteria ratings. The diversity concept has not been fully explored and utilized in most of the existing recommender systems. With the recent surge in collaborative similarity-based recommenders, such as Amazon.com, a number of multi-criteria ranking methods have

been proposed. Of significant importance to this research is work suggesting the importance of diversity-sensitive recommendation sets to provide a balance between diversity and optimality, and the need for presenting the user with a range of options and not with a homogeneous set of alternatives [1,5,6,7,9,17,26,31,41]. However, most recommender systems limit recommendations to those that are relevant to users' requests. Therefore, their recommendations are often similar to each other and do not provide enough diversity.

Several researchers have presented informal arguments that diversity of recommendations is often a desirable feature in recommender systems and could be as important as similarity in some cases [17,101]. For example, news article service Daily-Learner filters out items not only if they are too different from the users' preferences, but also if they are too similar to something the user has seen before [1]. In addition, [17,73,101] discussed measures to evaluate the novelty of a specific recommendation. Furthermore, [32] presented a statistical dispersion called the Gini coefficient to measure the effect of sales diversity in recommender systems. The Gini coefficient is a common measure of distributional inequality. This paper examined recommender systems effect on buying behavior and offered initial evidence that recommender systems do influence sales diversity. Also, [26] shows that a list of diverse recommendations scored lower in accuracy measure but users liked that diverse set more, compared to a none diverse set.

Presented in [38] are several GRASP algorithms for MDP and tested them on medium size datasets. Their optimum solution strategy required long processing time (20 hours of CPU time). Proposed in [31], are tabu search-based algorithms for MDP. The proposed algorithm is based on the tabu search methodology and incorporates memory structures for

both construction and improvement. The work [39] proposed different GRASP heuristics for MDP, which used “distinct construction procedures” that includes “a path-relinking” technique. However, their best technique took 10 sec to find the target solution for a set of 200 candidates. The work in [85] proposed a branch and bound algorithm, which on average took 12.51 sec to find an optimal solution on a set of 50 recommendation, which is much smaller than the number of composite recommendations we work with.

The work presented in [17,72,101] details several algorithms for selecting diverse recommendation sets based on the similarity of individual attributes. Three different approaches are presented in Figure 2.2 [17], namely BoundedRandomSelection, GreedySelection, and BoundedGreedySelection where diversity is given by compromising similarity in a way that optimizes the similarity-diversity trade-off.

BoundedRandomSelection is randomly chooses from a bounded result set that is the most similar to the target result. The other approach is called GreedySelection, where selection is picked from all domains of results; the most similar recommendation is picked first. Consequent recommendations are chosen based on their quality. Quality is determined as a combination of how similar the recommendation in question is to the target and how diverse the recommendation is from already picked recommendations. This approach is expected to provide more diversity than the previous approach because the result set is larger and the chances of missing a more diverse recommendation in the BoundedRandomSelection approach are more likely to happen. The third approach is a combination of both approaches mentioned, where a diverse set of recommendations is constructed from a bounded result set. Of interest is GreedySelection approach, where diversity of a set is

```
t: target query, C: case-base, k: # results, b: bound

1.   define BoundedRandomSelection (t, C, k, b)
2.   begin
3.     C' := bk cases in C that are most similar to t
3.     R := k random cases from C'
4.      return R
5.   end

1.   define GreedySelection (t, C, k)
2.   begin
3.     R := {}
4.     For i := 1 to k
5.       Sort C by Quality(t,c,R) for each c in C
6.       R := R + First(C)
7.       C := C - First(C)
8.     EndFor
9.   return R
10.  end

1.   define BoundedGreedySelection (t, C, k, b)
2.   begin
3.     C' := bk cases in C that are most similar to t
4.     R := {}
5.     For i := 1 to k
6.       Sort C' by Quality(t,c,R) for each c in C'
7.       R  := R + First(C')
8.       C' := C' - First(C')
9.     EndFor
10.  return R
11.  end
```

Figure 2.2: Diversity approaches by Smyth [17]

calculated by adding the distance between every 2 members in the set. The total is normalized to give the total diversity of a set. The maximum diversity is not guaranteed with the GreedySelection approach as the first recommendation selected is always the one with the highest similarity to the target and, in every subsequent iteration, the recommendation selected is the one with the highest diversity with respect to the set of recommendations already selected during the previous iterations. As it stands this algorithm is expensive. For a case-base of n candidate recommendations, during each of the k iterations we must calculate the diversity of each remaining case relative to those recommendation(s) so far

selected. This means an average of $(n - k/2)$ relative diversity calculations in each iteration, or $k * (n - k/2)$ calculations overall. Similar to [72], the work by Zhang [73] used a similar approach with respect to calculating diversity. However, measuring similarity of recommendations was based on the set rather than individual recommendations within a set. A recommendation with low similarity to the target might make it to the list because the set it belongs to has a similarity score that is above a threshold with respect to the whole set. In addition, in problem two of the paper, a set is constructed under the constraint that the diversity of the set is greater than some diversity threshold and the total similarity between the elements of the set and the target is maximized. The recommendation set is calculated by engaging an external quadratic programming solver. It is not clear whether this approach can scale well in case of composite recommendation space as they introduce a binary vector y to indicate if the item is part of the final list or not. This raises the question if a programming solver is the best strategy or if a lightweight algorithm (heuristics) would be sufficient for finding a set with optimal or near-optimal diversity. However, this work is based on measures such as Euclidean distance or Hamming distance calculated for pairs of attribute values. The work done by Linden, et al [64], also suggests a diverse ranking algorithm. Like the Diversity layering algorithm presented in this paper, the method proposed in [64] also makes recommendations by finding solutions that optimize one attribute of the solution; however, the multi-criteria ranking method presented here optimizes each attribute while bounding the allowable degradation in overall utility. In this way, the recommendations made by the Diversity layering algorithm offers the user a broad view of the solution space while maintaining an acceptable overall utility.

Furthermore, authors of [33] discussed how diversity can be increased where similarity is fully preserved using layering technique. Also, [33] discussed the case where similarity is strictly relaxed to allow some loss of similarity to optimize diversity. In addition, [73] presented the competing goals of maximizing the diversity of the retrieved list while maintaining adequate similarity to the user query as a binary optimization problem, where the similarity measure of recommendations was based on a set rather than individual recommendations. Also, as observed in experiments described in Chapter 5, Section 4, the approach in [73] does not scale because it requires solving a binary combinatorial problem with a binary variable per each recommendation. The computational overhead of this approach becomes more pronounced in the case of composite recommendations where the number of candidate recommendations is large.

As adopted in this research, researchers in [17,72,73,101] used the definition of diversity as the average dissimilarity. However, their referenced work above provides diversity by compromising similarity to optimize the similarity-diversity trade-off, namely use the same space for similarity and diversity. This is in contrast to our approach where we separate utility space from diversity space. The separation would result in similarity and diversity complementing each other, and not competing over the same goal. Recommendations made by our diversity approach offer the user a broad view of the solution space while maintaining a superior score against the overall utility.

Finally, to the best of our knowledge, most work that has been published deal with atomic products and services, while our work is addressing recommendations for composite products and services, which makes the recommendation space very large (or infinite, for

continuous selected quantities). This is in particular important when we compare with solution strategies that use external solvers. With such techniques, the recommendation space is expected to be small or at most medium size to deliver good results in reasonable amount of time.

2.7 Utility function elicitation

Ideally, decision making should be based on full knowledge of the utility function of the decision maker. However, in many cases, this may not be possible all the time. Acquiring such knowledge may not be an easy task due to several reasons such as the size of the outcome space, the complexity of the utility elicitation process [113,118], and finally, the time allowed to elicit the utility function from the users' perspective [10,15]. This explains why, people and systems tend to make decisions with only partial utility information, therefore, whatever questions asked for the purpose of eliciting the utility need to be carefully chosen. Due to challenges stated above, we see a surge in research in the utility function elicitation.

In addition, there are two major categories of preference statements that we can be used to learn the preference of the user [110,111]:

1. Dyadic (comparative) statements, indicating a relation between two referents using the concepts such as "better","worse", and "equal in value to".

2. Monadic (classificatory) statements, evaluating a single referent using ordinal language concepts such as "good", "very bad", and "worst".

There are known techniques for learning with preference statements above such as ranking and rating respectively. However, Most of the preference statements that could be used may suffer from the "generalizing nature" where these statements usually can only capture the preference of a subset of attributes. This creates an ambiguity with respect to actual preferences of the user, an example would be the following: If we present the user 5 recommendations $A_1, \cdots, A_5$ then ask the user to rank or rate presented recommendations. If recommendation A_2 and A_4 stretch for example the saving dimension while recommendations A_3 and A_5 stretch on enjoyment dimension, then all of the above recommendations may have "other" attributes that the user may prefer or base their decision on, however those "other" attributes could or could not be captured in our utility function such as the location of the vacation or the number of days. So if the user indicates a preference toward recommendations A_2 (which stretches on saving) and A_5 (which stretches on enjoyment) but the two recommendations share some other attributes such as number of days or location then this preference choice of the user would make it harder for the utility function elicitation process to know the reason for the decision of the user, since number of day and/or location are not captured in the utility. Above example shows the need for careful selection of dimensions or attributes in the utility function.

There are several approaches for eliciting utility functions, most of which aim for semi-automated learning of a decision makers utility function. One approach is iterative learning and refinement of the users utility function using a value of information approach [113]. Another approach is by eliciting the utility function from a database of observed behavioral patterns [79,105,108,110]. A third approach is by eliciting the utility function from

a database of already elicited utility functions [111]. Most related to this research is the work by Jameson, et al. [53] and Linden, et al. [64] which presented models for eliciting partial utility and reasoning with such models with regard to certainty for domains such as shopping and airline reservations. Poh and Horvitz [83] presented the benefit of refining utility information. Work presented by Jimison et al. [54] discussed the value of explicitly representing uncertainty for some key utility attributes in medical decision models when interacting with users. Finally, [113] where they treat utility as a random variable that is drawn from a known distribution. The use medical databases of utility functions to estimate the distribution of utility functions in the population and then use this estimate as a prior when elicit utilities from the new users they advise. However, The space of possible outcomes depending on attributes used was only 70 which is extremely small compared to composite spaces.

While few recommender systems provide for estimating and refining the preferences of the user [68], works such as [81] have exemplified the need for such techniques. However, none of these works, to the best of our knowledge, work on recommendations for composite products and services, which makes the recommendation space very large (or infinite, for continuous selected quantities), and implicitly defined. In contrast, the DG-RCA framework deals with the recommendation space of composite products and services by using decision optimization when extracting recommendations that optimize specific axis in the utility space.

2.8 Summary of the Evaluation of Related Work

Recommender systems employ different representation models, ranking methods, and learning techniques to recommend solutions in a variety of domains. I have presented comprehensive analysis of recommender systems, and more details can be found in [1,22,68,108]. Most relevant to our work are those systems supporting multi-criteria ranking methods, utility function elicitation, dynamic learning of user preferences, and diversity of recommendations.

In summary of classifying recommender systems based on approach, Figure 2.3 presents the most common methods of recommendations such as content-based, collaborative, and knowledge-based recommendation approaches. For each technique, it lists:

- The background information needed beforehand about user and/or items.
- Input from the user during the recommendation process.
- Description of the recommendation technique.
- Example(s) of the technique.

The majority of recommender systems recommend only atomic products or services, and are designed for a single target domain and do not provide a general framework for the development of recommender systems. Complex recommendation models involving composite alternatives, such as product configurations and service packages, are rarely addressed. In addition, there is limited leverage of information in user and item profiles. Most recommendations are based on secluded matching between one or two features in

Technique	Background	Input	Process	Examples
Content-based	Features of items.	Ratings from the user of items.	Recommend an item to a user based on a description of the item and a profile of the user's interests	MyBestBetsTV, Newsweeder, Resturants
Collaborative	Ratings from users of items.	Ratings from the user of items.	Use the preferences of "similar" users, rather than the characteristics of an item, to make suggestions to the current user	Amazon
Demographic	Demographic information about users and their ratings of items.	Demographic information about the user.	Identify users that are demographically similar to you, and extrapolate from their ratings of i.	Grundy, Krulwich
Utility-based	Features of items.	A utility function over items that describes the user's preferences.	Apply the function to the items and determine i's rank.	*PersonaLogic, Tête-à-Tête*
Knowledge-based	Features of items. Knowledge of how these items meet a user's needs.	A description of the user's needs or interests.	Infer a match between items and the user's need.	EntreeC,

Figure 2.3: Summary of approach-based classification of recommender systems

either the user or item profiles [1]. Sophisticated analysis and knowledge of profiles are becoming a necessity to optimize individual criterion. The shift toward knowledge-based and utility-based recommender systems that can overcome the reliance on user and item profiles is still developing.

Multi-criteria recommender systems characterize recommendation alternatives as associated attribute-value pairs. Multi criteria decision making has been analyzed from several perspectives [11,42,120] in different science areas [50,77,106]. Authors of [116] listed

and analyzed several multi-criteria decision making Methodologies such as Analytic Hierarchy Process (AHP), Goal Programming, and Simple Multi-Attribute Rating Technique (SMART). However, the majority of recommender systems rely on a single ranking or utility score. In many applications, there are multiple criteria that need to be taken into account, such as price, quality and enjoyment. Recently, multi-criteria ranking has been explored in recommendation set retrieval [1,17,101]. These methods choose a set of alternatives based on a distance measure calculated for each of the multiple criteria. Case-based recommenders often evaluate recommendation alternatives according to their similarity to a target solution [17,72,91,101]. In contrast, utility-based recommenders make recommendations often based on a single utility score [22,68]. The DG-RCA recommender framework is used to construct utility-based recommenders; however, many of the DG-RCA components could be used to accommodate similarity-based recommendation. Multi-criteria ranking can help provide a balance between diversity and optimality. However, most recommender systems limit recommendations to those that are relevant to users requests. Therefore, their recommendations are often similar to each other and do not provide enough diversity. Diversity is important because it helps users become aware of choices they may not have thought of.

In addition, when diversity is used, the same space is used for utility and diversity resulting in "trade-off" between accuracy or utility optimality from one end, and diversity from the other end. Furthermore, all diversification techniques listed above diversify against all candidate set in a static way, while our diversification technique is adaptive where the scope of candidate set changes based on learning and the preference of each user.

There are several collaborative and diversity techniques that are published in the area of recommended systems. However, each of these techniques works independently, and not a in a hybrid way. According to an extensive survey of hybrid recommended systems by Burke [22], one of the hybrid areas that has not been explored yet is combining collaborative filtering technique with knowledge-based technique, which is what I am presenting in this research. To the best of my knowledge, this is the first attempt of a cascaded hybrid recommender to combine a collaborative filtering technique with a knowledge-based technique (adaptive diversity).

Finally, many recommender systems are intrusive where they require explicit feedback from the user and often at a significant level of user involvement. For example, before recommending any movies, MovieLens.org expects the user to rate a predefined number of movies (e.g., 20). This request comes with costs on the user [1]. However, DG-RCA does not require any explicit feedback prior to using the system. Initially, DG-RCA would recommend a set of alternatives to choose from or provide feedback on. In addition, DG-RCA uses a simple feedback extraction mechanism, where users are asked only to place recommendations in a stratum that is typically quick and easy. The number of recommendations to rank becomes smaller with each iteration. According to our user study in chapter 4 section 4, only 3 out of 30 people complained about the explicit feedback required. As will be discussed in more details in chapter 4 section 2, the utility function elicitation of DG-RCA allows for less intrusive learning. Initially, the DG-RCA is capable of recommending alternatives without the need to extract feedback. In addition, DG-RCA could work with previously calculated utilities. This utility can be obtained by domain experts

or calculated using historical data. Finally, the DG-RCA framework deals with the recommendation space of composite products and services by using decision optimization when extracting recommendations that optimize specific axis in the utility space. Such problems are more complex when the number of alternatives is large (possibly infinite); however, I noticed that our framework converges after two questions, as explained in the case study section. In general, the proposed approach is a unified framework to recommend composite products and services and incorporates multi-criteria ranking. The framework has a less intrusive user preference learning experience during the utility function elicitation process, and it has the ability to support the notion of a diverse recommendation set.

Chapter 3: Composite Alternatives Framework

3.1 Introduction

Recommender systems facilitates the decision-making process of users. Their value increases when interacting with large information spaces. They recommend items of interest to users based on preferences they have expressed, either explicitly or implicitly. While the state-of-the-art recommender systems focus on atomic products or services, this research focuses on developing a framework for recommending composite services and products based on decision optimization, and eliciting user preferences. The Decision-Guided Recommender with Composite Alternatives (DG-RCA) framework combines the flexibility of diversity ranking functionality with the information processing capabilities to learn and capture the preferences of the user through an iterative learning process.

The approach undertaken in this research involves extensions of the Alternative Recommendation Development (CARD) framework by Brodsky, et al, [5] to develop a recommender framework that supports composite product and service definitions. . The CARD framework leverages partial knowledge specification, decision optimization, and dynamic preference learning to select from composite recommendation alternatives - that is, multidimensional recommendations, made up of a number of subcomponents, which satisfy global decision constraints.

The framework is based on Decision Guidance Management System (DGMS) [18], DGMS is a productivity platform for the fast development of database applications requiring closed-loop data acquisition, learning, prediction, and decision optimization. DGMS data model extends the relational model with stochastic attributes over which a probability distribution function is defined, it leverages Decision Guidance Query Language (DGQL). DGQL, is used to specify several decision guidance views supporting prediction, optimization, and learning:

- Prediction: DGQL provides transformers to define new attributes. a transformer is a program that computes outputs from inputs. A prediction view is specified by a DGQL query, in which the WHERE and/or the SELECT clause involve a probabilistic logical formula that contains random variables.

- Decision optimization: uses choice-null (c-null) for some attributes in view definitions. A DG-SQL query that involves C-nulls essentially defines a set of (possibly stochastic) relations, each corresponding to a query answer with a different instantiation of values for C-nulls. A decision query returns an optimal answer, such as max or min.

- Learning, takes place using Learning null attributes or parameters (L-null). Learned parameters are unknown at the time of a transformers definition, but can be learned using a learning set produced by queries from the database.

This chapter is organized as follows. Section 3.2 describes the DG-RCA framework. Section 3.3 outlines the Composite Recommendation Knowledge Base (CRKB). Sections

3.4 3.5 and 3.6 describe the components of the framework and their interaction with the CRKB. Section 3.7 illustrates the CARD framework with a travel recommender example. Section 3.8 is the summary.

3.2 DG-RCA Framework

DG-RCA is a framework that supports composite product and service, top-k decision optimization, and dynamic preference learning. Subservices can be atomic or composite. DG-RCA combines the flexibility of diversity ranking functionality with the information processing capabilities to learn and capture the preferences of the user through an iterative learning process. The framework proposes and informs the user of what is available in terms of composite recommendations with minimum interaction from the user side. For example, travel packages are composed of many services including ground and air transportation, accommodations, and activities. Each atomic and composite service is associated with metrics, such as cost, duration, and enjoyment ranking. An example to demonstrate the use of the system is to consider a family that would like to reserve roundtrip air transportation, accommodations at two separate destinations, and a rental vehicle for transportation between the two destinations. The family can travel any time within a specified window and can travel to their destinations in any order as long as the flights arrive and depart from the first destination.

DG-RCA extends the CARD framework [5]. The framework consists of three steps:

- Clustering the recommendation space.

- Selecting the utility axis
- Diversifying the recommendations.

The recommendation process is depicted in Figure 3.1. As shown in the diagram, the process is initiated when the user submits a request to the Recommender that contains both the users decision constraints and profile data. When the Recommender receives the request, it uses the service instance data provided by external data sources and the recommendation definitions data stored in the Composite Recommendation Knowledge Base (CRKB) to start the clustering step to determine which cluster the user is interested in. The recommendation space is split into a number of clusters, where each cluster contains a number of packages (recommendations). Examples of clusters are: honeymooners, single, family, etc. The clusters are extracted from historical purchase data cross-referenced with user demographic data. However, this is beyond the scope of our research at this point. Currently, I just assume that these clusters exist and packages can therefore be targeted to specific user groups. The recommender will return a set of recommendations to the user. Each recommendation from the set will represent a cluster and maximizes the total utility function in its cluster. Initially I will give equal weight to metric attributes as we are in an early stage to conclude what the user might value more with respect to metrics attributes (e.g. saving, enjoyment). Domain knowledge could also be used to determine how to assign weights and selections of metric attributes, consequently calculating the global utility function. When the user indicates her preference and the chosen recommendation determines her preferred cluster, future recommendation space is limited to the chosen cluster.

Next, the Preference Learner starts the iterative process of learning the utility vector of metrics attributes, e.g., saving, location attractiveness, enjoyment. This step starts with presenting the user with a number of distinguishable recommendations in terms of utility vectors. Each recommendation returned will stretch the dimension it represents (e.g., saving) and relaxes on the other dimensions (e.g. enjoyment, location attractiveness, etc). The process continues iteratively updating the utility vector every time, based on the feedback of the user until an exit point is reached (e.g., indicating "no difference" between recommendations presented). Upon exit, the recommendation space will be constructed into the Recommendation View, according to the utility vector learned.

Finally, the Recommender constructs a set of diverse recommendations. Diversity is based on recommendation space, where each recommendation is characterized by a vector in n dimensional space. It starts with the recommendation that maximizes the utility function calculated, and each consequent recommendation represents a diverse choice within the recommendation space. The system provides the user with a given number of diverse alternatives to choose from. At this point, the user can select the most preferred alternative or she can provide a (partial) ranking of the suggested alternatives to the Feedback Extractor. Feedback from the user will be honored if she has chosen to view additional suggestions. Then ask the user to rank the set of recommendation presented, and weight vector of recommendation(s) ranked the highest will be used as a base to reconstruct the utility function and update preference parameters, consequently present a new set of diverse recommendation in an iterative fashion. The process ends if the user either chooses a recommendation (and consequently the services to implement the selected alternative are

invoked) or exits the recommendation process.

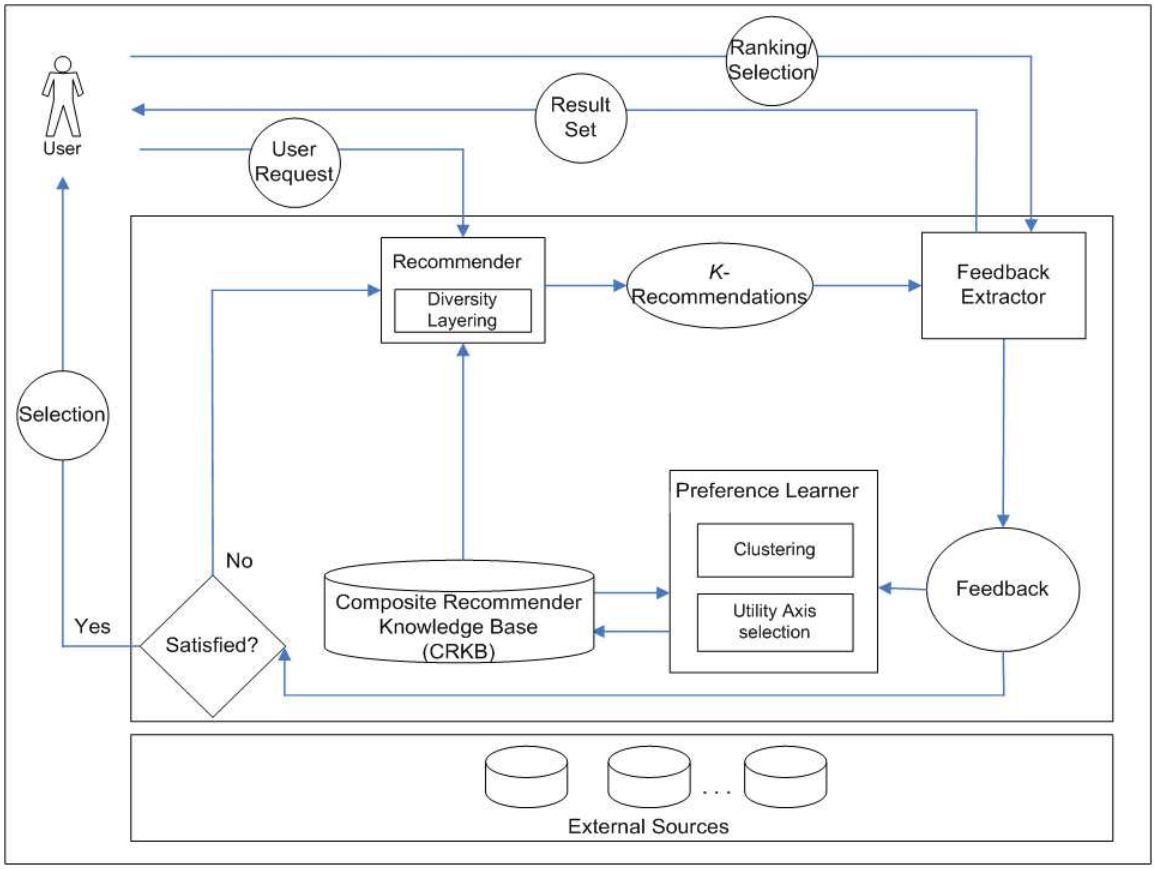

Figure 3.1: DG-RCA Framework

3.3 Composite Recommendation Knowledge Base (CRKB)

The CRKB stores the internal data and the modeling components used to construct recommendation sets [5]:

User Profile Maintains information about the current client and contains: domain-specific user data to select an appropriate recommendation template (demographic), and

Recommendation constraints, such as the number of dependents and the budget limit.

Service Metric Views Extracts information from raw tables based on the user profile and data, such as cost, duration, or enjoyment ranking that characterizes the service. Service Metric Views can be atomic or composite.

Recommendation Views Process and rank recommendations based on diversity ranking function (transformer). Each Recommendation View is associated with a Service Metric View. Transformers and Learned Transformer Parameters: Transformers are parametric functions that define metric attributes of a composite service given the values of metric attributes of the subservices. Some transformer parameters (e.g., coefficients) may not be known a priori, but learned, and they are called l-null.

Historical Preference Data and Current User Preference Data CRKB stores Historical Preference Data for each Recommendation View. For example, if r_1, r_2 are tuples from the Service View and have the same user attributes, if the user indicates that the service instance defined by r_1 is preferred over the service instance defined by r_2, then CRKB will capture this information for future learning.

3.4 Recommender

In order to provide recommendations, the Recommender materializes the corresponding Recommendation View, and returns an ordered set of recommendations. Recommendation Views are based on Service Metric Views, which are used to define, for every service instance, key attributes and metric attributes. Key attributes are the attributes selected to

uniquely identify a product or service instance within a Service Metric View. Metric attributes represent those attributes that can be used to measure the suitability of an alternative.

3.5 Feedback Extractor, Preference Learning

Raw preference data is collected each time a user interacts with the Feedback Extractor. For each Recommendation View, historical preference data is maintained. The first data set stores the key attribute-value pairs for all past recommendation alternatives. A corresponding second data set stores a history of the preferred recommendation alternatives. The Feedback Extractor elicits a partial ordering of the recommendation set, for example, what has been liked and what has not. Feedback is submitted to the preference learner along with the Recommendation View and Service metrics view. Learning transformers allow for the exploitation of domain expert knowledge in defining metric attributes. An example is the use of l-null to calculate coefficients.

3.6 Use Case - Travel Package Application

Consider a family that would like to reserve roundtrip air transportation, accommodations at two separate destinations, and a rental vehicle for transportation between the two destinations. The family can travel any time within a specified window and can travel to their destinations in any order as long as the flights arrive and depart from the first destination.

[5]. The Service Metric View hierarchy for the travel example, shown in Figure 3.2, includes the composite (root) view, Family Travel Metrics, and the three atomic views Travel Accommodation Metrics, Air Travel Metrics, and Rental Vehicle Metrics. Family Travel Metrics defines the composite recommendation model for the recommendation of family travel packages. Each of the atomic views represents an element of the family travel package recommendation model.

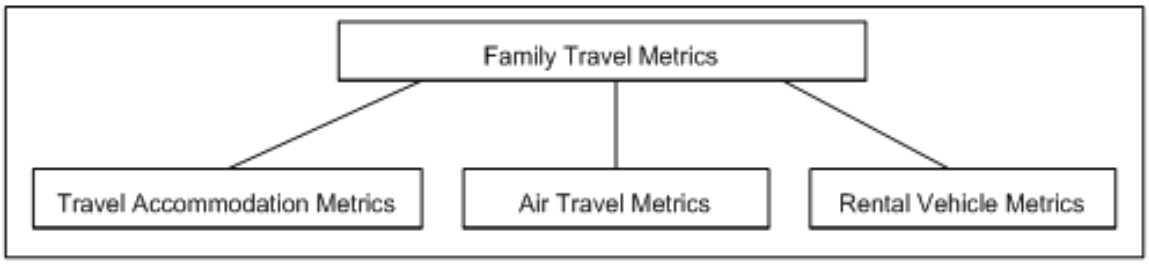

Figure 3.2: Family travel service metric view hierarchy [5]

The views in Figure 3.2 are constructed from the bottom up. The atomic views are populated by combining external service provider data with internal user profile data and learned preferences. The composed view is then populated using the data generated by the atomic views. The schemas for the source tables used in the family travel example are listed in Figure 3.3.

The Accommodation Service Metric Views: For each Atomic service we create a Service Metrics View, In order to demonstrate the construction of the family travel Service Metric View hierarchy, consider the Travel Accommodation Metrics view defined and depicted in Figure 3.4 we start with Accommodation service view: travelerID and accommID

AccommodationPackages	Flights	RentalVehicles	TravelerProfile
accommID	flight_no	vehicleID	travelerID
type	departDate	vehicleClass	type
location	source	capacity	location
amenities	destination	location	budgetLimit
activityName	class	costPerDay	earliestDepart
environment	no_hops		latestReturn
activityCost	flightTime		no_travelers
no_stars	cost		
costPerNight			

Figure 3.3: Source schemas [5]

are key attributes, the remaining metric attributes, amenities, no_stars are taken directly from the Accommodation Packages source. In contrast, accEnjoyment and accCost both represent dynamically computable expressions.

The flights Service Metric Views: The Air Travel Metrics view characterizes flights. As presented in Figure 3.5 : class, hops, flightTime, and airCost are direct projections of the source database. The metric attribute flightEnjoyment represents an expression preference for a specific flight using stored coefficients.

The Car rental Service Metric Views: It defines all feasible rental car alternatives. As

```
CREATE VIEW TravelAccommodationMetrics AS
SELECT TP.traveler_ID, AP.location,
       AP.accomm_ID, START.adate as checkin,
       END.adate as checkout,
       AP.amenities,   AP.no_stars,
       AP.type*AC.act_Type_coeff+
       AP.environment*AC.environment_coeff as
       accEnjoyment,
       (END.adate - START.adate)
       *AP.cost_Per_Night+AP.activity_Cost as
       accCost
FROM   Traveler_Profile TP,
       Accommodation_Packages AP,
       Accommodation_Coefficients AC,
       Dates START, Dates END
WHERE  (END.adate - START.adate)
       *AP.cost_Per_Night+AP.activity_Cost
       <= TP.budget_Limit
       AND END.adate <= TP.latest_Return
       AND START.adate >= TP.earliest_depart
       AND START.adate < END.adate;
```

Figure 3.4: Travel Accommodation Metrics view

presented in Figure 3.6, the attributes pickupDate and returnDate will be populated with all rental pickup and return dates that meet the specified constraints. The metric attribute rentCost is defined by an expression which dynamically calculates the total rental cost for each of the rental alternatives.

Family Travel Metrics view: Family Travel Metrics view definition is presented Figure 3.7 and it defines all feasible family vacation packages. This includes flights both to and from the first destination, accommodations at both the first and second destinations, and a rental vehicle for travel between the first and second destinations.

```
CREATE VIEW AirTravelMetrics AS
SELECT TP.traveler_ID, F.flight_no,
       F.SOURCE, F.DESTINATION,
       F.depart_Date, F.class, F.hops,
       F.flight_Time, F.cost
      (F.class*FC.class_coeff)+
      (F.hops*FC.hops_coeff)+
      (F.flight_Time*FC.flight_Time_coeff)
      As flightEnjoyment,
FROM  Traveler_Profile TP,
      Flights F,
      Flight_Coefficients FC
WHERE F.cost <= TP.budget_Limit AND
      F.depart_Date >= TP.earliest_Depart AND
      F.depart_Date <= TP.latest_Return;
```

Figure 3.5: Air Travel Metrics view

3.7 Summary

Recommender systems will continue to face the challenge of learning the preference of the user with minimum interaction from the user side. Consequently, return the user with a set of recommendations that are optimal or near-optimal with respect to the user preference.

Many recommender systems are intrusive where they require explicit feedback from the user and often at a significant level of user involvement. For example, before recommending any movies, MovieLens.org expects the user to rate a predefined number of movies

```
CREATE VIEW RentalVehicleMetrics AS
SELECT TP.traveler_ID, RV.VEHICLE_ID,
       RV.location, START.adate as pickup_date,
       END.adate as return_date,
       RV.class,
       (END.adate - START.adate) * RV.cost_Per_Day
       As rentCost
FROM   Traveler_Profile TP,
       Rental_Vehicles RV,
       Dates START,
       Dates END
WHERE (END.adate - START.adate) * RV.cost_Per_Day
      <= TP.budget_Limit AND
      RV.capacity >= TP.no_Travelers AND
      START.adate >= TP.earliest_Depart AND
      END.adate <= TP.latest_Return AND
      START.adate < END.adate;
```

Figure 3.6: Rental Vehicle Metrics view

(e.g., 20). This request comes with costs on the end-user [1]. However, DG-RCA does not require any explicit feedback prior to using the system. Initially, DG-RCA would recommend a set of alternatives to choose from or provide feedback on. In this chapter, we studied methods for providing recommendations on composite bundles of products and services, which are dynamically defined using database views extended with decision optimization using mathematical programming. We proposed a framework for (1) finding recommendation cluster, (2) user utility elicitation using decision optimization, and (3) The notion of presenting a diverse set of recommendations to extract a balanced set of both optimal and

```
CREATE VIEW FamilyTravelMetrics AS
SELECT TP.traveler_ID, TAM1.accomm_ID as accomm_ID1,
       TAM1.checkin as checkin1,
       TAM1.checkout checkout1,
       TAM2.accomm_ID as accomm_ID2,
       TAM2.checkin as checkin2,
       TAM2.checkout as checkout2,
       ATM1.flight_no as Flight_no1,
       ATM1.depart_Date as depart_date1,
       ATM2.flight_no as flight_no2,
       ATM2.depart_Date as depart_date2,
       RVM.VEHICLE_ID, RVM.location,
       RVM.pickup_Date, RVM.return_Date, RVM.Class,
       TAM1.accEnjoyment+ TAM2.accEnjoyment+
       ATM1.flightEnjoyment+ ATM2.flightEnjoyment
       As tEnjoyment,
       TAM1.accCost+TAM2.accCost+
       ATM1.Cost+ATM2.Cost+RVM.rentCost As tCost
FROM   Traveler_Profile TP,
       TravelAccommodationMetrics TAM1,
       TravelAccommodationMetrics TAM2,
       AirTravelMetrics ATM1,
       AirTravelMetrics ATM2,
       RentalVehicleMetrics RVM
WHERE  TAM1.accCost+TAM2.accCost+
       ATM1.Cost+ATM2.Cost+RVM.rentCost <=
       TP.budget_Limit AND
       TAM2.checkin >= TAM1.checkout AND
       ATM1.depart_Date = TAM1.checkin AND
       ATM2.depart_Date = TAM2.checkout AND
       RVM.pickup_Date = TAM1.checkout AND
       RVM.return_Date = TAM2.checkout;
```

Figure 3.7: Family Travel Metrics view

diverse recommendations.

There are few recommender systems that estimate and refine the preferences of the user [61], works such as [64] have exemplified the need for such techniques. The DG-RCA framework integrates support for the estimation and refinement of user preferences into the application development model [5]. The development of effective and accurate decision guidance systems must keep pace with the dynamic nature of the products and services.

Chapter 4: Iterative Utility Elicitation for Diversified Composite Recommendations

4.1 Introduction

DG-RCA is a recommender system that supports composite product and service, top-k decision optimization, and dynamic preference learning. DG-RCA combines the flexibility of diversity ranking functionality with the information processing capabilities to learn and capture the preferences of the user through an iterative learning process. The framework proposes and informs the user of what is available in terms of composite recommendations with minimum interaction from the user side. As explained in details in chapter 3, DG-RCA extends the CARD framework [5]. DG-RCA consists of three steps:

- Clustering the recommendation space.
- Selecting the utility axis.
- Diversifying the recommendations.

The process is initiated when the user submits a request to the system which contains both the users decision constraints and profile data. Then the recommendation space is split into a number of clusters, where each cluster contains a number of packages (recommendations). Examples of clusters are: honeymooners, single, family, etc. The clusters

are extracted from historical purchase data cross-referenced with user demographic data. However, this is beyond the scope of our research at this point. Currently, I just assume that these clusters exist and packages can therefore be targeted to specific user groups. The recommender will return a set of recommendations to the user. Each recommendation from the set will represent a cluster and maximizes the total utility function in its cluster. Initially I will give equal weight to metric attributes as we are in an early stage to conclude what the user might value more with respect to metrics attributes (e.g. saving, enjoyment). Domain knowledge could also be used to determine how to assign weights and selections of metric attributes, consequently calculating the global utility function. When the user indicates her preference and the chosen recommendation determines her preferred cluster, future recommendation space is limited to the chosen cluster. Next, the Preference Learner starts the iterative process of learning the utility vector of metrics attributes, e.g., saving, location attractiveness, enjoyment. This step starts with presenting the user with a number of distinguishable recommendations in terms of utility vectors. Each recommendation returned will stretch the dimension it represents (e.g., saving) and relaxes on the other dimensions (e.g. enjoyment, location attractiveness, etc). The process continues iteratively updating the utility vector every time, based on the feedback of the user until an exit point is reached (e.g., indicating "no difference" between recommendations presented). Upon exit, the recommendation space will be constructed into the Recommendation View, according to the utility vector learned. Finally, the Recommender constructs a set of diverse recommendations. Diversity is based on recommendation space, where each recommendation is characterized by a vector in n dimensional space. It starts with the recommendation that

maximizes the utility function calculated, and each consequent recommendation represents a diverse choice within the recommendation space. The system provides the user with a given number of diverse alternatives to choose from. At this point, the user can select the most preferred alternative or she can provide a (partial) ranking of the suggested alternatives to the Feedback Extractor. Feedback from the user will be honored if she has chosen to view additional suggestions. Then ask the user to rank the set of recommendation presented, and weight vector of recommendation(s) ranked the highest will be used as a base to reconstruct the utility function and update preference parameters, consequently present a new set of diverse recommendation in an iterative fashion. The process ends if the user either chooses a recommendation or exits the recommendation process.

The user has the option of restarting any of the main steps at anytime: Cluster, Optimize, and Diversify. Restarting the Diversify step uses original utility weight vectors obtained from the utility axes elicitation step. Currently, I am exploring the Recommender switching among 3 modes at different stages:

- During the clustering step where it presents a number of recommendations, where each recommendation represents a cluster.
- During utility axes selection, where the recommender uses n utility vectors, each represents a stratum formulated by Preference Learner.
- During the diversification step, where the Recommender returns a diverse set of recommendations. At this stage, the Recommender constructs one utility function from either:

- ◇ the utility axis selection step.
- ◇ After extracting feedback from the user on presented diverse recommendations.

The CARD Framework [5] supports composite product and service definitions, and recommendations are based on dynamically learned utility function and decision optimization. Composite services in CARD are characterized by a set of sub-services, which, in turn, can be composite or atomic. CARD uses a decision-guidance query language DG-SQL to define recommendation views, which specify multiple utility metrics, and the weighted utility function. However, the CARD framework has a number of limitations. First, it assumes the knowledge of the estimated utility function, whereas often this may not be available, but needs to be extracted from the user. Second, in some cases, a recommendation space may have different utility functions for different cluster of recommendation, which the CARD framework does not address. For example, a person may be considering different categories of vacation packages, such as family, romantic or business travel, and would apply different utilities for these categories. Furthermore, the diversity method of CARD has not been mathematically formalized or tested.

In this research, we adopt the CARD framework and resolve the limitations outlined above. More specifically, the contributions of this chapter are as follows. First, we propose a framework for finding a diverse recommendation set, when no prior knowledge on user preference is given. The framework involves interaction with the user to (1) choose a recommendation cluster the user is interested in, (2) dynamic elicitation of the weighted utility function, and then (3) generating a diverse set of recommendation that contains an

optimal recommendation in terms of the estimated utility function.

Second, we develop a method for utility function elicitation. It is based on an iteratively refining a set of axes in the n-dimensional utility space, starting from the utility space standard axes. At every step, the user is asked to rank a set of recommendations, each being optimal for one of the current axes. Based on the user feedback, the method refines the set of axes which become closer to each other, until the user cannot differentiate among them.

Third, we formalize the notion of a diverse recommendation set by defining the notion of m-layered recommendations. These recommendations contain one that optimizes the learned weighted utility function. Then, the space of all non-dominated[1] recommendations (the "skyline") is split into m layers, so that the first layer contains all recommendations whose utility is at least the maximum utility minus $\frac{1}{m}$ of the utility function range; the second up to $\frac{2}{m}$ and so on. Within each layer, a recommendation is chosen that optimizes one dimension of the utility space.

Fourth, we conducted a preliminary experimental study on the efficacy of the proposed framework, comparing precision and recall of ranked recommendations of a popular commercial travel site (called herein System A) vs. the DG-RCA framework using the same underlying set of flights and accommodations. The study showed that DG-RCA significantly outperformed System A. Furthermore, DG-RCA showed an average recall of 26% at rank 5 compared to 16% for System A, and an average precision of 100% at rank 1 compared to 36% for System A. While the preliminary study did not directly assess the level

[1] One recommendation dominates another if it is at least as good as the other in all respects.

of diversity provided by DG-RCA, we conjecture that recall studied is partly reflective of recommendations diversity.

This chapter is organized as follows: Section 4.2 describes the Utility axis selection of DG-RCA. Section 4.3 describes the Diversity Layering of the DG-RCA framework. Section 4.4 presents a case study for the purpose of validating the framework. Section 4.5 is the summary and possible extensions.

4.2 Utility Axis Selection

Now that we know what cluster the user is in, we will limit our recommendation space to the chosen cluster, and then deploy an iterative method to learn the user's preference with respect to n dimensional utility space (e.g., Enjoyment, Saving, Location attractiveness, etc.). Intuitively, at each step of the iterative process, we maintain a set of utility axes, which become "closer" to each other with every iteration until the user can no longer differentiate among them in terms of the preference. At that time the iterative process stops, and a final utility function is constructed. We first describe the overall process and then summarize it with an algorithm.

Recommendations space $\Re$, consists of composite products and services, each recommendation is represented by a tuple in a Service Metric View. A recommendation could be a vacation package the user can choose. Each recommendation is mapped to a utility vector $\vec{u}$, from an n dimensional utility space U, which is presented as $\mathbf{R}_{+}^{n}$, we denote this mapping by: $\vec{U} : \Re \rightarrow \mathbf{R}_{+}^{n}$. Components of a utility vector $\vec{u} = (u_1, u_2, \cdots, u_n)$, are

associated with metrics such as Enjoyment, Saving, Location attractiveness, etc. Each metric has an associated domain D_i, $1 \leq i \leq n$. For example $D_{Saving} = \mathbf{R}_+$, $D_{Enjoyment} = \{0, 1, \cdots, 10\}$, $D_{Location} = \{0, 1, \cdots, 10\}$. We assume the Location metric represents the attractiveness on a scale from 0 to 10 (this can be extracted from domain knowledge). Each domain D_i has a total ordering "better than" denoted $\succeq_{D_i}$. For example, for domain Saving, $a_1 \succeq_{Saving} a_2 \Leftrightarrow a_1 \geq a2$. The utility model assumes linearity with respect to each utility dimension; an increase of the value on any dimension results in a proportional increase of the total utility value. The ratio between the total utility value increase and the increase on a dimension is constant.

We model the relative importance the user places in each dimension by means of a vector of weights $\vec{w} = (w_1, w_2, \cdots, w_n)$, where $|\vec{w}| = \sqrt{\sum_{i=1}^{n} w_i^2} = 1$, which we call an axis. Each component w_i captures the weight of the i-th dimension. The total utility of a recommendation r_k w.r.t. axis $\vec{w}$ is defined as $U_{\vec{w}}(\vec{u}) = w_1u_1 + w_2u_2 + \cdots + w_nu_n = \vec{w} \cdot \vec{u}$. I point out that the utility elicitation method is robust to changes in measuring units on the dimensions. More precisely, the order imposed in the recommendations set by the learned utility axis is the same regardless on the units chosen on each dimension.

In the beginning, we assume no prior knowledge of the users subjective weights along each dimension, and would like to learn it as follows. We start with n axes, that represent the original dimensions in the utility space i.e.,

$$\vec{w_1} = (1, 0, \cdots, 0)$$
$$\cdots \tag{4.1}$$
$$\vec{w_n} = (0, 0, \cdots, 1)$$

In every iteration, a current set of axes is modified as follows. For each axis in the current set, we select a recommendation that maximizes the total utility according to that axis. For example, if we have 3 axes, we would present the user with 3 different composite recommendations, each exhibiting the highest utility w.r.t. the corresponding axis. Figure 6.1 exemplifies recommendations proposed to the user.

We then ask the user to partition these recommendations into up to k preference strata, where stratum 1 represents the best recommendations, stratum 2 the second best etc. Note that, each stratum may have 1 or more recommendations in it, and that 2 or more recommendations in the same stratum indicates that the user doesn't have a preference among them. Our goal from this step is to allow the user to inform the system of how to adjust the learned total utility function to better reflect her preference. The feedback extracted from the user, i.e., the preference strata, is used to move the current axis closer to the learned utility function.

We first replace each $\vec{w_i}$ as follows. Let r_i be a recommendation that maximizes $U_{\vec{w_i}}(\vec{u_i})$, where $\vec{u_i}$ is a utility vector associated with r_i. We then replace $\vec{w_i}$ with the axis $\frac{\vec{u_i}}{\|\vec{u_i}\|}$, where the notation $\|\cdot\|$ means the norm of the vector. Then, for every rank k, we calculate the normalized mean,

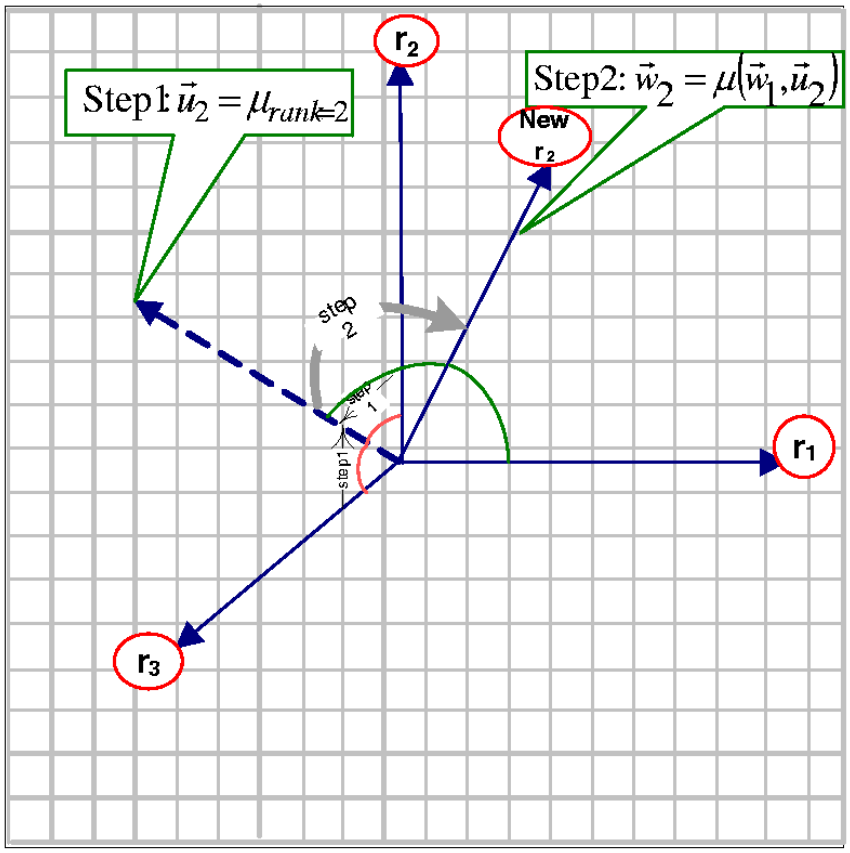

Figure 4.1: Example of Utility Axis Selection

$$\mu_i = \mu(\{\vec{w}_i | rank(r_i) = k\}) = \frac{\sum_{rank(r_i)=k} \vec{w}_i}{\left\| \sum_{rank(r_i)=k} \vec{w}_i \right\|} \tag{4.2}$$

We now build new axes $\vec{w}_1, \cdots, \vec{w}_k$, where k is the number of strata, as follows:

$$\text{For stratum 1,} \quad \vec{w_1} := \vec{\mu_1}$$

$$\text{For stratum 2,} \quad \vec{w_2} := \mu(\vec{w_1}, \vec{\mu_2})$$

$$\text{For stratum } i, \quad \vec{w_i} := \mu(\mu(\vec{w_1}, \cdots, \vec{w_{i-1}}), \vec{\mu_i}),$$

$$\text{where } 3 \leq i \leq k$$

Intuitively, after adjusting $\vec{w_1}, \cdots, \vec{w_{i-1}}$, we do not yet know the user's preference among them, but do know that they are preferable over strata i, represented with $\vec{\mu_i}$. Therefore, we create new axis $\vec{w_i}$ as the vector mean of $\mu(\vec{w_1}, \cdots, \vec{w_{i-1}})$ and $\vec{\mu_i}$, intuitively moving it toward $\mu(\vec{w_1}, \cdots, \vec{w_{i-1}})$ "half way". For example, assume the user is presented with 3 recommendations r_1, r_2, and r_3, according to utility vectors $\vec{u_1}$, $\vec{u_2}$, and $\vec{u_3}$ respectively. The user placed recommendation r_1 in stratum 1 and both recommendations r_2 and r_3 in stratum 2. First, for all recommendations, we replace $\vec{w_i}$ with the axis $\frac{\vec{u_i}}{\|\vec{u_i}\|}$. Second, we calculate the mean utility vector of r_2 and r_3 as $\mu_2 = \mu(\vec{w_{r2}}, \vec{w_{r3}})$. Third, we calculate $\vec{w_2} := \mu(\vec{w_1}, \vec{\mu_2})$ Finally, we use the resulting $\vec{w_2}$ to calculate the new recommendation r_2' as shown in Figure 6.1. The iterative process continues each time with a new set of axes, until all proposed recommendations, optimal w.r.t. the current axes, are in a single stratum 1. This means recommendations presented are indifferent, i.e., the user can not differentiate among recommendations suggested. As a final step, we calculate the normalized mean one more time of the resulting axes to be used as the utility weight vector in the next step of our framework, described in this chapter in section 5, that is diversity layering. Algorithm 1 captures the process of utility axis selection.

Algorithm 1 Algorithm of utility axis selection

for $i = 1$ to n **do**
  $\vec{w_i}$ = the vector with 1 on the i-th component and 0 everywhere else
end for
$p = n$
while $p > 1$ **do**
  for $i = 1$ to p **do**
    r_i = a recommendation which maximizes $U_{\vec{w_i}}$
    Recalculate w_i using the weights of presented r_i
  end for
  Ask the user to place each recommendation presented in a stratum where 1 is the best, 2 is next best, etc;
  MaxRank = the max stratum label assigned by the user;
  for $k = 1$ to MaxRank **do**
    Collect all recommendations labeled k;
    $\vec{u_k}$ = the mean of the weight vectors for recommendations labeled k;
  end for
  for $k = 2$ to MaxRank **do**
    $\vec{w_k} := \mu(\mu(\vec{w_1}, \cdots, \vec{w_{k-1}}), \vec{\mu_k})$
  end for
  $p =$ MaxRank
end while
return $\vec{w_1}$

4.3 Diversity Layering

Now that we know more about the user in terms of the utility axis, we construct the global utility function where weights given to each metric attribute reflect the attractiveness of recommendations to the user. However, giving recommendations by the utility learned may not provide sufficient diversity of recommendations. We would like to return a diverse set of recommendations with a range of options that are not too similar and which are ranked by the learned utility. For example, a person whose utility is mostly in favor of low price may decide to take a very attractive travel package even if the price is not minimal. In this section, we develop diversity layering method to provide diverse recommendations sorted by the utility. This is done by a recommendation view. The syntax template of a composite

diverse recommendation MyRecommendation
select $V.*$, $w_1 * u_1 + \cdots + w_n * u_n$ **as** *utility*
from MVview V
where user_constraints
order by *utility*, $u_1, \cdots, u_n$
[**layers** m]
limit k;

Figure 4.2: Recommendation view

Recommendation View is depicted in Figure 4.2.

Each Recommendation View is associated with the corresponding Service Metric View (SMV), which appears in the **from** clause. Examples of SMV are Rental vehicle, Airline flights, Travel Accommodation, or a combined travel package. The **where** clause contains user constraints, e.g., the maximum budget and duration of travel. The **select** clause returns all the key and metric attributes of the service instance, along with *utility* that has been learned in the utility axis selection step. We introduce a new optimal clause **layers** which indicates how many layers to split the recommendation space into. The **limit** indicates the number of recommendations the user would like to be presented with. The **limit** value could be a configuration or user-defined parameter. Intuitively, to reach diversity we start with the optimal recommendation (in terms of the learned utility) and then dynamically partition the recommendation space into m layers. Recommendations in the first layer have the utility function close to the max utility up to $\frac{1}{m}(U_{max} - U_{min})$, i.e. their utility is in the interval $[U_{max} - \frac{1}{m}(U_{max} - U_{min}), U_{max}]$. Recommendations in the i-th layer have utility in the interval $[U_{max} - \frac{i}{m}(U_{max} - U_{min}, U_{max}]$. Within each layer we select n recommendations

to maximize each dimension of the utility space in turn. Finally, we return the user a set of k recommendations (in terms of the learned utility) chosen from the m layers, after removing duplicates and sorting them by utility. We first illustrate the diversity layering method using an example, and then present a formal definition of diversity layering. Consider an example depicted in Figure 4.3, for three layers, i.e., $m = 3$ in the **layers** clause and given **order by** *utility*, u_1, u_2. There are two dimensions, u_1 and u_2 of the utility space (i.e. metrics relevant to selection), and U_w is the learned global utility. For example, u_1 can stand for (total-budget cost), i.e., Saving, and u_2 for the Location attractiveness factor of family travel. The two-dimensional polyhedral set in the figure depicts all possible utility vectors of recommendations.

We note that recommendations (e.g. $r_0, r_{11}, r_{12}, r_{21}, r_{22}, r_{31}, r_{32}$) residing on the "skyline" are the non-dominated choices of the recommendation space. For example, recommendation $\underline{d}$ in the figure is dominated by r_0, the utility vector of r_0 is higher than that of $\underline{d}$ in both dimensions. First, we eliminate all dominated recommendations, and thus are left with the "skyline", which is denoted with the thick line. From the **order by** clause, the user indicated that u_1 is more important than u_2. The recommendation r_0 maximizes the global utility U. Then the skyline is split into three layers. The first is corresponding to the area above the highest dashed line which corresponds to recommendations with utility $U \geq U_{max} - \frac{1}{3}(U_{max} - U_{min})$, and select recommendations r_{11}, r_{12} that maximize dimensions u_1 and u_2 respectively. Similarly, the second and third layers correspond to $U \geq U_{max} - \frac{2}{3}(U_{max} - U_{min})$ and $U \geq U_{min}$ respectively. As a result, we extract recommendations $r_{21}, r_{22}, r_{31}, r_{32}$. If the user requests four recommendations (i.e., **limit** = 4), then

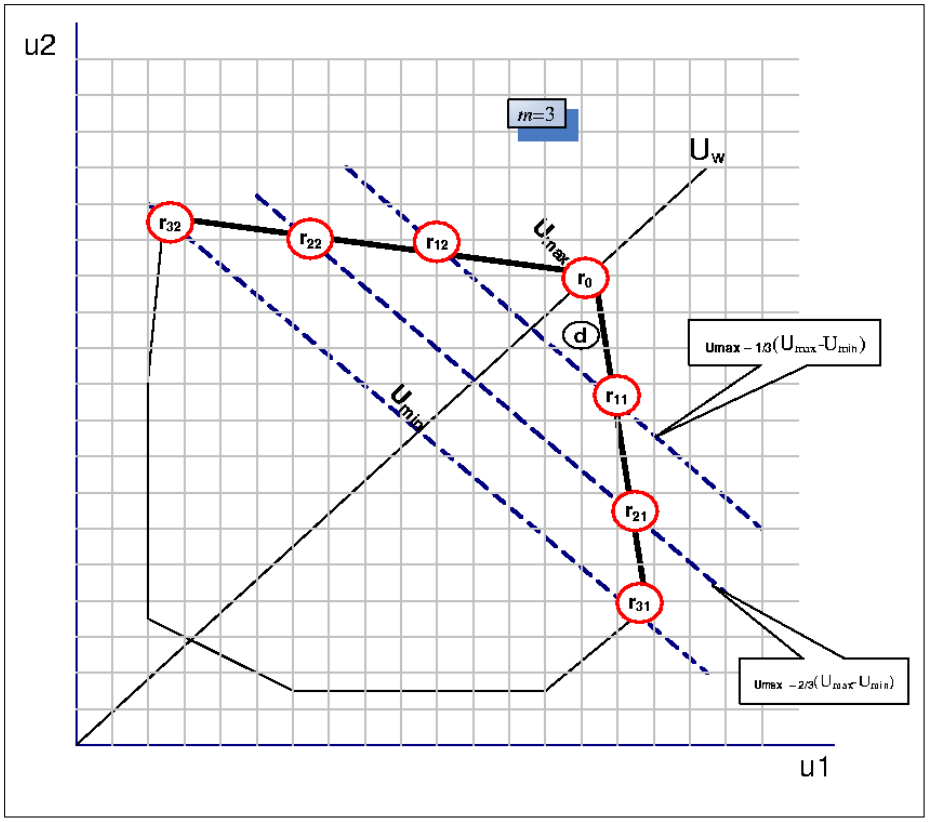

Figure 4.3: Diversity Layering Example

$(r_0, r_{11}, r_{12}, r_{21})$ will be returned in this order, and if **limit** = 6, $(r_0, r_{11}, r_{12}, r_{21}, r_{22}, r_{31}, r_{32})$ will be returned. Intuitively, maximizing each metric component in turns gives diversity, while restricting the global utility within its layer controls the distance from the optimal global utility. More formally, given two recommendations r_1, r_2 in $\Re$ and the corresponding utility vectors $\vec{u} = (u_1, u_2, \cdots, u_n)$ and $\vec{v} = (v_1, v_2, \cdots, v_n)$ respectively in $\mathbf{R}^n_+$, we say that r_1 dominates r_2, denoted $r_1 \succeq r_2$, if $u_i \geq v_i$ for all i, $1 \leq i \leq n$. Intuitively, one recommendation dominates another if it is at least as good as the other in all respects. We denote by $\bar{\Re}$ the set of all non-dominated recommendations i.e.,

$$\bar{\Re} = \{r | \neg(\exists r' \in \Re) r' \succeq r\} \tag{4.3}$$

As before, $\vec{U}$ is the utility mapping, $\vec{U} : \Re \rightarrow \mathbf{R}^n_+$. Below we use U_{max}, U_{min} defined as:

$$U_{max} = \max U_w(\vec{U}(r)) \text{ s.t. } r \in \bar{\Re} \tag{4.4}$$

$$U_{min} = \min U_w(\vec{U}(r)) \text{ s.t. } r \in \bar{\Re} \tag{4.5}$$

Definition 1. *An m-layered recommendation set is a set:*

$\{r_0, r_{11}, \cdots, r_{1n}, r_{21}, \cdots r_{2n}, \cdots r_{m1}, \cdots r_{mn}$ *of recommendations such that:*

1. $r_0 \in \bar{\Re}$, $r_{ij} \in \bar{\Re}$ *for all* $1 \leq i \leq n$, $1 \leq j \leq m$ *(i.e. only non-dominated recommendations are included).*

2. $U(r_0) \geq U(r_{11}) \geq \cdots \geq U(r_{1n}) \geq \cdots \geq U(r_{m1}) \geq \cdots \geq U(r_{mn})$ *for all* $1 \leq i \leq n$, $1 \leq j \leq m$.

3. $r_0 =$ *argmax*$_r U_w(\vec{U}(r))$ *s.t.* $r \in \bar{\Re}$.

4. *For every* $1 \leq i \leq n$, $1 \leq j \leq m$, $r_{ij} =$ *argmax*$_r U_w(\vec{U}(r))$ *s.t.* $r \in \{=$ *argmax*$_r \vec{U_j}(r) | r \in \bar{\Re} \wedge U_w(\vec{U}(r)) \geq U_{max} - \frac{1}{m}(U_{max} - U_{min})\}$.

An m*-layered* k*-recommendation is a sequence* $(r_0, r_1, \cdots, r_{k-1})$ *such that:*

- $r_0, r_1, \cdots, r_{k-1} \in \{r_0, r_{11}, \cdots, r_{mn}\}$
- *All* $r_0, r_1, \cdots, r_{k-1}$ *are different*
- $r_0, r_1, \cdots, r_{k-1}$ *are sorted in lexicographical order of* U_w, u_1,$\cdots$, u_n.

Finally, a k*-recommendation is an* m*-layered* k *recommendation, where* m *is selected to be the minimum of the number of layers that produce at least* k *recommendations. Note that a* k*-recommendation is returned when* **layers** m *clause is omitted.*

4.4 Validation - User Case Study

In order to evaluate our proposed recommender system, we conducted a user study of 30 users (with HSRB approval provided in Appendix B). The objective of the study was to verify the following hypotheses:

1. Our system achieves a better recall and precision than a non-personalized travel recommender system.

2. The interactive elicitation of the utility axis imposes an acceptable overhead to the users.

3. The vertical diversity layering step increases the recall.

Hypothesis 1 is justified by the widespread adoption of recall and precision as a standard measure for validation in Information Retrieval(IR) [e.g. 12,26,27,28]. Recall is the percentage of correctly predicted “high“ ratings among all the ratings known to be “high“ [1], while precision is the percentage of truly “high“ ratings among those that were predicted to be “high“ by the recommender system [1]. The reason we chose Hypothesis 2 is to measure the burden caused by our interactive framework and determine if it is acceptable in view of the perceived benefits. Finally, we wanted to test Hypothesis 3 so that we can assess the usefulness of the vertical diversity layering. We loaded our systems database with real data about vacation packages extracted from a popular travel commercial website, which we will call System A.

We conducted the user study aiming to estimate the recall and precision of our system. Specifically; we submitted a request for a three week vacation in Los Angeles, California starting on May 1, 2009, including roundtrip airfare from Washington Dulles Airport. We then extracted all packages returned by System A, keeping just the cost and number of stars (enjoyment) of each package. Since we wanted to evaluate the quality of the top results returned by our system against System A, we limited the number of results shown to the user to five. We surveyed a total of 30 users, all working professionals.

For DG-RCA, in the first phase we learned the users utility function through a two step dialog: at each step, we present the user with two choices, one with a better enjoyment (number of stars), and the other with a smaller cost, as described in this chapter in section 4. Depending on the user's answers, their inferred utility function can reflect the following:

- a strong sensitivity to price (PP)
- a moderate preference for less expensive packages (PQ)
- a moderate preference to higher quality packages (QP)
- a strong bias towards high quality packages (QQ)

The distribution of answers was as follows:

- 6 users in the PP category
- 18 users in the PQ category
- 4 users in the QP category
- 2 users in the QQ category

In the second phase, I computed five recommendations using the diversity layering method described in this chapter in section 5 and presented them to the user in descending order of their utility (according to the personalized utility function estimated in the first phase). For System A, we just presented the top five recommendations in the order suggested by the website.

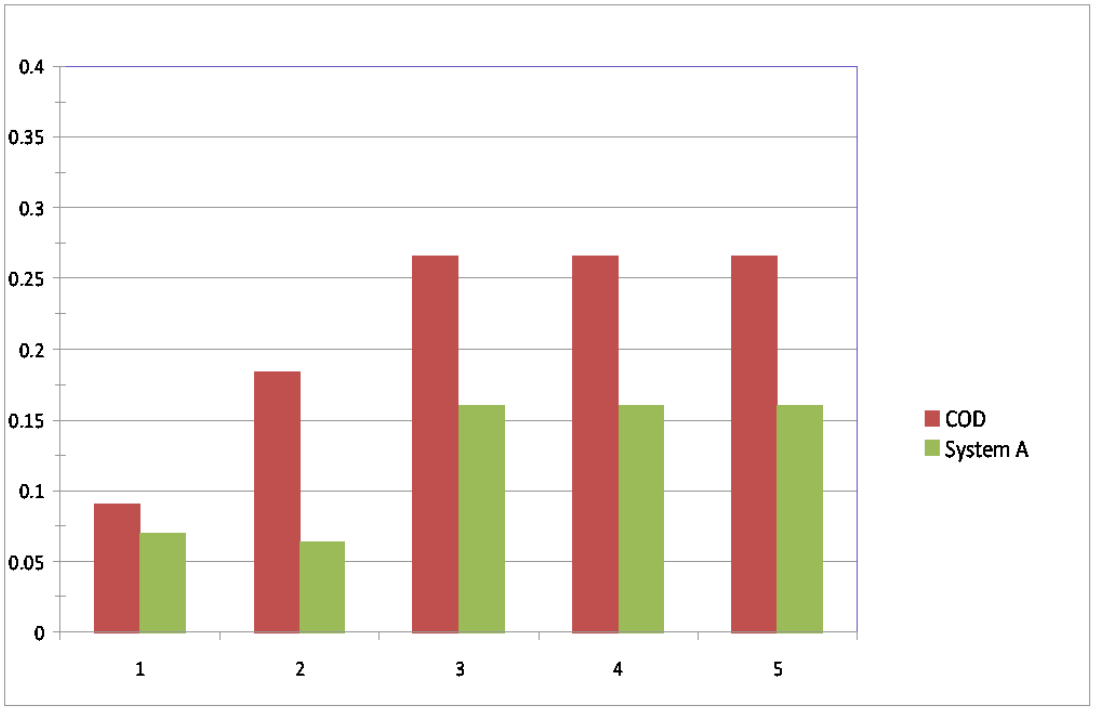

Figure 4.4: Average Recall vs. Rank

We then asked the users to rate each of the ten recommendations on a scale of 1 to 5, where:

- 5 means “definitely buy”,
- 4 means “likely to buy”,
- 3 means “neutral”,
- 2 means “unlikely to buy” and

- 1 means “definitely not buy”.

We point out that users surveyed did not know which recommendation set came from which system.

In order to estimate recall at a given rank, we gathered all the packages rated 4 or 5 by any user, call that set “Buy“ (this is the set of all recommendations which the users considered buying). Then, we counted how many of these recommendations were present in the top k results returned by DG-RCA system and System A. Formally,

$$recall(k) = \frac{|\{r \in Buy | rank(r) \leq k\}|}{|Buy|} \tag{4.6}$$

For each of the two systems we then computed the average recall at each rank k by taking the average of $recall(k)$ across all the 30 users. The results are summarized in Figure 7.2. As we can see, already at rank 2, our system returned 18% of the relevant packages compared to 6% for System A. Moreover, our system returned 26% of the relevant recommendations in the top 5 results. By contrast, System A returned only 16% of the relevant results in the top 5.

In order to estimate precision, we counted how many of the recommendations in the top k results were actually in the set Buy. Formally,

$$precision(k) = \frac{|\{r \in Buy | rank(r) \leq k\}|}{k} \tag{4.7}$$

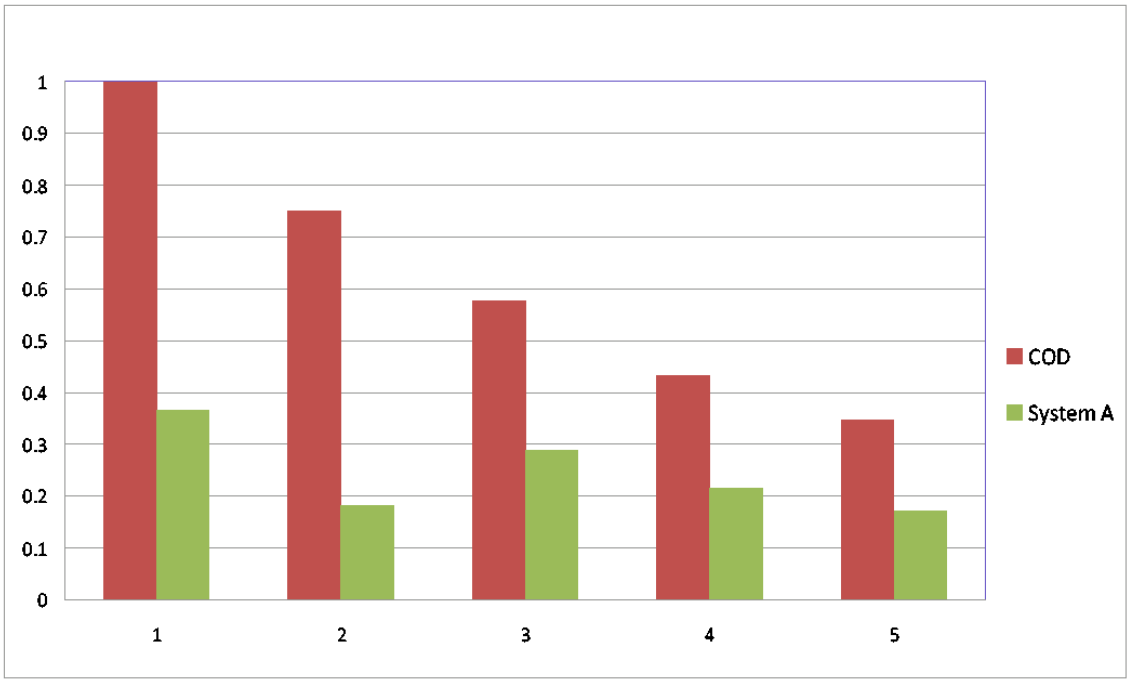

Figure 4.5: Average Precision vs. Rank

For each of the two systems we computed the average precision at each rank k by taking the average of $precision(k)$ across all the 30 users. The results are summarized in Figure 7.1. As we can see, at rank 1, all of the recommendations returned by DG-RCA in the top position were actually relevant, compared to 36% for System A. At rank 2, 75% of our recommendations were relevant compared to 18% for System A. In fact, at every rank our system considerably outperformed System A with respect to precision. In order to determine the statistical significance of our results, we assume a uniform distribution of

ratings over the set of available packages, meaning that a package selected at random has an equal chance of receiving any of the 5 user ratings. Under this assumption, the probability of a randomly selected package to be rated “buy” (rating 4 or 5) is 2/5. DG-RCA selected 52 packages rated Buy out of 150 trials (5 results for 30 users), which can occur by chance with a probability of 2.76%. Therefore, hypothesis 1 is confirmed with adequate statistical significance (p-value of 0.0276). We believe the better quality of our results comes from personalizing the utility function according to the learned user preferences. While System A returns the same set of results to every user, we make an attempt to learn more about what each particular user is interested in. Since this extra step is imposing an extra burden on the end user, we included at the end of our survey the following question: “Would you be willing to spend a few more minutes answering few questions so that the system can learn about you and, consequently, provide you more personalized recommendations, considering the amount of the transaction?”. The vast majority of the surveyed users (27 out of 30) answered “yes”, which confirms hypothesis 2.

Finally, for hypothesis 3, we examined the distribution of the “buy” ratings within the vertical layers in the ranked result lists. Here, we observed that all the “buy” ratings were restricted to the first position and the top-most layer (positions 2 and 3), and all the recommendations in the bottom layer were rated either “neutral” or “not buy”, therefore not improving the recall. We believe this is due to the rather low value we chose as a threshold for utility in this particular experiment. More work is needed to study whether a carefully calibrated threshold leads to improved recall beyond the top-most layer.

4.5 Summary

In this chapter, we studied methods for providing recommendations on composite bundles of products and services, which are dynamically defined using database views extended with decision optimization using mathematical programming. We proposed a framework for finding a diverse recommendation set, when no prior knowledge on user preference is given, which includes (1) finding recommendation cluster, (2) user utility elicitation using decision optimization, and (3) partitioning the recommendation space into layers to extract a balanced set of both optimal and diverse recommendations. We also conducted a preliminary experimental study, which showed that the DG-RCA framework significantly outperforms a popular commercial system in terms of precision and recall.

Many recommender systems are intrusive where they require explicit feedback from the user and often at a significant level of user involvement. For example, before recommending any movies, MovieLens.org expects the user to rate a predefined number of movies (e.g., 20). This request comes with costs on the end-user [1]. However, DG-RCA does not require any explicit feedback prior to using the system. Initially, DG-RCA would recommend a set of alternatives to choose from or provide feedback on. In addition, DG-RCA uses a simple feedback extraction mechanism, where users are only asked to place recommendations in a stratum which is typically quick and easy, furthermore, the number of recommendations to rank becomes smaller after the previous iteration. According to our user study, only 3 out of 30 people complained about the explicit feedback required.

As discussed in more details in section 4.2, the utility function elicitation of DG-RCA

allows for less intrusive learning. Initially, DG-RCA is capable of recommending alternatives without the need to extract feedback. In addition, DG-RCA could work with previously calculated utilities. This utility can be obtained by domain experts or calculated using historical data. Finally, the DG-RCA framework deals with the recommendation space of composite products and services by using decision optimization when extracting recommendations that optimize specific axis in the utility space. Such problems are more complex when there is a large number of alternatives (possibly infinite), however, we noticed that our framework converges after 2 questions as explained in the case study section.

As we saw in section 4.4, the diversity layering method did not improve recall according to the user case study. Specifically, while all recommendations ranked 1 and 2 by DG-RCA were rated 5 or 4 by users, corresponding to "definitely buy" and "likely to buy" respectively, most recommendations presented to users ranked 3 and more were mostly rated "unlikely to buy" or "definitely not buy" by users. Intuitively, the reason for a low rating received in rank 3 and more is because the same space which is utility space is used for similarity and diversity. Diversification is based on utility space. In Chapter 5, I will address this limitations and refine the currently developed method for diversifying a recommendation set. I introduce a new approach for diversifying the recommendation space: an n-dimensional recommendation space is constructed and used for diversification. This space is separate from the utility space that is used for utility elicitation, where a distance function is constructed to be used for diversification while using the learned utility function.

Many research questions remain open. They include (1) identify the right balance between optimality of recommendations (in terms of the learned utility) and diversity, and refining algorithms to reflect that balance; (2) developing efficient algorithms for diversity layering queries, which take advantage of simultaneous optimization of multiple constraint problems; (3) expanding the diversity layering to incorporate users feedback on diverse recommendations.

Chapter 5: A Randomized Algorithm for Maximizing the Diversity of Recommendations

5.1 Introduction

The vast number of products and services offered via the web inevitably results in information overload for users. The shift toward customized product bundles and service compositions will only compound this problem in the future.

This chapter focuses on the maximum diversity problem (MDP) to recommend composite products and/or services. Recent popular surveys (e.g., [1,22,68])) classify the current generation of recommendation methods into three main categories: content-based, collaborative, and hybrid recommendation approaches. Content-based systems often employ classifier techniques that rely on information retrieval and information filtering to recommend an item to a user based upon a description of the item and a profile of the users interests. In contrast, Collaborative recommenders use the preferences of "similar" users, rather than the characteristics of an item, to make suggestions to the current user. The hybrid approach combines two or more methods to produce a recommendation set. More recently, utility-based, knowledge-based, and demographic systems have each suggested different techniques for providing recommendations. Systems employing one or more of these techniques have been proposed to recommend flights, movies, restaurants, and news articles

(e.g., [1,12,17,31]). Most of today's recommender systems recommend only atomic products or services. Complex recommendation models involving composite alternatives, such as product configurations and service packages, are rarely addressed. In addition to other desirable research areas, most research in this field is focused on improving the accuracy of the recommendation set with respect to the query submitted, but less attention is given toward improving the utility of the recommendation set [73]. Diversity is important because it helps users to be aware of choices they may not have thought of, and allows returning a set of recommendations with a range of options that are not too similar yet score high with respect to the utility (e.g., represented by the query). For example, a person whose utility is mostly in favor of low price may decide to buy an attractive travel package even if the price is not minimal as long as the price is within a reasonable range. It is wise to assume that users want to be returned a set of alternatives that are similar to the query but not similar to each other. Assume you submit a query for a laptop computer with a specific price range and all models returned are from the same manufacturer. If you cannot purchase this brand for some reason, then all presented recommendations are useless [17].

Recent work (e.g., [5,6,17,33,73,101]) on diversity assumed a trade-off between similarity and diversity. The reason for this assumption is the use of the same space, which is utility space, for similarity and diversity simultaneously. The more diverse the result set is, the less similar that set to the query submitted. For example, if the user submits a query for a laptop with price of $1500, then introducing diversify to the returned set would result in presenting the user different variation of prices, because we are using the same utility space for similarity and diversity. However, some presented recommendations may cost

more than what the customer has stated. In this paper, we question the similarity-diversity trade-off assumption for a new reason. We suggest that when multiple recommendations are retrieved for a given target query (or utility), the similarity of these cases (relative to the utility), as well as their diversity (relative to each other), must be both explicitly maximized. To the best of our knowledge, all diversity techniques use the same space for similarity and diversity resulting into two goals competing with each other.

We are proposing a method for diversifying a recommendation set using the separation of **utility space**, e.g., in travel domain, utility can be represented by metrics such as *Saving* and *Enjoyment*, from **diversity space**. Diversity space would be represented by metrics that are different from those of utility space, for example, *Activity types* and *Duration* of the trip. Typically, utility space is low dimensional, whereas diversity space is of considerably higher dimensionality. This way, we can guarantee that all returned recommendations score high (possibly highest) against the utility function but are also diverse compared to each other. It is important to point out that the formal definition of the MDP problem is concerned with returning a diverse set of alternatives to the user, regardless of the space used. In this paper, our focus is to introduce and then separate two spaces: the utility space and the diversity space. Throughout the paper, we restrict our focus to the diversity space, since all the candidate recommendations already score high with respect to utility. In addition, since the determination of relevant attributes of the Diversity space is not included in the scope of the formal research problem, we informally discuss the selection criteria of attributes and their characteristics.

Working with campsite products and services has the challenge of the selecting from

larger space, but also gives an opportunity to diversify using higher-dimension diversity space (where features of composite products or services are numerous). In addition, we introduce and apply a new approach in this application domain for diversification, which outperformed the popular state-of-the art diversity algorithms that have been published [33,73,101].

The CARD Framework [5] supports composite product and service definitions, and gives recommendations that are based on dynamically learned utility function and decision optimization. Composite services in CARD are characterized by a set of sub-services, which, in turn, can be composite or atomic. CARD uses a decision-guidance query language (DGQL) to define recommendation views, which specify multiple utility metrics, and the weighted utility function.

DG-RCA [6] is based on CARD framework [5]. DG-RCA suggested efficient technique to elicit utility function and a diversity method using the same utility space. The case study conducted in DG-RCA (Chapter 4 Section 4) showed strength in utility function elicitation. However, its diversity method did not improve recall. Specifically, while most recommendations ranked 1 and 2 by DG-RCA were rated "definitely buy" or "likely to buy" by users, most recommendations presented to users ranked 3 and more were mostly rated "unlikely to buy" or "definitely not buy". Intuitively, the reason for low rating received in rank 3 and more is because we used the same space, which is utility space for both similarity and diversity. In this paper, we adopt the DG-RCA framework and resolve the limitations outlined above. The key idea of this paper is to separate the utility space (to be used for utility elicitation) from the higher-dimensional diversity space (to be used for

diversification). More specifically, the contributions of this paper are as follows.

First, we introduce a new approach for diversifying a set of recommendations. An m-dimensional diversity space is constructed and used, which is separate from the utility space that is used for utility elicitation. In the diversity space, a distance function is constructed to be used for diversifying recommendations that score high in terms of the learned utility function. That is, these recommendations are optimal or near-optimal in terms of the learned utility function and chosen to be the most diverse from each other.

Second, we present a randomized algorithm that provides a competitive solution with respect to finding a diverse set from candidate recommendations. The algorithm is lightweight yet efficient. The idea of the algorithm is to iteratively relax the selection by the Greedy algorithm [17,101], with an exponential probability distribution. This relaxation allows our algorithm to identify a better solution with high probability.

Third, we conducted extensive experimental studies on the efficacy of the proposed algorithm to compare precision and scalability of our ranked recommendations with the two state-of-the art algorithms: Greedy [17,101] and Eigensolver [73]. Experiments suggest that the proposed algorithm outperforms Greedy and Eigensolver with respect to the quality of results. Furthermore, we were able to verify that for k up to 7 and n up to 100, our algorithm converges to optimal solution in under 100ms. In addition, we present different series of experiments designed to evaluate the proposed algorithms' parameters for the exponential probability distribution and the number of repetition of the algorithm.

This chapter is organized as follows: Section 5.2 discusses Diversity space with Utility space. Section 5.3 presents and explains the proposed diversity algorithm. Section 5.4

covers the validation part. Section 5.5 is the summary and possible extensions.

5.2 Utility space vs. Diversity space

The new surge of current mobile computing devices such as PDAs and WAP-enabled mobile phones with screen size that could be 200 times smaller than that of a PC, is adding another pressure on recommender systems to move beyond the accuracy measure when producing recommendation list [17]. This reduces the number of recommendations that can be returned in a single search. If the few returned recommendations are similar to each other, then it is unlikely the user will be satisfied. In addition, [96] makes the argument that diversity is also important for news aggregator sites because the resulting diverse set makes many people as possible feel that their viewpoint is heard. Furthermore, the utility is only an estimate to users' preferences, because users may have not explained their preference accurately, or they did but the recommender system could not capture the preference precisely. For this reason, many practical systems are interactive, allowing the user to scroll through a set of choices to make a decision [73].

We assume that composite products and/or services from the set $\Re$, called the recommendation space, are offered for purchase by a recommender system. Recommendations from $\Re$ could be a vacation package the user can choose, and could be represented in many ways, e.g., a tuple in a Service Metric View [5,6]. Each recommendation is associated with a user-specific utility vector $\vec{u} = (u_1, u_2, \cdots, u_n)$ from an n-dimensional utility space

$U = U_1 \times \cdots \times U_n$ where $U_1, \ldots, U_n$ are "basic" domains. This is denoted by a mapping $\vec{u} : \Re \rightarrow U$. Components of the utility vector $u_1, u_2, \cdots, u_n$ are associated with metrics such as *Enjoyment, Saving, Location* attractiveness, etc. Each metric has an associated domain U_i, $1 \leq i \leq n$. For example $U_{Saving} = \mathbf{R}_+$, $U_{Enjoyment} = \{0, 1, \cdots, 10\}$, $U_{Location} = \{0, 1, \cdots, 10\}$. We assume the Location metric represents the attractiveness on a scale from 0 to 10 (this can be extracted from domain knowledge). Each domain U_i has a total ordering "better than" denoted $\succeq_{U_i}$. For example, for domain Saving, $a_1 \succeq_{Saving} a_2 \Leftrightarrow a_1 \geq a2$.

As presented in DG-RCA [6], the utility function is elicited for individual users by using a method that is based on iteratively refining a set of axes in the n-dimensional utility space, starting from the utility space standard axes. The DG-RCA paper also gave a diversity algorithm working in the same utility space. The idea was to iteratively relax the distance ϵ from the maximum utility and optimize different utility metrics, e.g., Saving and Enjoyment within the relaxation ϵ. However, as experimentally observed, although relaxing ϵ beyond a certain small value increased diversity (in the utility space), it quickly makes recommendations irrelevant. This gave us the key idea of this paper, which is to introduce a separate space for diversity while limiting recommendations to be sufficiently close (up to a small ϵ) to the maximum utility.

Thus, in this chapter we propose that each recommendation also be associated with a "diversity" vector $\vec{v} = (v_1, v_2, \cdots, v_m)$, from an m-dimensional diversity space $V = V_1 \times \cdots \times V_m$ where $V_1, \ldots, V_m$ are "basic" domains. Components of diversity vector $v_1, v_2, \cdots, v_m$ are associated with metrics such as *activity type*, *hotel location* e.g., the

geographical position (Latitude, Longitude), *package description*, etc. Furthermore, we assume that each V_i, $1 \leq i \leq m$ has a distance function $d_i : \ V_i^2 \ \rightarrow \ R_+$. For example, distance d_i could be Euclidian distance for hotel location, the depth difference on activity taxonomy hierarchy for the activities, or $1-$ cosine TF/IDF keyword similarity for package description.

Finally, we define the distance between two diversity vectors $\vec{x} = (x_1, x_2, \cdots, x_m)$ and $\vec{y} = (y_1, y_2, \cdots, y_m)$ by combining the distances on the basic diversity domains, for example $\sqrt{d_1(x_1, y_1)^2 + d_2(x_2, y_2)^2 + \cdots, d_m(x_m, y_m)^2}$, and the distance between two recommendations r_x and r_y as the distance between their diversity vectors $\vec{v}(r_x)$ and $\vec{v}(r_y)$.

We calculate the diversity of a set similar to [6,17,33,73,101], that is the average distance between every two members in a set s. $Diversity(s) = \frac{\sum_{i=1}^{k} \sum_{j=i+1}^{k} d(i,j)}{\frac{k}{2} * (k-1)}$, where $k = |S|$.

Diversifying using the diversity space will address the issue that has been raised by [5,6,17,33,73,101] which is that the joint goal of offering a set S of high diversity and of high matching value, stand in opposition to each other.

The proper separation of metrics attributes into utility and diversity spaces requires domain expertise. There are some properties of dimensions that make them more suitable to be used as either utility or diversity dimensions. We suggest that only a small number of dimensions is used as utility dimensions, otherwise, it would be difficult for users to precisely express their individualized utility functions. In addition, metrics of utility space need to have totally ordered domains so the user can compare choices. On the other hand,

the diversity space could have a large number of dimensions since more diversity dimensions will enable a higher diversity. In addition, metrics of the diversity space tend to be more categorical and are not required to have totally ordered domains, as long as we have a distance function well-defined.

Depending on how interactive the utility function elicitation is, a diversity metric could be redefined to become utility metric. For example, if we initially use the hotel *activity type* as a diversity metric, if the recommender system notices that a user shows significant interest in beach activities, then *activity type* could be dynamically redefined to be a utility metric with "beach" ranked highest.

5.3 Diversity Approach

Our proposed diversity approach is a lightweight yet efficient randomized algorithm that both incorporates and relaxes Greedy by selecting from the best candidate points according to an exponential probability distribution [12]. This relaxation allows our algorithm to identify a better solution with high probability. The proposed approach can be used and applied on any space provided it is multi-dimensional. The proposed approach is not limited to be used with DG-RCA framework, but as a continuation of our work, we are using DG-RCA framework to present the approach. In addition, The proposed approach can be applied with any utility function. However, in our framework, the utility function is constructed during the utility axis selection step [6] where weights given to each metric attribute reflect the attractiveness of recommendations to the user. It is obvious that poor

utility function would result in a retrieval set that will not satisfy the user regardless of how diverse the set is [73]. In addition, giving recommendations by the utility learned may not provide sufficient diversity of recommendations. We would like to return a set of composite alternatives (recommendations) that are optimal or near-optimal with respect to utility yet each is unique in terms of other package characteristics.

In this section, we discuss two variations of the MDP (with respect to the first point selection) and the proposed algorithm to address them. In addition, we present pseudocode of two flavors of the algorithm.

Problem Definition [MAXDIV]: Given a set $\Re$ of N items and one distinguished item r in $\Re$, find a subset $S \subset \Re$ such that:

- $r \in S$
- $|S| = k$
- $Diversity(S) = max_{T \subset \Re \, , \, |T|=k} \, Diversity(T)$

One of the most popular heuristics to address this problem is the Greedy approach. Two heuristics were presented in [101], the best of which with respect to diversity is the "greedy selection". The first point to be selected is always the one with the highest similarity to the target. During each subsequent iteration, the case selected is the one with the highest combination of: (1) similarity to the target and (2) diversity with respect to the set of cases selected during the previous iteration. Presented in [33], how diversity can be increased using layering. In addition, [33] discussed the case where similarity is strictly relaxed to optimize diversity. However, [33] achieved less diversity than Greedy. As it stands, Greedy

will always choose the case that has maximum distance to what has been chosen so far, but this may not provide the optimum solution toward the end. Let us consider the example presented in Figure 5.1.

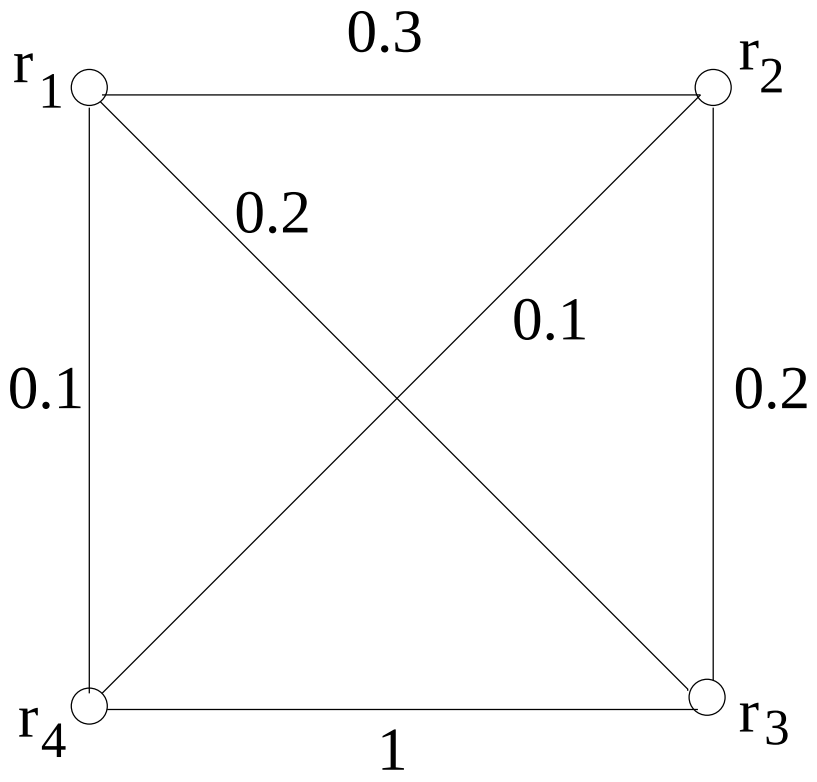

Figure 5.1: MDP Example

In this example, we show how the Greedy can get a solution arbitrarily far from optimum. We have a set of four candidate recommendations $\{r_1, r_2, r_3, r_4\}$ and we would like to pick the most diverse three recommendations from this set with diversity distances presented by edges between every two points[1]. As assumed by [17,33,101], there will be a starting point, say r_1, that maximizes the utility. Let us examine the operation of the

[1]For visual clarity, edges length do not represent the actual distance between any two points

Greedy algorithm, step by step:

1. Start with r_1.

2. Pick the point with maximum distance from r_1. Since the distances are $d(r_2, r_1) = 0.3$, $d(r_3, r_1) = 0.2$, and $d(r_4, r_1) = 0.1$, Greedy must pick r_2 at this step.

3. Pick the point with maximum combined distance to the points already selected. Since the combined distances for the two possible candidates are $d(r_3, r_1) + d(r_3, r_2) = 0.2 + 0.2 = 0.4$, and $d(r_4, r_1) + d(r_4, r_2) = 0.1 + 0.1 = 0.2$, Greedy will pick r_3.

Therefore, the complete solution found by Greedy is the set $\{r_1, r_2, r_3\}$. The resulting diversity of this solution is the average distance between any two points, which is 0.2333. However, the optimum solution for this diversity problem is $\{r_1, r_3, r_4\}$ with a diversity of 0.4333. The reason Greedy missed the optimum solution is because it greedily picks r_2 at step 2, when a better choice would have been to pick r_3, which enables the optimum solution.

Algorithm 2 RandDivFixed$(R, r, k, n_{rep}, \alpha)$

```
1: for rep = 1 to n_rep do
2:     sol = {r1}
3:     for i = 2 to k do
4:         S ← Select(R − sol, sol, L)
5:         m ← Pick(S, P, α)
6:         sol ← sol ∪ {m}
7:     end for
8:     if Diversity(sol) > Diversity(best_sol) then
9:         best_sol ← sol
10:    end if
11: end for
12: return best_sol
```

Inspired by the above example, we are proposing a randomized algorithm. We start

with describing the algorithm and the pseudocode is given in Algorithm 2. The parameters are as follows: R is a set of candidate recommendations (already selected such that their utility is within a fixed threshold ϵ from the maximum utility), a fixed recommendation r to be used as starting point (r is the recommendation with maximum utility), k is the number of recommendations that need to be returned, n_{rep} is the number of iterations, and α is a fixed attenuation factor. The main loop repeatedly builds a solution sol and retains the best solution found so far in $best\_sol$. At each iteration the solution is first initialized to the fixed recommendation r. For each of the $k-1$ remaining slots, a recommendation is picked from a list of the top candidate recommendations, ordered by their combined distance to the points selected so far (see function Select in Algorithm 3).

Algorithm 3 Select(C, sol, t)

```
1: for i = 1 to t do
2:    V[i] ← 0
3: end for
4: t ← min(t, |C|)
5: for all r ∈ C do
6:    if |sol| > 0 then
7:       d ← Σ_{s∈sol} dist(r, s)
8:    else
9:       d ← max_{s∈C} dist(r, s)
10:   end if
11:   for i = 1 to t do
12:      if d > V[i] then
13:         for j = t down to i + 1 do
14:            V[j], S[j] ← V[j − 1], S[j − 1]
15:         end for
16:         V[i], S[i] ← d, r
17:         break
18:      end if
19:   end for
20: end for
21: return S
```

Thus, the top candidate in this list is picked with a given probability P, the second best

with a smaller probability αP, the third best with probability $\alpha^2 P$, and the i-th best candidate with probability $\alpha^{i-1} P$. The last candidate is picked if none of the better candidates in the list is picked (see function Pick in Algorithm 4).

We repeat this process n_{rep} times and in every iteration we calculate the diversity of the resulting set and compare it with the highest diversity found so far, and finally return the best solution found. Intuitively, a smaller P allows the algorithm to take more risk at each step, and a larger P limits this risk. In Section 5.4 we show the impact of varying this parameter on the number of steps until converging to an optimal (or near-optimal) solution.

Going back to the example in Figure 5.1, we will analyze the probability that our approach would find the optimum solution. Figure 5.2 presents possible recommendation selection paths and their probability according to our algorithm. When we start with r_1 as a starting point, there will be three different choices to pick from with different probability for each. Thus, r_2 is picked with probability p, r_3 with probability αp and r_4 with the residual probability $1 - p - \alpha p$. Let us denote the optimum solution by $q = \{r_1, r_3, r_4\}$, and calculate the probability that we can find it with certain number of repetition for the algorithm. Clearly, at each iteration of the algorithm, there are two possibilities to get to optimum solution q, specifically $\langle r_1, r_3, r_4 \rangle$ and $\langle r_1, r_4, r_3 \rangle$. For $\langle r_1, r_3, r_4 \rangle$ the probability is αp^2 and for $\langle r_1, r_4, r_3 \rangle$, the probability is $(1-p-\alpha p)p$. In each iteration of the algorithm, the solution q will be found with probability $\alpha p^2 + p - p^2 - \alpha p^2 = p(1-p)$. Therefore, in n_{rep} iterations, the probability to find q in *any* of the iterations is $1-$ the probability to miss q in *every* iteration, that is: $1 - (1 - p(1-p))^n_{rep}$. For $p = 0.7$ and $n_{rep} = 50$ this amounts to over 99.999%

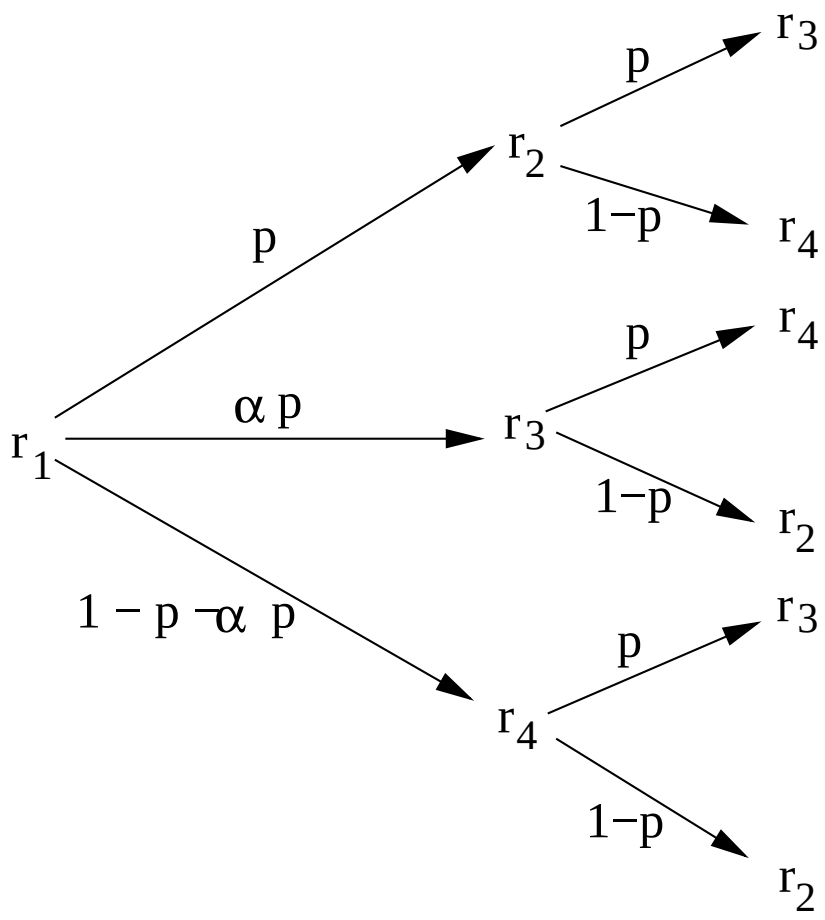

Figure 5.2: Probabilities of Recommendation Selections

The following problem is a variation of the MAXDIV, which removes the constraint on a fixed point. We call this problem MAXDIV-U (short for maximum diversity unconstrained). The domain of applicability of this problem is larger than recommender systems. Consider, for example, a wireless phone company which wants to select locations for installing k new cell towers from a set R of candidate locations, in such a way as to maximize the coverage. Clearly, the more diverse the set of selected locations is, the better the coverage (not considering other factors like elevation, obstructing buildings, etc.).

Algorithm 4 Pick(S, P, α)

```
1: t ← |S|
2: rand ← rnd() {between 0 and 1}
3: for i = 1 to t − 1 do
4:    if rand < P then
5:       return S[i]
6:    else
7:       return P+ = α · P
8:    end if
9: end for
10: return S[t]
```

Problem Definition [MAXDIV-U]: Given a set $\Re$ of N items, find a subset $S \subset \Re$ such that:

- $|S| = k$
- $Diversity(S) = \max_{T \subset \Re \, , \, |T|=k} \; Diversity(T)$

Several methods (e.g., [31,38,39,73,88]) for solving MAXDIV-U have been proposed in recent years. Clearly, exhaustively enumerating all subsets of $\Re$ with k elements is exponential in k and thus impractical.

The Greedy algorithm can be used with the starting point r selected such that it maximizes $\max_{s \in \Re} d(s, r)$. However, as we have shown, Greedy can obtain solutions far from optimal.

Another approach is proposed in [73] which relaxes MAXDIV-U to a trust-region problem. There, the proposed solution involves solving a parameterized eigen value problem and then quantize the resulting real solution to a binary one by retaining the k largest eigen values found, which we call here EigenSolver.

We extend our proposed randomized algorithm by allowing it to pick the starting point

from a list of L recommendations that have the largest maximum distance to other recommendations. The pseudocode for this algorithm is given in Algorithm 5.

Algorithm 5 RandDivFloat(R, k)

```
1: for rep = 1 to n_rep do
2:     sol = ∅
3:     for i = 1 to k do
4:         S ← Select(R − sol, sol, L)
5:         m ← Pick(S, P)
6:         sol ← sol ∪ {m}
7:     end for
8:     if Diversity(sol) > Diversity(best_sol) then
9:         best_sol ← sol
10:    end if
11: end for
12: return best_sol
```

5.4 Experimental Evaluation

In order to evaluate the effectiveness and efficiency of the proposed approach, we used both synthetically generated datasets as well as data extracted from a publicly available travel site. The synthetic data was obtained by generating random distances in a set of 1000 points, first with skewed distribution then with a uniform distribution. All generated distances were between 0 and 1. The travel data consisted of a set of 714 hotel records including location (latitude, longitude), activities and description. For the hotel dataset, we defined a distance function by combining the following attribute-specific distances: Euclidian distance for hotel location, the depth difference on activity taxonomy hierarchy for the activities, and $1-$ cosine TF/IDF keyword similarity for description, then pre-computed distances between every two hotels.

We start by evaluating RandDivFixed against Exhaustive and Greedy. Then we compare the effectiveness and performance of RandDivFloat against EigenSolver [73].

5.4.1 Evaluation of RandDivFixed

For the first experiment, we used the hotel dataset and we ran 100 tests on randomly selected samples using a machine with Intel Core 2 Duo CPU 2.53GHz and 3GB RAM. We used a sampling ratio of $\frac{1}{7}$, thus each sample consisted of about 100 hotels. For each run, we run the Exhaustive algorithm to calculate the optimum diversity for that sample. Then, we ran RandDivFixed with $P = 0.4$ for k from 3 to 7. We could not run tests for higher values of k because Exhaustive already took 18 hours to complete the 100 tests for $k = 7$ (for $k = 8$ it would take 100 times more time). For each test we measured the percentage of time RandDivFixed reaches the optimum solution after a variable number of iterations (the parameter n_{rep} in Algorithm 2). We show the results in Figure 5.3.

This experiment shows that RandDivFixed always converges to the optimum solution in at most 1500 iterations, with a total running time of at most 0.1151 seconds per test (see Table 5.1 for details). We repeated this experiment with various values for the parameter P (0.4, 0.5, 0.6, 0.7) in Algorithm 2 to study the impact on the convergence to the optimal solution. We noticed that for higher values of P the algorithm achieves a higher quality faster but takes more iterations to converge to the optimum solution. Conversely, with low values of P the algorithm starts with much poorer solutions in the first few iterations, but converges to the optimal solution within 1500 iterations. The practical relevance of this

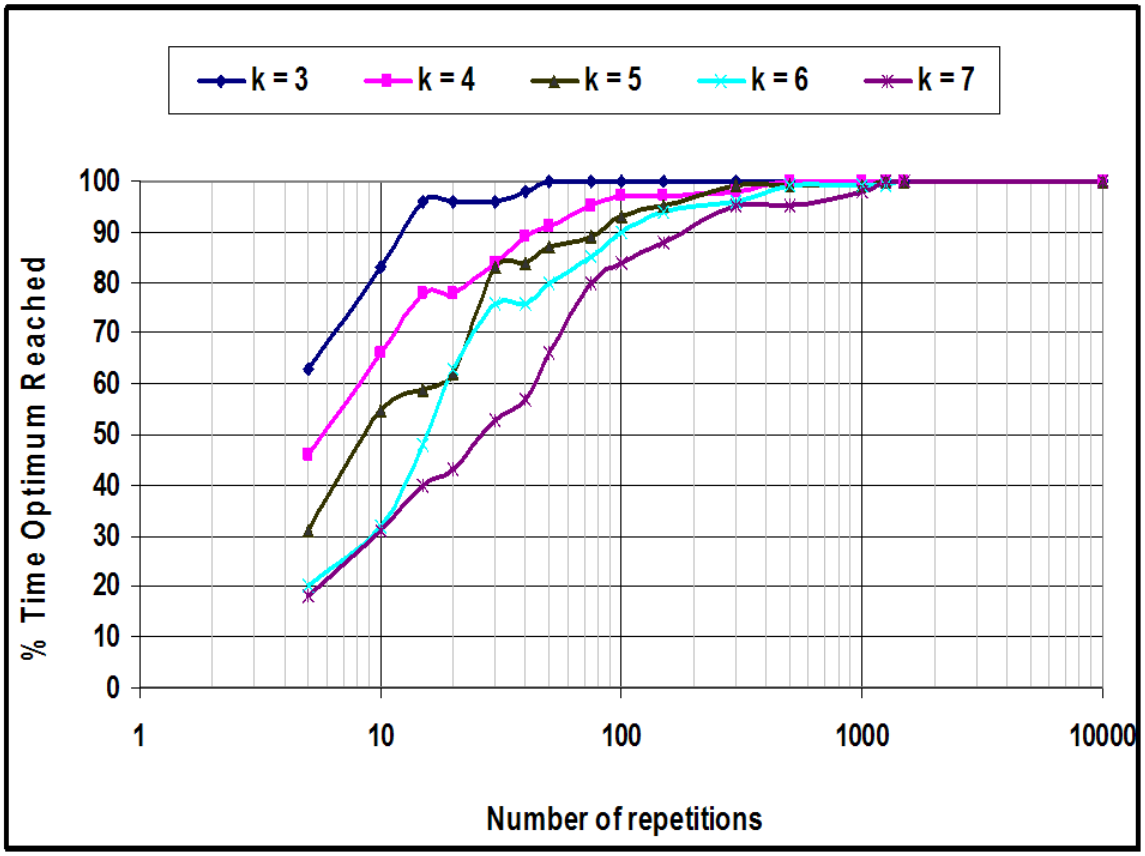

Figure 5.3: Convergence to Optimum Solution

observation is that one can adjust the value of P to fit within a time budget: under time pressure, it would be better to run with a high value of P to improve the chances that a better solution is found faster; conversely, under more abundant resources, one might want to use a low value of P to obtain the optimal solution towards the end.

Table 5.1: RandDivFixed Running Time

$k = 3$	$k = 4$	$k = 5$	$k = 6$	$k = 7$
0.0319 s	0.0492 s	0.0720 s	0.0942 s	0.1151 s

In the same experiment we also ran the Greedy algorithm and recorded the percentage of time RandDivFixed obtains a better solution than Greedy, using different numbers of repetitions. The results are shown in Figure 5.4. Notice that this time we were able to compare with Greedy using larger values for k (up to 15).

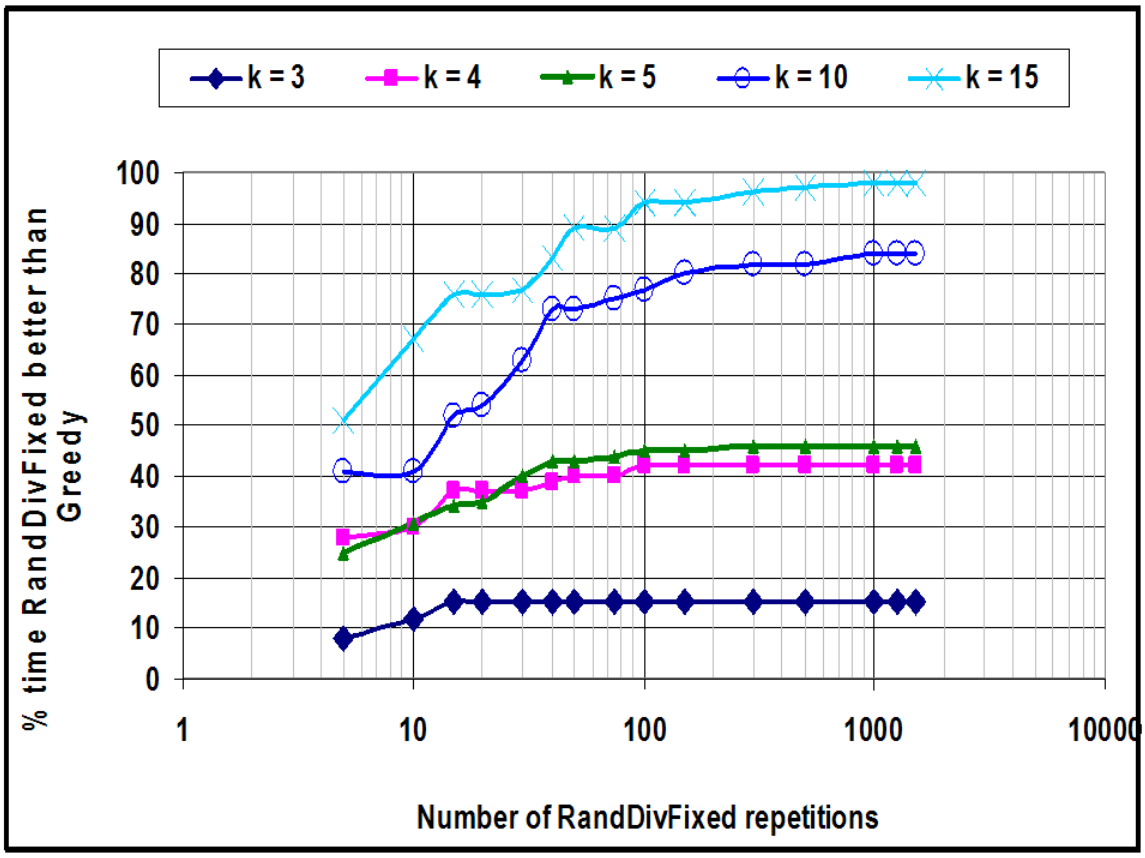

Figure 5.4: RandDivFixed versus Greedy

This experiment shows that RandDivFixed always produces a solution at least as good as Greedy, and for all values of k, a solution strictly better than Greedy is found between 15% (for $k = 3$) and 95% (for $k = 15$) of the time, when RandDivFixed is repeated 1500 times.

From the above experiment we noticed that the absolute diversity values found by the algorithms are dependent on the data distribution of the distances between the points in the dataset. The next experiment presents how changes in the distance distribution affects the ratio between the diversity found by Greedy compared to RandDivFixed. For this experiment we used a synthetically generated dataset of 100 points with a skewed data distribution for the distances. The relative diversity value obtained by Greedy when comparing with RandDivFixed as a baseline is shown in Figure 5.5. For this dataset, the diversity value obtained by Greedy is between 5% (for $k = 3$) and 13% (for $k = 15$) off the diversity value obtained by RandDivFixed. For fairness, we repeated this experiment with uniformly distributed random dataset and Greedy was only between 2% and 3% off RandDivFixed. This shows that in practice, the absolute diversity improvement will be dependent on the actual data and distance function used. However, across all tests we ran, RandDivFixed was consistently outperforming Greedy.

5.4.2 Evaluation of RandDivFloat

As discussed in Section 5.3, the unconstrained diversity maximization problem (MAXDIV-U) can be solved using RandDivFloat or the EigenSolver solution proposed in [73]. In this section, we compare the two approaches both for effectiveness and running time. For this experiment shown in Figure 5.6, we used the same hotel dataset.

This experiment shows that RandDivFloat almost always produces a solution strictly

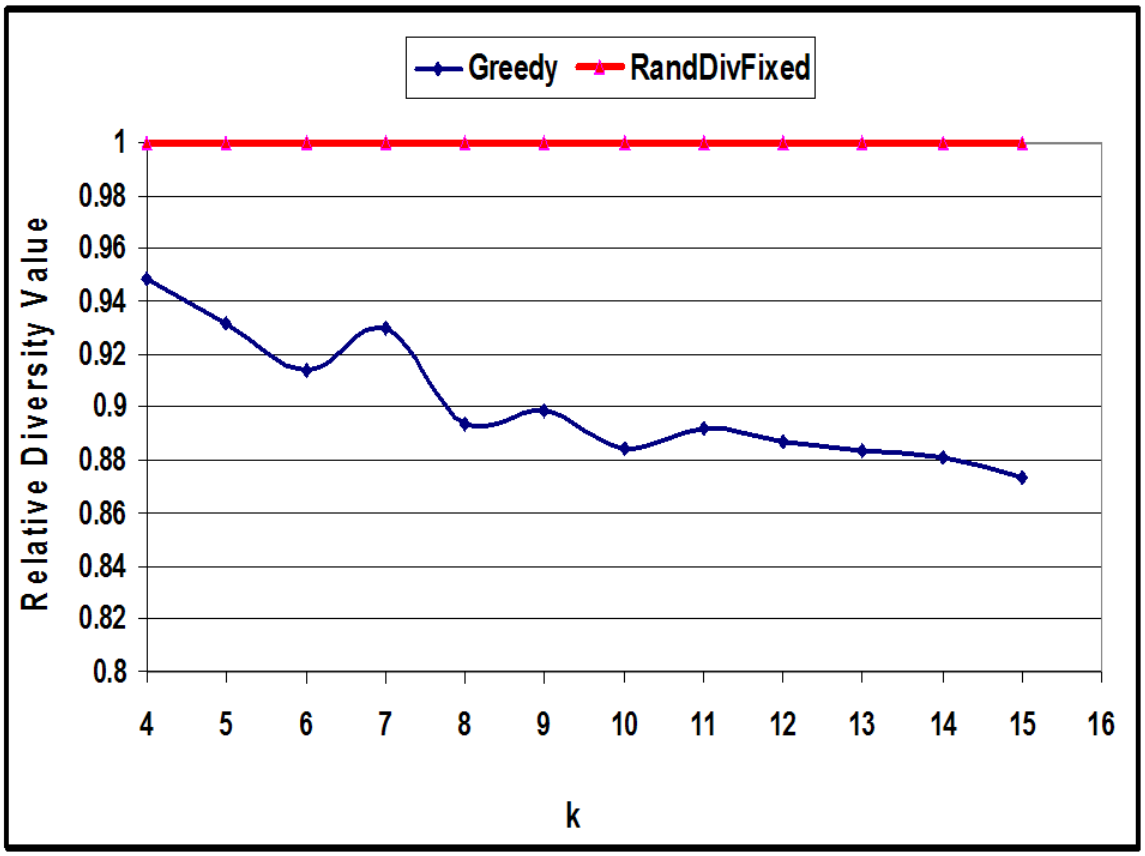

Figure 5.5: Relative Diversity Value

better than EigenSolver, even when RandDivFloat is repeated as little as 10 times. Moreover, for $k \geq 6$, RandDivFloat always outperforms EigenSolver.

Next, we wanted to examine the scalability of both algorithm with respect to the size of the dataset. For this purpose, we varied the sampling ratio for the dataset selection between $\frac{1}{7}$ to $\frac{7}{7}$ (at $\frac{7}{7}$, all 714 hotels are selected). We ran the experiments with different k values (between 3 and 15) and the results are not much different (between 3 and 15), so we show only the running time for $k = 5$ for both algorithms in Figure 5.7. The maximum running time for RandDivFloat is about 1 second (for 714 points), whereas EigenSolver takes over

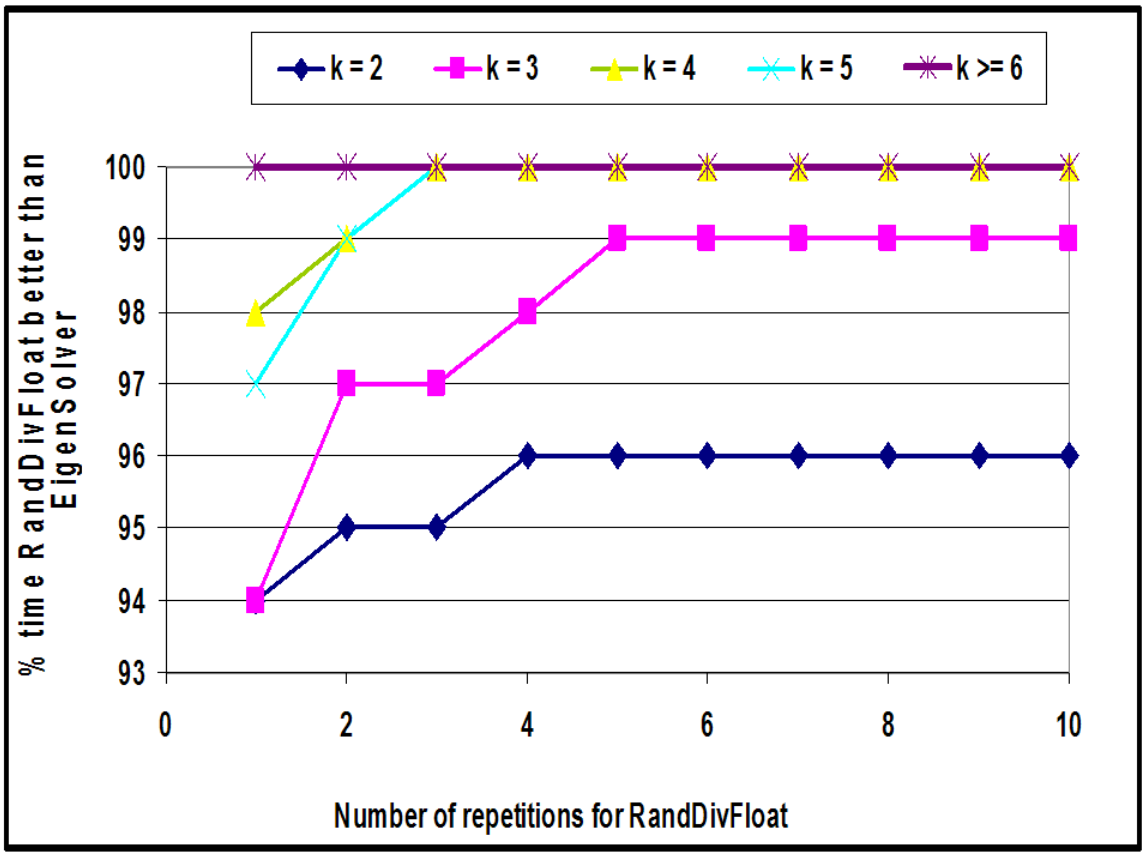

Figure 5.6: RandDivFloat versus EigenSolver

225 seconds on the same dataset.

5.5 Summary

In this paper, we introduced a new approach for diversifying a set of recommendations, where an m-dimensional diversity space is constructed and used. In addition, we proposed and validated a randomized algorithm to address the Maximum Diversity Problem (MDP) which is both highly effective and scalable in our experiments. Many research questions

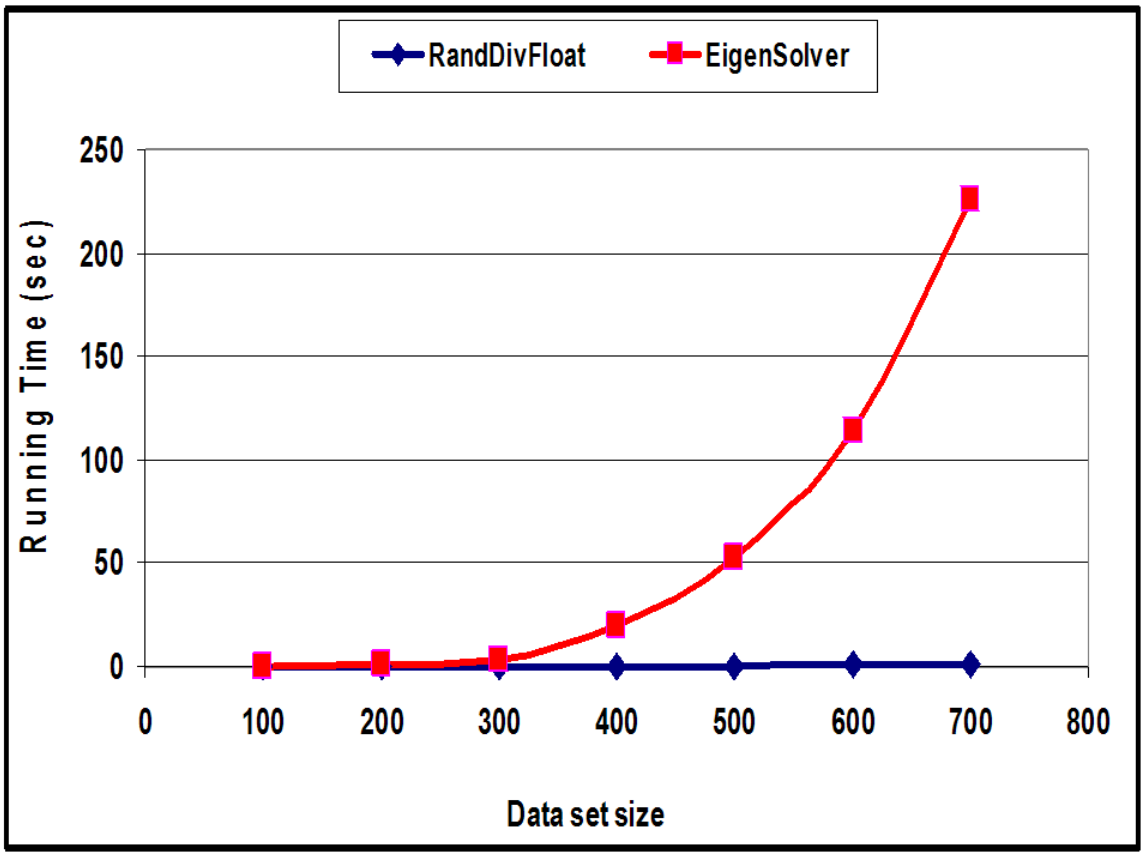

Figure 5.7: Execution Times of EigenSolver and RandDivFloat

remain open. They include (1) how to select the probability P and the factor α for the randomized algorithm in order to converge in the fewest number of repetitions; (2) how to efficiently detect when the best found solution is in fact optimal, or at least within a given factor of the optimal solution; and (3) how does the choice of a distance function affect the quality of the solutions found by each algorithm.

Chapter 6: An Adaptive Utility Learning Method for Composite Recommendations

6.1 Introduction

This chapter focuses on learning and eliciting user preferences to recommend composite services and products. Most of today's recommender systems rely on a single ranking or utility score, whereas, in many applications, there are multiple criteria that need to be taken into account, such as price, quality and enjoyment. Recently, multi-criteria ranking has been explored in recommendation set retrieval (e.g. [5,6,7,9,68,108]). These methods choose a set of alternatives based on a distance measure calculated for each of the multiple criteria. Multi-criteria ranking can help provide a balance between diversity and optimality.

Recent popular surveys (e.g., [1,22,68]) classify the current generation of recommendation methods into three main categories: content-based, collaborative, and hybrid recommendation approaches. Another way to classify recommender systems is based on techniques: either heuristic-based or model-based [1]. The main difference between model-based techniques and heuristic-based techniques is how to calculate the utility (rating) predictions. In the heuristic-based approach, calculation of predicted utility (rating) is based on some ad hoc heuristic rules, whereas in the model based approach, calculation is based on a model learned from the underlying data using statistical learning techniques [1]. For

comprehensive analysis of recommender systems refer to [1,68]. Most relevant to our work are those systems supporting multi-criteria ranking methods, utility function elicitation, and dynamic learning of user preferences. There are several approaches for eliciting utility functions, most of which aim for semi-automated learning of a decision makers' utility function [113]. One approach is iterative learning and refinement of the users' utility function using a value of information approach. Another approach is by eliciting the utility function from a database of observed behavioral patterns. A third approach is by eliciting the utility function from a database of already elicited utility functions. While few recommender systems provide for estimating and refining the preferences of the user [68]. However, none of these works, to the best of our knowledge, work on recommendations for composite product and services, which makes the recommendation space very large (or infinite, for continuous selected quantities), and implicitly defined.

The CARD Framework [5] supports composite product and service definitions, and gives recommendations that are based on dynamically learned utility function and decision optimization. Composite services in CARD are characterized by a set of sub-services, which, in turn, can be composite or atomic. CARD uses a decision-guidance query language (DGQL) to define recommendation views, which specify multiple utility metrics, and the weighted utility function. DG-RCA [6.7.9] is based on CARD framework. DG-RCA suggested a technique to elicit utility function and a diversity method. However, the DG-RCA framework has a number of limitations. Up to this point, DG-RCA doesn't leverage historical information about users and therefore has to always start the utility elicitation process from standard utility axes. In addition, DG-RCA uses a static (non-adaptive)

method to converge axes after extracting feedback from the user. In this paper we address limitations outlined above, more specifically, the contributions of this chapter are as follows.

First, we incorporated a regression analysis technique to predict the user preference. The preference learning is based on historical multi criteria rating submitted by the same user on similar products or services. There is a consistent family of criteria $m_1, \cdots, m_n$, each criterion is represented by a rating given by the user as a real value m_i in the range of $[m_i*, m_i^*]$ where m_i* and m_i^* are the worst and the best level of the i-th criterion respectively.

Second, we enhanced the DG-RCA method for utility function elicitation, that is based on an iteratively refining a set of axes in the n-dimensional utility space. The starting point of utility function of the prospective user is learned by history mining. Based on the learned weights, an adaptive technique is used to adjust the standard axes toward learned axes depending on confidence degree. Then the user is presented with a set of recommendations each being optimal for one of the current axes.

Third, we conducted extensive experimental studies on the efficacy of the proposed algorithm to compare precision of our ranked recommendations with the previously used [6] standard axes technique. We found and used a dataset in the Yahoo! Movies Web site (http://movies.yahoo.com) to demonstrate system's performance. The study suggests that significant improvements in the utility of recommendations can be obtained using the proposed method.

This chapter is organized as follows: Section 6.2 describes the Adaptive Selection of

Initial Utility Axes, and explains the motive and benefits of using an adaptive axes selection compared with standard one. Section 6.3 presents experimental studies for the purpose of validating the framework using Yahoo movie dataset. Section 6.4 is the summary and possible extensions.

6.2 Adaptive Selection of Initial Utility Axes

As noted in Chapter 4, Section 2, the original utility axes selection method presented in [6] has not leveraged historical information to learn the preference of the user, resulting in the use of standard utility axes. In addition, the adjustment of axes has been static. In this section, we describe the new proposed adaptive algorithm for selecting the initial utility axes for the iterative utility elicitation for a given user. The algorithm has three phases. First, we analyze the historical information consisting of multi-criteria ratings of previously purchased products, to produce an estimated utility axis for the user. Second, depending on the computed confidence on this derived utility axis, we adjust the standard utility axes toward the learned utility axis. The degree by which the axes are altered is proportional to the confidence in the learned utility. Third, for each of the modified axes, we select a recommendation which exhibits the best utility according to that axis and present them to the user. Therefore, the number of recommendations presented to the user is equal to the number of criteria (or axes).

Figure 6.1 exemplifies recommendations proposed to the user.

Phase I: Learning the User Preferences

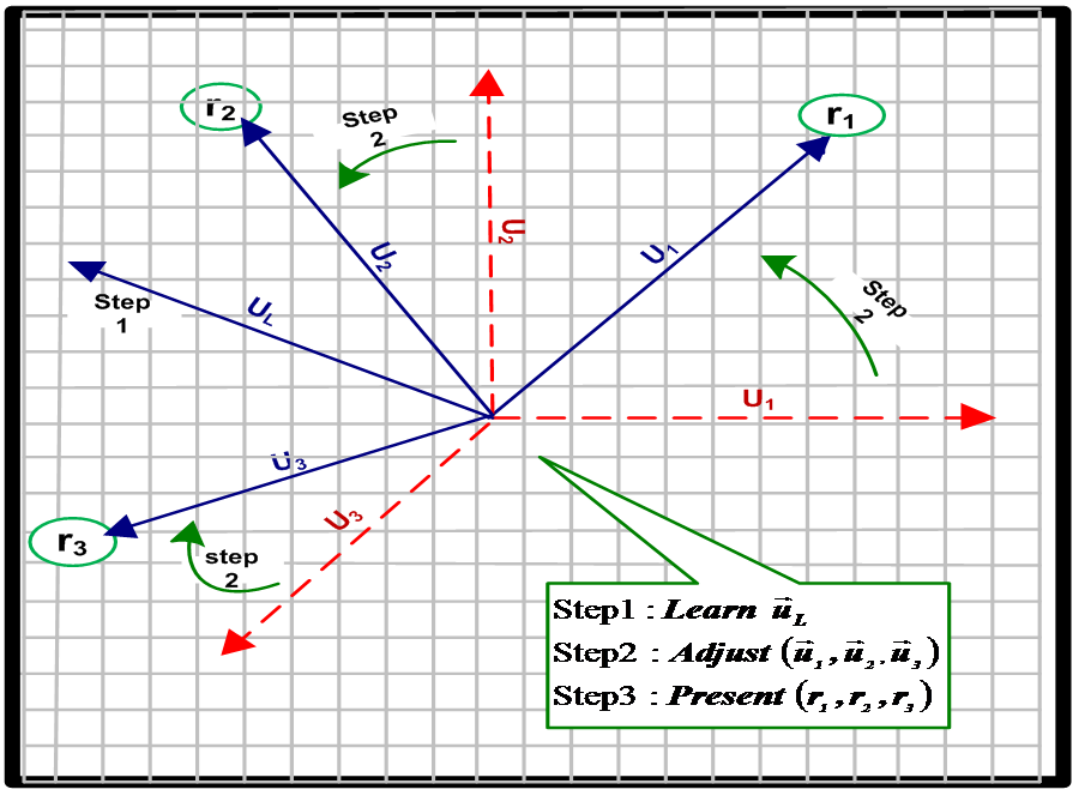

Figure 6.1: Example of Learned Utility Axes

In this phase, we collect all the previous ratings R_u on previously purchased products from the current user u. These include ratings on each of the individual criteria as well as an overall rating. One rating is a tuple of the form $r = (u_1, \cdots, u_n, u)$, where $u_1, \cdots, u_n$ are the ratings on the individual criteria and u is the overall rating. Subsequently, we express the overall utility function $U(r)$ as a weighted sum of marginal utility values on each of the individual criteria: $U(r) = \Sigma_{i=1,n} w_i * u_i$. Learning the user preferences amounts to approximating the overall rating by the overall utility function applied on the individual criteria ratings. This reduces to finding the values of the weights $w_1, \cdots, w_n$ that minimize

the approximation error:

$$error(w_1, \cdots, w_n) = \Sigma_{r \in R_u}(u - \Sigma_{i=1,n} w_i * u_i)^2$$

Finding the values of $w_1, \cdots, w_n$ that minimize $error(w_1, \cdots, w_n)$ can be done for example using a non-linear solver package, or though a variety of other methods [60]. In our implementation, we used an fast randomized heuristic approach that computes a good approximation of the optimum weights. The user preferences are thus expressed through a utility axis $\vec{w} = (w_1, \cdots, w_n)$.

Phase II: Deriving Utility Axes

In the second phase, we derive n utility axes by altering the n standard axes $s\vec{a}_1, \ldots, s\vec{a}_n$ towards the utility axis $\vec{w}$ learned in Phase I. The amount by which the standard axes are altered is proportional to the confidence α in the learned utility. We compute the confidence as follows:

$$\alpha = 1 - error^*$$

where $error^* \in [0, 1]$ is the normalized approximation error:

$error^* = error(w_1, \cdots, w_n)/(|R_u| * max(u)^2)$.

The algorithm for computing the modified axes is given in Algorithm 6.

Algorithm 6 Computing modified axes

1: **for** $i = 1$ to n **do**
2: $\quad m\vec{a}_i = (1 - \alpha) \cdot s\vec{a}_i + \alpha \cdot \vec{w}$
3: **end for**

Phase III: Selecting Recommendations

Once the n modified axes are computed in Phase II, we select n recommendations from the set of Candidate Recommendations (CR) as follows: for each modified axis $\vec{ma}_i$ we pick one recommendation r which maximizes the estimated utility with respect to utility axis $\vec{ma}_i$:

$$U_{\vec{ma}_i}(r) = \Sigma_{j=1,n} ma_{ij} \cdot r_j$$

6.3 Experimental Evaluation

In order to evaluate our proposed recommender system, we used the publicly available dataset in Yahoo! Movies Web site (http://movies.yahoo.com) to demonstrate system's performance. The dataset has rating information submitted by users for several movies. These ratings include four criteria which are a) story, b) acting, c) direction and d) visuals, in addition to an overall rating, user ID, and movie ID. The collected data came from randomly selected movies encoded with a serial number from 1 to 48. Each user has rated at least 7 movies up to 48 movies. The rating values as given by the users were in range of 1 to 13. The dataset has 34,800 ratings for 1,716 users. Finally, the entire data set as described is separated into two disjoint sets, the training set used for the purpose of learning the preference of the user, and the test set used for evaluating the overall utility of the recommendations.

For each user we generated 4 recommendations which score best on the standard axes and 4 recommendations using the modified axes computed using the adaptive method described in Section 6.2.

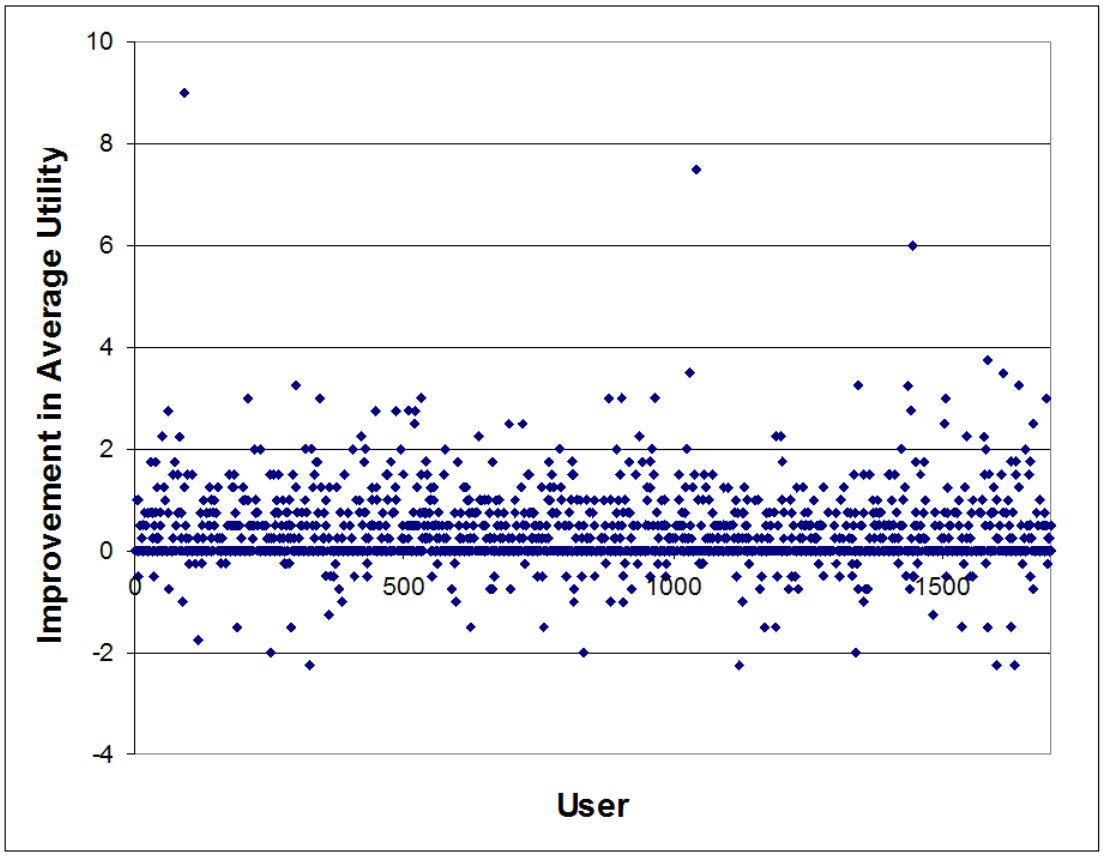

Figure 6.2: Average Utility Improvement for Each User

Figure 6.2 plots the improvement in average utility of the recommendations generated using the adaptive method over the recommendations generated using the standard axes method. As we can see, for the vast majority of the users, the adaptive method produces significant improvements in utility (as high as 9 points on the 1 to 13 scale, with 1 to 4 points typical improvement).

6.4 Summary

In this paper we refined a method for providing recommendations of products and services. The method learns the users' preferences from historical data, and according to the confidence degree, adjusts standard utility axes in the n-dimensional utility space to reflect what has been learned, and finally presents the user with a number of recommendations, each of which is optimal for an axis. Many research questions remain open. They include (1) how to enhance the method for users with no or low history information; (2) how to enhance the learning of the current user's preferences to incorporate learning from other similar users ("collaborative filtering").

Chapter 7: A Confidence-Based Recommender with Adaptive Diversity

7.1 Introduction

This chapter focuses on presenting a collaborative filtering (CF) technique and the impact of adaptive diversity to recommend composite products and/or services. Content-based systems often employ classifier techniques that rely on information retrieval and information filtering to recommend an item to a user based upon a description of the item and a profile of the users interests. In contrast, Collaborative recommenders use the preferences of "similar" users, rather than the characteristics of an item, to make suggestions to the current user. The hybrid approach combines methods from collaborative and content-based approaches. More recently, utility-based, knowledge-based, and demographic systems have each suggested different techniques for providing recommendations. Systems employing one or more of these techniques have been proposed to recommend flights, movies, restaurants, and news articles (e.g., [1,12,17,31]).

There are several hybrid recommender exists where two or more techniques are combined to produce better recommendation results. According to a popular hybrid recommender survey [22], and to the best of our knowledge, there is no hybrid recommender system that combines CF and knowledge-based in such sequence. DG-RCA is the first

recommender system that utilizes hybridization techniques which are based on a cascaded collaborative recommender and knowledge-based. DG-RCA framework presents a hybrid recommender system that is a cascaded collaborative recommender and knowledge-based. DG-RCA learns the utility of the user with a collaborative filtering technique then cascade the result set to be refined with a knowledge-based technique based on diversity space, and finally present the user with a diverse set of composite products and services.

Content-based and Collaborative-based techniques both have limitations, however, it is been noted that CF techniques outperform those of Content-based, most of the time [1,98,104]. The basic principle of CF is that if user X and user Y shared the same taste or preference in the past, then to a degree, we can predict what they might like in the future. Recent surveys [1,104] classify CF methods into memory-based and model-based. Memory-based methods use the rating data of users to calculate the similarity with other users and make predictions for recommendations according to those calculated similarity values, an example is http://www.amazon.com/. Memory-based methods could result in less reliable similarity measures when data are sparse and common items are few between users. This limitation can be addressed using Model-based CF methods. Model-based methods use the pure rating data to estimate or learn a model to make predictions [1,104], an example is MovieLens and Netflix prize.

Recently, multi-criteria ranking has been explored in recommendation set retrieval [1,5,6,7,8,17,68]. These methods choose a set of alternatives based on a distance measure calculated for each of the multiple criteria. Most research in the field of recommender systems is focused on improving the accuracy of the recommendation set with respect to

the query submitted, but less attention is given toward improving the quality of the recommendation set [73]. Multi-criteria ranking can address this issue and help provide a balance between diversity and optimality. Limiting recommendations only to those that are relevant to users' requests, would result in returning a set of alternatives that are often similar to each other and do not provide enough diversity. Diversity is important because it helps users to be aware of choices they may not have thought of, and allows returning a set of recommendations with a range of options that are not too similar yet score high with respect to the utility (e.g., represented by the query). For example, a person whose utility is mostly in favor of low price may decide to buy an attractive travel package even if the price is not minimal as long as the price is within a reasonable range. It is wise to assume that users want to be returned a set of alternatives that are similar to the query but not similar to each other. Assume you submit a query for a laptop computer with a specific price range and all models returned are from the same manufacturer. If you cannot purchase this brand for some reason, then all presented recommendations are useless [17].

DG-RCA [6,8] suggested an efficient technique to elicit utility function on utility space and a diversity method using the diversity utility space. The case study conducted in DG-RCA showed strength in utility function elicitation. In addition, DG-RCA [7] presented the separation of **utility space**, e.g., in travel domain, utility can be represented by metrics such as *Saving* and *Enjoyment*, from **diversity space**. Diversity space would be represented by metrics that are different from those of utility space, for example, *Activity types* and *Duration* of the trip. Typically, utility space is low dimensional, whereas diversity space is of considerably higher dimensionality. However, DG-RCA did not examine the quality

of recommendation when diversity is introduced. In addition, DG-RCA has not leveraged learning from users' historical information which resulted in using the standard utility axes when starting the utility elicitation process. Furthermore, the presented diversity technique was static where the used diversification technique did not leverage learning to introduce adaptability when diversifying the result set. To the best of my knowledge, this is the first attempt to incorporate learning from historical data into the diversity process where the scope of candidate recommendation space is refined to reflect users' preferences.

In this chapter, we adopt the DG-RCA framework and resolve the limitations outlined above. The key idea of this paper is to introduce a collaborative filtering technique in the framework, and measure the quality of recommendation set when using an adaptive diversification technique. More specifically, the contributions of this chapter are as follows.

First, we introduce a new technique for collaborative filtering to learn the preference of the user from history rating data, and then estimate similarity among users based on confidence measure. The technique is light yet efficient, and is based on a threshold for the number of co-rated items. It enforces a positive correlation threshold between any two users. Our Confidence-based collaborative filtering technique is able to identify recommendations that are optimal or near-optimal in terms of the learned utility.

Second, we refined a randomized diversity algorithm that provides a competitive solution with respect to finding a diverse set from candidate recommendations. The randomized algorithm is adjusted to be adaptive based on learning, and limits the scope of candidate result set for diversity. The algorithm is to iteratively relax the selection by the Greedy algorithm [17,101], with an exponential probability distribution.

Third, we conducted extensive experimental studies on the efficacy of the proposed CF method proposed to compare precision of our ranked recommendations with a baseline, where a prediction for an item is that item's mean rating, and the item-based k-Nearest Neighbor (kNN) algorithm. In addition, we measured the impact of adaptive diversity on the quality of the result set presented. Experiments suggest that the proposed CF algorithm consistently identifies result sets with an average precision of 90% and Mean Absolute Error of less than 8% . In addition, we show how is the proposed adaptive diversity technique achieved mean actual rating of 4.469 out of 5.0 and over 84% of diversity on a dataset of 10 million ratings for 10681 movies by 71567 users from Movielens (movielens.umn.edu).

This chapter is organized as follows: Section 7.2 presents and explains the proposed CF algorithm. Section 7.3 discusses the idea and impact of Adaptive Diversity. Section 7.4 covers the validation part. Section 7.5 is the summary and possible extensions.

7.2 Collaborative Filtering Technique

DG-RCA [6,7,8] is mainly based on leveraging multi-criteria for decision making. The notion of n-dimensional utility space and m-dimensional diversity space is direct representation of moving beyond single criteria decision making. However, one of the limitations of the DG-RCA framework is the lack of users' preference learning. To alleviate this, in this work we are focusing on exploiting the available historical information gathered from users' past interactions with the system. Specifically, we propose a collaborative filtering technique that incorporates a lightweight learning of similarity among users.

When deciding which users to consider for collaborative filtering, it is important to select users that are sufficiently similar to the current user. There are several aspects to this: first, in order for a user to be considered as relevant for the current user, she needs to have at least a certain number of rated items in common with the current user; second, the ratings on the common items need to be positively correlated; finally, the correlation among the ratings from the two users needs to be sufficiently strong. The correlation between the ratings of two users x and y is defined as the standard Pearson correlation coefficient:

$$corr(x,y) = \frac{\sum_{s \in Sxy} (r_{x,s} - \bar{r_x})(r_{y,s} - \bar{r_y})}{\sqrt{\sum_{s \in Sxy} (r_{x,s} - \bar{r_x})^2 \sum_{s \in Sxy} (r_{y,s} - \bar{r_y})^2}}$$

where S_{xy} are the common items rated by users x and y and $\bar{r_x}$ denotes the average of all of user x's past ratings.

Formally, if we denote by $r_{c,s}$ the rating given to item s by user c, we define the set $\hat{C}$ of relevant users for a given user c as follows:

$$\hat{C} = \{c' : |S_{xy}| \geq min\_support \wedge corr(s', s) > min\_corr\}$$

where $min\_support$ is a threshold for the number of co-rated items and min_corr is a positive threshold on the correlation between two users.

Once we established the set $\hat{C}$ of relevant users for a given user c, we will now focus

on deriving a predicted rating for a new item from the ratings given by relevant users. Intuitively, the ratings coming from relevant users need to be combined to produce a predicted rating. One possible way of deriving a predicted rating for item s' is to simply take the average of all ratings $r_{c',s'}$ coming from relevant users $c' \in \hat{C}$. However, while simple, this method has several drawbacks. First, it doesn't account for differences in the effective rating scales of different users: some users may be more willing to give high ratings while others are more strict. Second, it treats all relevant users' ratings the same way, even though some users are more strongly correlated with the current user than others.

Our proposed CF technique aims to use the similarity between relevant users and the current user as a confidence measure indicating the degree to which we rely on these users' ratings when predicting ratings for the current user. We call our CF technique *Confidence-based Collaborative Filtering* (CCF). Specifically, CCF combines the ratings from relevant users as follows:

$$r_{c,s} = \bar{r_c} + \alpha \sum_{c' \in \hat{C}} corr(c', c)^k \times (r_{c',s} - \bar{r_{c'}})$$

where $\alpha = 1/\sum_{c' \in \hat{C}} corr(c', c)^k$ is a normalization factor and k is an attenuation parameter that determines the degree of confidence in the ratings learned from relevant users.

Once the predicted ratings are computed for each movie, the method returns the top 5 items in order of their predicted rating as recommendation set to the user.

The advantage of this method is twofold: first, it gives more weight to ratings coming from strongly correlated users; second, it also accounts for the different rating scales by appropriately translating the ratings from relevant users c' into the rating scale of the current

user c.

7.3 Adaptive Diversity Approach

The new surge of current mobile computing devices such as PDAs and WAP-enabled mobile phones with screen size that could be 200 times smaller than that of a PC, is adding another pressure on recommender systems to move beyond the accuracy measure when producing recommendation list [17]. This reduces the number of recommendations that can be returned in a single search. If the few returned recommendations are similar to each other, then it is unlikely the user will be satisfied. In addition, [96] makes the argument that diversity is also important for news aggregator sites because the resulting diverse set makes many people as possible feel that their viewpoint is heard. Furthermore, the utility is only an estimate to users' preferences, because users may have not explained their preference accurately, or they did but the recommender system could not capture the preference precisely. For this reason, many practical systems are interactive, allowing the user to scroll through a set of choices to make a decision [73].

Inspired by the idea of the example presented in [33], we illustrate our adaptive diversity approach for increasing diversity while preserving similarity in Table 7.1. We compare utility and diversity for three different approaches : Confidence-based Collaborative Filtering (CCF), Static Diversity Set (SDS), Adaptive Diversity Set (ADS). CCF is defined as the set of 5 recommendations that have the highest predicted rating, SDS is defined as the

set of 5 recommendations that are most diverse (based on certain criterion) among candidate recommendations without learning from user preferences. ADS is the set that is most diverse from the candidate result set and adaptive to the learned user preferences.

Table 7.1: Adaptive Diversity Example

ID	Utility	Genre	Visuals	CCF	SDS	ADS
5	1	Drama	A	1	1	1
7	1	Drama	B	1	1	1
3	0.67	Comedy	B	1		
15	0.67	Drama	D	1	1	1
20	0.67	Crime	D	1		
33	0.67	Drama	C		1	1
2	0.67	Drama	D			1
1	0.67	Drama	D			
11	0.33	Sci-Fi	E		1	
40	0.33	Sci-Fi	E			
MU				**0.802**	**0.734**	**0.802**
DIV				**0.8**	**1.00**	**0.9**

The example we are using is an artificial case in the movie domain. Let us assume the result set presented in Table 7.1 contains the top 10 movies with respect to Utility score. We would like to recommend a set of 5 movies form each method (CCF, SDS, ADS) to the user based on her purchase history. The 1's in the last three columns show the selected 5 movies for each method. We calculate the mean utility (MU) by averaging the utility of the items in the result sets identified. for example, the set of movies chosen by CCF is $\{$5, 7, 3, 15, 20$\}$, the mean utility MU $= \frac{1+1+0.67+0.67+0.67}{5} = 0.802$. We calculate the Diversity for each approach using the standard method presented in [3,5,9,18,22]. The diversity of a set is the average distance between any two members in a set s. $DIV(s) = \frac{\sum_{i=1}^{k}\sum_{j=i+1}^{k} d(i,j)}{\frac{k}{2}*(k-1)}$, where

$k = |S|$. For simplicity of illustration,We calculate the distance d between any two items i and j to be 1 if they are different on the Visuals criterion and 0 otherwise. for example, the diversity (DIV) of CCF (based on **Visuals**) $= \frac{(1+1+1+1)+(0+1+1)+(1+1)+(0)}{\frac{5}{2}*4} = 0.8$. The CCF method (described in Section 7.2) selects the top 5 movies according to the predicted utility. The SDS method diversifies according to the Visuals criterion, so it picks the top most movie for each distinct Visuals value. The ADS method is leveraging the learned genre preferences of this user (e.g. the user has shown a strong preference to the Drama genre) by diversifying on Visuals while at the same time staying within the preferred genres.

The reason for SDS to get relatively lower score with regard to mean utility is the fact that diversification was based on unconditional candidate recommendation space (namely choosing the most diverse recommendation set on Visuals). The adaptive diversification technique allowed us to choose a set of recommendations that scored the highest with respect to mean utility of 0.802 (same as CCF), and at the same time scored high with respect to average diversity with score of 0.9. This is an example that shows leveraging multi-criteria ranking can lead to optimal or near-optimal solutions with respect to mean utility and diversity at the same time.

We start with describing our proposed adaptive diversity set approach (ADS). ADS is a lightweight yet efficient randomized algorithm that both incorporates and relaxes Greedy by selecting from the best candidate points according to an exponential probability distribution [7]. This relaxation allows our algorithm to identify a better solution with high probability. The proposed approach can be used and applied on any space provided it is

multi-dimensional. The proposed approach is not limited to be used with DG-RCA framework, but as a continuation of our work, we are using DG-RCA framework to present the approach. In addition, The proposed approach can be applied with any utility function. However, in our framework, the utility measure is based on our Confidence-based Collaborative Filtering (CCF). It is obvious that a poor CF technique would result in a retrieval set that will not satisfy the user regardless of how diverse the set is [88]. In addition, giving recommendations by the utility learned may not provide sufficient diversity of recommendations. Our goal is to return a set of composite alternatives (recommendations) that are optimal or near-optimal with respect to utility yet each is unique in terms of other package characteristics.

ADS algorithm is an enhanced version of the RandDivFixed algorithm presented in [7]. The difference between AdaptiveDiversity (ADS) and SDS (referred in [7] as RandDivFixed) lies in the way we identify candidate sets at each step: in the case of AdaptiveDiversity, the function AdaptiveSelect constructs a set of candidates based on the learned preferences of the user, whereas the basic Select function does not incorporate any learned information. The pseudocode for the Adaptive Diversity algorithm is shown in Algorithm 7.

The parameters are as follows: R is a set of candidate recommendations (already selected such that their utility is within a fixed threshold ϵ from the maximum utility), a fixed recommendation r to be used as starting point (r is the recommendation with maximum utility), k is the number of recommendations that need to be returned, n_{rep} is the number of iterations, and α is a fixed attenuation factor. The main loop repeatedly builds a solution sol and retains the best solution found so far in $best\_sol$. At each iteration the solution is

Algorithm 7 AdaptiveDiversity$(R, r, k, n_{rep}, \alpha)$

```
1: for rep = 1 to n_rep do
2:     sol = {r1}
3:     for i = 2 to k do
4:         S ← AdaptiveSelect(R − sol, sol, L)
5:         m ← Pick(S, P, α)
6:         sol ← sol ∪ {m}
7:     end for
8:     if Diversity(sol) > Diversity(best_sol) then
9:         best_sol ← sol
10:    end if
11: end for
12: return best_sol
```

first initialized to the fixed recommendation r. For each of the $k-1$ remaining slots, a recommendation is picked from a list of the top candidate recommendations, ordered by their combined distance to the points selected so far (see function AdaptiveSelect in Algorithm 8). We point out that $C_{Adaptive}$ will contain the list of recommendations that satisfy conditions from user preference learning.

After AdaptiveSelect returns a list of candidates, the top candidate is picked with a given probability P, the second best with a smaller probability αP, the third best with probability $\alpha^2 P$, and the i-th best candidate with probability $\alpha^{i-1} P$. The last candidate is picked if none of the better candidates in the list is picked. (see function Pick in Algorithm 9).

We repeat this process n_{rep} times and in every iteration we calculate the diversity of the resulting set and compare it with the highest diversity found so far, and finally return the best solution found. Intuitively, a smaller P allows the algorithm to take more risk at each step, and a larger P limits this risk.

Algorithm 8 AdaptiveSelect(C, sol, t)

```
1: for i = 1 to t do
2:     V[i] ← 0
3: end for
4: t ← min(t, |C|)
5: for all r ∈ C do
6:     if r ∈ C_Adaptive then
7:         if |sol| > 0 then
8:             d ← Σ_{s∈sol} dist(r, s)
9:         else
10:            d ← max_{s∈C} dist(r, s)
11:        end if
12:        for i = 1 to t do
13:            if d > V[i] then
14:                for j = t down to i + 1 do
15:                    V[j], S[j] ← V[j − 1], S[j − 1]
16:                end for
17:                V[i], S[i] ← d, r
18:                break
19:            end if
20:        end for
21:    end if
22: end for
23: return S
```

Algorithm 9 Pick(S, P, α)

```
1: t ← |S|
2: rand ← rnd() {between 0 and 1}
3: for i = 1 to t − 1 do
4:     if rand < P then
5:         return S[i]
6:     else
7:         P+ = α · P
8:     end if
9: end for
10: return S[t]
```

7.4 Experimental Evaluation

In order to evaluate the effectiveness of the proposed approach, we used a dataset of 10 million ratings for 10681 movies by 71567 users from Movielens (movielens.umn.edu). This dataset was split into training data and test data. Each experiment was run 100 times and the variation on results was (+/- 1%) on average. The ratings are on a scale from 1 to 5 (5 being the best).

7.4.1 Collaborative Filtering Evaluation

In this section we do a comparative study between our proposed CF technique CCF (confidence-based collaborative filtering) described in Section 7.2 and two other CF techniques: BAS - a baseline, where a prediction for an item is that item's mean rating, and KNN - the item based k-nearest neighbor as described in [1]. Since there are a large number of CF techniques in existence, we cannot compare with all, and we are limiting our comparison to the two above mentioned techniques. All three methods select the top 5 movies in order of their predicted rating for each user.

As evaluation metrics we use the established MAE (mean absolute error) [1,104] which is a measure of accuracy of the predicted rating compared to the actual rating in the test dataset, as well as MAR (mean actual rating), which is simply the average actual rating of the recommendation result set computed by each technique. The results are shown in Table 7.2.

As we can see from this table, CCF outperforms the other two techniques both in terms

Table 7.2: Comparison of CF techniques

	BAS	**KNN**	**CCF**
MAR	4.097	4.114	4.469
MAE	0.266	0.211	0.075

of mean actual rating (almost 9% better than KNN) and mean absolute error (three times smaller error than KNN).

7.4.2 Adaptive Diversity Evaluation

We now focus on evaluating the additional benefit of the adaptive diversification method (ADS) described in Section 7.3. The scope of the comparison in this section is as follows: we are comparing ADS against a static (non-adaptive) diversification technique SDS as presented in [7] and our own utility-based CCF (no diversity). For fairness, all three techniques (ADS, SDS and CCF) use the same predicted rating mechanism described in Section 7.2. The only difference among them is the way the results are diversified: ADS diversifies recommendations while being mindful of the learned user preferences (e.g. interest in specific movie genres); SDS maximizes the diversity of the result set irrespective of any learned user preferences; and CCF does not attempt to diversify the result set and simply returns the top 5 movies, according to their predicted rating.

To begin with, we use the same MAR and MAE evaluation metrics as before (Section 7.4.1). In addition, we are also measuring the diversity (DIV) of the result sets as described in Section 7.3. The CCF results are obtained from the same run as before, and

the ADS and SDS are ran on the same data as CCF, for fairness. The results are shown in Table 7.2.

Table 7.3: Comparison of diversification techniques

	CCF	**SDS**	**ADS**
MAR	4.469	4.293	4.466
DIV	0.768	0.997	0.841
MAE	0.075	0.128	0.074

The results show that ADS is almost identical with CCF in terms of utility metrics (MAR and MAE) while at the same time offering a much better diversity than CCF. The SDS algorithm, on the other hand, while producing the most diverse result sets, exhibits a significant degradation in terms of utility metrics (MAR and MAE). This is, in part, due to the fact that SDS maximizes diversity even if that sacrifices the utility.

Next, we compare the popular classification accuracy metrics of average precision and recall [1,104] at rank k ($k = 1, \cdots, 5$) of the three methods. In our experiments, we consider a movie with an actual rating of 4 and above in the test data set to be a hit.

Average precision at a given rank k is the average percentage of hits presented in the top k results. The measured precision at ranks 1 to 5 is shown in Figure 7.1. As reference point, we calculated the average precision for an algorithm that selects movies at random (RND).

As we can see, the average precision of all three methods (CCF, SDS, and ADS) is over 80% at all ranks. This is attributable to the quality of the predicted ratings from our CCF technique. However, the average precision of the SDS method degrades at ranks 3, 4 and

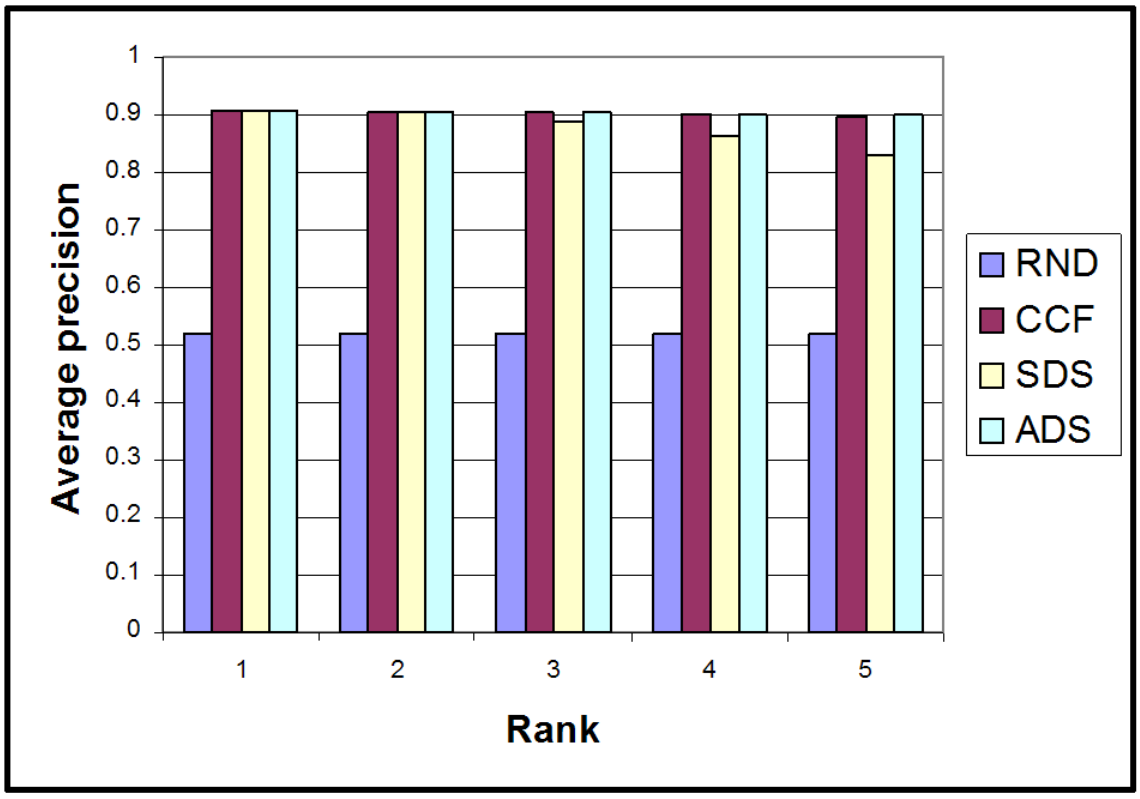

Figure 7.1: Average Precision at Rank k

5 because at this point the impact of sacrificing utility for the sake of diversity becomes noticeable. On the other hand, ADS does not suffer from this issue because it leverages the multi-criteria ranking where the result set was diversified by fixing one criterion to the learned values and restricting the diversification scope to the remaining criteria.

Average recall at a given rank k is the average percentage of all the movies rated above 4 by the current user (in the test data set) which are present in the top k results of that user. As reference point, we also calculated the theoretical best possible recall for each user by simulating a hypothetical optimum algorithm (OPT) which recommends only hits to each

user. Thus, the recall at rank k for OPT is k divided by the total number of movies in the test dataset. The measured recall at ranks 1 to 5 is shown in Figure 7.2.

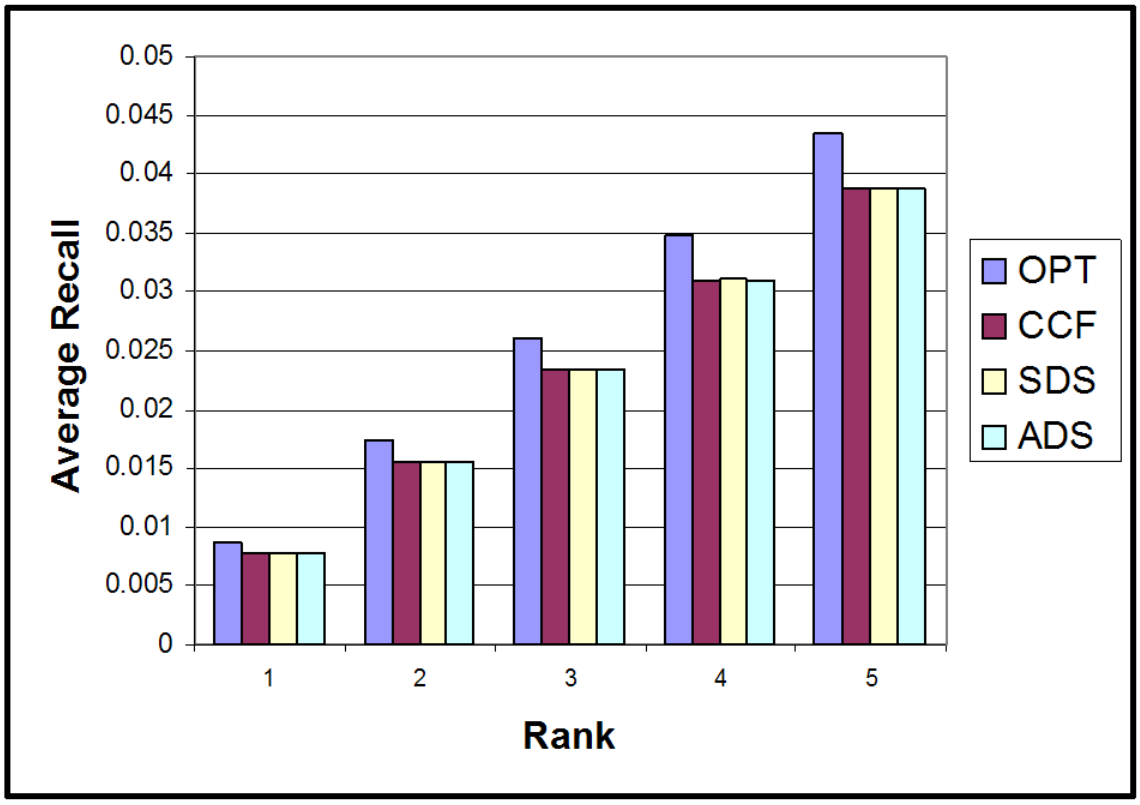

Figure 7.2: Average Recall at Rank k

At first glance, the absolute recall values may seem low (less than 5%). However, one should put these numbers into perspective by comparing them with the recall obtained by the theoretical best algorithm (OPT). Moreover, what is important is that the average recall is increasing with rank which is an indication that on average many of the results that are presented are in fact hits (as confirmed by the precision results too).

7.5 Summary

Multi-criteria ranking continues to enhance the quality of results when used in recommender systems. In our previous work [7], we have presented a case where multi-criteria learning of utility can enhance the quality of recommendations by 9% compared to single criterion learning. In this validation, we have shown how a better recommendation can be presented after multi-criteria ranking enabled adaptive diversity. One of the most popular recommender systems surveys have stated [1] the need to move beyond single criterion to extend capabilities of recommenders, for example, many restaurant guides, such as Zagats Guide, provide three criteria for restaurant ratings: food, decor, and service. Most of today's recommender systems recommend only atomic products or services. Complex recommendation models are rarely addressed, and one of the main enablers for working with composite alternatives is the use of multi-criteria. Many research questions remain open, such as how can we incorporate our confidence-based CF technique into our Utility Axis Selection in [6]. In addition, we would like to examine if Adaptive Diversity technique could be enhanced to provide recommendations with higher Utility.

Chapter 8: Conclusions and Future Research Directions

This chapter summarizes the research presented in this dissertation and suggests directions for future work.

8.1 Conclusions

I have discussed the challenges related to recommender systems, and presented a unified framework for recommending composite products and services. The framework incorporates an iterative process of 1) utility function elicitation and 2) diversification of a recommendation set. While there are several recommender systems exist (e.g., [1,22,104]), their scope is atomic products and services. There are several research (e.g., [91,92,94,103]) and commercial (Expedia.com, and Travelocity.com) recommender systems, however, they either provide partial product bundling, or the composition step is completed manually by the user. The proposed framework supports composite product and service definitions, and recommendations are based on dynamically learned utility function and decision optimization. The framework involves fast and easy interaction with the user to (a) choose a recommendation cluster the user is interested in, (b) dynamic elicitation of the weighted utility function, and then (c) generating a diverse set of recommendation that contains an optimal recommendation in terms of the estimated utility function.

In the case where no historical data exists to learn from, we developed a method for utility function elicitation. It is based on an iteratively refining a set of axes in the n-dimensional utility space, starting from the utility space standard axes. At every step, the user is asked to rank a set of recommendations, each being optimal for one of the current axes. Based on the user feedback, the method refines the set of axes which become closer to each other, until the user cannot differentiate among them. To add breadth to my research, I have expanded learning of utility function with a new technique for collaborative filtering to learn the preference of the user from history rating data, and then estimate similarity among users based on a confidence measure. The technique is light yet efficient, and is based on a threshold for the number of co-rated items. It enforces a positive correlation threshold between any two users. Our Confidence-based collaborative filtering technique is able to identify recommendations that are optimal or near-optimal in terms of the learned utility. In addition, preference learning has been leveraged with the existence of historical multi criteria rating data submitted by the same user on similar products or services. Instead of initially starting with the standard utility axes in the utility axes elicitation step, preference learning is added resulting in an adaptive utility learning. We used a regression analysis technique to predict the user preference when there is a consistent family of criteria $m_1, \cdots, m_n$, each criterion is represented by a rating given by the user as a real value m_i in the range of $[m_i*, m_i^*]$ where m_i* and m_i^* are the worst and the best level of the i-th criterion respectively.

The learning process and utility function refinement continues as feedback is extracted from the user. Methods proposed are based on a low-dimensional utility space (as opposed

to the high-dimensional recommendation space). Because we are working in composite products and services space, the number of attributes is very large due to multitude of composite services, consequently, the "feature/diversity space" is too high-dimensional to learn the recommendation utility accurately and needs exponentially large learning set and time for accurate learning. The method is based on iteratively refining a set of axes in the n-dimensional utility space, starting from the utility space standard axes. At every step, the user is asked to rank a set of recommendations, each being optimal for one of the current axes. Based on the user feedback, the method refines the set of axes that become closer to each other. Consequently, the utility function is constructed. Works of Nielsen et al, Suryadi et al, Russell et al, and Chajewska et al [79,105,108,110] elicit the utility function from a database of observed behavioral patterns, while Chajewska, [111] focuses on eliciting the utility function from a database of already elicited utility functions. All of these approaches are targeted to produce a utility function from a database. While few recommender systems provide for estimating and refining the preferences of the user [68], works such as Pazzani [81] have exemplified the need for such techniques. However, none of these works, to the best of my knowledge, work on recommendations for composite product and services, which makes the recommendation space very large.

We have defined methods for diversifying a recommendation set using the separation of utility space from recommendation space. Working with campsite space has the challenge of selecting from larger space, but also gives an opportunity to diversify using higher-dimension diversity space. Giving recommendations by the utility learned may not provide sufficient diversity resulting in a recommendation set where members are often similar to

each other.

In addition, I have developed a research prototype for the different components of DG-RCA to model highly-complex service compositions for the movies domain and the travel domain, which includes accommodations, rental vehicles, and air transportation. Some prototype travel domain systems exist (e.g., ATA , ITR [91],SPIRE [51]); however, all of these prototypes recommend atomic travel services, except for ITR, which works with composite products, but the composition is performed manually by the user. The travel tool proposed will provide alternative solutions of travel packages for users and decision makers based on the different models and methods defined.

Furthermore, I conducted extensive experimental studies to prove the efficacy of DG-RCA approach by developing specific performance measurements of algorithms proposed, and compared with state-of-the art available algorithms (e.g., [17,73,101]). Most if not all experimental studies conducted shows that the proposed framework significantly outperforms in many aspects of quality and scalability. Our validation datasets included publicly available data sets with user ratings such as movielens.org, as well as data extracted from popular commercial systems. I have also completed a case study that required interviewing 30.

Finally, the framework presented a hybrid recommender system that is a cascaded collaborative recommender and knowledge-based hybrid. Working with composite products and services mandates that algorithms and techniques used to be efficient and can scale well due to the large size of candidate recommendation space. According to a popular hybrid

	Weighted	Mixed	Switching	Feature Combination	Cascade
CF/CN	P-Tango	PTV, ProfBuilder	DailyLearner	(Basu, Hirsh & Cohen 1998)	Fab
CF/DM	(Pazzani 1999)				
CF/KB	(Towle & Quinn 2000)		(Towle & Quinn 2000		DG-RCA
CN/CF					
CN/DM	(Pazzani 1999)			(Condliff, et al. 1999)	
CN/KB					
DM/CF					
DM/CN					
DM/KB					
KB/CF					EntreeC
KB/CN					
KB/DM					

(CF = collaborative, CN = content-based, DM = demographic, KB = knowledge-based)

Redundant

Not possible

Figure 8.1: Possible and actual hybrid recommender systems

recommender survey [22], and to the best of my knowledge, DG-RCA is the first recommender system that utilizes hybridization techniques based on cascaded collaborative filtering and knowledge-based. As presented in Figure 8.1, DG-RCA is the only recommender system that learns the utility of the user with a collaborative filtering technique then cascade the result set to be refined with a knowledge-based technique that is based on using the diversity space to employ our diversification technique and finally present the user with a diverse set of composite products and services.

8.2 Future Research Directions

Many questions remain for future research. DG-RCA has several components and each component could be looked at from different aspect. In this section, I point out some focus areas to explore further. They mainly include issues related to:

Expand diversity step to enhance learning of utility function: In current research, the preference learning ended when the utility elicitation step ended. Specifically, once we proceed to the diversity step, the utility function elicitation does not incorporate users feedback to improve the learned utility function. Consequently, not use what has been learned from the diversity step with regard to feedback extracted. One of the future research areas is how to incorporate users feedback from diversity step to reconstruct the utility function. The goal is to make the utility function elicitation a continues process, where learning and refining the axes is not limited to the utility axes selection step but also expanded to leverage the users feedback from the recommendation diversity phase.

Examine utility dimensions represent what matters: In my research, it is assumed that utility and diversity dimensions represent the true dimensional preference of the user based on domain expertise knowledge. However, this assumption may not hold accurate all the time, one way to resolve this issue is to examine and refine utility dimensions themselves and not only their weights. Below is a flow chart in Figure 8.2 that represents the major steps and decision points for providing the user with a diverse set of recommendation(s) based on learning from feedback extracted. The focus of the flowchart is more toward dimensions and whether they represent what all what the user base their judgment

on or not, and if those dimensions are measurable or not.

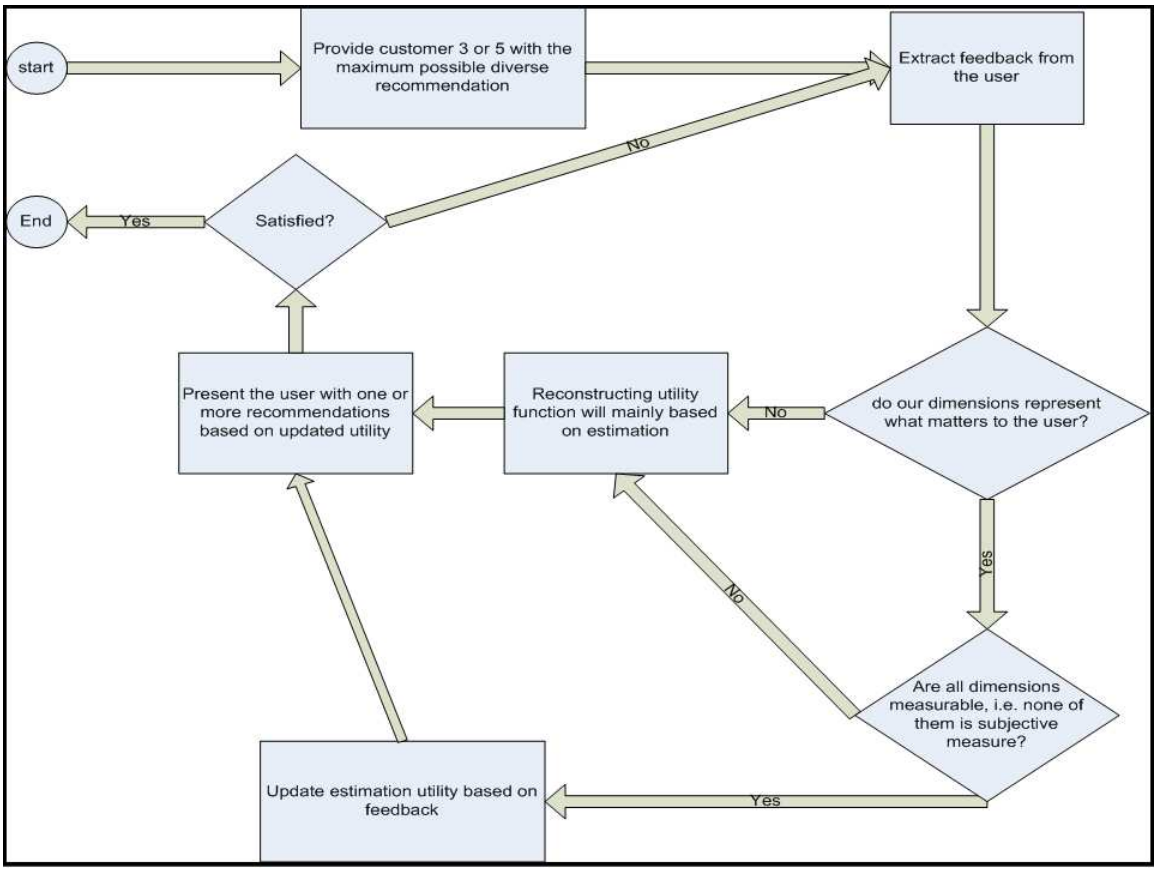

Figure 8.2: Flowchart for initial learning of dimensions

Group decision making: This is a popular research area [44,70,114], where the input of several participants is combined to produce or recommend one product or service that captures the preference of most and hopefully all participants. Juan et. al [57] introduced a novel method of making recommendations to groups based on existing techniques of collaborative filtering and taking into account the group personality composition. The scope of my research was mainly focused on single user interaction, and there is a value in expanding it to address and incorporate inputs from more than one user. Conflict among users

is one of the issues to consider when dealing with group decision making. One proposed way is to extract a representative sample of the dataset and then apply a regression analysis technique to determine the solution with least contradictions among users' rating or ranking. This solution can be later mapped to a utility function that can be applied against the complete of solutions and measure its' success. This process can be iterative until we identify the best utility function with respect to precision.

Interaction with DGQL: A key modeling technology for DG-RCA is Decision Guidance Query Language (DGQL). Significant effort and time would be required to modify, extend, and build a new recommender system with composite alternatives from scratch. Therefore, It is desirable to extend the DGQL with views for recommending composite alternatives in terms of syntax and formal semantics, and develop efficient algorithms to evaluate DGQL queries. Traditional Database Management Systems have shown efficient handling of data with solid integrity, but lack the ranking capability needed for answering Information Retrieval (IR) queries. IR provides mechanisms for effective fuzzy ranking. Rank-aware query processing has emerged to support top-k queries in DB (e.g., [20,21,48]). Efficient rank aggregation and combining scoring functions have been proposed in [36]. In addition, [48] has suggested an approach for top-k results based on joining multiple inputs. However, these inputs are atomic products and services. Our focus is producing top-k results of composite products that are composed of another composite or atomic products or services.

Clustering technique to determine the appropriate domain: One of the areas to consider and explore more is the clustering step. Currently, We start with the clustering

step to determine what cluster the user is interested in. Our recommendation space is split into a number of clusters, where each cluster contains a number of packages (recommendations). Examples of clusters are: honeymooners, single, family, etc. Basically, clusters are extracted from historical purchase data cross-referenced with user demographic data. However, this is beyond the scope of our research at this point. Currently, we just assume that these clusters exist and packages can therefore be targeted to specific user groups. We then return a package to represent the corresponding cluster. Each recommendation returned is the highest total utility function in its cluster. Initially, we will give equal weight to metric attributes as it is too early to conclude what the user might value more with respect to metrics attributes (e.g. Saving, Enjoyment). However, domain knowledge could also be used to determine how to assign weights and selections of metrics attributes, consequently calculating the global utility function. The clustering mechanism could be one of the following: 1) As simple as distinct selections from recommendation space on package type. 2) Another way is to construct a linear function by combining features from the underlying atomic services such as rental car, airline flight type, or number of stars of the hotel. 3) Using supervised learning and classification techniques such as Support vector machines (SVM) [117].

Appendix A: Research Publications

This research has resulted in the following publications:

- Alodhaibi, Khalid. Brodsky, Alexander. Mihaila, George A.: COD: Iterative Utility Elicitation for Diversified Composite Recommendations. HICSS, 2010 43rd Hawaii International Conference on System Sciences, 2010: p.1-10.

- Alodhaibi, Khalid. Brodsky, Alexander. Mihaila, George A.: A Randomized Algorithm for Maximizing the Diversity of Recommendations. HICSS, 2011 44th Hawaii International Conference on System Sciences, 2011.

- Alodhaibi, Khalid. Brodsky, Alexander. Mihaila, George A.: An Adaptive Utility Learning Method for Composite Recommendations. ICMIA November 30 - December 2, 2010, Seoul, Korea.

- Alodhaibi, Khalid. Brodsky, Alexander. Mihaila, George A.: A Confidence-Based Recommender with Adaptive Diversity. IEEE SYMPOSIUM SERIES ON COMPUTATIONAL INTELLIGENCE, APRIL 11-15, 2011, PARIS, FRANCE: P.36-43.

Appendix B: Human Subjects Review Board (HSRB)

Office of Research Subject Protections
4400 University Drive, MSN 4C6, Fairfax, Virginia 22030
Phone: 703-993-4121; Fax: 703-993-9590

Human Subjects Review Board (HSRB)

New Submission Checklist

To avoid delay in the processing of HSRB applications, please ensure that the following are included in your application. Applications cannot be reviewed until all of the following checklist items are submitted.

YES	NO	N/A	ITEM
☒	☐	☐	Application with **ALL** sections completed (including check boxes on first page)
☒	☐	☐	Application signed by Principal Investigator
☒	☐	☐	CITI Training completed by all researchers including research assistants
☒	☐	☐	**Proposed Consent Form** (See Template Consent and Consent Guidelines)– All instructional language removed, written at the appropriate reading level for participants
☐	☐	☒	**Proposed Assent Form** (If minors are involved) – Written at the appropriate reading level for the age group (Contact ORSP for a sample of a 6th grade Assent Form)
☒	☐	☐	**Instrumentation** – All surveys, questionnaires, standardized assessment tools, interview questions, focus group questions/prompts or other instruments of data collection
☐	☐	☒	**Recruitment Materials** – Letters to potential participants, advertisements, flyers, listserve postings, emails, brochures, SONA postings, telephone scripts, presentation scripts, etc.
☐	☐	☒	**Grant Applications** – If the research is funded, include the grant application as submitted to the funding agency (Please note that the HSRB application title must match the grant application title.)
☐	☐	☒	**Debriefing Form** – If the study proposes to use deception or incomplete information to participants
☐	☐	☒	**Cultural Contact Information** – If the study is being conducted outside the US, the HSRB must inquire about the conduct of research in that country. Submit the name and contact information of an individual who can provide that information.

Applications can be reviewed without the following items, but if they are applicable to the study, they must be submitted before approval can be given.

☐	☐	☐	Research in Mason Classrooms – Submit permission from the instructors when course credit is given
☐	☐	☐	Research in School Systems – Submit approval letter from the school district Human Subjects Review Board
☐	☐	☐	Research in Universities – Submit approval letter from the University Human Subjects Review Board
☐	☐	☐	Research in Hospitals – Submit approval letter and approved consent document from the hospital Human Subjects Review Board
☐	☐	☐	Research in Institutions/Organizations without Human Subject Review Boards – Submit permission letter from the institution/organization
☐	☐	☐	If George Mason is the primary recipient of funding, submit Human Subjects Review Board approval from subcontractors conducting human subjects research
☐	☐	☐	Psychology Department – Sign off by the Chair of the Department
☐	☐	☐	School of Management (SOM) – Submit SOM routing form with all approval signatures
☐	☐	☐	Other Mason Committee Oversight– If your study involves the use of blood or other human biological specimens, submit **Institutional Biosafety Committee** approval. If your study involves sources of ionizing radiation or Xray producing devices, submit **Radiation Safety Committee** approval.

George Mason University

Human Subjects Review Board
Application for Human Subjects Research Review

For ORSP Use Only	GMU
Protocol No. _____	Proposal No. _____
Classified: ☐ Exempt ☐ Non Exempt ☐ Expedited	
Signature _____	Date _____

Federal Regulations and George Mason University policy require that all research involving humans as subjects be reviewed and approved by the University Human Subjects Review Board (HSRB). Any person, (GMU faculty member, staff member, student, or other person) wanting to engage in human subject research at or through George Mason University must receive written approval from the HSRB before conducting research. Approval of this project by the HSRB only signifies that the procedures adequately protect the rights and welfare of the subjects and should not be taken to indicate University approval to conduct the research.

Please complete this cover page AND provide the Protocol information requested on the back of this form. Forward this form and all supporting documents to the Office of Research Subject Protections, MS 4C6. If you have any questions please feel free to contact ORSP at 703-993-4121.

Project Title:	**COD: Iterative Utility Elicitation for Diversified Composite Recommendations**	
	Principal Investigator (Must be Faculty)	**Co-Investigator / Student Researcher***
Name	Dr. Alexander Brodsky	Khalid Alodhaibi
Department	Department of Computer Science	Department of Computer Science
Mail Stop	4400 University Drive, MS 4A4 Fairfax, Virginia 22030-4444	11446 Abner Ave Fairfax, VA 22030
Phone	703-993-1529	703-865-5789
Email	brodsky@gmu.edu	kalodhai@gmu.edu

*Student researchers should provide a mailing address rather than campus address. Additional researchers should be listed on a separate page.

Type of Project:
- ☐ Faculty/Staff Research
- ☒ Doctoral Dissertation
- ☐ Masters Thesis
- ☐ Student Project (Specify Grad or Undergrad):
- ☐ Other (Specify):

VULNERABLE POPULATION:	**PERSON IDENTIFIABLE DATA:**	**RESEARCH DESIGN:**
☐ Fetuses/Abortuses/Embryos	☐ Audio taping	☐ Questions on harm to self or others
☐ Pregnant women	☐ Video taping	☐ Questions on illegal behavior
☐ Prisoners	☒ Data collected via email	☐ Deception
☐ Minors	☐ Data collected via Internet	☐ Human/computer interaction
☐ Mentally disabled	☐ Confidential electronic records	☐ Collection/analysis of secondary data
☐ Emotionally disabled	☐ Coded data linked to individuals	Funding: ☐ Yes ☒ No
☐ Physically disabled	☐ Human biological materials	Source:
☐ Undergrad student pool (Psych/SOM)	Biosafety Project #:	OSP Proposal #:
☐ Other:	(If yes, please attach a copy of the grant application)	

I certify that the information provided for this project is correct and that no other procedures will be used in this protocol. I agree to conduct this research as described in the attached supporting documents. I will request and receive approval from the HSRB for changes prior to implementing these changes. I will comply with the HSRB policy for the conduct of ethical research. I will be responsible for ensuring that the work of my co-investigator(s)/student researcher(s) complies with this protocol.

Brodsky / Prof. Alex Brodsky / 10/6/2010

Principal Investigator Signature	Date

ABSTRACT

1. Describe the aims and specific purposes of the research project and the proposed involvement of human participants.

   A methodology is proposed to capture the preference of the user when recommending products or services. In order to evaluate our proposed recommender system, we will conduct a user study of 30 users. The objective of the study was to verify the following hypotheses:
   1. Our system achieves a better recall and precision than a non-personalized travel recommender system.
   2. The interactive elicitation of the utility axis imposes an acceptable overhead to the users.
   3. The vertical diversity layering step increases the recall.

2. Describe the characteristics of the intended sample (number of participants, age, sex, ethnic background, health status, etc.).

   Participants are not expected to have any unique characteristics as long as they are employed to prevent the impact of not having an income which might influence their responses when it comes to recommendation prices. The purpose of the survey is to verify a method that balances multiple criteria, and eventually derives a preference level to measure how successful the module in capturing preferences and then suggest recommendations to the participant. Simple few steps process until the module gives recommendations. Since one of the criteria used is price, it is desirable to have someone that has income regardless of how much it is. Otherwise anyone can be used for the survey. As an IBM employee, I am looking for engaging IBM co-workers as participants. The plan is to have about 30 participants. There are enough participants that I can find within IBM that meet requirements. For now, I am only inviting co-workers.

3. Identify the criteria for inclusion or exclusion. Explain the rationale for the involvement of special classes of participants (children, prisoners, pregnant women, or any other vulnerable population).

   In IBM, we have online Instant Messenger tool called SameTime All IBM employees (300k) have accounts in this tool, and when they login to their computers, it shows them as available if they wish to chat. Any IBM employee can contact any other IBM employee using this tool regardless if they know them or not because within IBM community, we know this tool can only be used by IBM staff.
   The identification of participants is quite simple; any employee that shows available using IBM IM tool is "contactable" for the purpose of my survey. I will keep looking for participants until I reach the number I need (30 participants).

4. Describe your relationship to the participants if any.

   Participants are co-workers.

PROTOCOL – Involving Human Participation

1. If there are direct benefits to the participants, describe the direct benefits and also describe the general knowledge that the study is likely to yield. If there are no direct benefits to the participants, state that there are no direct benefits to the participants and describe the general knowledge that the study is likely to yield.

   There are no direct benefits, it is a quick and easy mathod to balance two different criteria to infer the user's total utility function, participants get to see how the method works and how in fact it could give them better recommendations (travel packages) when compared with exisiting methodologies in recommender system domain.

2. Describe how participants will be identified and recruited. Note that all recruitment materials (including ads, flyers, letters to participants, emails, telephone/presentation scripts, SONA postings) for participants must be submitted for review for both exempt and non-exempt projects.

   As stated, being IBM employee, we have Instant messenger tool that i can contact candidates instantly online. If participants have the time and desire to participate, they will communicate their ability instantly. Once I greet them, and they respond to my greetings, i use this text to invite participants: "Hello, my name is Khalid Alodhaibi, i am a PhD student at George Mason University in Computer Science field. I would like to take few minutes of your time, it is quick simple questions, related to my school project, it will take 10 minutes at most". Participants can agree or not to participate. I will keep looking for participants until I reach the number I need (30 participants).

3. Describe your procedures for obtaining informed consent. Who will obtain consent and how will it be obtained. Describe how the researchers will ensure that subjects receive a copy of the consent document.

   Once I contact participants in the IM tool, and they respond to my greetings, I will post to them:

   "Hello, my name is Khalid Alodhaibi, i am a PhD student at George Mason University in Computer Science field. I would like to take few minutes of your time, it is quick simple questions, related to my school project, and it will take 10 minutes at most"

   If they say Yes, and agree to participate, then i will send them email with the content of the consent form (attached) and ask them to reply back using their email account with "yes, I accept" or "No, I don't". Participation is completely voluntary and once the consent forms are read and approved, the study starts by either sending it in email or completing the study instantly using the Instant Messenger tool; however, those who did not wish to be part of the study will not have their results used for the purpose of the research. Participants have the right to withdraw at anytime without the need to give reasons; their results will not be used in the research.
   Once i get their acceptance reply in email, i start the survey process (by pasting the script attached) , using IM i start presenting the participant with questions A to D attached.

4. State whether subjects will be compensated for their participation, describe the form of compensation and the procedures for distribution, and explain why compensation is necessary. State whether the subjects will receive course credit for participating in the research. **If yes**, describe the non-research option for course credit for the students who decide not to participate in the research. The non-research option for course credit must not be more difficult than participation in the research. Information regarding compensation or course credit should be outlined in the Participation section of the consent document.

   Compensation: Compensation will not be provided. No participant will receive course credit for participation in this project.

5. If minors are involved, their active assent to the research activity is required as well as active consent from their parents/guardians. This includes minors from the Psychology Department Undergraduate Subject Pool. Your procedures should be appropriate to the age of the child and his/her level of maturity and judgment. Describe your procedures for obtaining active assent from minors and active consent from parents/guardians. *Refer to the Guidelines for Informed Consent for additional requirements if minors from the Psychology Subject Pool are involved.*

   No minors will be involved.

6. Describe the research design and methods. What will be done to participants during the study? Describe all tests and procedures that will be performed. Include an estimate of the time required to complete the tests and procedures.

   In order to measure precision and recall of our propsed method, we ask each participant to rate 5 recommendations resulting from our propsed method and also other 5 recommendations resulting from a standard method that does not incorporate any intelligance. For our system, in the first phase we learn the users utility function through a two step dialog: at each step, we present the user with two choices, one with a better enjoyment (number of stars), and the other with a smaller cost. Depending on the user's answers, their inferred utility function can reflect either: a strong sensitivity to price (PP), a moderate preference for less expensive packages (PQ), a moderate preference to higher quality packages (QP), or a strong bias towards high quality packages (QQ). In the second phase, we compute five recommendations using our method and present them to the user in descending order of their utility (according to the personalized utility function estimated in the first phase). In addition, we present the user the top five recommendations in the order suggested by the standard method. We then asked the users to rate each of the ten recommendations on a scale of 1 to 5, where 5 means "definitely buy", 4 means "likely to buy", 3 means "neutral", 2 means "unlikely to buy" and 1 means "definitely not buy". We point out that users do not know which recommendation set come from which system. using IBM IM, participants will be presented with questions A to D (attached) and asked to give answers using IM tool. The time should no more than 10 minutes for the survey.

7. Describe how confidentiality will be maintained. If data will be collected electronically (e.g. by email or an internet web site), describe your procedures for limiting identifiers. Note that confidentiality may have to be limited if participants are asked questions on violence toward self or others or illegal behavior. Contact the Office of Research Subject Protections for assistance.

   IBM has a secure email system, there is no need to and will not be identifying indivisuals, simply their identidy is irrelevant therfore will not be collected nor needed.

8. Describe in detail any potential physical, psychological, social, or legal risks to participants, why they are reasonable in relation to the anticipated benefits and what will be done to minimize the risks. Where appropriate, discuss provisions for ensuring medical or professional intervention in case participants experience adverse effects. Where appropriate, discuss provisions for monitoring data collection when participants' safety is at risk.

   Potential Risks: There are no invasive procedures involved and no potential physical, social, or legal risks to the participants that would result from being involved in the study

9. If participants will be audio-or video-taped, discuss provisions for the security and final disposition of the tapes. *Refer to Guidelines for Informed Consent.*

   There will not be audo-taping

10. If participants will be misinformed and/or uninformed about the true nature of the project, provide justification. Note that projects involving deception must not exceed minimal risk, cannot violate the rights and welfare of participants, must require the deception to accomplish the aims of the project, and must include a full debriefing. *Refer to Guidelines for Informed Consent.*

    Participants will not be misinformed and/or uninformed about the true nature of the project

11. Submit a copy of each data collection instrument/tool (including questionnaires, surveys, standardized assessment tools, etc.) you will use and provide a brief description of its characteristics and development. Submit scripts if information and/or questions are conveyed verbally.

attached

12. **INFORMED CONSENT:** Attach appropriate Proposed Informed Consent document(s). **See Guidelines for Informed Consent and the Template Informed Consent Document for additional information.**

13. **APPROVAL FROM COOPERATING INSTITUTION/ORGANIZATION**: If a cooperating institution/organization provides access to its patients/students/clients/ employees/etc. for participant recruitment or provides access to their records, Attach written evidence of the institution/organization human subjects approval of the project.

PROTOCOL - Involving Existing Records

For the study of existing data sets, documents, pathological specimens, or diagnostic specimens.

1. Describe your data set.

   Travel vacation packages are collected from travelocity.com, for each package we only keep price and enjoyment.

2. Provide written permission from the owner of the data giving you access for research purposes at George Mason University if the data set is not publicly available.

   Data set is publicly available at www.travelocity.com

3. Describe how you will maintain confidentiality if the data set contains person identifiable data.

   data set does not contain any person identifiable data

4. Describe what variables you are extracting from the data set.

   No existing data sets will be used

Script for recruiting participants

Hello, my name is Khalid Alodhaibi, I am a PhD student at George Mason University in Computer Science field. I would like to take few minutes of your time, it is quick simple set of questions, related to my school project, and it will take 10 minutes at most

Script for the survey to use with participants

This is about a recommender system for travel package, assume you planning on vacation for 3 weeks in LA, and i am only showing you the cost and the enjoyment of the package,

Question A:

From these, which one would you choose ?

1-Cost $4173, Enjoyment: 4.5 stars
2-Cost $1472, Enjoyment: 3 stars

1 or 2?

Question B:

If 1 then choose between:

1.1- Cost $7877, Enjoyment: 5 stars
1.2- Cost $2802, Enjoyment: 4 stars

1.1 or 1.2 ?

If 2 then choose between:

2.1- Cost $1489, Enjoyment:3 stars
2.2- Cost $1167, Enjoyment: 2.5 stars

2.1 or 2.2 ?

Question C:

Based on answers above, how would you rate each one of the following?, Give a rating for each of the recommendations, you can give the same rating to more than one recommendation:

5 definitely buy
4 likely to buy
3 neutral

2 unlikely to buy
1 definitely not buy

The first 5 are coming from system A, the second set is coming from system B, based on your answer above, you will select the corresponding set B from below.

System A

A 4.5 stars	$4,173
B. 4 stars	$5,962
C. 3 stars	$2,146
D. 4 stars	$4,718
E. 4 stars	$6,575

System B

If Choice is 2.1:

A. 3 stars	$1,472
B. 2.5 stars	$1,167
C. 4.5 stars	$4,173
D. 3.5 stars	$4,159
E. 5 stars	$7,877

If Choice is 2.2:

A. 2.5 stars	$1,167
B. 3 stars	$1,472
C. 4 stars	$2,802
D. 4.5 stars	$4,173
E. 5 stars	$7,877

If Choice is 1.2:

A. 4.5 stars	$4,173
B. 5 stars	$7,877
C. 2.5 stars	$1,167
D. 3 stars	$3,893

If Choice is 1.1:

A. 5 stars	$7,877
B. 4.5 stars	$4,173
C. 3 stars	$1,472
D. 2.5 stars	$1,167

Now that users took the survey, and my system navigated through their preferences, and finally presented them results, we want to check if they think the amount of time they spent interacting with the system to give them more personalized results is worth the better (or worse) result they got, I ask them this question:

Question D:

"Would you be willing to spend a few more minutes answering few questions so that the system can learn about you and, consequently, provide you more personalized recommendations, considering the amount of the transaction?" (Y/N)

INFORMED CONSENT FORM

Project title: **COD: Iterative Utility Elicitation for Diversified Composite Recommendations**

RESEARCH PROCEDURES
This study will try to evaluate and combine the flexibility of diversity ranking functionality with the information processing capabilities to learn and capture the preferences of the user through an iterative learning process, where we propose and inform the user of what is available in terms of composite recommendations, with minimum interaction from the user side.

If you agree to participate, you will be asked to complete a short survey (about 8-10 minutes), that will not require any personal identification information.

RISKS and BENEFITS
There are no foreseeable risks for participating in this research and no deception will be employed. There are also no benefits to you as a participant.

CONFIDENTIALITY
The data in this study will be confidential. Confidentiality will be maintained by creating a pseudonym for each participant and through the codification of data. Once the results have been transcribed and checked, they will be deleted from all records with only the unidentifiable results remaining. There will be no identifying markings on the surveys. While it is understood that no computer transmission can be perfectly secure, reasonable efforts will be made to protect the confidentiality of your transmission.

PARTICIPATION
Your participation is voluntary, and you may withdraw from the survey at any time and for any reason. If you decide not to participate or if you withdraw from the survey, there is no penalty or loss of benefits to which you are otherwise entitled. There are no costs to you or any other party.

CONTACT
This research is being conducted by Khalid Alodhaibi, a PhD student at The Volgenau School of Information Technology and Engineering at George Mason University, he may be reached at kalodhai@gmu.edu or (703)865-5789 for questions or to report a research-related problem. This study is being supervised by Dr. Brodsky, a professor at George Mason University, who may be reached at brodsky@gmu.edu or 703-993-1529.

This research has been reviewed according to George Mason University procedures governing your participation in this research. You may contact the George Mason University Office of Research Subject Protections at 703-993-4121 if you have questions or comments regarding your rights as a participant in the research.

CONSENT

The George Mason University Human Subjects Review Board has waived the requirement for a signature on this consent form. However, if you wish to sign a consent, please contact Khalid Alodhaibi
I have read this form and agree to participate in this study.

Bibliography

Bibliography

[1] Adomavicius, G. and A. Tuzhilin, Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. Knowledge and Data Engineering, IEEE Transactions on, 2005. 17(6): p. 734-749.

[2] A.I. Schein, A. Popescul, L.H. Ungar, and D.M. Pennock, Methods and Metrics for Cold-Start Recommendations, Proc. 25th Ann. Intl ACM SIGIR Conf., 2002.

[3] Ajantha Dahanayake. Personalized Information Retrieval and Access: Concepts, Methods, and Practices. IGI Publishing. 2008. Books24x7. $< http : //common.books24x7.com/book/id_1 8640/book.aspx >$

[4] Alan Said. 2010. Identifying and utilizing contextual data in hybrid recommender systems. In Proceedings of the fourth ACM conference on Recommender systems (RecSys '10). ACM, New York, NY, USA, 365-368

[5] Alexander Brodsky, Sylvia Morgan Henshaw, Jon Whittle, CARD: a decision-guidance framework and application for recommending composite alternatives. RecSys 2008: 171-178.

[6] Alodhaibi, Khalid. Brodsky, Alexander. Mihaila, George A.: COD: Iterative Utility Elicitation for Diversified Composite Recommendations. HICSS, 2010 43rd Hawaii International Conference on System Sciences, 2010: p.1-10.

[7] Alodhaibi, Khalid. Brodsky, Alexander. Mihaila, George A.: A Randomized Algorithm for Maximizing the Diversity of Recommendations. HICSS, 2011 44th Hawaii International Conference on System Sciences, 2011.

[8] Alodhaibi, Khalid. Brodsky, Alexander. Mihaila, George A.: An Adaptive Utility Learning Method for Composite Recommendations. ICMIA November 30 - December 2, 2010, Seoul, Korea (to appear).

[9] Alodhaibi, Khalid. Brodsky, Alexander. Mihaila, George A.: A Confidence-Based Recommender with Adaptive Diversity. IEEE SYMPOSIUM SERIES ON COMPUTATIONAL INTELLIGENCE, APRIL 11-15, 2011, PARIS, FRANCE: P.36-43

[10] Armstrong, James E.. Introduction to Systems Engineering. John Wiley and Sons. 2000. Books24x7. $< http : //common.books24x7.com/book/id_8618/book.aspx >$

[11] Andriole, Stephen J.. Best Practices in Business Technology Management. Auerbach Publications. 2009. Books24x7. $< http : //common.books24x7.com/book/id_26395/book.aspx >$

[12] Angel A. Juan, Daniel Riera, David Masip, Josep Jorba, Javier Faulin: A Simulation-based Methodology to Assist Decisionmakers in Real Vehicle Routing Problems. ICEIS (2) 2009: p.212-217.

[13] Asela Gunawardana and Christopher Meek. 2009. A unified approach to building hybrid recommender systems. In Proceedings of the third ACM conference on Recommender systems (RecSys '09). ACM, New York, NY, USA, 117-124

[14] Badrul Sarwar, George Karypis, Joseph Konstan, and John Reidl. 2001. Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th international conference on World Wide Web (WWW '01). ACM, New York, NY, USA, 285-295.

[15] Bart P. Knijnenburg and Martijn C. Willemsen. 2009. Understanding the effect of adaptive preference elicitation methods on user satisfaction of a recommender system. In Proceedings of the third ACM conference on Recommender systems (RecSys '09). ACM, New York, NY, USA, 381-384.

[16] Basu, C., Hirsh, H. and Cohen W.: 1998, Recommendation as Classification: Using Social and Content-Based Information in Recommendation. In: Proceedings of the 15th National Conference on Artificial Intelligence, Madison, WI, pp. 714-720.

[17] Bradley, K. and B. Smyth, Improving Recommendation Diversity. Proceedings of the Twelfth Irish Conference on Artificial Intelligence and Cognitive Science, Maynooth, Ireland, 2001: p. 85-94.

[18] Brodsky, A. and X.S. Wang, Decision-Guidance Management Systems (DGMS): Seamless Integration of Data Acquisition, Learning, Prediction and Optimization. Hawaii International Conference on System Sciences, Proceedings of the 41st Annual, 2008: p. 71-71.

[19] Brook Wu (eds), Yi-fang. Handbook of Research on Text and Web Mining Technologies. IGI Global. 2009. Books24x7. $< http : //common.books24x7.com/book/id_33360/book.aspx >$

[20] BRUNO,N., CHAUDHURI, S., ANDGRAVANO, L. 2002. Top-k selection queries over relational databases: Mapping strategies and performance evaluation. ACM Trans. Datab. Sys. 27, 2, 369-380.

[21] BRUNO, N., GRAVANO, L., AND MARIAN, A. 2002. Evaluating top-k queries over web-accessible databases. In Proceedings of the 18th International Conference on Data Engineering. 153-187.

[22] Burke, R., Hybrid Recommender Systems: Survey and Experiments. User Modeling and User-Adapted Interaction, 2002. 12(4): p. 331-370.

[23] Burke, R., Hammond, K., and Young, B.: 1997, The FindMe Approach to Assisted Browsing. IEEE Expert, 12 (4), 32-40.

[24] Burke, R.: 2000, Knowledge-based Recommender Systems. In: A. Kent (ed.): Encyclopedia of Library and Information Systems. Vol. 69, Supplement 32.

[25] C. D. Manning, P. Raghavan, and H. Schutze. Introduction to Information Retrieval. Cambridge University Press, 2008.

[26] C. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen: Improving recommendation lists through topic diversification. In Proceedings of the 14th International Conference on WWW 2005. p.22-32.

[27] Carmel Domshlak, Thorsten Joachims, Efficient and non-parametric reasoning over user preferences Received: 31 October 2005 / Accepted in revised form: 4 September 2006 / Published online: 26 January 2007 Springer Science+Business Media B.V. 2007.

[28] Chavas, Jean-Paul. Risk Analysis in Theory and Practice. Academic Press. 2004. Books24x7. $< http://common.books24x7.com/book/id_28065/book.aspx >$

[29] Claude Ghaoui. Encyclopedia of Human Computer Interaction. IGI Global. 2006. Books24x7. $< http://common.books24x7.com/book/id_14703/book.aspx >$

[30] Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D. and Sartin, M.: 1999, Combining Content-Based and Collaborative Filters in an Online Newspaper. SIGIR 99 Workshop on Recommender Systems: Algorithms and Evaluation. Berkeley, CA. $< URL : http : //www.cs.umbc.edu/\ ian/sigir99 - rec/papers/claypool_m.ps.gz >$

[31] D. Abraham. Mart, Rafael: Tabu search and GRASP for the maximum diversity problem. European Journal of Operational Research 178(1) p. 71-84 (2007).

[32] D. Fleder and K. Hosanagar. Recommender systems and their impact on sales diversity. In Proceedings of the 8th ACM conference on EC p. 192-199, 2007.

[33] D. McSherry.: Diversity-conscious retrieval. In Proceedings of the 3rd European Conference on Case-Based Reasoning. p. 219-233, 2002.

[34] D.W. Oard and J. Kim, Implicit Feedback for Recommender Systems, Proc. Recommender Systems. Papers from 1998 Workshop, Technical Report WS-98-08, 1998.

[35] Davies, John. Information Retrieval: Searching in the 21st Century. John Wiley and Sons (UK). 2009. Books24x7. $< http : //common.books24x7.com/book/id_3 3746/book.aspx >$

[36] DWORK, C., KUMAR, S. R., NAOR, M., AND SIVAKUMAR, D. 2001. Rank aggregation methods for the web. In Proceedings of the 10th International World Wide Web Conference. 613-622.

[37] Felix Hernandez del Olmo, Elena Gaudioso, Evaluation of recommender systems: A new approach, Expert Systems with Applications, Volume 35, Issue 3, October 2008, Pages 790-804.

[38] Geiza C. Silva, Luiz Satoru Ochi, Simone L. Martins: Experimental Comparison of Greedy Randomized Adaptive Search. Procedures for the Maximum Diversity Problem. WEA 2004 p.498-512.

[39] Geiza C. Silva, Marcos R. Q. de Andrade, Luiz Satoru Ochi, Simone L. Martins, Alexandre Plastino: New heuristics for the maximum diversity problem. J. Heuristics 13(4) p. 315-336 (2007).

[40] Goh, Dion, and Schubert Foo (eds). Chapter XVI - Hybrid Recommendation Systems: A Case Study on the Movies Domain. Social Information Retrieval Systems: Emerging Technologies and Applications for Searching the Web Effectively. $< http : //common.books24x7.com/book/id_2 3246/book.asp >$

[41] Graen (ed), George B.. Dealing with Diversity. Information Age Publishing. 2003. Books24x7. $< http : //common.books24x7.com/book/id_8 834/book.aspx >$

[42] Gupta, Jatinder N.D.. Decision Making Support Systems: Achievements, Trends, and Challenges for the New Decade. IGI Publishing. 2003. Books24x7. $< http : //common.books24x7.com/book/id_5 300/book.aspx >$

[43] Halpern, Joseph Y.. Reasoning About Uncertainty. The MIT Press. 2003. Books24x7. $< http : //common.books24x7.com/book/id_7 010/book.aspx >$

[44] Hinsz, V.B., and G.S. Nickell. "Positive Reactions to Working in Groups in a Study of Group and Individual Goal Decision-Making." Group Dynamics 8 (2004): 253-264.

[45] Hofmann, T., 2004. Latent semantic models for collaborative filtering. ACM Transactions on Information Systems, Vol. 22, No. 1, January 2004, Pages 89-115.

[46] Honeycutt, Jerry. Knowledge Management Strategies. Microsoft Press. 2000. Books24x7. $< http : //common.books24x7.com/book/id_1 493/book.aspx >$

[47] Ido Guy, Alejandro Jaimes, Pau Agull, Pat Moore, Palash Nandy, Chahab Nastar, and Henrik Schinzel. 2010. Will recommenders kill search?: recommender systems - an industry perspective. In Proceedings of the fourth ACM conference on Recommender systems (RecSys '10). ACM, New York, NY, USA, 7-12.

[48] Ihab F. Ilyas, Walid G. Aref, Ahmed K. Elmagarmid,Hicham G. Elmongui, Rahul Shah, and Jeffrey Scott Vitter. Adaptive rank-aware query optimization in relational databases, ACM Transactions on Database Systems (TODS),Volume 31 , Issue 4 (December 2006),ISSN:0362-5915.

[49] In Lee. "Chapter 124 - A Linguistic Recommender System for Academic Orientation". Encyclopedia of E-Business Development and Management in the Global Economy. IGI Global. 2010. Books24x7. $< http : //common.books24x7.com/book/id_34727/book.aspx >$

[50] Institution of Mechanical Engineers (IMechE), The. Design Methods for Performance and Sustainability. Professional Engineering Publications. 2001. Books24x7. $< http : //common.books24x7.com/book/id_11047/book.aspx >$

[51] J. J. Daniels and E. L. Rissland. What you saw is what you want: Using cases to seed information retrieval. In ICCBR, pages 325336, 1997.

[52] J. R. Sean M. McNee and J. A. Konstan. Accurate is not always good: How accuracy metrics have hurt recommender systems. In extended abstracts on Human factors in computing systems(CHI06) p. 1097-1101.

[53] Jameson, A., Schafer, R., Simons, J., and Weis, T. 1995. Adaptive provision of evaluation-oriented information: Tasks and techniques. In Proc. IJCAI-95, 1886-1893.

[54] Jimison, H. B., Fagan, L. M., Shachter, R. D., and Shortliffe, E. H. 1992. Patient-specific explanation in models of chronic disease. AI in Medicine 4:191-205.

[55] John Wang. Encyclopedia of Data Warehousing and Mining, Volume II, I-Z. Idea Group Publishing. 2006. Books24x7. $< http : //common.books24x7.com/book/id_15518/book.aspx >$

[56] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. 1999. An algorithmic framework for performing collaborative filtering. In Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '99). ACM, New York, NY, USA, 230-237.

[57] Juan A. Recio-Garcia, Guillermo Jimenez-Diaz, Antonio A. Sanchez-Ruiz, and Belen Diaz-Agudo. 2009. Personality aware recommendations to groups. In Proceedings of the third ACM conference on Recommender systems (RecSys '09). ACM, New York, NY, USA, 325-328.

[58] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, Eigentaste: A Constant Time Collaborative Filtering Algorithm, Information Retrieval J., vol. 4, no. 2, pp. 133-151, July 2001.

[59] K. Nehring and C. Puppe: A theory of diversity. Econometrica, 70(3) p. 1155-1198, May 2002.

[60] Kleanthi Lakiotaki and Nikolaos Matsatsinis. UTA-Rec: A Recommender System based on Multiple Criteria Analysis. ACM RecSys08. Pages 219-226

[61] Konstan, J.A., et al., GroupLens: applying collaborative filtering to Usenet news. Communications of the ACM, 1997. 40(3): p. 77-87.

[62] L.H. Ungar and D.P. Foster, Clustering Methods for Collaborative Filtering, Proc. Recommender Systems, Papers from 1998 Workshop, Technical Report WS-98-08 1998.

[63] Lam, Shyong, Frankowski, Dan, Riedl, John: Do You Trust Your Recommendations? An Exploration of Security and Privacy Issues in Recommender Systems, Book Title: Emerging Trends in Information and Communication Security, Copyright: 2006, Springer Berlin / Heidelberg

[64] Linden, G., S. Hanks, and N. Lesh, Interactive assessment of user preference models: The automated travel assistant. Proceedings of the Sixth International Conference on User Modeling, 1997. 6778.

[65] Liu, N. N., and Yang, Q., 2008. EigenRank: a rankingoriented approach to collaborative filtering. In SIGIR 08, 83-90.

[66] Liu, N. N., Zhao, M., and Yang, Q., 2009. Probabilistic latent preference analysis for collaborative filtering. In CIKM 09, 759-766.

[67] Luiz Pizzato, Tomek Rej, Thomas Chung, Irena Koprinska, and Judy Kay. 2010. RECON: a reciprocal recommender for online dating. In Proceedings of the fourth ACM conference on Recommender systems (RecSys '10). ACM, New York, NY, USA, 207-214.

[68] Manouselis, N. and C. Costopoulou, Analysis and Classification of Multi-Criteria Recommender Systems. World Wide Web, 2007. 10(4): p. 415-441.

[69] Masoud Mohammadian. Intelligent Agents for Data Mining and Information Retrieval. IGI Publishing. 2004. Books24x7. $< http : //common.books24x7.com/book/id_7081/book.aspx >$

[70] Maznevski, M.L. "Understanding Our Differences: Performance in Decision-Making Groups with Diverse Members." Human Relations 47: 531-542.

[71] Mehdi Khosrowpour. Challenges of Information Technology Management in the 21st Century: 2000 Information Resources Management Association International Conference. IGI Publishing. 2000. Books24x7. $< http : //common.books24x7.com/book/id_2900/book.aspx >$

[72] McGinty, L. and B. Smyth, Comparison-Based Recommendation. Proceedings of the Sixth European Conference on Case-Based Reasoning (ECCBR-02), 2002

[73] Mi Zhang and Neil Hurley. - Avoiding Monotony: Improving the Diversity of Recommendation Lists. ACM RecSys08. Pages 123-130.

[74] Min Xie, Laks V.S. Lakshmanan, and Peter T. Wood. 2010. Breaking out of the box of recommendations: from items to packages. In Proceedings of the fourth ACM conference on Recommender systems (RecSys '10). ACM, New York, NY, USA, 151-158.

[75] N.J. BELKIN, R.N. ODDY, H.M. BROOKS, (1982) "ASK FOR INFORMATION RETRIEVAL: PART I. BACKGROUND AND THEORY", Journal of Documentation, Vol. 38 Iss: 2, pp.61 - 71

[76] Nava Tintarev, Judith Masthoff, "A Survey of Explanations in Recommender Systems," icdew, pp.801-810, 2007 IEEE 23rd International Conference on Data Engineering Workshop, 2007

[77] Nkechi J. Nnadi. 2009. Applying relevant set correlation clustering to multi-criteria recommender systems. In Proceedings of the third ACM conference on Recommender systems (RecSys '09). ACM, New York, NY, USA, 401-404.

[78] Neal Lathia, Stephen Hailes, Licia Capra, and Xavier Amatriain. 2010. Temporal diversity in recommender systems. In Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval (SIGIR '10). ACM, New York, NY, USA, 210-217.

[79] Ng, A. Y., Russell, S., 2000. Algorithms for inverse reinforcement learning. In: Proceedings of the Seventeenth International Conference on Machine Learning. pp. 663-670

[80] Nigro, Hctor Oscar, Sandra Elizabeth Gonzlez Csaro, and Daniel Hugo Xodo (eds). "Chapter VIII - Evaluating the Construction of Domain Ontologies for Recommender Systems Based on Texts". Data Mining with Ontologies: Implementations, Findings, and Frameworks. IGI Global. 2008. Books24x7. $< http : //common.books24x7.com/book/id_23220/book.aspx >$

[81] Pazzani, M. and D. Billsus, Learning and Revising User Profiles: The Identification of Interesting Web Sites. Machine Learning, 1997. 27(3): p. 313-331.

[82] Pazzani, M. J.: 1999, A Framework for Collaborative, Content-Based and Demographic Filtering. Artificial Intelligence Review, 13 (5/6), 393-408.

[83] Poh, K. L., and Horvitz, E. 1993. Reasoning about the value of decision-model refinement: methods and application. In Proc. UAI-93, 174-182.

[84] Qusai Shambour, Jie Lu: Integrating Multi-Criteria Collaborative Filtering and Trust Filtering for Personalized Recommender Systems. IEEE SYMPOSIUM SERIES ON COMPUTATIONAL INTELLIGENCE, APRIL 11-15, 2011, PARIS, FRANCE: P.44-51

[85] Panayiotis Zaphiris. Human Computer Interaction: Concepts, Methodologies, Tools and Applications. IGI Global. 2009. Books24x7. $< http : //common.books24x7.com/book/id_28820/book.aspx >$

[86] R.B. Statnikov and J.B. Matusov, Multicriteria Optimization and Engineering. Chapman and Hall, 1995.

[87] R.J. Mooney and L. Roy, Content-Based Book Recommending Using Learning for Text Categorization, Proc. ACM SIGIR 99 Workshop Recommender Systems: Algorithms and Evaluation, 1999.

[88] Rafael Mart, Micael Gallego, Abraham Duarte: A branch and bound algorithm for the maximum diversity problem. European Journal of Operational Research 200(1) p. 36-44 (2010).

[89] Rendle, S., Freudenthaler, C., Gantner, Z., and Schmidt-Thieme L., 2009. BPR: Bayesian personalized ranking from implicit feedback. In UAI 09, 452-461.

[90] Resnick, P., Varian H.R. (1997). Recommender systems, (Vol. 40). New York: ACM Press.

[91] Ricci, F., et al., ITR: a case-based travel advisory system. 6th European Conference on Case Based Reasoning, ECCBR, 2002: p. 613-627.

[92] S. Fickas, B.R. Helm, "Knowledge Representation and Reasoning in the Design of Composite Systems," IEEE Transactions on Software Engineering, pp. 470-482, June, 1992

[93] Sage, Andrew P.. Systems Engineering. John Wiley and Sons. 1992. Books24x7. $< http : //common.books24x7.com/book/id_8639/book.aspx >$

[94] Sage, Andrew P.. Decision Support Systems Engineering. John Wiley and Sons. 1991. Books24x7. $< http : //common.books24x7.com/book/id_8602/book.aspx >$

[95] Sean, J. R. McNee, M. and Konstan, J. A. . Accurate is not always good: How accuracy metrics have hurt recommender systems. In extended abstracts on Human factors in computing systems(CHI06) p. 1097-1101.

[96] Sean Munson, Daniel Xiaodan Zhou, Paul Resnick. Sidelines: An Algorithm for Increasing Diversity in News and Opinion Aggregators. Proceedings of ICWSM09 Conf. San Jose, CA.

[97] Schubert Foo. Social Information Retrieval Systems: Emerging Technologies and Applications for Searching the Web Effectively. IGI Publishing. 2008. Books24x7. $< http://common.books24x7.com/book/id_23246/book.aspx >$

[98] Shi, Yue , Larson, Martha , and Hanjalic, Alan. 2010. List-wise learning to rank with matrix factorization for collaborative filtering. ACM

[99] Shimazu, H. Expertclerk: navigating shoppers' buying process with the combination of asking and proposing, IJCAI'01.

[100] Shiwan Zhao, Michelle X. Zhou, Quan Yuan, Xiatian Zhang, Wentao Zheng, and Rongyao Fu. 2010. Who is talking about what: social map-based recommendation for content-centric social websites. In Proceedings of the fourth ACM conference on Recommender systems (RecSys '10). ACM, New York, NY, USA, 143-150

[101] Smyth, B. and P. McClave, Similarity vs. Diversity. Case-Based Reasoning Research and Development: 4th International Conference on Case-Based Reasoning, ICCBR 2001, Vancouver, BC, Canada, July 30-August 2, 2001: Proceedings, 2001.

[102] Smyth, B. and Cotter, P. : 2000, A Personalized TV Listings Service for the Digital TV Age. Knowledge-Based Systems 13: 53-59.

[103] Stuart E. Madnick, Y. Richard Wang: Evolution Towards Strategic Applications of Databases through Composite Information Systems., Journal of Management Information Systems, Vol. 5, No. 2 (Fall, 1988), pp. 5-22.

[104] Su, X. and Khoshgoftaar, T. M., 2009. A survey of collabora-tive filtering techniques. Advances in Artificial Intelligence, no. 421425, 19 pages.

[105] Suryadi, D., Gmytrasiewicz, P. J., 1999. Learning models of other agents using influence diagrams. In: Preceedings of the Seventh International Conference on User Modeling. pp. 223-232

[106] Suzana Dragi. Collaborative Geographic Information Systems. IGI Publishing. 2006. Books24x7. $< http : //common.books24x7.com/book/id_13285/book.aspx >$

[107] Theodoros Lappas and Dimitrios Gunopulos. 2010. Interactive recommendations in social endorsement networks. In Proceedings of the fourth ACM conference on Recommender systems (RecSys '10). ACM, New York, NY, USA, 127-134.

[108] Thomas D. Nielsen, and Finn V. Jensen, Learning a decision maker's utility function from (possibly) inconsistent behavior Artificial Intelligence Volume 160, Issues 1-2, December 2004, Pages 53-78

[109] Thomas, R. Roosevelt. Redefining Diversity. AMACOM. 1996. Books24x7. $< http://common.books24x7.com/book/id_2684/book.aspx >$

[110] U. Chajewska, Koller, D., Ormoneit, D., 2001. Learning an agents utility function by observing behavior. In: Proceedings of the Eighteenth International Conference on Machine Learning. pp. 35-42.

[111] U. Chajewska, Getoor, L., Norman, J., Shahar, Y., 1998. Utility elicitation as a classification problem. In: Cooper, G. F., Moral, S. (Eds.), Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence. pp.79-88.

[112] U. Shardanand and P. Maes, Social Information Filtering: Algorithms for Automating Word of Mouth, Proc. Conf. Human Factors in Computing Systems, 1995.

[113] Urszula Chajewska, Daphne Koller, Ronald Parr, Making Rational Decisions using Adaptive Utility Elicitation, Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-00),pages 363-369, Austin, Texas, August 2000.

[114] Van de Ven, A. and A. Delbecq. "The Effectiveness of Nominal, Delphi, and Interacting Group Decision-Making Processes." Academy of Management Journal 17 (1974): 147-178.

[115] Vijayan Sugumaran. Intelligent Information Technologies: Concepts, Methodologies, Tools, and Applications. IGI Global. 2008. Books24x7. $< http://common.books24x7.com/book/id_23490/book.aspx >$

[116] Virine, Lev, and Michael Trumper. "Appendix D - Multi-Criteria Decision-Making Methodologies". Project Decisions: The Art and Science. Management Concepts. 2008. Books24x7. $< http://common.books24x7.com/book/id_16221/book.aspx >$

[117] V. Vapnik and A. Chervonenkis, Theory of Pattern Recognition, Nauka, Moscow, 1974.

[118] Wartha, C., et al. Decision Support Tool - Supply Chain. in Winter Simulation Conference. 2002.

[119] William B. Frakes and Ricardo Baeza-Yates. Information Retrieval: Data Structures and Algorithms. Prentice Hall PTR, June 1992.

[120] William B.. Handbook of Systems Engineering and Management. John Wiley and Sons. 1999. Books24x7. $< http : //common.books24x7.com/book/id_8613/book.aspx >$

[121] Yang Xiang. Dynamic and Advanced Data Mining for Progressing Technological Development: Innovations and Systemic Approaches. IGI Global. 2010. Books24x7. $< http : //common.books24x7.com/book/id_34016/book.aspx >$

[122] Yin Fang, Robert Garfinkel, Ram Gopal, Arvind Tripathi, Design of a shopbot and recommender system for bundle purchases, Decision Support Systems, Volume 42, Issue 3, December 2006, Pages 1974-1986, ISSN 0167-9236.

[123] Hanjo Jeong, Hybrid Filtering in Semantic Query Processing, Ph.D. dissertation, George Mason University, 2011.

Curriculum Vitae

Khalid Alodhaibi received his Bachelors of Information Systems from King Saud University, Riyadh, Saudi Arabia. In 2006, he graduated from Santa Clara University, Santa Clara, California with Master of Business Administration. In that same year, Khalid started his Doctoral degree in Information Technology at George Mason University, and has published several research papers during his PhD studies.

Khalid has over 18 years of professional experience working in IT field. Most recently, he worked for AT&T in Pleasanton, CA as a senior system analyst. Since 2004, Khalid has joined IBM Corporation, and held several positions in the Software Group. In early 2010, he was assigned as a Manager in the software group at IBM, leading the support team of Content Manager product.