ANT COLONY INSPIRED MODELS FOR TRUST-BASED RECOMMENDATIONS

By **Deema Alathel**

B.S. in Computer Applications, June 2000, King Saud University, Saudi Arabia
M.S. in Computer Science, October 2005, King Saud University, Saudi Arabia

A Dissertation Submitted to

the faculty of
The School of Engineering and Applied Sciences
of The George Washington University
in partial fulfillment of the requirements
for the Degree of Doctor of Philosophy

May 17, 2015

Dissertation Directed by

Abdelghani Bellaachia
Associate Professor in the Computer Science Department

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346

The School of Engineering and Applied Science of The George Washington University certifies that Deema Alathel has passed the Final Examination for the degree of Doctor of Philosophy as of December 12th, 2014. This is the final and approved form of the dissertation.

ANT COLONY INSPIRED MODELS FOR TRUST-BASED RECOMMENDATIONS

Deema Alathel

Dissertation Research Committee:

Abdou Youssef, Professor of Engineering and Applied Science, Dissertation Director

Xiuzhen Cheng, Professor of Computer Science, Committee Member

Nan Zhang, Associate Professor of Computer Science, Committee Member

Hiroki Morizono, Associate Professor of Integrative Systems Biology and Pediatrics, Committee Member

DEDICATION

To Sarah…

ABSTRACT

ANT COLONY INSPIRED MODELS FOR TRUST-BASED RECOMMENDATIONS

Recommender Systems can be thought of as being information filtering systems that aid in predicting a user's preference or rating for a certain item. Collaborative filtering is a technique used by some recommender systems to filter the information and thus provide a prediction by means of collaboration among *similar* users in the network. Trust-based recommender systems generate the recommendations by making use of known, expressed trust between the users to increase the accuracy of the recommendations. In my thesis, I propose a nature-inspired framework based on our Trust-based Ant Recommender (T-BAR) algorithm, that is applied to trust-based recommender systems to further increase the accuracy and the coverage of the recommendations in the network. T-BAR is the first successful application of an algorithm from the swarm intelligence field to trust-based recommender systems. T-BARis a hybrid of the Ant Colony System (ACS) computational model and the Ant System (AS) algorithm that mimics the behavior of ants during their foraging for a *good* food source. My proposed hybrid algorithm's advantage over other known algorithms that have been used with Recommender Systems is that it considers all the target item ratings along the solution paths rather than just stopping and using the first rating found in the search process. The Epinions.com dataset is used for the empirical evaluation of Trust-based Ant Recommender (T-BAR) along with two different variations of it. The

performance of the algorithms is further analyzed by examining different views of the dataset to understand the algorithms' strengths and weaknesses. T-BAR and its variations proved their success by drastically improving the coverage of the recommendations while maintaining a reasonable level of accuracy of the results in general. T-BAR outperforms the basic collaborative filtering (CF) algorithm that uses the Pearson Similarity and Paolo Massa's MoleTrust (MT) by achieving a balanced trade-off between accuracy and coverage.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 RATIONALE

In the past few years there has been an increasing demand for personalizing users' experiences on the web and thus for filtering the vast amount of information available online in order to deliver the right piece of information to the right user. Recommender systems can assist in providing an adaptive web environment by suggesting to a user items, such as movies, books, music, jokes, articles, etc., that the user may find useful or interesting. Collaborative filtering techniques are considered to be the most popular approaches used in such systems which aim at finding users that are similar to the active user and then basing the recommendation of items on the item ratings provided by those like-minded users. But due to the inherent problems with recommender systems, such as cold start users and the lack of users' ratings in general, many researchers shifted their attention to trust-based recommender systems where users explicitly express how much they trust other users rather than relying on the system to implicitly predict the similarity between them. Many researchers prefer to deal with implicit trust claiming that it is easier to calculate and collect, but this dissertation stems from the belief that systems that rely on explicit trust from users should not be neglected but should rather be extensively investigated to utilize the trust and overcome these challenges.

The recommendation problem in trust-based recommender systems is considered to be an optimization problem since the goal is to reach and utilize as many useful, trust-

worthy users as possible to predict the rating of an unseen item. Many optimization algorithms have been explored and successfully applied to recommender systems, including nature-inspired algorithms. Ant algorithms have been recently considered in recommender systems, in which a group of decentralized agents mimic the behavior of ants in their colonies while searching for a good food source, leading to the emergence of a solution as a result of the collaborative behavior of the ants. Although successful, such algorithms have never been applied to trust-based recommender systems.

This dissertation presents a set of novel models that are based on ant algorithms to solve the recommendation problem in trust-based recommender systems.

1.2 CONTRIBUTIONS

The main focus of this dissertation is to illustrate how the application of ant colony algorithms to analyze and utilize trust relationships in web-based social networks can exploit additional information within the network that can lead to better exploration of the solution space and thus to better personalized recommendations which enhance the system's performance.

The contributions of this dissertation can benefit research in trust-based online communities, recommender systems, and artificial intelligence systems. Through the presented models, this research shows that exploiting trust relationships in web-based social networks can enhance a user's experience on the web.

In order to accomplish the goals of this dissertation, the presented research achieves the following:

- Design a model based on ant algorithms to increase the accuracy of recommendations in trust-based recommender systems.
- The designed model must exploit and utilize all useful information in the system, including implicit and explicit trust, item ratings, and user popularities in the recommendation process.
- Analyze the designed ant-based model to identify areas for improvement then deign the enhanced models to target specific problems in recommender systems, such as the cold start problem.

1.3 ORGANIZATION

For this dissertation, trust in web-based social networks has been chosen as a very specific area to study the larger issue of trust, reputation, and relationships in social networks. The decision to work with web-based social networks is imposed by the fact that they form a large, publicly available dataset with tremendous interest from the general public. Chapter 2 specifically defines what can be considered as a web-based social network, and then presents the results of an exhaustive survey of websites. Billions of user accounts spread across hundreds of websites were surveyed, with a wide subject range such as religious, dating, socializing, and entertainment. The description of the size of websites and their general categories is followed by an explanation of how users can add information to their social connections. In fact, several popular social networks have already incorporated a way for users to express trust among them within

the network, which increases the desirability to choose to work with these datasets.

Before making any computations with trust in social networks, it is crucially important to define what trust is and the properties it has. Within computer science, trust has been adopted by many subfields to mean many different things. For example, it has been used to describe security and encryption [80], used as a mean for authentication or digital signatures [8], and as an attack-resistance gauge [149]. It was not until recently that the computing community has begun to consider the more social aspect of trust as a relationship between humans. The difficulty behind combining trust with algorithms and mathematical analyses is that trust is difficult to define, let alone to express as a quantifiable way. Intense theoretical analysis and complex models have been used to address this issue. However, the real improvement to the fusion of these two topics has come in the form of web-based social networks that force people to quantify trust. In Chapter 3, a definition of trust is presented within the context of the philosophical, sociological, and psychological communities. The definition captures the nature of social trust relationships yet remains clear and simple enough to be used in social networks on the web. The chapter explains the different trust properties to facilitate how they are used later in the work presented in this dissertation.

A discussion about the basic components needed to devise a trust-based model is described in Chapter 4. The general definition of what trust metrics are and their importance is highlighted along the different types of trust algorithms. An overview of how trust is calculated in the literature is provided afterwards since they relate to the metrics presented in this research.

Chapter 5 provides a detailed explanation about Recommender Systems (RS), their building blocks, techniques, and applications. Recommender Systems gained a lot of popularity through their ability to recommend to users items that they will most likely find interesting. A deep understanding of recommender systems and their functionality paved the way for creating the proposed models in this dissertation. Most of the techniques and methods applied to RS stem from the field of Data Mining, thus Chapter 6 covers the most popular techniques that one would expect to come across when working with RS. The chapter presents algorithms in each of the main steps of a data mining process, i.e. Data Preprocessing, Data Analysis, and Results Interpretation.

Instead of following the traditional methodologies for building a RS, this dissertation opted to apply a nature-inspired algorithm inspired by the behavior of ants when foraging for food, which resulted in creating a novel approach for predicting the ratings in trust-based RS. Thus, a detailed explanation of the basic terminologies and steps used in Ant Colony Optimization (ACO) algorithms in general is presented in Chapter 7. The chapter justifies how the behavior of real ants can be successfully simulated by artificial agents to accomplish the same successful results reached through a decentralized system similar to the one that real ants create.

While analyzing ACO algorithms for this research, it was noticed that there is a window of opportunity to improve the performance of ACO algorithms by altering the way pheromone is initialized in such systems. The novel local pheromone initialization technique is explained in Chapter 8 as part of this dissertation's contributions. The chapter includes the extensive experiments conducted on the travelling salesman problem along with the analysis of results, which prove that the presented technique can

significantly affect the speed of the system's convergence to the optimal solution.

Before discussing the details of the presented models in this research, Chapter 9 covers the different attempts made in the literature to apply ant algorithms to RS. The presented overview supports the novelty of the presented models since they are the first successful application of ant algorithms to trust-based RS. For the sake of completion, the chapter also discusses the use of trust in different applications with ant algorithms. The chapter concludes by stating the problem to be solved by this research along with the formal definitions of the presented models' parameters.

Chapter 10 explains in details the specifications of the first presented model in this dissertation, Trust-Based Ant Recommender (T-BAR), with a description of how the model incorporates both local and global trust values, deals with encountered item ratings, enforces a path trust threshold, and utilizes the presented pheromone initialization technique. The model is tested and applied to the Epinions dataset and used to predict the rating for an unseen item by the user. The results are compared to the ones obtained from applying several other algorithms [93][94]. Chapter 11 elaborates on the details of these experiments providing a complete analysis and explanation of the results. In addition, the chapter discusses two variations of T-BAR that change the way the path trust quality is assessed and the results are compared to the ones previously obtained.

Based on the detailed analysis of T-BAR and its two variations in Chapter 11, it was determined that there is room for improvement in terms of increasing the importance of explicit trust in the system to compensate for the lack of item ratings for cold start users, and in terms of increasing the level of communication between the artificial agents by sharing more information about the constructed solutions and studying the effect of that

on the system's performance.

To study the effect of increasing the influence of trust in the solution construction process, Chapter 12 presents two localized T-BAR models that reflect the differences in trust levels between the users on the initial pheromone levels assigned to the edges connecting these users. The new local pheromone initialization technique presented in Chapter 8 calculates a single value using local information within each neighborhood to initialize the pheromone level on the edges within that neighborhood. However, the localized T-BAR models still use local information to calculate the initial pheromone level but they use the individual trust values to reflect each edge's importance. The chapter explains the details of two localized models: Simple Localized T-BAR (SLT-BAR) and Averaged Localized T-BAR (ALT-BAR).

On the other hand, Chapter 13 discusses how altering the way ants communicate and share information can have an effect on the system's results. The chapter introduces two dynamic T-BAR models: Dynamic Localized T-BAR (DLT-BAR) and Dynamic Averaged Localized T-BAR (DALT-BAR). The dynamic aspect of the two models stems from the dynamically changing local information within each neighborhood that is used by ants in their solution construction process. The results presented are compared to the ones obtained from other known algorithms in addition to the ones obtained from T-BAR.

The analysis of the models presented in this dissertation shows how an understanding of the properties of relationship types within systems can lead to effective algorithms for understanding the implicit and hidden relationships in those systems.

This dissertation can be considered as an extension and an enhancement to the work done on trust-based social networks and to the work done in the field of ACO algorithms. This research complements the growing body of work that is integrating social network analysis and trust into the user experience as well as the work done in the field of artificial intelligence.

Furthermore, the results here seem to suggest that a deeper understanding of the relationships in complex systems and methods for inferring information about them has the potential to lead to new discoveries in the social, biological, and physical sciences.

Ultimately, trust and social preferences can be integrated into any number of potential applications. Social networks are only one type of complex system, and trust is only one type of relationship, so the possibilities are limitless.

CHAPTER 2

WEB-BASED SOCIAL NETWORKS

2.1 INTRODUCTION

Web-based social networks (WBSN) have grown rapidly in number and scope since the mid-1990s. They present an interesting challenge to traditional ways of thinking about social networks. These networks are large, living examples of online user interactions. It has been rarely possible in the past to look at an actual network of millions of people without having to use models to fill in or simulate most of the network. It has always been a difficult problem to gather social information about a large group of users. With WBSN, there are many networks online with millions of users that need no generated data. These networks are also much more complex with respect to the types of relationships they could have within them. It is common to have information qualifying and quantifying aspects of the social connections between people in these systems. This means there is a potential for a much deeper analysis of the network.

The term *social network* has become looser as interest in social networking has increased. Many sites promote themselves as social networks while they do not maintain any data that would be useful for a network analysis. This chapter presents a set of criteria for a system to qualify as a WBSN and another set for determining when information can be considered part of a relationship.

2.2 THE STUDY OF SOCIAL NETWORKS

Most of the fundamental work in the analysis of social networks, and the major advances in the 20th century have been carried out in the fields of sociology, psychology, and communication [13][153][151]. With the goal of understanding the function of relationships in social networks, and how they affect the social systems within the networks, the conducted research has been mainly both theoretical and applicable. Labor markets [102], public health [23], and psychology [112] are just a few of the areas where social network analysis has generated interesting results.

In the last ten years or so, there has been an increasing interest in the structure and dynamics of social networks to complement the work already being done in social network theory. Although one of the first and most popular papers in this area, *Six Degrees of Separation* [99], was done by Milgram who is a social scientist, but the topic has attracted attention from physical scientists as well. Their studies have covered issues such as mathematical analyses of the structure of small world networks [152], community structure [50], and how social network structure affects the spread of disease [71][106].

As the web emerged and expanded, online communities and social networks became a source of interesting data. Garton, *et al.* [49] presented a detailed explanation on traditional social network analysis methods that could be applied to these online communities. Work in this area was also adopted by the interdisciplinary field of human-computer interaction (HCI), which resulted in interesting work related to the design and support of online communities [117] and their application to problems such as collaborative filtering [78] and electronic commerce [75].

Social networks on the web offer new opportunities for researchers across the several disciplines. For example, social networks provide a new, large source of data for mathematical and structural analysis that can be applied to network topologies extracted from the web. At the same time, users are constantly participating online in rich social environments while building these networks. That creates a rich area for scientists interested in the general function of social interactions, however close attention should be paid to the contexts of these networks as they tend to be very restricted and thus they can serve as a window into specific communities.

2.3 CHARACTERISTICS OF WBSN

There are several ways in which social networks can be represented on the web. For example, a group of users can be considered to form a social network if they are connected through online transactions or if they post messages within the same thread on a news group or message board. There is a great misunderstanding about the properties of social networks among the common users and online communities. This led to many communities online to falsely claim to be or support social networks, while they lack some of the properties one may expect of a social network. A web-based social network must meet the following criteria [51]:

1. The network must be easily accessible online through a web browser. This criterion eliminates networks where users would need to download special software on their computers in order to participate in them. It also excludes social networks based on other technologies, such as mobile devices such as networks formed on applications like WhatsApp or BBM.

2. Users must explicitly state their relationship with other people in the network. Although social networks can be extracted by analyzing implicit interactions between the users, a WBSN is more than just a potential source of social network data; it is a website or framework that has the goal of developing an explicit social network; i.e. users are aware that they are creating social connections with other users. This criterion excludes social networks that are based on events that link people through a connection created as a side effect of another process, such as auction transactions and co-postings.
3. The system must have built-in support for users' social connections. In other words, the system should be specifically designed to support a user's ability to add another user to their social circle in the network. So a group of friends who maintain a simple HTML page with a list of their friends would not qualify as a WBSN because HTML does not have explicit built-in support for making social connections. A WBSN must have a unifying structure that connects the data and manages how it is presented and formatted.
4. Relationships must be apparent and browsable. That does not necessarily mean that the data has to be public to anyone on the web, but it should be at least accessible to the registered users of the system. Websites where users maintain private lists of contacts or ones that allow users to bookmark the profiles of other users or maintain address books are ruled out as WBSN. Although these lists are explicit expressions of social connections, but they would not qualify a system as a WBSN if they cannot be seen and browsed by other users.

Based on these criteria, many of the major social networking websites like Tickle and LinkedIn would qualify as WBSN. For the same reasons, many dating sites, like Match.com, and other online communities that connect users, such as Craig's List or MeetUp.com, would be excluded.

Within web-based social networks, users are often able to say more about their relationships than simply stating they exist. Yet, the functionality of a WBSN can be easily confused with the actual information about a relationship. Therefore, it is helpful to have a set of criteria to establish when an action or data qualifies as information about a relationship in the social network:

1. A basic social networking connection between users must exist before adding any additional information about it. In order to use the additional information about a relationship, there must be a relationship between the individuals in the first place.
2. The information must be persistent. Many websites allow users to send messages or mini-messages (such as "winks" or "pokes") to indicate interest without really establishing any form of connection. Since these are sent and do not persist as a label on the relationship, they do not qualify as a piece of information about a relationship. However, comments or endorsements about a person do persist on the website and are considered as free text descriptions of a relationship.
3. The relationship information must be visible and editable by the user who added it. That does not necessarily mean that the information has to be publicly visible, some data, like trust ratings, are personal and users usually do not want to share them with others.

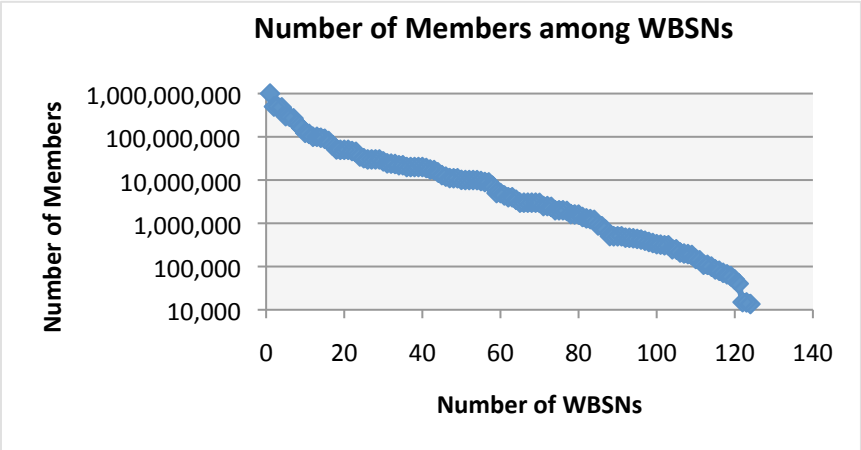


Figure 2.1: Web-based social networks membership ranked by population.

2.4 A Survey of Web-based Social Networks

This section outlines the current most popular social networks available on the web. The number of registered and active users and primary purpose of each website, along with the launch date is described for each site. Sites that require an invitation to join are not included. The list of current WBSN is constantly growing and changing. As of December 16, 2013 there are over 150 active social networks online originating from different countries around the world. One of the earliest online social networks is classmates.com, which was launched in 1995 and is still active to this day with over 50 million registered users.

2.4.1 Size

The size of the current WBSN varies greatly. The 10 most popular networks have over 100 million registered users, as shown in Table 2.1. Figure 2.1 shows the membership of sites ranked according to size. It is obvious that there is an exponential decrease in the membership of the sites moving from the largest to the smallest.

Table 2.1: Web-based social networks with over 100 million members

Name	Focus	Registered Users	Active Users	Date Launched	Country of Origin	Date of Statistics
Facebook	Photos, videos, blogs, and apps.	> 1 billion	1 billion	Feb. 2004	United States	Sep. 2013
Tencent Qzone	Photos, videos, blogs, and apps.	> 597 million	150 million	2005	China	Sep. 2012
Google+	General.	> 784 million	712 million	June 2011	United States	Oct. 2013
Twitter	Micro-blogging, RSS, updates.	> 500 million	> 200 million	Mar. 2006	United States	Dec. 2012
Sina Weibo	Social micro-blogging.	> 500 million	> 100 million	Aug. 2009	China	Feb. 2013
Odnoklassniki	Connecting with old classmates in Russia.	> 205 million	148 million	Mar. 2006	Russia	Apr. 2013
LinkedIn	Business and professional networking.	> 225 million	160 million	May 2003	United States	June 2013
VKontakte	Music sharing.	> 220 million	100 million	Sep. 2006	Russia	Apr. 2013
Tumblr	Micro-blogging.	> 110 million	100 million	Feb. 2007	United States	May 2013
Dropbox	File sharing	> 100 million	100 million	Sep. 2008	United States	Nov. 2012

2.4.2 Categorization

With the vast amount of WBSN online, one may categorize these sites from different perspectives, such as location, purpose, and size. For example, the networks can be categorized based on purpose into: blogging, business, dating, photo sharing, religious, and social/entertaining. However, some networks may belong to multiple categories such as sites that serve both religious and dating purposes.

2.4.3 Relationship Data

As mentioned in the previous section, some WBSN allow users to add information about their relationships. However, it turns out that only one third of existing sites had some method for describing their connections. For the rest, the only method of describing relationships was through free-text comments. With a few exceptions (such as LinkedIn which is a professional networking site), most of those non-describing sites were dating or social/entertainment sites where testimonials were in the form of friends writing about their friends. On the other hand, the sites that allow users to describe their relationships usually do so in a more restricted way. Most of them include options for users to categorize their relationships such as Facebook where a relationship can be further categorized as a friend, cousin, family, work, city, etc. In some networks, relationship types can be user-defined, but usually users opt to choose from the site's supplied list. Other sites provide users with the ability to rate aspects of their relationships using a numeric scale. For instance, the social website Orkut allows a user to rate another user based on three criteria: trust, attractiveness, and coolness. Each is rated on a scale from 0 – 3, which could supply a social network analyst a deeper understanding about the

qualities of each relationship.

Typically, an analyst begins by constructing a graph representation of the social network and using the rating numbers as labels on edges. It is important to understand the functional properties of the relationship characteristic. Determining whether the characteristic is symmetric between users, transitive or composable for example dictate the types of algorithms and mathematical methods that could be used to get a better insight about the indirect relationships between people in social networks.

2.5 IMPORTANCE OF ANALYZING WBSN

WBSN provide analysts with a magnified view of real living, evolving networks. Users manipulate their relationships frequently by adding, removing, and changing connections all the time. Also, the growth rate of such networks is remarkable with popular sites gaining literally hundreds and thousands of members everyday. Analysts can easily track new members and how they establish connections with existing members in the network at regular intervals, providing them with an insight on how social networks expand and evolve. In addition to the analysts' ability to track the type of friends added to a person's network, they can also keep records of when those relationship types change.

Computationally, there is an opportunity to develop algorithms that can analyze the connections within a social network's graph. This can lead to recommendations for new connections and further understanding of the existing relationships between users. It is also possible to integrate users' social preferences into applications, especially in networks that are open data sources.

Such data sources form the infrastructure for this dissertation, which aims to achieve online personalization through social intelligence.

CHAPTER 3

TRUST: DEFINITION AND PROPERTIES

3.1 INTRODUCTION

Trust is a major factor among individuals in any functioning society [47][28][146] therefore it is natural to expect the same to be true in online social communities. When building an online system, there are several strategies to increase a user's trust in the system and in other users in general [137]. In WBSN where users explicitly define their trust relationships, otherwise known as trust-based social networks (TBSN), the goal is not necessarily just to build trust between members in the network but also to make useful computations with the existing data. In human societies, trust depends on several factors that are not necessarily easily modeled in a computational system.

When a person is faced with the decision of whether or not to trust someone, several aspects govern that decision. These aspects include past experiences with that person and experiences with his or her friends, personal judgment of opinions or actions taken by that person, personal psychological reasons that probably are unrelated to the other person, rumors, other people's opinions about that person, and possible advantages and disadvantages of extending trust to that person. Applying a usable notion of trust to social networks in a way that is computationally beneficial requires a precise, clear definition of trust that preserves the properties that we are all familiar with in our social lives.

In order for trust to be measured in social networks, the definition of trust must be explicit and clear. Average users of online social networks need a simple definition that

they can understand and relate to so that they can accurately describe their trust in others. Such a definition ensures avoiding muddled expressions of trust within the network due to confusion in understanding what the term refers to.

3.2 DEFINITION OF TRUST

There are different definitions for trust across many disciplines, such as in the areas of sociology, psychology, economics, political science, history, philosophy, and computer science. A major problem with defining trust is that it could mean something different to each person even if it is considered within the same context [34][135].

Since the focus of this dissertation is to utilize trust in TBSN for recommendation purposes, it is logical to focus on the perception of trust in the computer science field. One of the most referenced works in the literature is Marsh's *Formalising Trust as a Computational Concept* [91]. In his work, Marsh sheds the light on the different elements affecting trust, from the biological to the sociological ones, in order to allow agents to interact in a distributed manner using an underlying trust model. His trust model is purely theoretical and complex. In addition to the difficulties faced with Marsh's implementation, it does not seem to be appropriate for use in social networks. In social networks, users assign trust as a single rating describing their connection to others without explicit context or history, while in his work the focus was on interacting agents that could maintain information about history and observed behaviors. In order to use his trust model in a social network setting, much of the necessary information is missing.

Web-based social networks are popular among the average web users. Thus, the definition of trust in TBSN must be straightforward, uncomplicated, and clear enough so

that an average web user can understand what he is expressing and therefore can express it accurately. The most frequently referenced definition of trust is the one given by Deutsch in [33] where he states that trusting behavior occurs when a person (say Alison) encounters a situation where she faces an uncertain path. The result of following the path can be good or bad, and the occurrence of the good or bad result is dependent on the action of another person (say Bill). Moreover, the negative implication of the bad result is greater than the positive implication of the good result, which encourages Alison to make the correct choice. If she chooses to follow the path, then she has made a trusting choice; i.e. she trusts that Bill can lead the way and take the necessary steps to ensure the good outcome. The requirement indicating that the bad outcome must have greater negative impact than the positive impact of the good outcome was imposed in [55]. Sztompka [144] also defines trust in a simple manner, similar to that of Deutsch, by presenting trust as a gamble on the future that relies on the actions of others. There are two main components in both definitions: *belief* and *commitment*. A person *believes* that the trusted person will act in a certain way, then that belief is used as the motivation for *committing* to a particular action. The two components complement each other because belief or commitment alone is not enough to indicate trust. Believing in someone's behavior without making commitments to specific actions based on that belief does not necessarily indicate trust. In the same manner, committing to an action that happens to be similar to or appears to be (by chance) dependent on the actions of someone else without having belief in his behavior does not imply trust either.

In order to define trust to be computationally used in her dissertation in the area of computer science, Golbeck [51] extracted the main social aspects of trust from the

definitions above to define trust as follows: *Alison trusts Bill if she commits to an action based on a belief that Bill's future actions will lead to a good outcome*. The same definition is assumed in this research since it coincides with the context of Golbeck's work.

The action of the trusted person and the commitment by the trusting individual do not have to be significant and they rely on the context in which trust is being examined and defined. For example in the context of movies, Alison trusts Bill if she decides to see a movie (commits to an action) based on Bill's recommendation (based on her belief that Bill has good taste in movies which happens to be similar to hers).

The justification for the belief component of the definition may vary among people even within the same context. In the previous example, Alison believes that Bill has similar taste to hers and that is why she decides to trust his taste, while another person (say Mary) believes that Bill is popular and many people follow his recommendations so she will trust his taste based on that. People may base their belief on personal pervious experiences that have nothing to do with the other person, on a history of interactions with that person, or on information gathered from an outside source.

It is important to pay attention to the fact that trust is not necessarily represented by a single value, but could be expressed as a group of values each referring to a different aspect within a single context. Referring again to the movies example, a person may have an overall opinion on how much he trust another person's general taste in movies, but it would be more accurate and precise if a person is given the opportunity to express how much he trusts the other person's taste in each movie genre. The process can be taken one step further by breaking down genres by period (50s, 60s, etc.), and so on. The

possibilities are endless when it comes to the different ways we can break down trust when used in TBSN, however it is crucial to maintain a balance between the accuracy of the trust expression and the complexity of the expression.

The adopted definition of trust serves as the foundation for understanding the properties of trust, identifying where trust exists in social networks, and how it can be used in computations.

3.3 PROPERTIES OF TRUST

3.3.1 Transitivity

The most important trust feature in the context of this dissertation is transitivity. With respect to trust, transitivity is not treated in the same way as it is treated in mathematics. For example, just because Alison highly trusts Bill, and Bill highly trusts Richard, that does not necessarily imply that Alison will highly trust Richard. However, there is a perception that trust can be passed between people. In our daily lives, it is common for us whenever we meet a new person to ask our trusted friends about how much they trust him. When we consider a trusted friend's opinion about someone, we are using her opinion and integrating it with whatever knowledge we have to form an initial personal opinion about that person.

The adopted definition of trust in this research supports the idea of transitivity. In the definition, trust involves a belief that the trusted person will take an action that will lead to a good outcome. For example, if Alison asks Bill whether or not Richard is a good mechanic, she is going to rely on Bill's answer to guide her action of whether or not to use Richard because she believes Bill will give her information that will lead to a good

service outcome. If Bill suggests to Alison that she should trust Richard, Alison will rely on her trust in Bill to form some trust in Richard. So, it is obvious that Bill's recommendation becomes the foundation for Alison's *belief* component in her new trust for Richard since she developed some preliminary trust in him based on Bill's information.

This argument can be extended to longer chains of trust. For instance, when Alison asks Bill about his opinion in Richard, Bill may not know anything about Richard. So, Bill may turn to his trusted friend Mary to get her recommendation about Richard, and then he passes the recommendation he receives to Alison. Hence, the chain becomes: Alison → Bill → Mary → Richard. The definition of trust will not change along the chain and the two components (belief and commitment) are still intact at every step: Alison trusts Bill to provide her with information that will lead to a good outcome (which is her decision on whether or not she should trust Richard), and Bill trusts Mary to give him good information about Richard that will lead to a good result (which is the ability to provide Alison with reliable information about Richard). When we have a chain of trust, Bill's future action is considered to be his decision to resort to Mary for her opinion about Richard. Therefore, this shows that it is possible to pass trust along a chain of trusting people. This is illustrated in part (a) of Figure 3.1.

Since trust as a concept is not perfectly transitive, it is reasonable to expect it to deteriorate as it is passed down along a chain of connections. It is rational for Alison to have more trust in Bill's information if he knew Richard directly, since she trusts Bill, rather than having the information passed down to her through a chain of people who trust each other, because she simply does not know whether or not they have the same

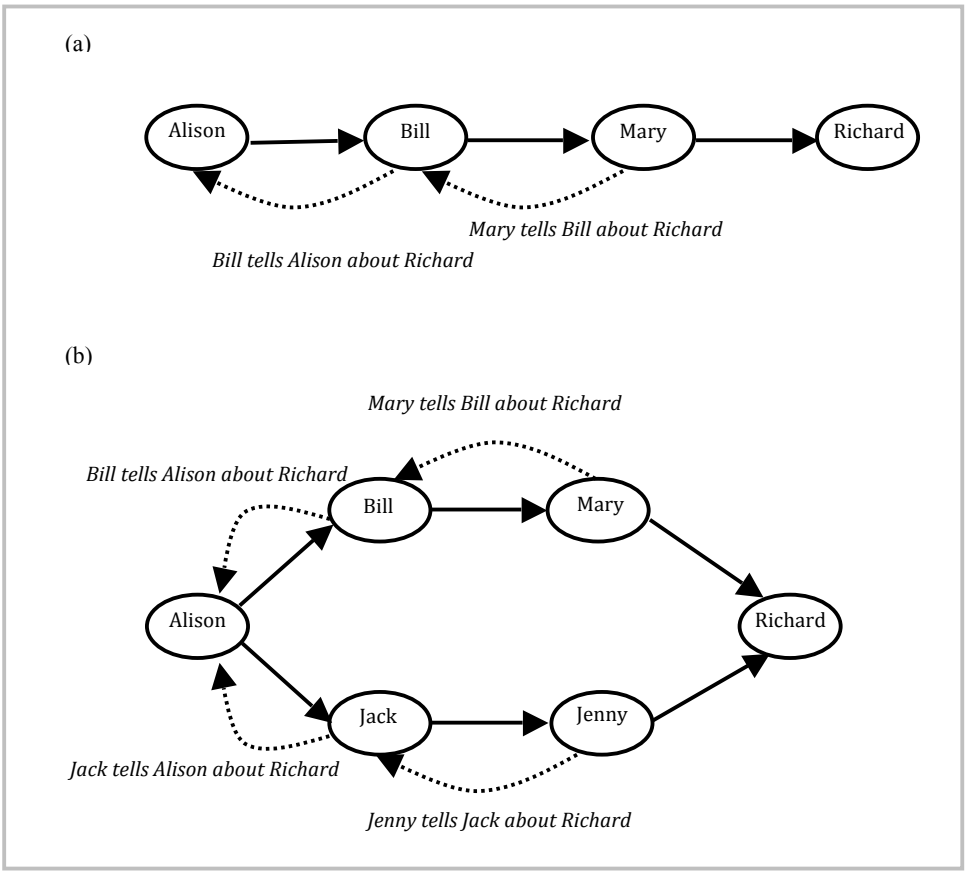


Figure 3.1: Path structures for finding trust.
Part (a) a simple chain of people where Alison forms an opinion about Richard based on the information passed from Mary to Bill, then from Bill to Alison.
Part (b) a more complex structure where Alison receives information about Richard from two people and she must come up with an opinion by composing the information she receives.

standards she has when it comes to trusting people. The idea of propagating trust and exploiting the concept of transitivity along a chain of acquaintances has been the focus of many publications in the literature [59][61][73][74][127][159].

Transitivity is an important aspect of trust in this dissertation because it contributes to reaching more possibly trust-worthy users that can assist in the recommendation process.

3.3.2 Composability

Transitivity justifies the possibility of passing down trust information along a chain of people trusting each other. However, there are cases where the trust recommendations can be supplied by multiple chains leading to the unknown person. Of course the recommendations generated by the chains will not necessarily be identical. In fact, there might be diversity in the recommendations received by the different chains. This case is shown in part (b) of Figure 3.1, where many people are making recommendations about how much to trust Richard. In such a situation, Alison must compose the information received from the multiple chains to decide whether or not to trust Richard. For that reason, trust composability is another important property for making trust computations.

The importance of trust composability becomes clear when the different trust recommendations are considered as sources that supply the belief component of trust. Receiving the information from multiple sources provides more reasoning and justification to support the belief. Usually, the trust values of each neighbor and their recommendations about the unknown person are all used as input for a composition function. The function's details vary depending on the context and situation at hand. This is another computational issue that will be addressed in this work since a user may receive a prediction for an item's rating from more than one source. The dissertation's proposed integration of transitivity into its model along with the composition function used to assist in the recommendations have attributed to the high quality of the obtained results.

3.3.3 Personalization and Asymmetry

Trust is subjective because it is an expression of personal opinion. Two people may have very different opinions about the trustworthiness of the same person due to several factors, such as previous experiences and interactions with that person. Therefore, it is crucial to personalize trust by computing it from each user's perspective.

The definition of trust includes a belief that the actions of the trusted person will lead to a good outcome. What distinguishes an outcome from being good or bad depends on the perspective of the person. For example, when two teams play against each other, what qualifies as a good outcome depends on which team the person is rooting for. Everyone has interests, priorities, and opinions that may clash with the interests, priorities, and opinions of others, therefore when and how much one would trust people will vary accordingly. From that perspective, it would be almost impossible to have a universal measure of the trustworthiness of a person. As a result, trust calculations must take into consideration the interests of the person requesting the information.

The asymmetry aspect of trust is also important and is related to the fact that trust is subjective. Just because two people trust each other does not necessarily imply that they trust each other at the same level (trust is not necessarily identical in both directions). Nevertheless, trust could be a one-sided relationship where only one person in the relationship trusts the other [63][28] but this may be viewed as an extreme case. The truth of the matter is that most of the trust expressed between people is mutual [63] although there might be differences in how much they trust one another. This is due to differences in personal experiences, psychological backgrounds, and histories leading to the reasonable conclusion that any two people will evaluate their trust in one another

differently. A clear example is the relationship between parents and their children where they trust each other at different levels simply because there are certain tasks that children are not capable of. Another example is the relationship between employees and their supervisors; most employees would probably trust their supervisors more than the supervisors trust their employees. These types of relationships can be found in a variety of hierarchies [156]. Since asymmetric trust relationships can arise in any relationship, the trust models used in social networks must allow for these differences to be represented.

3.4 THE VALUE OF TRUST

Trust provides information about a social relationship so in a WBSN it is represented as a label on that relationship. There is great flexibility in how to format that label. However, trust is a fairly new concept in social networks and only a handful of the existing ones provide a way for expressing trust in one way or another.

One of the simplest approaches for representing trust was observed in the social network eCademy in which users have two options: either state that an individual is *trusted*, or do not provide any trust statement about an individual. The technique used in this network does not allow trust to vary within a range; either it exists or it does not. Also, a user's decision of not to issue a trust statement for another user implies that the user has no opinion or information about the other user. Therefore, their system lacks the ability to indicate untrustworthiness. It simply lets users indicate which people they trust.

There are certain scenarios in which a relationship does not require any form of variance. For instance, relationships expressing whether or not you know someone,

whether you ever met at least once before, if you are related, or if you are co-workers would either exist or not. However, based on the common notion and understanding of what trust is, it can never be expressed as simple as that without having some variance in its degree or strength [48][90][91].

There are other approaches for representing the levels of trust. Epinions.com provides users with two labels: trust (represented by a 1) and distrust or block list (represented as -1). Richardson *et al.* [127] used a continuous range of 0 – 1 to allow for a precise trust expression. The social network Orkut allows trust to be expressed within a discrete range from 0 to 3. Many networks use labels to differentiate between the different levels of trust rather than using numbers (e.g. *very low trust*, *low trust*, *moderate trust*, *high trust*, and *very high trust*). There are many possibilities for labeling trust and expressing its degree, although most of them have never been used in any existing WBSN. It is also possible to use a ranking system to provide a relative, rather than absolute, value for trust. This could be combined with preference elicitation mechanisms [79][17] to build a profile of a user's trust model. In general, the most common way of expressing the variance in trust in most WBSN (that support the trust model) is to use a direct rating scheme that passes the burden of expressing trust to the users and allows for quick information extraction. The proposed models in this research are designed to work with the standard explicit rating of trust.

CHAPTER 4

TRUST INFERENCE APPROACHES

4.1 INTRODUCTION

One of the goals in this dissertation is to develop a model that can utilize trust within a social network to generate good system outcomes. Rather than just using the explicit trust values in the system, the presented work in this research exploits trust between people that do not necessarily have a direct link between them in the network. This requires an understanding of the structure of trust-based social networks and analyzing how trust behaves within them. The term *source* is used to refer to the person whom we are trying to predict his trust in another person (regardless whether they are directly connected or not). The term *destination* describes the person whom we are trying to express trust towards.

4.2 ONLINE SOCIETIES AS TRUST-BASED SOCIAL NETWORKS

With the increasing popularity of online social networks, it has become common to interact with unknown people within the so-called *global village*. Therefore, a need has emerged for new tools to assist in deciding whether or not to trust someone online. Several solutions have been proposed in the literature, but the most useful ones where the techniques that use a decentralized collaborative assessment of the trustworthiness of unknown users [92]. The recent trends in TBSN support this idea by allowing each user to express her opinion in others by means of a trust rating.

Based on the adopted definition of trust in Chapter 3, a trust statement can be defined as the explicit opinion expressed by a user about another user with respect to the perceived quality of a certain characteristics of that user. For example, on a site where users provide reviews about products, users could be asked to express a positive trust statement on a user whose reviews and ratings they have consistently found to be useful and a negative trust statement on reviewers whose contributions are found to be consistently offensive, inaccurate, or in general useless. For instance, in the Epinions dataset trust T is expressed from user x towards a user y as a value within the range [0, 1], where $T_{xy} = 0$ indicates that user x has issued a statement expressing his degree of trust in user y as the minimum, i.e. x totally distrusts y, while $T_{xy} = 1$ means that user x totally trusts user y. This simple example is a reminder that trust is subjective so a user may receive different, and sometimes contradictive, trust values from different users. Also since trust is asymmetric then if x trusts y as 0.8, this does not necessarily mean that y has to trust x with the same degree, that is if y trusts x in the first place. In most TBSN, a user usually issues trust statements towards a small portion of the users in the network. The remaining users are considered unknown to the source user.

A graph that represents the societies created by trust relationships can be created by aggregating the trust statements expressed by every user in the social network. A simple trust network is illustrated in Figure 4.1. The network is represented as a directed, weighted graph whose nodes represent users and edges indicate issued trust statements between the users. An edge would be pointing from the user who issued the trust statement (source) to the user whom trust is being expressed towards (destination). An edge is labeled with the associated (explicit) trust level.

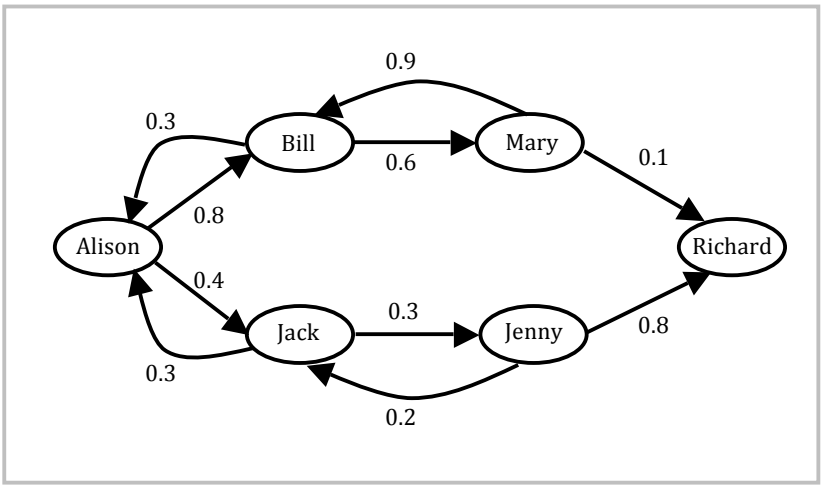


Figure 4.1: A simple trust network.
Nodes represent users and edges represent trust statements. The value associated with each edge reflects the level of trust in the direction of the edge.

4.3 TRUST ALGORITHMS

Given a social network, information about trust can be provided to users in many ways. In most settings, the goal usually is to recommend to one node how much to trust another node in the network. However, this is not the goal of this dissertation; trust plays only a part in this research which exploits the trust information in the network and incorporate trust algorithms as part of the bigger recommendation model. However, a description of the different trust algorithms is essential to understanding their functionality in the presented model.

There are two different types of trust algorithms: *global* and *local*. *Global algorithms* compute a general trust value for each person in the network, without any form of personalization from the perspective of the source. Hence, regardless of who asks for a trust recommendation, the same value is provided to everyone since each destination has a universal trust value. On the other hand, *local trust algorithms* calculate trust from the

perspective of the person asking for the trust recommendation (source) and therefore the results are personalized (and may be different) for each user.

Since trust is a personal opinion, it can be affected by many factors and thus it is reasonable for it to vary between two people. Based on that notion, it is only logical to predict that the personalization of trust, through a local algorithm, should improve the accuracy of the results. The calculation of the local trust relies heavily on the properties of transitivity and composability. Transitivity provides the ability to pass down information from the destination all the way back to the source, while composability provides the source with the power to combine the received information from multiple paths.

4.4 LOCAL TRUST VS. GLOBAL TRUST

A trust metric is a measurement that can be used to predict the trust level in unknown users. There are different techniques for accomplishing this task [53][84] but typically it involves controlled aggregation and/or propagation of trust over the trust network. In accordance to how trust algorithms are categorized, trust metrics are also classified into global and local ones [93][158]. *Local trust metrics* reflect the subjective opinions of the source user when predicting the trust he places in unknown users, while *global trust metrics* compute a single trust value that approximates how much the community as a whole trusts a specific user (the destination), independent of any specific user opinion. Usually such a global trust value can be thought of to reflect a user's *reputation* [123].

For example, in the social network of Figure 4.1, a global trust rating for Richard would aggregate (probably average) the trust values issued from Mary and Jenny

resulting in a global trust value of 0.45 that is provided to any user in the network asking about Richard's reputation. However, a local trust metric is used to predict the trust that each user could express towards Richard. So, if we average trust along the path leading from the source user to the destination, Bill's predicted trust in Richard would be 0.35 while Jack's predicted trust in Richard would be 0.55.

PageRank is probably one of the most popular global trust metrics available [111]. It is one of the algorithms used behind the Google search engine to generate a single rank for each Web page independently of the browsing user's preferences. Although local trust metrics can be more precise and adaptive to a user's personal views and opinions, they are however computationally more expensive since they need to be computed for every single user whereas global ones are generated once for the whole community. Nevertheless, a major advantage of using local trust metrics is that they can be attack-resistant [84] by excluding malicious users from trust propagation and thus and they do not influence the results of a user's personalization. Gori and Witten [57] show that malicious exploitation of links is an immanent and unavoidable problem for global trust metrics. In addition, the increasing popularity of link-farms worsens the problem for global trust metrics. The majority of trust metrics proposed in the literature are global, however there are a few local trust metrics such as the ones proposed by Golbeck [51], Ziegler [158], and Massa and Avesani [94].

There are certain scenarios where it is practical or useful for the system to favor one type of trust over the other. For example, in non-critical systems it may be cost and time effective to consider global trust rather than wasting resources personalizing every single trust recommendation, especially in social networks with a huge number of users. On the

other hand, there are cases where the accuracy of predicted trust affects the users' trust in the system's performance, thus the system favors to personalize the trust to gain users' confidence.

In this dissertation, it is believed that the two trust types are equally important as they provide different insight about the trust information in the network. The two trust types do not conflict with one another and therefore the proposed model incorporates the two trust types to complement each other. To be more realistic, the influence of the local trust in this research is higher than the influence of the global one to reflect the importance of personalization in the presented model.

4.5 Dealing with Trust in Computer Science

4.5.1 Peer-to-Peer Systems

Peer-to-peer (P2P) file sharing networks are similar to social networks in the sense that a peer is connected to another if they had interacted at least once. Thus, trust plays a major role in these networks.

Nejdl [105] used trust to describe access control policies in P2P networks. If an agent can show proof that it meets the requirements in the access control policy, then the agent would be trusted to access information. For a P2P system to work, each node must correctly implement the network protocols and provide access to uncorrupted files. If a node is not reliable (i.e. points to a corrupted file or does not conform to the policy), it can affect the usefulness of the entire network. Thus, the trustworthiness of an agent reflects the quality of its participation in the P2P network. In an attempt to filter out the bad nodes from the network, several contributions in the literature have addressed the

issue of predicting the trustworthiness of unknown nodes. The EigenTrust algorithm [76] expresses trust as a function of corrupt files as opposed to valid files that the node provides. A peer continuously updates the trustworthiness of other peers with which it has interacted based on the proportion of good files it has received from that peer so far. The EigenTrust algorithm calculates trust as a variation of the PageRank algorithm [111] and it generates a globally accepted trust rating over a series of iterations using a matrix representation of the trust values. EigenTrust has been proven to be highly resistant to attack. There are other approaches to manage trust and reputations in P2P networks such as [75] in which they focus on how to share trust assessments in a distributed way.

There is a major difference between trust in P2P networks and trust in social networks. In P2P networks, trust is based on the dependability of a node to conform to perfectly correct parameters. Thus, trust can be viewed as having a binary value of 0 or 1 because a file is either corrupted or it is not; there is no such thing as a *slightly corrupted* file. However, in social networks two people may hold extremely different opinions about a topic (such as religion or politics), therefore there is no absolute truth to determine which one should be trusted and which one should not; a person decides how much to trust another based on personal opinion. In P2P networks the calculated trust provided by one peer represent the absolute truth for all peers. As a result, the need for personalization of a trust rating is minimized because each peer is expected to have the same experience as every other peer.

It is worth noting that the use of trust in this dissertation is as a measure of reputation and not as one that leads to actions such as eliminating nodes from the system.

4.5.2 Public Key Infrastructure

Trust has been used in the Public Key Infrastructure (PKI) in a similar manner to the way it is used in social networks. In order to execute secure transactions, it is crucial to map a name to a public key or, conversely, to find the public key associated with a certain user. In the absence of a centralized authority to map keys and names, the process of authentication can be accomplished by combining information from a path of authorities. The reliability of the chains may deteriorate if any of the intermediate authorities has poor information. Trust values can be combined over paths of authorities to determine the confidence in the authority at the end point. Several researchers have discussed the metrics for calculating trust over such paths including Tarah and Huitema [145], Mendes and Huitema [98], Maurer [96], and Reiter and Stubblebine [121].

The inputs and outputs used with these metrics could vary depending on the algorithms, approaches, and applications. For example, Maurer computes the confidence ratings and combines them with explicit trust statements and authenticity measures to infer authenticity information. Although his approach can be considered indirect and complex, others like Beth, Borcherding, and Klein's metric [16] used a simpler technique that take as input binary trust values, the source, and the destination, and generate a calculated trust value as output.

4.5.3 Online Communities

On the web, trust has ignited a lot of issues relating to security, authentication, and digital signatures. However, there have been substantial contributions that focus on the social aspects of trust. Levin *et al.* [85] used the Advogato website as a benchmark for

their research on trust metrics. Advogato is a community discussion board and resource for free software developers where a set of authoritative users collectively calculate the trust ratings for the other users using a network flow model. Levin's trust metric constitutes certifications between members to determine the trust level of a person, and eventually their membership within a group. Access to post and edit website content is controlled by these certifications. Advogato is considered a global trust algorithm since the authoritative nodes are used to make calculations for every user, however it can be modified to carry out personalized calculations by using a single authoritative node, which converts the metric into a local one.

Richardson *et al.* [127] use trust-based social networks to calculate a user's belief in a statement by finding possible paths from the user (source) to any other user that holds an opinion about the statement in question. Trust values are concatenated along each path to produce a recommended belief in that statement. The values are then aggregated from the different paths to calculate the final trust value for the statement. They deliberately did not define a specific concatenation function for calculating trust between individuals, preferring to present a general framework as their main result. Grishchenko [60] adds to the work of Richardson *et al.* by addressing some issues and presenting applications related to their work.

The problem with algorithms such as Kamvar [76], Zeigler and Lausen [159], and Richardson *et al.* is that they are all based on finding the principal eigenvector, which means that trust must be normalized first to function within the matrix. The normalization would affect the trust values because they will be dependant on the number of ratings that user has issued. If the user has made many trust ratings then the normalized trust value

will be lower than if he had rated only a few people. Other researchers, like Guha *et al.* [61] shifted their focus to study distrust and whether it can be propagated and inferred like trust. In their work they converted continuous ratings to binary values representing the two polar extremes of trust and distrust.

Since the above contributions focus on utilizing trust in TBSN, they can be considered to align perfectly with the work done in this dissertation. In fact, they were the starting point for exploring the possible contributions in the early stages of this research.

CHAPTER 5

RECOMMENDER SYSTEMS

5.1 INTRODUCTION

Recommender Systems (RS) are one of the most popular applications used in web-based social networks. RS are useful software tools and techniques that suggest to users items that may be of interest to them [88][122][21] and therefore assist users in online communities in making decisions about those items. The items could be movies, music, articles, books, or even jokes. It would be rare to find a single RS that suggests a variety of items to users (such as Amazon.com); most RS focus on a certain item type to be suggested (e.g. Netflix.com) because that greatly determines the system's design, user interface, and the recommendation technique. Many users interact with RS on a daily basis without being aware of what goes on at the back-end. Websites such as Amazon, Netflix, YouTube, and TripAdvisor rely internally on a RS to deliver the right information to the users requesting (or needing) it.

The motivation behind developing RS stems from the observation that users tend to rely on recommendations provided by others to assist them in making basic daily decisions. For example, before deciding on which movie to watch, many people prefer to read reviews provided by others or by relying on critics' reviews in the local newspaper to aid their decision. Even at a bookstore, some feel overwhelmed when it comes to picking up a book to read and they end up purchasing one of the best sellers since they seem to be popular among the general public.

5.2 Recommender Systems Evolution and Functionality

Early RS were designed to benefit from the ratings provided by other users in the system to produce a list of recommended items to an active user, i.e. the user requesting recommendations. Such lists are usually aggregated from lists of items liked by users similar to the active user. The term *similar users* refers to users that have a similar rating pattern for items rated by the active user. The rationale behind this approach is that if the active user agreed in the past on the ratings of certain items with specific users, then the ratings provided by these similar users should be close to the active user's taste and preferences. Such systems are known as collaborative filtering (CF) RS.

When e-commerce websites started to emerge and gain popularity, there was a strong demand for an infrastructure that can filter the large repository of available items in order to guide the users (or customers) to easily locate the items they are interested in. Despite the efforts made by developers at the time, users still found it difficult to choose the item that would best suite their needs from the wide range of available alternatives.

RS can be thought of as being the most effective solution for providing an adaptive web environment for users overwhelmed by the increased online information overload on e-commerce websites. RS have the ability to quickly filter the information to suite the needs of different users and therefore to personalize a user's experience on the web. This personalization process results in different users receiving different item recommendations that match their tastes. Non-personalized recommendations have always been available and are fairly easy to generate. Such recommendations can be in the form of the top *n* best-seller books, top *m* music tracks that have been downloaded, or the most read news articles during that week. Although such lists may be useful for a

certain category or group of users, they are not the focus of RS as they do not provide any form of personalization for users.

In their simplest form, RS provide lists of suggested items, but the difference between such lists and the ones just mentioned is that these lists are generated to match a user's preferences and criteria, hence providing a personalized experience. Users' preferences can either be expressed explicitly by the user, such as rating items in the system, or inferred implicitly by interpreting the user's actions, such as navigating to a certain product page or purchasing an item.

As the variety of information available on the web started to rapidly grow, especially with the wide spread of e-commerce websites and services, users had the tendency to make poor decisions simply because they were overwhelmed. In psychology that can be thought of as the implication of having too much freedom to select from the available options which turns later into a misery-inducing tyranny [132].

RS have become so sophisticated over the years and they can be relied upon to direct a user to the items that are of utmost relevance to his needs or preferences. The added advantage of RS is that they most often introduce the user to new unexplored items that the user would most probably never come across if he had to navigate through the items himself. Regardless of the different approaches or techniques applied to implement the various types of RS, they all share the same basic core function: analyze the user's profile, which may include his preferences, needs and rating patterns, then compile a list of items that suites the user's needs by utilizing the various types of knowledge and data about the users and items. In some RS, the user has the freedom to browse through the recommendations and may or may not accept them. Sometimes, the user's feedback is

collected (either in an implicit form such as purchasing the same item again, or in an explicit manner such as when the user rates the item based on his experience with it). Such feedbacks are useful for a RS because it allows it to gain more knowledge about the user's preferences and thus to fine-tune the future recommendations to better suite the user's taste.

5.3 RECOMMENDER SYSTEMS ROLES

When someone is first introduced to RS, they can be falsely tricked into thinking that they only serve a single purpose: easing a user's mission for finding certain items or services. But after a deep understanding of how RS are implemented and used, it immediately becomes evident that they have two major roles; the RS's role on behalf of the service provider and the RS's role on behalf of the user. For example, a travel services website would utilize a RS in a way that guarantees that it maximizes its turnover by selling more hotel rooms or increasing the number of travelers to a certain unpopular destination [124]. But a user of such a system is not aware of the service's goals and is only concerned about booking a hotel room within a certain budget or exploring and finding interesting destinations.

In fact, there are several reasons to use RS from both the service provider's perspective and the user's perspective. But ultimately, a RS must balance between the two sides' needs in order to provide a valuable service for both.

From a service provider's point of view, the following sums up the most important reasons for it to benefit from such a technology [126]:

- *Increase the number of sold items*

  This can be considered the most important function of a RS for a service provider. For a commercial RS, its ultimate goal is to provide a user with items that closely match what he is looking for in order to increase the user's probability of purchasing the item. Even non-commercial websites are interested in increasing the traffic on their site, although they do not gain profit from a user's selection of a recommended item. So in both cases the ultimate goal from a service provider's point of view is to increase the conversion rate, i.e. the number of users that accept the recommendation and consume an item, compared to the number of site visitors that simply browse through the items available.

- *Sell a variety of items*

  Another major advantage for a service provider to use a RS is to increase users' exposure to unpopular items. For example in a movie renting RS, the service provider is interested in profiting from renting all the movies available in its repository and not just the popular ones. Therefore, by utilizing a RS such movies can be suggested to the appropriate users based on their known needs and preferences.

- *Increase user satisfaction*

  A well-designed RS can produce reliable, accurate, and useful recommendations for its users, which in turn increases the user's satisfaction with the system and therefore increases the probability of using the system more frequently and accepting the recommendations.

- *Increase user fidelity*

  Due to the nature of RS and how they work, the system takes advantage of the acquired information from past user interactions which allows the system to treat a returned user as a valuable one by providing more accurate recommendations. As a result, the longer a user uses the system, the more refined his profile becomes which leads to producing a set of more customized recommendations that match his taste.

- *Gain a better understanding of users' needs*

  A RS has the ability to describe a user's preferences, either by explicitly collecting them or by inferring them from recorded actions. Service providers can benefit from the increasingly growing knowledge about the users and therefore improve the management of its items or services provided. For example, a travel services website can gain a better understanding of what its users usually look for when booking a vacation. Such an understanding aids the system in providing the correct advertisement to the proper users, which in turns improves users' satisfaction with the system's ability to meet their needs.

However, from the user's side, a RS can be useful for different reasons and purposes [67], such as:

- *Finding some good items*

  Most RS provide the user with a list of items that the user will most likely find interesting. Usually such lists are sorted by their ranks, where items with a high probability of being liked appear higher on the list. Some systems even augment

the items on the list with the prediction of how much the user would like them (predicted rating). Typically, such ratings usually fall within a scale of 1 to 5.

- *Finding all good items*

  In some systems, it is desirable and even crucial for the RS to generate all the possible items that match the user's needs. This is usually true in systems where the number of items is relatively small and in critical systems such as the ones used in the medical and financial fields. In these systems, the user can even benefit from any extra information the RS can provide him with, such as the rationale behind displaying these items or the user's search criteria or profile specification that resulted in retrieving each item.

- *Annotation in context*

  Within a specific context the RS can further highlight the items that closely match the user's preferences based on the user's long-term transaction or preference history. An example on that would be an electronic program guide that can emphasize or highlight the shows that would be worth watching, based on the user's profile.

- *Recommend a sequence*

  Some RS do not simply recommend an item to a user, but rather keep recommending items afterwards that fall within the same context of the user's previously recommended items. Examples include recommending a compilation of music tracks or recommending a book about the same topic that a previously

recommended book was about. The sequence of recommended items usually includes items of the same type (books, music tracks, movies, etc.) as suggested by Shani *et al.* [133] and Hayes *et al.* [65].

- *Recommend a bundle*

  In some systems, the recommendations are not presented as single items, but rather offered as a package of different items that can fit well together. For instance, a travel services website may present the user, based on his preferences, with a bundle composed of a suggested airline or flight, a certain hotel, a rental car, and offers for some of the attractions in the desired destination [125].

- *Just browsing*

  A user may simply want to browse the repository of items available without any intention of making a purchase afterwards. The RS should be able to understand the user's purpose behind browsing the catalog in that session to better assist him in meeting his browsing needs.

- *Test a recommender's credibility*

  Some users are on the fence when it comes to trusting a RS to provide them with recommendations that they could trust. Those users usually prefer to play around with the system in order to test it and get a feel of its recommendation ability. Some systems provide separate functions for those users to allow them to test the system's behavior in addition to the basic recommendation functionality.

- *Improve profile*

  A RS's ability in allowing the user to provide as much information about their preferences as they desire is a crucial aspect that should never be neglected. A user of the system is expected to understand that the more input they provide to the system, the stronger the benefit gained from the recommendations in subsequent sessions. Otherwise, if the system has no or little knowledge about the user's likes or dislikes, then it will not be able to provide a personalized recommendation and thus the recommendation would be composed of the most popular items or the items that appeal to an average user of the system.

- *Express self*

  For some reason, some users do not care about the recommendation aspect of the system, but rather get a lot of satisfaction by providing the system with their ratings and feedback about the items, thus feeling the freedom to express their opinions and beliefs. Those users can still be useful for the system because as they start feeling connected to it, this will lead to increasing traffic to the service provider's website.

- *Help others*

  Some users feel obliged to help others by providing their ratings and reviews about their recent experiences with the recommended items. Those users are very important in systems in which users are not expected to use that often, such as a car dealer's RS. A user that provides feedback about his recently purchased car knows that this feedback will not be likely used by the system to guide him in his

future car purchase, but would be rather more helpful for other customers.

- *Influence others*

  There are cases in which users are using the system with the sole purpose of affecting the decisions taken by other users. Although not always the case, but most of the times those are malicious users who are either trying to promote the sales, usage, or popularity of a specific item, or are trying to discourage the users and drive them away from certain items.

5.4 DATA AND KNOWLEDGE SOURCES

Due to the different roles that a RS can play within an information system, there could be several sources for the information collected during the recommendation process. Typically, knowledge and data are collected from information about the items, users, and the users' interactions with the system (i.e. transactions). Not all RS are expected to exploit all the knowledge sources available. In its simplest form, a RS may only use the item ratings to generate the recommendations. On the other hand, advanced RS are more dependent on the available knowledge such as users' demographical information, the context of items, and even the users' transaction patterns. Follows is a detailed explanation of the three main knowledge sources for RS.

5.4.1 Items

The items are the objects that are recommended by the system. They can be represented as a complex unit with a set of attributes describing the item, or simply referred to using a handle or a single ID code. Items with low complexity and value

include DVDs, movies, music tracks, news articles and books. Items with higher complexity and value are mostly electronics such as cameras, laptops and cell phones. The items with the highest level of value possible are jobs, financial investments, and travel packages, just to name a few [101].

Regardless of its complexity, an item will always have a value in the system. Useful items for a user are considered to have a positive value while useless items that the user wrongfully selects are considered to have a negative value. It is worth noting that when a user is trying to find an item, he may incur two types of costs: a cognitive cost associated with the time and effort spent to locate the item, and a monetary cost in the case where items must be purchased.

A RS may make use of several item attributes to further increase its understanding of the items' structure and thus provide better recommendations. For instance, a movie RS may use information about the movie's genre along with information about the director, actors, and year of production to increase the movie's value in the system, i.e. increasing the movie's chances of having a positive value when recommended to a user.

5.4.2 Users

The main purpose of any RS is to provide personalized recommendations to its users. In order to accomplish that, the system needs to build and exploit a user profile so that it can assist it in making useful recommendations. The profile in its simplest form consists of the different item ratings provided by the user. In more complex systems the profile may be composed of several attributes obtained from demographical information (age, gender, education, etc.), behavioral patterns (browsing pattern in a web-based RS, travel search history in a travel services RS, etc.) and/or relationships between the users in the

form of trust (whether it is explicitly defined by the users or implicitly inferred from the interaction history among the users).

5.4.3 Transactions

The term refers to the interactions that occur between the users and the system, stored in the form of a log. The logs include useful information for the recommendation process such as the items selected by a user, the captured description of the recommendation request, and/or the feedback provided by the user about their experience with the selected item in the form of a rating or review.

Item ratings can be considered the most important piece of information in the log. It may be explicitly provided by the user or implicitly implied or calculated by means of the system's analysis of the user's interaction history with the item. Schafer *et al.* [131] indicates that ratings can be specified in several forms, such as:

- Numerical ratings usually in the scale from 1 to 5
- Ordinal ratings in which the user is asked to indicate his opinion, i.e. whether he *strongly agrees*, *agrees*, *neutral*, *disagrees*, or *strongly disagrees* with the system's recommendation of the item.
- Binary ratings in which the user either "agrees" or "disagrees"

5.5 RECOMMENDATION TECHNIQUES

To implement and reach the goal of any RS, a system must have the capability to predict the items that would appeal to the user's needs and taste by analyzing the items and their usefulness for that user in order to be used in the system's prediction process.

This approach is considered generic as it can be applied to simple RS as well as complex ones. Simple RS may lack knowledge about the users' preferences or descriptions about the items. In such systems the most popular items are recommended since they would have a higher probability of appealing to an average user, as opposed to recommending a random item. The process of figuring out the most popular items that were chosen by other users can be considered as a form of *analyzing* the items to determine the most suitable ones. The analysis process is more obvious though in complex systems that have access to users' profiles, preferences, and/or description of the items.

There have been several attempts to categorize RS but the most widely used taxonomy is the one proposed by Burke [21], which differentiates between six different categories of recommendation techniques:

5.5.1 Collaborative Filtering Recommender Systems

The collaborative filtering (CF) technique is the most popular approach among RS researchers [131]. The basic CF techniques recommend to a user items that were liked by other users with a similar taste. The similarity in taste between two users is computed by comparing their rating history. We can consider users sharing similar rating profiles to be part of a neighborhood, thus having a strong correlation between the users (neighbors), which is sometimes referred to as *user-user similarity*. This neighborhood-based approach can also be applied on the item level as well in order to recommend items similar to other items liked by the same user, i.e. *item-item similarity*. Figure 5.1 illustrates an example of a simple CF technique that uses user-user similarity while Figure 5.2 presents the same example using item-item similarity.

	i_1	i_2	i_3	i_4	i_5	…	i_m
u_1	5	3	4	4	?	…	2
u_2	3	1	2	3	3	…	1
u_3	4	3	4	3	5	…	5
u_4	3	3	3	5	4	…	4
…	…	…	…	…	…	…	…
u_n	3	1	3	2	1	…	1

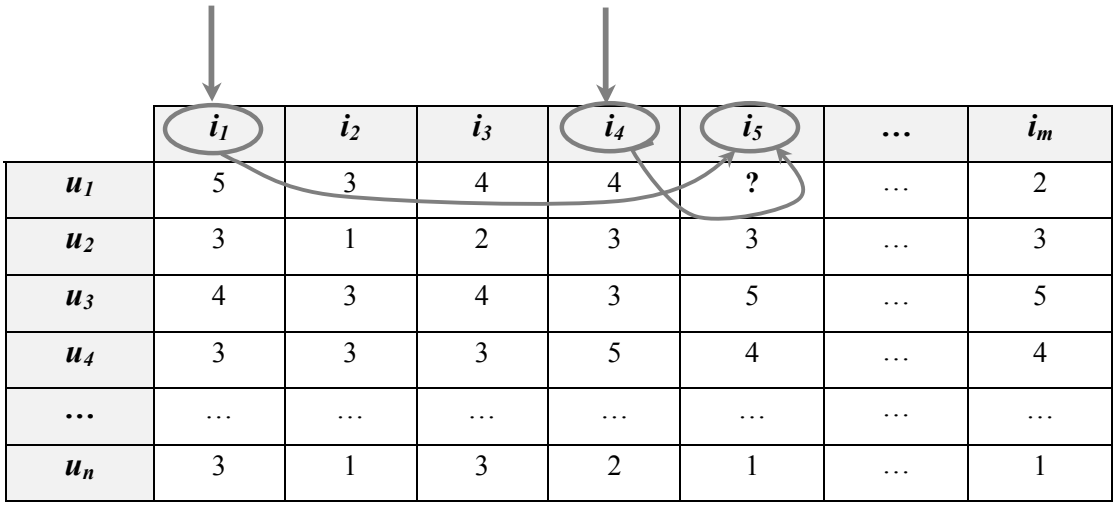
Figure 5.1: The user-item ratings matrix in a simple CF RS using user-user similarity. User-user similarity is used to predict the ratings for unrated items. To predict the rating of item i_5 for user u_1, the user's rating profile is compared to other users' profiles and a similarity measure is calculated. For example, if the Pearson Similarity is used then u_2 and u_4 would be the most similar to u_1 and the predicted rating for i_5 is calculated using the similar users' ratings for that item.

By applying a neighborhood-based approach, *nearest neighbor* algorithms gained popularity in the area of RS due to their simplicity and ability to provide personalized recommendations with good accuracy, yet it is worth noting that they have their share of downsides such as data sparsity and coverage issues.

5.5.2 Content-Based Recommender Systems

This type of RS attempts to learn a user's preferences by analyzing the items rated by him in the past and then recommending items that are similar to his highly rated items. For example, if there's a tendency for a user to give high ratings for songs by a certain artist, then the system will recommend unrated songs in the future by the same artist since they would have a high probability of being liked by that user.

In such systems, access to information about the items, in addition to users' ratings of items, is crucial for the recommendation process.



	i_1	i_2	i_3	i_4	i_5	…	i_m
u_1	5	3	4	4	?	…	2
u_2	3	1	2	3	3	…	3
u_3	4	3	4	3	5	…	5
u_4	3	3	3	5	4	…	4
…	…	…	…	…	…	…	…
u_n	3	1	3	2	1	…	1

Figure 5.2: The user-item ratings matrix in a *simple* CF RS using item-item similarity.

Item-item similarity is used to predict the ratings for unrated items. To predict the rating of item i_5 for user u_1, the item's rating profile (across the users) is compared to other items' profiles and a similarity measure is calculated. For example, if the Cosine Similarity is used, i_2 and i_4 have a similar rating profile to i_5's across all users so the predicted rating for i_5 is calculated using the similar items' ratings for the target user u_1.

5.5.3 Demographic Recommender Systems

Such systems have access to demographical information associated with each user and therefore suggest items that appeal to a certain demographic. For example, certain recommendations could be based on the age of the user, while others rely on the user's location to provide proper suggestions, such as the restaurant reservation website OpenTable.com. Little attention has been given to this technique in RS research [89].

5.5.4 Knowledge-Based Recommender Systems

By having access to domain-specific knowledge, knowledge-based techniques exploit how well certain item features can match a user's needs and preferences and to what extent can an item be useful for a user. Two of the most popular knowledge-based

systems are *case-based* systems and *constraint-based* systems. In both, the user explicitly provides the system with his requirements as an input, the system provides suggestions to the user to resolve conflicting requirements, and an explanation is provided for why the suggested items were recommended. In *case-based* systems [20][125], the system estimates how well a recommended item matches a user's needs by means of a similarity function. The similarity score is then used to determine the usefulness of the item to the user. *Constrained-based* systems are more complicated since predetermined rules define how to relate a user's requirement with an item feature. Such rules make up the knowledge base, which is utilized by the system in the recommendation process. Knowledge-based systems must have a learning component embedded in them to guarantee their successful functionality.

5.5.5 Community-Based Recommender Systems

These RS rely on a user's social network to provide the proper recommendations. The basic principle behind community-based systems is that people tend to prefer recommendations provided by their friends (even if they have different tastes) rather than recommendations given by similar anonymous users [138]. With the increased popularity of WBSN in the past years, these RS gained a lot of interest from researchers in the field, hence referring to them as social-based systems [53]. Several attributes can be used to construct a social network between users, such as the number of interactions (e.g. emails exchanged) or the explicit expression of trust among the users.

5.5.6 Hybrid Recommender Systems

All of the above techniques have their pros and cons; therefore there were attempts to combine multiple techniques within a single system in order to take advantage of one while fixing the flaws of another. For instance, CF techniques suffer from the cold-start problem, i.e. the system's inability to provide recommendations for items that have no ratings or for users that did not rate enough items. Yet, a content-based system does not suffer from this problem since the recommendations are based on an item's features (rather than its number of ratings). So, it is obvious that by creating a hybrid system that combines both CF and content-based techniques, it can overcome such a problem while benefiting from the advantages provided by both techniques [21].

The presented work in this dissertation is considered a hybrid RS because it incorporates features inspired by CF techniques in addition to using a community-based RS where trust is chosen to define connections between users. This research refers to such RS as trust-based recommender systems (TBRS) to emphasize the role of trust in the recommendation process.

5.6 RECOMMENDER SYSTEMS APPLICATIONS

In addition to the theoretical contributions of recommender systems, researchers focused on their commercial applications with an emphasis on the practical aspects of the implementation of these systems. These aspects affect different stages in the RS's life cycle, such as the design, implementation, and maintenance.

In the design phase, these aspects include factors (such as the application's domain) that determine the choice of algorithmic approach that should be applied.

Montaner *et al.* [101] classify existing RS applications with respect to specific application domains as:

- *Entertainment* - recommendations for movies, music, TV shows.
- *Content* - personalized newspapers, recommendation for documents, recommendations of Web pages, e-learning applications, and article recommendations.
- *E-commerce* - recommendations for consumers of products to buy such as books, cameras, PCs, beauty products, etc.
- *Services* - recommendations of travel services, recommendation of houses to rent, or matchmaking services.

The increased popularity of recommender systems expanded the scope of their possible advantages when applied to new applications, such as recommending friends or tweets to follow. Therefore, the above classification can be considered as an initial taxonomy of the existing types of RS application domains, since it is not expected to cover the new domains that are constantly being explored and added.

In order to select the proper recommendation algorithm for a certain domain and to design an effective user interface, it is crucial for a RS developer to understand the domain's specific characteristics, requirements, challenges, and limitations. In addition, it is important to analyze the available knowledge sources, which can greatly affect the algorithm choices as well.

5.7 RECOMMENDER SYSTEMS EVALUATION

The process of evaluation is required during the different stages of any system's life cycle [21][4]. For instance, evaluation is necessary during the design phase of a RS to ensure the selection of the proper recommendation approach. Usually offline evaluations are conducted by running several algorithms on the same dataset and comparing their performance to the actual values obtained from user interactions (i.e. ratings). Such evaluations are typically performed on either existing public benchmark data (if available), or on collected data. Bailey [12] stresses on the importance of carefully designing offline experiments in order to ensure reliable results.

Evaluation is also required after the system has been launched. The algorithms might be very accurate in predicting user ratings, but for some reason the system may not be accepted by users because the system's performance is not as expected. In such cases, it is usually useful to perform an online evaluation with real users of the system and analyze the system logs to further enhance the system performance. In addition, most of the algorithms include parameters, such as weights, thresholds, and number of neighbors that require constant adjustment and calibration regardless whether the evaluation is done online or offline.

There are cases though where an online evaluation is too risky or not feasible. Therefore, the evaluation process would require a focused user study, where a controlled experiment is planned and a small group of users are asked to perform different tasks with different versions of the system. Later, questionnaires are distributed among the users to reflect on their experience. By analyzing the performance and feedback, the system's quantitative and qualitative information can be collected and summarized.

Ideally, it would be beneficial to evaluate the implementation of existing real systems to determine their applicability, constraints, and challenges with respect to the new system. Unfortunately, many commercial RS owners are not willing to share their implementation or practice insights since it may give their competitors an advantage by revealing their trade's secrets. The same obstacle is faced with benchmark data, as some owners are also unwilling to share their data or user interactions' details even if it is for academic purposes. This was one of the main problems in this research since it was hard to find a dataset that shares both item ratings and trust information among users. Many dataset owners feel that sharing trust information would compromise user confidentiality. The Epinions dataset was the only publically available dataset that matched the needs of this research at the time of conducting the experiments for this dissertation.

CHAPTER 6

DATA MINING TECHNIQUES FOR RECOMMENDER SYSTEMS

6.1 INTRODUCTION

Recommender Systems apply techniques and methodologies inspired by the ones used in related areas such as Human Computer Interaction (HCI) and Information Retrieval (IR). Typically, these techniques can be perceived as an instance of a Data Mining (DM) technique. In general, a DM process consists of three steps: *Data Preprocessing*, *Data Analysis*, and *Results Interpretation* as depicted in Figure 6.1. Follows is an overview of how the techniques used in each step can be useful in the study of RS.

6.2 PREPROCESSING TECHNIQUES

The term *data* generally refers to a collection of *objects* and their *attributes*, where an attribute is a property or a characteristic of an object. Other common names for an object include item, record, point, sample, instance, or observation. An attribute is sometimes referred to as a variable, feature, characteristic, or field.

Real-life data is rarely used in DM techniques in its raw format, but rather needs to be transformed into a usable format. Specifically, three main issues need to be taken into consideration when designing a RS; namely similarity measures, sampling techniques, and dimensionality reduction.

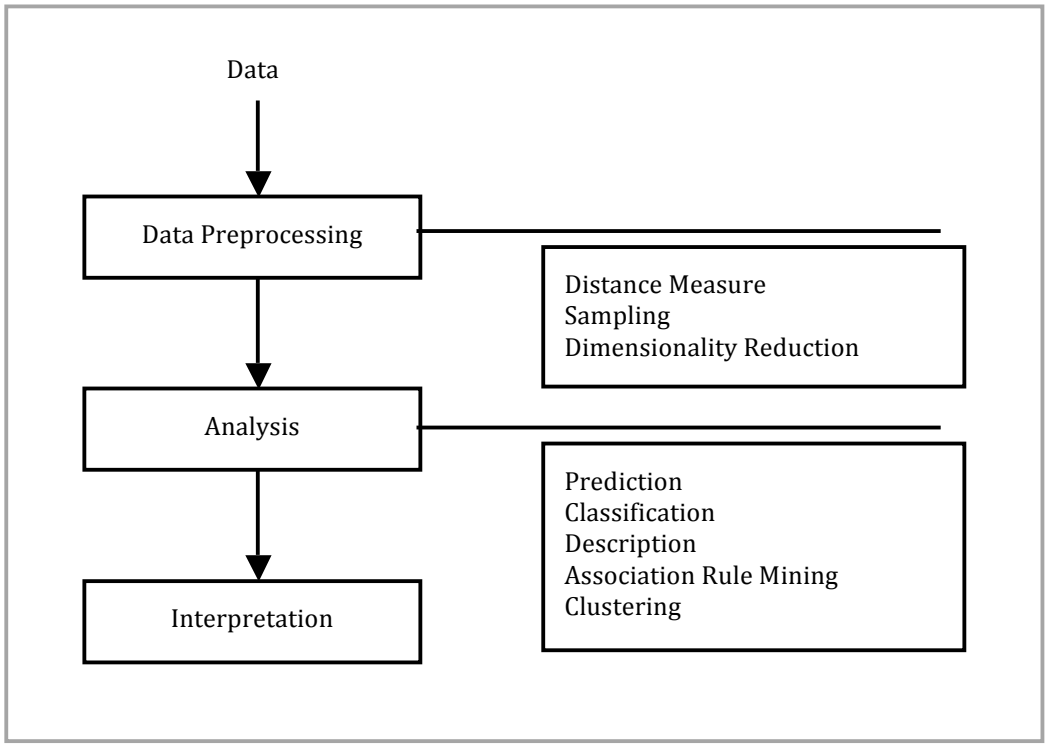


Figure 6.1: Main steps and methods in a data mining problem.

6.2.1 Distance and Similarity Measures

Most classifiers and clustering methods rely on defining a proper similarity or distance measure. Perhaps the simplest example of a distance measure is the *Euclidean Distance*, which is defined as:

$$d(x,y) = \sqrt{\sum_{k=1}^{n}(x_k - y_k)^2} \tag{6.1}$$

where n is the number of dimensions (or attributes) and x_k and y_k refer to the k^{th} attribute of the data objects x and y, respectively.

The *Minkowski Distance* is a generalization of the Euclidean Distance of the form:

$$d(x,y) = \left(\sum_{k=1}^{n}\left|x_k - y_k\right|^r\right)^{\frac{1}{r}} \tag{6.2}$$

where r is the degree of the distance. In fact, the Minkowski Distance has specific names

depending on the value of r. When $r = 1$, it is usually known as the *city block*, *Manhattan*, *taxicab*, or *L1 norm* distance. The Euclidean Distance is the special case when $r = 2$. As $r \rightarrow \infty$ it is known as the *supremum*, L_{max} *norm*, or L_{∞} *norm* distance, which is usually used to compute the maximum difference between any dimensions of the data objects.

Another commonly used distance is the *Mahalanobis Distance* that is defined as:

$$d(x,y) = \sqrt{(x-y)^{\sigma^{-1}}(x-y)^{T}} \tag{6.3}$$

where σ is the covariance matrix of the data.

A widely used approach is to use the *Cosine Similarity* by representing the items as document vectors in an n-dimensional space then computing the similarity as the cosine of the angle that 2 vectors form:

$$\cos(x,y) = \frac{(x \bullet y)}{\|x\|\|y\|} \tag{6.4}$$

where the $\bullet$ operator refers to the vector dot product and $\|x\|$ is the norm of vector x. Sometimes the Cosine Similarity is referred to as the L2 Norm.

It is also common to measure the similarity between items by their *correlation*, i.e. the measurement of the linear relationship between them. The *Pearson Correlation* is probably the most commonly used correlation coefficient, among the many that may be applied. It is computed by:

$$Pearson(x,y) = \frac{\sum(x,y)}{\sigma_x \times \sigma_y} \tag{6.5}$$

where Σ is the covariance of data points x and y, and σ is their standard deviation.

The most commonly used similarity measures in RS have been the *Cosine Similarity* and the *Pearson Correlation* (or any of their many variations). However, depending on

the context of the RS and its data, any of the other distance measures can be applied. Spertus *et al.* [140] conducted a study in which they evaluated six different similarity measures applied to the Orkut social network. Although their results are biased by the context of their study, but they concluded that the best recommendation results were obtained when the Cosine Similarity was used. A similar study was carried by Lathia *et al.* [82], in which they concluded that in general the prediction accuracy of a RS is not affected by the similarity measure being used. In fact, an interesting observation in the context of their work was that sometimes using a random similarity measure provided better results than those of the well-known approaches.

6.2.2 Sampling

Sampling is one of the main techniques used in both the preprocessing and final data interpretation steps. It typically involves selecting a subset of relevant data from a large set of data. There are several reasons behind the necessity for this step. For example, there are cases where it is too computationally expensive to process the entire dataset. Sampling can be also used to create training and testing datasets where the training dataset is used to learn the parameters or configure the algorithms, while the testing dataset is used to evaluate the configuration obtained in the training phase.

The key factor in sampling is to find a proper subset of the original dataset that can be used as a representation of the entire set. A good representative dataset should have almost the same properties of interest of the whole set. Random Sampling is the simplest sampling technique where each item has an equal probability of being selected, yet there are many other sophisticated approaches. For example, in the *Stratified Sampling* technique, the data is divided into several groups based on specific attributes after which

random sampling is applied independently on each group.

There are two main approaches to sampling in general: sampling *without replacement* and sampling *with replacement*. In the former, when an item is selected to be part of the sample, it is removed from the population. In the latter, an item remains to be part of the population even after being selected for the sample, increasing its probability to be added to the sample more than once.

Basic random sampling without replacement is the most common approach applied. Usually an 80/20 proportion is used when dividing the data into training and testing sets respectively. In other words, random sampling without replacement is used to select 20% of the data for testing while leaving the remaining 80% for training purposes. However, the 80/20 proportion is not a strict standard that must be followed but it is only a common rule of thumb. Usually, any proportion that maintains the training set as more than $2/3^{rd}$ of the dataset would be appropriate.

Sometimes, sampling may cause an over-specialization to a particular division of the training and testing datasets. Therefore, training should be repeated K several times using K different combinations of training and testing datasets in each. After that, the average performance of the K learned models is reported. Such a process is known as *cross validation*. There are many variations to cross validation techniques. For instance, in *repeated random sampling*, a standard random sampling process is carried out K times. In *n-fold cross validation*, the dataset is divided into n folds such that one of the folds is used for testing the model and the remaining $n - 1$ folds are used for training. The process is then repeated n times with each of the n subsamples used exactly once as validation data. Another popular technique is the *leave-one-out* (*LOO*) approach which

can be considered as an extreme case of *n*-fold cross validation. In LOO, *n* is set to the number of items in the dataset resulting in the algorithm being run as many times as the number of data points. In each run, one of the data points is used for testing while the rest are used for training. Usually cross validation techniques provide reliable results when the dataset is very large [68].

When cross validation is used to sample ratings in RS, several issues need to be considered that may bias the sampling process. For instance, we may wish to consider the most recent ratings for sampling since those ratings represent the current trends. Or we may need to impose that the random sampling is done on a per user basis to preserve a certain proportion of ratings per user. Such issues can be considered to relate to the problem of RS evaluation, which is still a rich topic for research and discussion.

6.2.3 Dimensionality Reduction

Two common problems in most RS datasets are sparsity and dimensionality. In other words, most datasets have features that define a high dimensional space with very sparse information in that space. The *curse of dimensionality* greatly affects clustering and outlier detection since the density and distance between points become less meaningful in highly dimensional spaces. Dimensionality reduction techniques can overcome this problem by transforming the original high-dimensional space into a lower-dimensional one.

Sparsity and the curse of dimensionality can exist in the simplest settings in RS since we are likely to have a sparse data matrix with thousands of rows and columns (corresponding to users and items) most of which are zeros or undefined. Therefore,

dimensionality reduction fits naturally in the preprocessing steps of RS. The advantage of applying dimensionality production is that its results can be directly used in the computation of the predicted recommendation values. Thus, many systems designers are encouraged to consider it as part of their RS design process rather than a preprocessing step.

The most widely used dimensionality reduction algorithms in the context of RS are *Principal Component Analysis* (PCA) and *Singular Value Decomposition* (SVD). Both can be used in isolation or as a preprocessing step for any of the other earlier mentioned techniques.

Principal Component Analysis (PCA) [70] is a classical statistical method to find patterns in high dimensionality datasets. PCA can provide an ordered list of components with the largest variance from the data in terms of least square errors, such that the amount of variance captured by the first component is larger than the amount of variance on the second component and so on. The dimensionality of the data can be reduced by abandoning the components with a small contribution to the variance.

Singular Value Decomposition [56] is also a powerful technique for dimensionality reduction, which is similar to PCA. The major challenge in SVD is to find a lower dimensional feature space where the new features represent concepts and the strength of each concept in the context of the collection is computable.

Although current trends seem to prefer using SVD and similar techniques (such as Non-Negative Matrix Factorization) but earlier works in the literature used PCA. For instance, Goldberg *et al.* proposed in [54] the use of PCA in the context of an online joke recommendation system. In their system, Eigentaste, they start by selecting a subset of

items from the standard user-item matrix for which all users had a rating. This new matrix subset is then used to compute the global correlation matrix where a standard 2-dimensional PCA is applied.

SVD is known to being used as a tool to improve CF for some time. Sarwar *et al.* [130] present two different ways to use SVD in this context. One way would be to use SVD to uncover latent relations between customers and products, which is accomplished by first filling the zeros in the user-item matrix with the item's average rating and then normalizing by subtracting the user's average rating. This matrix is then factored using SVD and the resulting decomposition can be used directly to compute the predictions. The second approach is to use the low-dimensional space resulting from the SVD to improve neighborhood formation.

6.3 CLASSIFICATION TECHNIQUES

A classifier is a mapping between a feature space and a label space, where the features represent characteristics of the elements to be classified and the labels represent the classes. A simple example would be a restaurant RS, which can be implemented by a classifier that classifies restaurants into one of two categories (good, bad) based on a number of features that describe it.

There are many types of classifiers, but they can be generalized as being either *supervised* or *unsupervised* classifiers. In *supervised* classification, a set of labels or categories is known in advance and we have a set of labeled examples that make up a training set. In *unsupervised* classification (or clustering), the labels or categories are unknown in advance and the task is to properly categorize the elements according to

some criteria. Some of the major classifiers used with RS include nearest neighbors, decision trees, and rule-based classifiers.

6.3.1 Nearest Neighbors

Nearest Neighbors (NN) are considered instance-based classifiers. They function by storing training records and using them later to predict the class label of unseen cases. A basic example is the *Rote-Learner* classifier, which memorizes the entire training set and classifies only if the attributes of the new record match one of the training examples exactly. A more complex and popular instance-based classifier is the *Nearest Neighbor* classifier (*k*NN) [30]. When the *k*NN classifier is provided with a point to be classified, it finds the *k* closest points (i.e. nearest neighbors) from the training records and then assigns a class label to that point according to the class labels of its nearest neighbors. The underlying idea is that if a record falls within a particular neighborhood where a class label is predominant then it is most likely for the record to belong to that very same class.

The most challenging issue in *k*NN is how to choose the value of *k*. If *k* is too small, the classifier will be sensitive to noise points. Yet if *k* is too large, the neighborhood might include too many points from other classes with no obvious prominent class.

*k*NN classifiers are of the simplest of all machine learning algorithms. Since they do not explicitly build models, they are considered lazy learners. Also, classifying unknown records can be relatively expensive since *k*NN classifiers defer many decisions to the classification step.

Due to its simplicity, NN is one of the most common approaches to CF, and thus in designing RS. In fact, it would be rare to come across an overview of RS that does not

include a discussion about the use of the NN algorithm in the context of RS. The major advantage of this classifier over others is that its concept is closely related to the idea of CF. Finding like-minded users (or similar items) is essentially equivalent to finding neighbors for a given user or an item. Another advantage is related to the fact that the *k*NN classifier is a lazy learner and it does not need to learn and maintain a given model. As a result, the system can adapt to rapid changes in the user-ratings matrix. The downside of this is the constant need to re-compute the neighborhoods and the similarity matrix values (similarity measures). To overcome this specific problem, Amatriain *et al.* [6] proposed a neighborhood model that uses a reduced set of experts as the source for selecting neighbors.

Although the *k*NN approach is simple and intuitive, it has shown good accuracy results and is very accommodating to improvements. As a matter of fact, its supremacy as the *de facto* standard for CF recommendations has only been challenged recently by new approaches based on dimensionality reduction.

6.3.2 Decision Trees

Decision trees are classifiers on a target attribute (or class) in the form of a tree structure [119][128]. The observations (or items) to classify are composed of attributes and their target values. The nodes of the tree can be either *decision nodes*, in these nodes a single attribute-value is tested to determine which branch of the subtree applies, or *leaf nodes*, which indicate the value of the target attribute.

Decision tree induction can be accomplished through many algorithms, such as *Hunts Algorithm*, CART, ID3, C4.5, SLIQ, and SPRINT. The *Hunt algorithm*, which is one of the earliest and easiest to understand, is a recursive algorithm that relies on the test

condition applied to a given attribute that discriminates the observations by their target values. Once the partition induced by the test condition has been found, the algorithm is recursively repeated until a partition is empty or all the observations have the same target value. Decision tree induction usually stops once all observations belong to the same class. However for practical reasons, most decision trees implementations use pruning by which a node is not split any further if the number of observations in the node are below a certain threshold.

The main advantages of building a classifier using a decision tree is that it is inexpensive to construct and it is extremely fast at classifying unknown instances. Another important advantage of decision trees is that they can be used to produce a set of rules that are easy to interpret while maintaining an accuracy that is comparable to the other basic classification techniques.

Decision trees may be used in a model-based approach for a RS. One way to do it is to use content features to build a decision tree that models all the variables involved in the user preferences. Bouza *et al.* [18] implement this idea to construct a decision tree using semantic information available for the items. The tree is built after the user has rated only two items, where the features for each of the items are used to build a model that represents the user ratings. They used the information gain of every feature as the splitting criteria.

It is only logical to determine that it is very difficult and unpractical to build a decision tree that tries to explain all the variables involved in the decision making process. However, decision trees may be used to model a particular part of the system, as demonstrated by the work of Cho *et al.* [25] and Nikovski *et al.* [107]. Another option to

utilize decision trees in a RS is to use them as a tool for item ranking, which has been studied in several settings [11][24].

6.3.3 Rule-Based Classifiers

Rule-based classifiers classify data by using a collection of "`if`… `then`… " rules. The rule *antecedent* or condition is an expression made of attribute conjunctions. The rule *consequent* is a positive or negative classification.

A rule *r covers* a given instance *x* if the attributes of the instance satisfy the rule condition. The *coverage* of a rule is defined as the fraction of records that satisfy its antecedent. The *accuracy* is defined as the fraction of records that satisfy both the antecedent and the consequent. A classifier is said to contain *mutually exclusive rules* if the rules are independent of each other, i.e. every record is covered by at most one rule. A classifier is considered to have *exhaustive rules* if they account for every possible combination of attribute values, i.e. each record is covered by at least one rule.

One of the advantages of rule-based classifiers is that they are extremely expressive since they are symbolic and operate with the attributes of the data without the need for any transformation. Rule-based classifiers are easy to interpret, easy to generate, and they can classify new instances efficiently. However, it is very difficult to build a complete recommender model based solely on rules. This is probably the main reason behind this method not being very popular in the context of RS. Deriving a rule-based system requires explicit prior knowledge of the decision making process, but a rule-based system can be useful for improving the performance of a RS by feeding the system with partial domain knowledge or business rules.

Anderson *et al.* [7] implemented a music CF RS that improves its performance by applying a rule-based system to the results of the CF process. For instance, if a user gives an album by a given artist a high rating, the predicted ratings for all other albums by this artist will be increased. Basu *et al.* applied in [14] an inductive approach using the Ripper system [26] to learn rules from data. They report slightly better results when using hybrid content and collaborative data to learn rules compared to a pure CF approach.

6.4 CLUSTERING TECHNIQUES

Clustering, which is also known as unsupervised learning, consists of assigning items to groups such that the items within each group are similar to one another [64]. In other words, the items in the same groups are more similar than items in different groups. The main objective of clustering is to find meaningful groupings that exist within the dataset. Similarity between items is regulated by distance measure, such as the Euclidean distance or the Mahalanobis distance. The goal of a clustering algorithm is to minimize intra-cluster distances while maximizing inter-cluster distances.

A major problem facing CF classifiers when being scaled is that the amount of distance computations (when using *k*NN for instance) drastically increases. One of the possible solutions to overcome this problem is to reduce the dimensionality, but even with that approach we may still have many objects for which the distances have to be computed. This is where clustering algorithms become useful because they can greatly improve the efficiency by reducing the number of operations that needs to be carried. However clustering techniques are unlikely to increase the accuracy of the results, unlike dimensionality reduction techniques. Hence, if a RS designer decides to apply a

clustering technique, attention must be paid to weigh the trade off between the decrease in accuracy and the increase in the performance efficiency.

There are two main categories of clustering algorithms: hierarchical and partitional. *Partitional clustering* algorithms divide data items into non-overlapping clusters such that each data item can belong to exactly one cluster. *Hierarchical clustering* algorithms successively cluster items within found clusters, producing a set of hierarchically nested clusters that can be organized as a hierarchical tree.

An ideal clustering algorithm would consider all possible partitions of the data and produce the partition that would minimize the objective function. Usually an objective function for a clustering algorithm measures the quality of the clustering. Clustering is not an easy problem and finding the optimal solution is often impossible. In fact it is considered an *NP*-hard problem, which is the reason why many clustering algorithms employ heuristics. Usually the decision to apply a particular clustering algorithm and the choice of parameters, such as the similarity measure, rely on many factors that mainly stem from the characteristics of the data. The *k*-means clustering algorithm and its variants are the most commonly used clustering techniques in RS.

6.4.1 *k*-Means

k-Means is a widely used partitional clustering method. The function partitions the N items of a dataset into k disjoint clusters (subsets) S_j, such that each cluster contains N_j items that are as close to each other as possible (in accordance to some distance measure). Each cluster in the partition is defined by its N_j members and by its centroid λ_j. The *centroid* for each cluster is defined as the point to where the sum of distances from all

items in that cluster is minimized. Therefore, the *k*-means algorithm can be defined as an iterative process to minimize *E*:

$$E = \sum_{1}^{k} \sum_{n \in S_j} d(x_n, \lambda_j) \tag{6.6}$$

where x_n is a vector representing the n-th item, λ_j is the centroid of the items in S_j, and d is the distance measure. The *k*-means algorithm moves items between clusters until E cannot be minimized any further. The algorithm works by randomly selecting k centroids then each item is assigned to the cluster whose centroid it is the closest to. The centroid of each cluster needs to be updated to reflect the addition and removal of items and the membership of moved items needs to be updated as well. This process continues until there are no further items to be moved between clusters. In practice, it has been noticed that most of the item convergences to their final partitions take place during the initial iterations of the algorithm, thus the stopping criterion is usually changed to one where a relatively few number of items change their clusters, rather than waiting for all items to converge, to improve efficiency.

The basic *k*-means algorithm is a very simple and efficient algorithm. Yet, it has its share of shortcomings. For example, in order to choose the appropriate number of clusters k, prior knowledge of the data is required. Also, the selection of the initial k centroids has a major influence on the final clusters reached. Outliers greatly affect the algorithm, not to mention that *k*-means has some limitations when it comes to producing clusters of different sizes or densities.

Xue *et al.* [154] illustrate a typical use of clustering in the context of a RS by applying the *k*-means algorithm as a preprocessing step to assist in neighborhood formation. In their work, they do not restrict the neighborhood to the cluster the user

belongs to but rather use the distance from the user to different cluster centroids as a pre-selection step for the neighbors. They also proposed a smoothing technique in which missing values for users within a cluster are replaced by cluster representatives. Their results show that their approach performs slightly better than standard *k*NN-based CF. Similarly, Sarwar *et al.* [129] present an approach to implement a scalable *k*NN classifier in which they partition the user space by applying the bisecting *k*-means algorithm and then they base the neighborhood formation process on these clusters. As expected, their results show a decrease in accuracy of around 5% as compared to standard *k*NN CF, but their method provided a significant improvement in efficiency.

Connor and Herlocker [27] shifted their focus to clustering items instead of users. They tested different algorithms using the Pearson Correlation similarity measure. Their clustering improved efficiency, but all of their clustering techniques resulted in accuracy and coverage that are worse than a non-partitioned baseline.

6.4.2 *k*-Means Alternatives

- *Density-based clustering algorithms*

  DBSCAN [126] is an example of a density-based clustering algorithm, which utilizes the density as the number of points within a specified radius. The algorithm defines three kinds of points: *core points* that have more than a specified number of neighbors within a given distance; *border points* having fewer than the specified number but belong to a core point neighborhood; and *noise points* which are neither core nor border. The algorithm iteratively removes noise points and performs clustering on the remaining points.

- *Message-passing clustering algorithms*

These are graph-based clustering methods that instead of staring the algorithm with the centroids as seeds, initially consider all points as centers, called exemplars. During the execution of the algorithm, the points exchange messages until clusters gradually emerge. *Affinity Propagation* is an example of such algorithms [46] which defines two types of messages between points (nodes): *responsibility*, which reflects how well-suited a receiving point is to serve as an exemplar of the point sending the message, considering other potential exemplars in the process; and *availability*, which is sent from a candidate exemplar to the point and reflects how appropriate it would be for the point to choose the candidate as its exemplar, considering support from other points that are choosing that same exemplar. Affinity propagation has demonstrated very good results in DNA sequence clustering, face clustering in images, and text summarization.

- *Hierarchical clustering*

  As mentioned earlier, hierarchical clustering generates a set of nested clusters organized as a hierarchical tree (known as a *dendogram*). A major advantage of hierarchical clustering over partitional ones is that it does not need to specify a particular number of clusters in advanced. Any desired number of clusters can be extracted by selecting the tree at the proper level. In addition, hierarchical clusters can also sometimes correspond to meaningful taxonomies. Typically, hierarchical algorithms use a similarity or distance matrix and merge or split the clusters one at a time. There are two main approaches to hierarchical clustering: agglomerative and divisive clustering. *Agglomerative hierarchical clustering* starts with the points as individual clusters then at every step merges the clusters that are the

closest to each other until only one cluster (or k clusters) remains. On the other hand, *divisive hierarchical clustering* starts with one cluster that includes all points then at each step splits a cluster until each cluster contains a single point (or there are k clusters left).

It is worth noting that k-means alternatives are rarely applied to RS because of the simplicity and efficiency of the k-means algorithm. So far, density-based and hierarchical clustering algorithms have not shown any signs of usefulness in the context of RS. However, message-passing algorithms have proved to be more efficient and that they can be easily translated to many RS problems.

6.5 Association Rule Mining

Association Rule Mining aims at finding rules that can predict the occurrence of an item based on the occurrences of other items. When two items are found to be related this indicates a co-occurrence, not a causality. It is common to confuse this technique with rule-based classifiers.

An *itemset* is defined as a collection of one or more items.. A k-itemset is one that has k items. The frequency of a given itemset is known as the *support count* and the *support* of the itemset is the fraction of transactions that contain it. A *frequent itemset* is an itemset with a support that is greater or equal to a minimum support threshold, *minSupThreshold*. An association rule is an expression of the form $X \Rightarrow Y$, where X and Y are itemsets. In this case the *support* of the association rule is the fraction of transactions that have both X and Y. On the other hand, the *confidence* of the rule is how often items in Y appear in transactions that contain X.

So, given a set of transactions *T*, the goal of association rule mining is to find all rules having *support* ≥ *minSupThreshold* and *confidence* ≥ *minConfThreshold*. A brute-force approach would list all possible association rules, computes the support and confidence for each rule and then gets rid of rules that do not satisfy both conditions. However, this is computationally expensive. Thus, a two-step approach is usually adopted where first all frequent are generated then high confidence rules from each frequent itemset are generated.

Association rule mining has been effective in uncovering patterns and driving personalized marketing decisions for some time [5]. Although there is an obvious relation between this method and the goal of a RS, yet they have not become conventional yet. Probably this is due to the fact that association rule mining is similar to item-based CF but with less flexibility since it requires an explicit notion of transactions (co-occurrence of events in a given session). Despite all that, there have been several attempts in the literature to incorporate them into RS.

For instance, Mobasher *et al.* [100] presented a web personalization system based on association rules mining. Their system identifies association rules from page views co-occurrences based on users' navigational patterns. Their method outperformed a *k*NN-based recommendation system with respect to both precision and coverage. Lin *et al.* [86] designed a new association-mining algorithm that adjusts the minimum support of the rules during mining in order to obtain an appropriate number of significant rules. Their measured accuracy outperformed previously reported values for correlation-based recommendation. Cho *et al.* [25] combined Decision Trees and Association Rule Mining in a web shop RS. In their system, they derived association rules in order to link

related items, then a recommendation is computed by intersecting association rules with user preferences. They later tracked the association rules in different transaction sets such as purchases, basket placement, and click-through. They also applied a heuristic for weighting rules coming from each of the transaction sets, such as giving purchase association rules higher weights than click-through association rules.

CHAPTER 7

ANT COLONY OPTIMIZATION

7.1 INTRODUCTION

The field of ant algorithms focuses on models derived from real ants' behavior to solve a variety of optimization problems. Recommender systems are viewed as an optimization problem since the objective is to enhance the quality of recommendations by aiming to reach and utilize as much of the available information in the system as possible. Ant algorithms have only been recently applied to RS but they have not been considered to deal with trust-based recommender systems. The novelty of this dissertation is that it presents successful models based on ant algorithms to solve the recommendation problem in trust-based social networks.

Ant algorithms is a family of algorithms that belongs to swarm intelligence methods, which are based on the collaboration between independent, decentralized, self-organizing agents that can lead the system to an emergent *intelligent* solution. The behavior of these artificial agents is usually one that stimulates behavior observed in nature, such as the behavior within colonies of ants, schools of fish, flocks of birds, or herds of land animals.

Ants in nature are self-organized and highly coordinated in their colonies. Many different characteristics of the behavior of ants, such as foraging, division of labor, and brood sorting, inspired a variety of algorithms that are referred to as ant algorithms. The ants are capable of coordinating their activities via *stigmergy*, which is a form of indirect communication accomplished by modifying their environment. For instance, an ant

foraging for food would deposit a chemical on the ground, otherwise known as a *pheromone*, which would indicate to other ants that this path is a good one and thus increasing the probability of other ants reaching that food source. Biologists have concluded that stigmergic, indirect communication allows ants (and other social insects) to achieve self-organization. Therefore, any ant algorithm can be based on a form of *artificial stigmergy* to coordinate work in societies of artificial agents.

This dissertation focuses on one of the most popular ant algorithms, *ant colony optimization* (ACO), which mimics the foraging behavior of ants in their colonies to solve discrete optimization problems.

7.2 Mimicking the Behavior of Real Ants

7.2.1 Foraging Behavior of Ants

Early research on ants' behavior showed that most of the communication among individuals, or an individual and the environment, is facilitated by the use of chemicals produced by ants, known as *pheromones*. Ants mark paths from found food sources to the nest creating pheromone trails. Other foraging ants can detect the pheromone on these trails to lead them a food source. Since more than one trail can be marked, ants tend to probabilistically choose the paths with strong pheromone concentration.

This ant behavior was investigated in a controlled environment by Deneubourg *et al.* [32] who conducted an experiment, known as the double bridge experiment, using one food source and one ant nest and connected them using a bridge with two branches of varying lengths, as depicted in Figure 7.1. Initially, ants were choosing one of the two

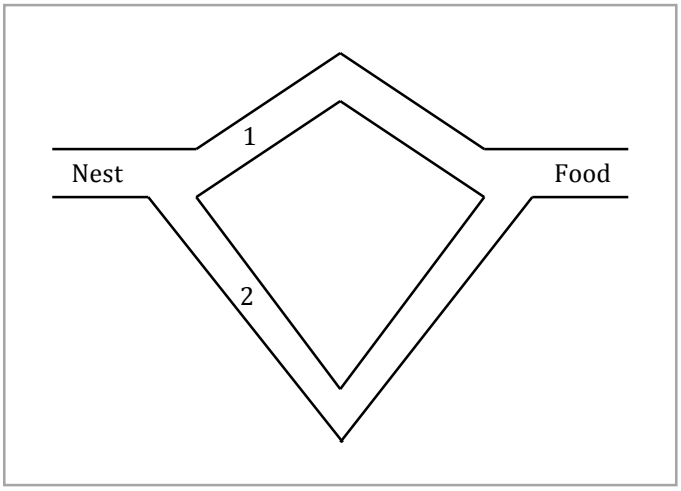


Figure 7.1: Experimental setup for the double bridge experiment
Experiment uses two branches of different lengths.

branches randomly since there is no pheromone (or the amount of pheromone is very low) and thus the ants would not have a certain preference, i.e. both branches would have the same probability. However, as time passes ants deposit more pheromone while crossing the branches, so the pheromone starts to accumulate faster on the shorter branch leading to more ants favoring it over the other branch until eventually the ants converge to that branch. This process of providing positive feedback illustrates the ants' self-organizing behavior.

An interesting trait about pheromones is that they evaporate with time. Although their evaporation rate is low, but they can encourage ants to forget the suboptimal paths found, especially in the early stages of the search process and thus avoids a rapid convergence of the algorithm towards suboptimal areas of the solution space. In short, pheromone deposit promotes *exploitation* of good paths while pheromone evaporation forces ants into the *exploration* of other areas in the solution space.

7.2.2 Artificial Ants

The double bridge experiment is a simple example that proves that ants have built-in optimization capabilities. These capabilities are evident from their use of probabilistic rules based on local information to find the shortest path between two points in their environment. Using this as an inspiration, Dorigo [41] was able to design artificial ants that, by moving on a graph model representing an environment, can find the shortest path between two nodes.

In such a setup, at each time step each ant moves from its current node to a neighboring node at a constant speed of one unit of length per time unit. As a result, ants add one unit of pheromone to edges they cross. Just like real ants, artificial ants move on the graph by choosing a path connecting to one of the neighboring nodes probabilistically.

Usually, artificial ants have a limited memory that can retain information regarding the partial paths that have been followed so far along with the cost of edges that have been crossed. According to this information, ants can dynamically adjust their behavior to build paths representing optimal solutions. Utilizing such a memory enables ant colony optimization algorithms to solve a wide range of optimization problems such as finding the minimum cost path between two nodes.

There are different variations to how ants behave and deposit pheromones in ant algorithms. In some models an ant deposits the pheromone as it is building the path while in others an ant deposits the pheromone on the path on its way back to the nest.

An ant can determine the cost of its solution by using the information in its memory. By evaluating the quality of its solution, an artificial ant can modulate the amount of

pheromone to be deposited. By incorporating the solution quality into the pheromone update function, subsequent ants can be strongly directed towards better solutions. Usually, shorter (or higher quality) paths get a higher amount of pheromone deposited on them.

Pheromone evaporation is modeled in ACO by carefully applying an evaporation rule. Pheromone evaporation is crucially important because it reduces the influence of the pheromones that were deposited in the early stages of the search. Hence, it allows ants to forget the suboptimal paths found, and to discover new and better paths.

7.2.3 Path-Searching Behavior

To copy the behavior of real ants leaving their nest to forage for food, several artificial ants, *K,* are dispatched from the source node (nest) to find a good solution to the problem (food source). Each ant builds its solution step-by-step, where at each step the ant k_i reads local information found on the current node and/or on the edge connecting the node to one of its neighbors. This information is used by the ant to probabilistically decide which node to move to next. At the very beginning of the search process, a constant amount of pheromone is assigned to all the edges in the graph. When positioned on a node x, an ant k_i computes the probability of moving to a neighboring node y by using the amount of pheromone on the connecting edge, denoted by τ_{xy}. This probability is computed in general as:

$$p_{xy}^{k_i} = \frac{\tau_{xy}^{\alpha}}{\sum_{z \in N_x^{k_i}} \tau_{xz}^{\alpha}} \tag{7.1}$$

where p^{ki}_{xy} refers to the probability of ant k_i moving from node x to node y, N^{ki}_x is the neighborhood of ant k_i at node x, and α is a parameter to control the influence of the

pheromone. Note that the neighborhood N^{ki}_x does not include nodes that cannot be part of the solution at this point, even if there are edges connecting to them from the current node. There are several reasons for not including certain nodes in the solution path; one reason could be because the node has already been added to the path and cycles are not allowed in the solution. Also note that different ACO algorithms have different variations of this probability.

An ant repeatedly moves from one node to another in the same manner until it reaches the destination node or a dead end. When a certain destination (target node) is to be reached by all ants, the ants travelling on shorter paths will reach that destination faster. Usually several trials (iterations) are run within a single round of the algorithm to allow the system to properly converge to the optimal solution.

7.2.4 Pheromone Update

Depending on the ACO algorithm variation being used, ants update the pheromone level on crossed edges at different times (after crossing an edge, after reaching the destination, etc.). Regardless of the timing of the pheromone update, an ant k_i deposits an amount of pheromone $\Delta\tau^{ki}$ on the edges of its constructed path. The pheromone level τ_{xy} on an edge connecting node x to node y is updated as follows:

$$\tau_{xy} \leftarrow \tau_{xy} + \Delta\tau^{ki} \quad (7.2)$$

So by applying this pheromone update rule, the pheromone level on the edge xy is increased which increases the probability of later ants crossing this edge.

The choice of the amount of pheromone to be deposited $\Delta\tau^{ki}$ is important and is usually dictated by the context of the problem at hand. In some cases, it would be sufficient to use a constant value $\Delta\tau$ to be used by all K ants. In other cases, the amount

of pheromone can vary by calculating the amount as a function of the edge length or weight. When $\Delta\tau^{ki}$ is calculated as a function of any of the available information, the main issue to pay attention to is that the function should be a non-increasing function of the parameter(s) used to ensure that, for example when edge length is used, shorter paths have higher pheromone levels than longer ones. Different ACO algorithms have different approaches and rules for updating the pheromone level on edges but they generally conform to the same process.

7.2.5 Pheromone Evaporation

Pheromone evaporation is an important trait of ACO algorithms because it can be considered as an *exploration mechanism* that avoids the quick convergence of ants to suboptimal solutions. As a matter of fact, decreasing the amount of pheromone on edges favors the exploration of different paths during the overall search process by forgetting the errors or poor choices made in the early stages and allowing continuous improvement of the learned problem structure.

Evaporation causes the pheromone level on edges to decrease at an exponential speed. In ACO, the pheromone evaporation may be interleaved with pheromone deposit (depending on the variation used). For example, in algorithms where the pheromone evaporation occurs after updating the pheromone level on an edge, the evaporation follows the general form:

$$\tau_{xy} \leftarrow (1 - \rho)\ \tau_{xy} \quad (7.3)$$

where $\rho \in (0, 1]$ is the pheromone evaporation coefficient. Just like pheromone deposit, each ACO algorithm has it own approach, rules, and equations for pheromone evaporation that abide by the general evaporation guidelines.

7.2.6 Setting the General Parameters of ACO Algorithms

Correctly tuning and setting the parameters is essential to the success of any algorithm, and ACO is not any different. In ACO, the parameters to be set include: the number of ants K, the number of iterations t, the initial level of pheromone on edges τ_0, the evaporation coefficient ρ, the influence parameters such as α, in addition to algorithms-specific parameters such as the search depth d.

Intuitively, increasing the number of ants provides better approximation and convergence to the optimal solution. However, each additional ant added increases the computational overhead not to mention that once the ants start to converge to the optimal path(s), any subsequent ants will not significantly contribute to reaching the optimal solution. On the other hand, using a small number of ants would cause fluctuations in path choices in the initial iterations, which could lead to strong enforcement of suboptimal paths. Therefore, a proper choice for the number of ants should accomplish a balance between the computational complexity of the system and quality of the solution.

The amount of pheromone $\Delta\tau^{ki}$ deposited on edges is crucial in ACO algorithms because it affects the speed of convergence to the optimal solution. The initial pheromone level τ_0 in ACO algorithms is usually set using a carefully chosen constant or using a pre-calculated value obtained from running a quick sub-optimal path construction algorithm [42], such as the nearest neighbor algorithm. Regardless whether a constant or a calculated value is used, the pheromone level on all the edges is always initialized using the same value τ_0. The reason behind having to carefully choose τ_0 is that if the initial pheromone values are too low then the search will quickly be biased by the first paths generated by the ants, which usually leads towards the exploration of inferior zones of the

search space. On the contrary, if the initial pheromone values are too high, then many iterations will be useless while waiting for the evaporation process to reduce the pheromone values and consequently for the effect of pheromone deposited by ants to bias the search.

The number of iterations t, the evaporation coefficient ρ, and the pheromone influence parameter α are usually determined by means of trial and error. Typically, α is set to a value of 1 because using larger values tend to amplify the influence of the initial random fluctuations. ρ is usually set to 0.1 for similar reasons. The maximum number of steps taken by each ant in an iteration, or the search depth d, should be carefully chosen too. Allowing ants to reach deeper levels while constructing the solution would increase the probability of all ants reaching the desired destinations. However, just like the number of ants, this comes at the expense of computational overhead.

7.3 THE ANT COLONY OPTIMIZATION METAHEURISTIC

Combinatorial optimization problems are captivating because they are often simple and easy to state but are very complex and difficult to solve. Many of the problems surfacing in applications are *NP*-hard, i.e. there is a strong belief that they cannot be solved to reach optimality within a polynomially bounded computation time. Therefore to find a practical solution, approximation methods are used to reach near optimal solutions in a reasonable amount of time. Such algorithms are known as *metaheuristics* and they usually use knowledge specific to the problem to build or improve the solution. Formally, a metaheuristic can be defined as a set of algorithmic concepts that can be used to guide or modify other heuristics to produce solutions beyond those normally generated when

searching for local optimality. In other words, a metaheuristic can be seen as a general-purpose heuristic method designed to guide an underlying problem-specific heuristic towards promising regions of the search space containing high quality solutions. In fact, the use of heuristics paved the way for finding such good solutions to hard-to-solve optimization problems in a relatively short time.

Ant colony optimization (ACO) is one of the successful metaheuristics inspired by the behavior of ants in their colonies. In ACO, artificial ants cooperate to find good solutions to difficult discrete optimization problems. The ants communicate indirectly by stigmergy (modifying their environment) thus these simple agents need to have computational resources allocated to them. Good solutions emerge as a result of the agents' cooperative interactions.

ACO can be applied to solve both *static* and *dynamic* combinatorial optimization problems. *Static optimization problems* are the ones in which the characteristics of the problem are set at the time of problem definition and do not change at run time. The *Travelling Salesman Problem* (TSP) is a popular example of such problems [69][83] [120], in which city locations and their relative distances do not change while searching for the solution. On the other hand, *dynamic optimization problems* are defined as a function of certain quantities whose values are determined by the dynamics of an underlying system. At run time, the problem instance changes and the optimization algorithm must adapt to the constantly changing environment. The *Network Routing Problem* is an example of such situations because the data traffic and the network topology change constantly.

As previously mentioned, as ants are walking the graph and constructing solutions, they deposit pheromones along the way. Sometimes nodes and edges may have a heuristic value η associated with them. This heuristic value represents *a priori information* about the problem instance. However, the deposited pheromone during the graph walk encodes information about the ant search process and can only be updated by the ants themselves, thus it can be considered as *posteriori information*. Typically η_{xy} represents the cost (or an estimation of it) associated with the edge *xy*, which can be added to the cost of the other edges, that are used to construct the solution, to determine the overall solution cost. In those cases, there are endless possibilities for encoding the problem heuristics using η_{xy}.

7.4 THE ANT COLONY OPTIMIZATION PROBLEM SPECIFICATION

ACO algorithms are typically applied to problems that can be represented as a connected graph $G = (V, E)$ where the nodes V represent the components of the problem and E represents the edges connecting the components according to the problem-specific information. In ACO, each ant k_i has the following properties:

- It can exploit the graph $G = (V, E)$ to search for the optimal solution.
- It has a memory that can be used to store information about the constructed (partial) path so far. This memory is essential for:
    1. Building feasible solutions that conform to the implemented problem constraints.
    2. Computing the heuristic values η.
    3. Evaluating the solutions found.

4. Retracing the path backward (if needed).

- It has a start state (node) and one or more termination conditions.
- When in a state (node) x, if none of the termination conditions is satisfied then the ant moves forward to the next node y that belongs to the neighborhood ${N^{ki}}_x$ and the new state becomes node y. However, if at least one of the termination conditions is met, then the ant stops.
- It selects a move by applying a probabilistic decision rule (such as Equation 7.1). This rule is a function of:
  1. The locally available pheromone levels.
  2. The heuristic values associated with the nodes and/or the edges.
  3. Information about the current node x and nodes belonging to the neighborhood ${N^{ki}}_x$.
- When adding a component (node) to the current state, it can update the pheromone level τ associated with it or with the corresponding edge.
- Once it builds a solution, it can retrace the path backwards and update the pheromone level on the used components.

Note that the K ants walk concurrently and independently of each other, and although each ant is capable of constructing a solution (that is not necessarily of a good quality), good quality solutions can only emerge as a result of the collective interactions among the ants. This interaction is accomplished by sensing and depositing pheromones on the constructed paths. This model represents a distributed learning process.

7.5 APPLYING ACO TO THE TRAVELLING SALESMAN PROBLEM

Most researchers working on ACO algorithms opt to test their work on the *Travelling Salesman Problem* (TSP) for several reasons. The main reason is that TSP is an important *NP*-hard optimization problem that emerges in many applications. The problem can also be easily defined and understood which makes it suitable for testing ACO algorithms since the algorithm behavior will not be obscured by unknown factors. It also forms a standard testbed for new algorithms to be developed; hence an algorithm that proves its effectiveness on TSP does not usually need further proof of its usefulness.

TSP is defined as a problem faced by a salesman who has to visit all customer cities, starting from his hometown, following a path that represents the shortest route possible to pass by all the cities (only once) and back to his hometown. The problem has been intensively studied in the literature and has attracted a lot of research efforts.

7.5.1 Problem Definition and Representation

The TSP optimization problem can be represented using a weighted graph $G = (V, E)$ with V being the set of n cities (nodes) and E being the set of edges connecting these cities. Each edge $(x, y) \in E$ is assigned a weight d_{xy} that corresponds to the distance between cities x and y. In this example, the distance between any two cities is symmetric regardless of the direction of the walk, i.e. $d_{xy} = d_{yx}$.

The TSP problem has the goal of finding the minimum length Hamiltonian cycle that constitutes a closed tour visiting each node in G only once. A solution to the problem is reported as a cyclic permutation of the cities (or their indices). The position of a city in the permutation is not important at all, but rather the relative order of cities is what matters. Thus, there are n permutations that represent the same solution.

7.5.2 Problem Formulation and Specification

The only constraints in TSP are that all cities must be visited and that each city is visited at most once. These constraints affect the steps taken by each ant since a feasible step would be one moving the ant from its current city to an unvisited city. The pheromone level τ_{xy} reflects the desirability of visiting city y when located at city x. The heuristic information η_{xy} reflects the distance between two cities d_{xy} and is usually set to be inversely proportional to it. Most ACO algorithms for TSP define the heuristic as a function of the distance, such as using $\eta_{xy} = 1/d_{xy}$.

Generally, in ACO algorithms an ant constructs a solution for TSP in a single iteration as follows:

1. Choose an initial city, according to some criterion, for the ant to start its tour.
2. Use pheromone τ_{xy} and heuristic η_{xy} values to probabilistically determine the order of cities to be visited by iteratively adding an unvisited city at each step, until all cities have been visited.
3. Return to the initial city. When all other ants finish constructing their tours, determine, based on the solution quality, the amount of pheromone to be deposited along each (or only good) tours depending on the algorithm variation used. Usually pheromone evaporation occurs afterwards.

This is roughly a high level description that applies to most ACO algorithms for TSP. Each ACO algorithm has its own way of dealing with the different aspects of the process, such as when to update the pheromone level on edges or how to calculate the probability of moving from one node to another. But regardless of the details, the algorithm repeats over several iterations t in which all ants construct their solutions. At the end of each

iteration, the tours are evaluated to determine the best ones found so far and the ants with the best solutions so far further increase the pheromone levels on those trails. At the end of the last iterations, the system converges to the optimal solution found as a result of the collaboration between the ants.

CHAPTER 8

IMPROVING PHEROMONE INITIALIZATION IN ACO ALGORITHMS

8.1 INTRODUCTION

Most, if not all, of the work done in the literature with respect to ACO algorithms and their applications focuses on the details of solution construction, probability calculation, and finding new approaches for improving the way ants communicate and process information. However, pheromone initialization has been neglected by researchers who simply opt to use a constant value for the initialization, whether it has been pre-calculated or estimated. But properly initializing the pheromone level affects the speed of the system's convergence to the optimal solution, and this research believes that this issue deserves more attention than it has been given so far. Before discussing the application of ACO algorithms to trust-based RS, an approach for locally initializing the pheromone level on edges in ant algorithms is presented and the conducted experiments' results highlight the advantages of adopting such an approach in ACO algorithms in general, and consequently in the presented models for this dissertation.

8.2 INITIAL PHEROMONE LEVEL IN TRADITIONAL ACO ALGORITHMS FOR TSP

Typically, an ideal value for the initial pheromone level τ_0 on edges in ACO algorithms would be one that is as close as possible to the average pheromone level

expected to be deposited by an ant k_i on an edge during one iteration. Choosing a very small value will slow the convergence process and may result in the system not getting a chance to reach the optimal solution (but rather a suboptimal one). Initializing the pheromone level using a large value will cause a fast convergence, which means the system will not take its time to explore other possible paths and eventually it will be stuck in a suboptimal solution. Also the choice between whether to use a constant value for τ_0 or to pre-calculate it using a quick suboptimal path construction algorithm [42] affects the quality of the solution. However, in both cases, the pheromone level on all edges in the system is initialized using the same value τ_0.

There are different approaches for pre-calculating a value for τ_0. Even within the context of a certain problem there is no standard approach to be followed. However, the standard application of each ACO algorithm for the TSP has a set of suggested initializations associated with each that have been proven to yield good results. For example, when the *Ant System* (AS) algorithm is applied to the TSP problem, it has been suggested by Dorigo *et al*. [40] to initialize τ_0 as follows:

$$\tau_0 = K/C_{nn} \quad (8.1)$$

where K is the number of ants and C_{nn} is the length of a tour constructed by applying the nearest neighbor algorithm. In *Elitist Ant System* (EAS) [35][41] more parameters are considered in the initialization, such that:

$$\tau_0 = (n + K)/\rho C_{nn} \quad (8.2)$$

where n is the number of cities and ρ is the evaporation coefficient used in the pheromone evaporation process, which in this specific case is usually set to 0.5. *MAX-MIN Ant System* [142][143][141] (MMAS) suggests an initialization that only

Table 8.1: Suggested parameter settings for ACO algorithms when applied to TSP.

ACO Algorithm	α	β	ρ	K	τ_0
AS	1	2 to 5	0.5	n	K/C_{nn}
EAS	1	2 to 5	0.5	n	$(n + K)/\rho C_{nn}$
MMAS	1	2 to 5	0.02	n	$1/\rho C_{nn}$
ACS	1	2 to 5	0.1	10	$1/nC_{nn}$

considers the nearest neighbor tour length in addition to the pheromone evaporation coefficient:

$$\tau_0 = 1/\rho C_{nn} \quad (8.3)$$

Similarly, the *Ant Colony System* algorithm (ACS) uses the number of cities n instead of the evaporation coefficient, as follows:

$$\tau_0 = 1/nC_{nn} \quad (8.4)$$

Table 8.1 summarizes the suggested parameter values used by these algorithms.

8.3 THE LOCAL PHEROMONE INITIALIZATION TECHNIQUE

While reviewing the literature for this dissertation, it was evident that all of the proposed ACO algorithms for the TSP focus on devising new ways for the agents to interact and probabilistically move on the graph while neglecting any attempts to improve the way the basic parameters are initialized. Although the suggested parameters summarized in Table 8.1 have been proven to provide good results, but that does not mean that there is no room for improvement. That was the motivation behind the initial phase of this research in which the possibility of improving the performance of ACO

algorithms by using a new pheromone initialization approach was investigated.

The suggested pheromone initialization techniques in the literature, regardless whether a constant or a pre-calculated value is used, can be considered as what this research refers to as a *global pheromone initialization* technique since the value is calculated once and applied to all edges. However, this research categorizes (and names) the presented technique as a *local pheromone initialization* technique since it calculates the initial pheromone levels locally within each node's neighborhood rather than using a *one-size-fits-all* approach.

In ACO algorithms in order for the system to converge properly in a timely manner, the initial pheromone level on edges should be a value close to what an ant is expected to deposit on an edge during a single iteration. The incentive behind the presented local initialization approach is that since we are looking for a value that is close to what an ant would deposit, and since ants determine the amount of pheromone to be deposited based on *local* calculations that yield different deposited amounts based on the quality of each path, then it does not make sense to initialize all edges using the same value especially knowing that not all edges are of equal quality in the path construction process.

In the remainder of this chapter, the new local pheromone initialization technique is defined and the experiments that were carried are presented along with a comparison to a standard application of the ACS algorithm to the TSP.

8.4 Applying The Local Pheromone Initialization Technique to ACS for TSP

In this research, the ACS algorithm was chosen to experiment with its pheromone initialization when the algorithm is used to solve the TSP. The choice of ACS was not

arbitrary; AS is the first ACO algorithms to be proposed in the literature while ACS is an extension to it that has a better exploitation for the local search and it performs pheromone evaporation and deposit on the best solutions found so far rather than updating all paths, so it would make more sense to consider ACS rather than AS. EAS and MMAS both use an initial pheromone value that is a function of the pheromone evaporation coefficient ρ, which is a parameter that needs to be controlled. If EAS or MMAS were chosen for this experiment it would have been hard to determine whether the results obtained are attributed to the new local initialization technique or to the value used for ρ.

In the new local initialization approach, a *local initialization* technique is followed in which ants initialize the edges locally within each neighborhood before deciding which one to cross (upon their first encounter). The goals are:

1. Use local information from the surrounding neighborhood N^{ki}_x to determine the proper initial pheromone level on the neighboring edges to reduce the effect of external factors (from distant nodes).
2. Avoid initializing edges that are never encountered.
3. Reduce the number of iterations needed (time spent) for the system to converge.

The basic idea behind the new technique is to use local information from the surrounding neighborhood to initialize the pheromone level on each edge xy before calculating the probability p^{ki}_{xy} of crossing the edge for the first time. Therefore an initial local value τ^{0}_{xy} is used to initialize the pheromone level on an edge xy instead of initializing with the constant τ_0. In the new approach, τ^{0}_{xy} is defined as the inverse of the sum of weights (distances) associated with uninitialized edges $y \in N^{ki}_x$ that can be

potentially crossed from the current node (city) x as follows:

$$\tau_{xy}^{0} = \frac{1}{\sum_{z \in N_x^{ki*}} w_{xz}} \tag{8.5}$$

where w_{xz} refers to the weight associated with the edge (distance between two cities) x and z and N^{ki*}_x is the neighborhood of potential nodes, from the perspective of ant k_i, with *uninitialized* connected edges to node x. Note that $N_x^{ki*} \subseteq N_x^{ki}$.

To be more specific, imagine the following scenario: an ant k_i is on node x and wants to determine which node y in its neighborhood N^{ki}_x should it move to next. The probability of moving to any node y involves the pheromone level τ_{xy} as dictated by Equation 7.1 (or any variation of it). If any of the edges in the neighborhood does not have pheromone associated with it, then the ant needs to initialize the pheromone level first before proceeding with the probability calculation.

An ant k_i can face one of three possible cases:

- <u>*Case 1:*</u> None of the encountered edges have been initialized yet.

  In this case, all potential edges within N^{ki}_x are initialized using Equation 8.5.

- <u>*Case 2*</u>: Some potential neighboring edges have *not* been initialized while the rest have pheromone values associated with them.

  When faced with such a scenario, the algorithm only considers the potential *uninitialized* edges in N^{ki*}_x when applying Equation 8.5 to initialize.

- <u>*Case 3*</u>: All potential neighboring edges have associated pheromone level values (i.e. they have already been initialized).

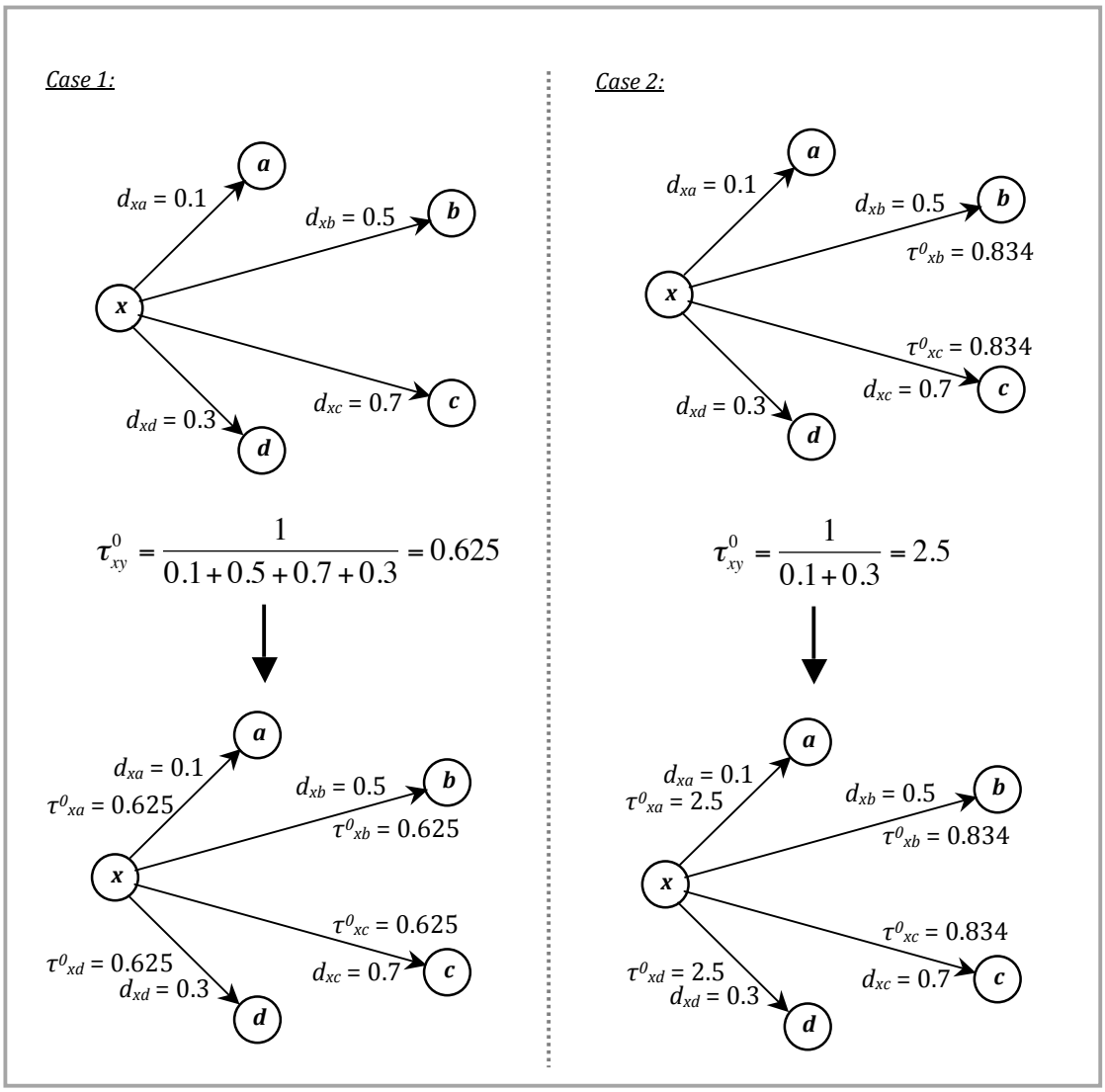


Figure 8.1: Local pheromone initialization example.
Ant k_i is located at node x for the TSP.
Case 1: None of the encountered edges has been initialized so the weight on all edges are used to calculate the initial pheromone level.
Case 2: Some of the encountered edges has been initialized so the weight on the uninitialized edges are used to calculate the initial pheromone level while the initialized edges remain unchanged.

In such a case, the ACS algorithm proceeds with the calculation of the probability p^{ki}_{xy} of moving to a potential neighboring node. Figure 8.1 shows examples corresponding to cases 1 and 2.

Note that in the presented local initialization approach, not all edges will be initialized during the first iteration of the system; it is possible to have pheromone initializations in every iteration until the system converges. It is expected however for the percentage of the initialized edges per iteration to decrease in the final iterations since that's when the system starts to converge to the optimal solution.

8.5 LOCAL PHEROMONE INITIALIZATION FEASIBILITY TESTING

8.5.1 Preliminary Experiments

In order to test the feasibility of the new local pheromone initialization technique, a synthetic TSP dataset with 50 cities and randomly generated distances among them was used. For the sake of comparison, three different algorithms were tested using the local pheromone initialization technique on the randomly generated dataset:

- ACS^{nn}: The ACS algorithm using the length of tour calculated using the nearest neighbor algorithm to initialize τ_0 (Equation 8.4).
- ACS^{avg}: The ACS algorithm using a pre-calculated constant value that initializes τ_0 using:

  $$\tau_0 = 1/(n * avgDist) \quad (8.6)$$

  where n is the number of cities and *avgDist* refers to the average distance between any two cities in the used dataset.
- ACS^{local}: The ACS algorithm initializing τ_0 using the new local pheromone initialization technique (Equation 8.5).

Table 8.2: Summary of preliminary results obtained by applying ACS^{avg}, ACS^{nn}, and ACS^{local} to a randomly generated TSP dataset with 50 cities over 2500 iterations.

Algorithm	No. of ants	Length of best tour found	# of iterations to find best tour	Time (milliseconds) to find best tour
ACS^{avg}	10	1838	784	17515
	20	1870	1582	68562
	30	1884	1603	106875
ACS^{nn}	10	1914	336	9703
	20	1883	1118	60422
	30	1927	1273	97328
ACS^{local}	10	1947	316	8688
	20	2012	2015	101516
	30	1923	850	64672

8.5.2 Analysis of Preliminary Results

A quick glance at Figures 8.2 and 8.3 shows that ACS^{local} has a great potential for improving the results for the TSP. By examining the results in Table 8.2, it can be seen how the local initialization algorithm achieved reasonable results that are slightly worse than ACS^{avg} and ACS^{nn} by increasing the best tour length found by ~1% in the case of 20 ants (ACS^{local} vs. ACS^{nn}). However, ACS^{local} exceeded the performance of the other two algorithms with respect to both the number of iterations needed and the time required to reach the solution. For example, ACS^{local} has a ~2% increase in tour length when compared to ACS^{nn} using 10 ants, but it yielded a ~ 6% reduction in the number of iterations needed and ~10% reduction with respect to time. This analysis is also applicable when comparing ACS^{local} to both ACS^{avg} and ACS^{nn} algorithms using 10 and 30 ants. Probably due to some anomalies, ACS^{local} did not perform well when 20 ants were used.

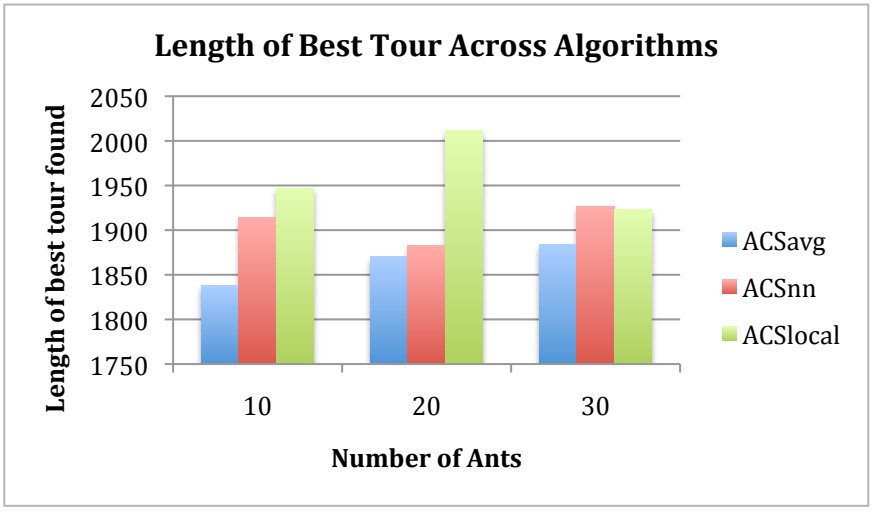


Figure 8.2: Comparing the length of the best tour found for TSP by applying ACS^{avg}, ACS^{nn}, and ACS^{local}.

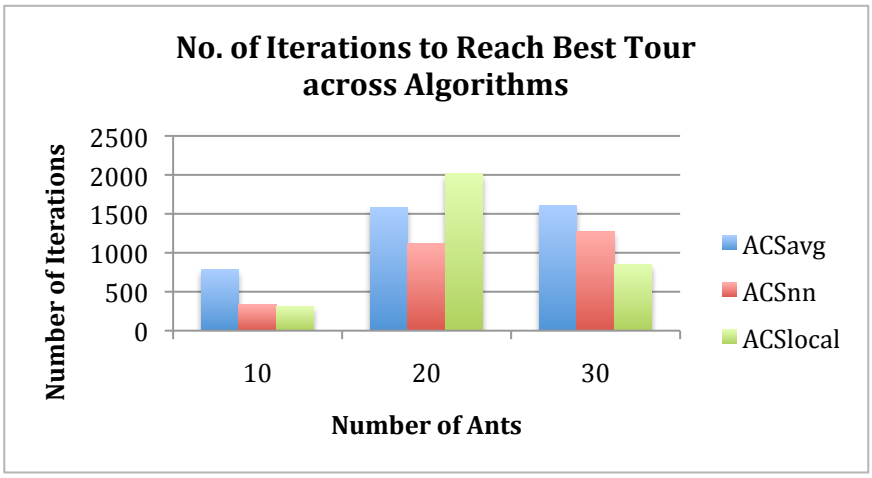


Figure 8.3: Comparing the number of iteration needed to find the best tour for TSP by applying ACS^{avg}, ACS^{nn}, and ACS^{local}.

So, taking into consideration the size of the dataset, the experiment has shown that the local pheromone initialization technique seems to be promising enough to be tested on larger datasets.

8.6 ADDITIONAL EXPERIMENTS AND THEIR RESULTS

Since the new local pheromone initialization technique showed some promise, additional experiments were curried using larger datesets. Table 8.3 compares the results obtained by applying ACS^{nn} and ACS^{local} on 11 different dataset. The datasets provide a variety in size (the number next to each dataset corresponds to the number of cities), which provides a way to monitor the effect of the dataset size on the algorithms' performances. The results highlighted in red in Table 8.3 indicate that ACS^{nn} performed better while numbers highlighted in green mark the datasets in which ACS^{local} performed better.

There was not an obvious trend or parameter that could have affected the results. For example, both datasets *pr76* and *eil76* have 76 cities however ACS^{local} performed better in one while ACS^{nn} generated better results in the other which could be attributed to the range of distances between the cities in the datasets. Overall, ACS^{local} resulted in a better performance (or tied ACS^{nn}) in 8 out of the 11 tested datasets. Also, the percentage of edges that were not crossed and hence were not initialized in ACS^{local} can be further analyzed to extract information that may be useful for improving the performance of the system as a whole. For example note that in *burmal14*, *ulysses22*, and *pr76*, ACS^{local} provided better results while initializing 91% of the edges for the two former datasets and 93% of the edges in the latter.

Based on the results obtained with the extensive experiments carried to validate the feasibility of using a new local pheromone initialization technique when applying the ACS algorithm to the TSP, ACS^{local} has demonstrated the ability to improve the results either by successfully constructing shorter routes. Even in cases in which the algorithm

Table 8.3: Summary of experimental results obtained by applying ACS^{nn} and ACS^{local} to 11 TSP datasets* using 50 ants, β = 2, and 50 iterations.

Database	Algorithm	Min Length	Max Length	Avg Length	Ratio of Initialized Edges
eil51	ACS^{nn}	**430**	551	**500**	100%
	ACS^{local}	443	601	517	95%
kroA100	ACS^{nn}	23122	28885	26047	100%
	ACS^{local}	**22862**	29484	26260	96%
att48	ACS^{nn}	33274	41296	37153	100%
	ACS^{local}	**32564**	41594	**26994**	95%
burmal14	ACS^{nn}	**25**	32.1392	**26.73**	100%
	ACS^{local}	**25**	32.1397	26.8	91%
bayg29	ACS^{nn}	8681	10697	9704	100%
	ACS^{local}	**8445**	10831	**9695**	94%
berlin52	ACS^{nn}	**7311**	9356	**8320**	100%
	ACS^{local}	7477	10135	8778	92%
eil76	ACS^{nn}	**556**	705	**639**	100%
	ACS^{local}	597	791	690	94%
pr76	ACS^{nn}	116437	144605	**131607**	100%
	ACS^{local}	**114064**	156760	135816	93%
st70	ACS^{nn}	696	866	**780**	100%
	ACS^{local}	**690**	893	788	95%
ulysses16	ACS^{nn}	**51**	72	57	100%
	ACS^{local}	**51**	72	**56**	90%
ulysses22	ACS^{nn}	54	75	**59**	100%
	ACS^{local}	**53**	77	60	91%

** http://iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp/*

did not improve the results, the presented local initialization managed to accomplish reasonable results in a shorter time or a fewer number of iterations. Hence, local pheromone initialization in ACO algorithms is adopted as the pheromone initialization approach in this dissertation to increase the accuracy of recommendations in TBRS.

CHAPTER 9

ANT ALGORITHMS IN RS AND PROBLEM DEFINITION

9.1 INTRODUCTION

There have been some efforts in the literature to apply algorithms from the area of swarm intelligence to recommender systems, such as the proposed RS in [4] that uses a particle swarm optimization algorithm and the fuzzy genetic RS presented in [3]. A few attempts involve the use of ant algorithms and there are only a limited number of successful applications. These applications vary from providing recommendations in different types of RS to using ant algorithms for clustering and classification purposes. Up until the day the research of this dissertation was proposed, no one has ever attempted to apply any form of ant algorithms to RS that use explicit trust values among the users (i.e. TBRS).

9.2 RELATED WORK

9.2.1 Ant Algorithms and Recommender Systems

Only a few researchers have attempted to apply ant algorithms to recommender systems in general. As a result, each attempt is completely different from the other in the sense that, for example, some deal with certain RS techniques (content-based, collaborative filtering, etc.) and each applies ant algorithms for a different purpose (clustering, rule-based classification, etc.).

Ant based clustering was first introduced by Deneuborg [32], where in his model ants discriminate between different kinds of items and spatially arrange them within the context of a content-based RS. The proposed approach by Kanade and Hall [77], combines ant based clustering and fuzzy *c*-means. Their model was employed by Nadi *et al.* who presented a fuzzy ant-based recommender system [104] that provides online users with a list of recommendations based on the comparison of the user's navigational behavior with other user's data. Their model combines ACO algorithms and fuzzy logic to generate the recommendations. Sharma *et al.* [136] developed an ACO algorithm called Ant-Recommender with the aim of recommending items within clusters of user profiles. The ants in their algorithm sense pheromone found on clusters rather than on individual paths to determine the best cluster to provide a recommendation from. The ant colony metaphor was also used by Sobecki [139] for selecting optimal solutions in his hybrid recommendation method and by Bedi *et al.* [15] who presented a recommender system based on the collaborative behavior of ants.

9.2.2 Ant Algorithms and Trust

Ant algorithms have been successfully used in several applications dealing with trust. However, there has not been a single application (at the time of this research) that utilizes ant algorithms to deal with explicit trust in TBRS. The only work in the literature that *claims* to do so is Bedi and Sharma's proposed algorithm, Trust Based Ant Recommender System (TARS) [15] in which they allege that their ant algorithm provides recommendations by integrating trust between users while in fact what their model does is compute trust between users based on profile similarity rather than use explicitly stated

trust among users. This dissertation believes that the way they name and refer to their algorithm is misleading.

Some of the recent work proposed in academia that incorporates trust into the application of ACO algorithms include Lin *et al.*'s proposed trust model [86] that adopts an ACO algorithm to simulate trust relationships between cloud entities in cloud computing. Also, Yang *et al.* [155] developed a trust evaluation model in P2P networks where the reputation level of a target peer depends on the trust values on the different recommendation paths. Ant algorithms have been applied to mobile ad hoc networks as well to estimate the degree of trust between nodes [150]. ACO algorithms are ideal for reaching optimality in grid scheduling problems, such as the work presented in [81] in which their ant-based module predicts trust throughout a network grid by forecasting a trust value for each entity to determine its self-protection capabilities as well as its reputation.

ACO algorithms have inspired many solutions in diverse applications, but since it is a newly adopted approach, especially in the field of RS, it makes it hard to find suitable datasets or to compare any significant contribution to similar work in the literature.

9.3 Problem Definition

9.3.1 Problem Statement

The previous chapters covered different areas that fall within the scope of this dissertation, which ties all these areas through its main goal. This research's main purpose is to present a model capable of providing a user with a rating prediction for an unseen/unrated item by utilizing the rating information provided by other users in the

network that are not necessarily directly trusted by that user.

Formally, the problem addressed in this dissertation is stated as:

Obtain an item rating r^x_i *for user* x *with respect to item* i *from users* y *that do not necessarily belong to user* x*'s web of trust* WOT_x.

9.3.2 Model Parameters

This research deals with trust-based recommender systems (TBRS) since trust is incorporated in the prediction process. The parameters the presented model consist of:

- A set of n users, U

  $U = \{u_1, u_2, u_3, \ldots, u_n\}$

- A set of m items, I

  $I = \{i_1, i_2, i_3, \ldots, i_m\}$

- A set of nxm item ratings, R

  $R = \{r^x_i\}, x \in U, i \in I$

- A set of $n(n-1)$ trust values, T

  $T = \{T_{xy}\}, x \in U, y \in U, x \neq y$

- A set of n webs of trust (neighborhoods), C

  $C = \{WOT_x\}, x \in U$

  $WOT_x = \{T_{xy}\}, x \in U, y \in U, x \neq y$

- A set of n popularity values, P

  $P = \{P_y\}, y \in WOT_z \; \forall \; z \in U$

Note that the set R is used to construct the [nxm] *user-item matrix* while C is used to populate the [nxn] *user-trust matrix*. The web of trust of a user x, WOT_x, refers to the set

of users that user x explicitly issued trust statements for. The popularity of each user P_y is pre-calculated as a function of the trust statements T_{zy} that were issued from users z towards user y.

The presented model deals with additional parameters consisting of:

- d, the maximum search depth, i.e. the maximum length of a trust path between any two users x and y.
- $n(I_{xy})$, the number of co-rated items between users x and y such that $y \in WOT_x$.
- PT_s, the path trust of a solution s. The path trust PT_s reflects the propagated trust along the solution path, which is calculated as a function of the trust T_{xy} between adjacent users on that path. The path trust can also be a function of additional parameters depending on the way it is calculated.

The only constraints that are enforced in the model are:

- Only consider solutions with trust path $PT_s \geq PT_{threshold}$ because this research believes that high quality solutions will result from paths with a high trust propagation value.
- Solutions must not exceed a maximum depth d.

CHAPTER 10

T-BAR: A TRUST-BASED ANT RECOMMENDER

10.1 INTRODUCTION

Trust-Based Ant Recommender (T-BAR) is the main presented model in this dissertation. T-BAR is a nature-inspired hybrid algorithm spawned from ACO algorithms for predicting recommendations (ratings) in trust-based recommender systems. The presented model is considered a pioneer approach to applying ACO algorithms to the area of trust-based RS.

T-BAR is a dynamic model based on the probabilistic methodology followed by ACO algorithms. The results achieved prove that T-BAR can produce good results when recommending items to users, while balancing the tradeoff between results accuracy and data coverage. The Epinions dataset is used as a testbed to verify the accuracy of the obtained results.

10.2 TRUST NETWORK AND INPUT REPRESENTATION

The trust network in T-BAR is modeled as a digraph $G=(V, E)$, where the set of nodes V represents users and the set of directed edges E represents the trust statements issued between the users. The values on the edges indicate the issued trust values. In the presented model the input consists of two matrices:

1. [*n*x*m*] *item-ratings matrix* that holds the ratings given by the users to different items in the past, with the *n* rows being the users in the system and the *m* columns representing the items in the system.
2. [*n*x*n*] *user-trust matrix* that holds the trust statements issued between the *n* users in the system with the rows representing the source users issuing the trust statements and the columns being the target users whom the trust statements are issued for.

10.3 T-BAR'S SPECIFICATIONS

T-BAR is a hybrid algorithm derived from both the AS and ACS algorithms proposed by Dorigo *et al*. in [42] and [40]. Follows is a detailed description of how ants behave and communicate in T-BAR, in addition to the specifics of how the system deals with pheromone deposit and evaporation. The discussion includes an explanation of what values are assigned to parameters.

10.3.1 The Artificial Ants and Edge Selection

T-BAR dispatches a predetermined number of *K* ants from the active user (source) with the goal of reaching as many *good* users as possible. *Good users* are users that can be reached through the active user's extended web of trust (i.e. friends of friends) and have a rating *r* for the target item *i*. Each of the *K* ants moves in the network by first calculating the probability p^{ki}_{xy} of crossing the edge connecting the current node *x* to each neighboring node *y* using the following equation inspired by the probability calculation in AS algorithms:

$$p_{xy}^{ki} = \frac{(\tau_{xy})^{\alpha}(\eta_{xy})^{\beta}}{\sum_{z \in N_x^{ki}} (\tau_{xz})^{\alpha}(\eta_{xz})^{\beta}} \quad (10.1)$$

where N^{ki}_x refers to the feasible neighborhood when ant k_i is located at node x, and where y is part of that neighborhood.

Note that the probability p^{k}_{xy} highly depends on two parameters in Equation 10.1: τ_{xy} which is the *pheromone level* on the edge xy, and η_{xy} which is the *desirability of the move* from node x to node y. α and β are both parameters that control the influence of τ_{xy} and η_{xy} respectively. Recall that in complex ACO algorithms, and specifically in ACS, η_{xy} is considered as a priori *desirability* computed by heuristics while τ_{xy} is a posteriori indication about the *goodness* of the move. Once the probabilities p^{ki}_{xy} are calculated the ant k_i moves along the edge that yielded the highest probability.

In T-BAR the ants stop their solution construction either when each ant reaches a set search depth d or if no more edges can be traversed. Once all K ants stop, the process repeats by dispatching the K ants again from the active user (source). The repeated process stops completely after a certain number of iterations t. During the last iteration each ant k_i keeps a record of *good* users that it came across while constructing its last path. Remember that *good* users are those with a rating r for the target item i.

In this dissertation, and after extensive trials and experimentations, the parameters of Equation 10.1 are defined as:

$$\eta_{xy} = T_{xy} \quad (10.2)$$

$$\beta = P_y \quad (10.3)$$

$$\alpha = 1 \quad (10.4)$$

where T_{xy} is the trust value issued from user x to user y, and P_y is the popularity (reputation) of user y computed as the average trust issued to user y by users z that have user y in their *web of trust* WOT_z:

$$P_y = \frac{\sum_{y \in WOT_z} T_{zy}}{n(z)} \tag{10.5}$$

where $n(z)$ is the number of users z that issued a trust statement towards user y. Note that Equations 10.2 and 10.3 represent how both *local* and *global* trust metrics have been incorporated into the presented model respectively, unlike other approaches in the literature that solely depend on one of the two metrics [51][94][123][158].

10.3.2 Pheromone Update Mechanism

In real life, while ants forage for food they deposit pheromone along their path to inform other ants that the path has been discovered. Other ants can smell the pheromone on the paths and subsequently tend to follow, probabilistically, the paths with a higher pheromone concentration. The pheromone build-up on the path leading to a good food source results in all the ants at the end converging to that path. However, an important trait about pheromones is that they evaporate as time passes by and the evaporation mechanism reduces the influence of the pheromone deposited by early ants and favors the *exploration* of new paths rather than *exploiting* the already discovered ones. In other words, the evaporation prevents the ants from converging to poor paths discovered during the early stages of the search.

The pheromone update mechanism in T-BAR interleaves the process of pheromone deposit and evaporation, which is similar to the way it is accomplished in ACS algorithms. In T-BAR, pheromone update is achieved on two levels: a *local* one and on a *global* one.

The *local pheromone update* occurs as each ant k_i traverses an edge xy. The pheromone level τ_{xy} on the edge is adjusted by:

$$\tau_{xy} = (1 - \rho).\tau_{xy} + \rho.\tau_{xy}^{0} \tag{10.6}$$

where τ_{xy} is the pheromone level on the edge xy, ρ is the pheromone evaporation coefficient, and τ^{0}_{xy} is the initial pheromone level on the edge xy initialized using the local initialization algorithm presented earlier (Equation 8.5). Typically in ACS, ρ is usually set to 0.1 [42].

The *global pheromone update* takes place at the end of each iteration when all the K ants finish constructing their solutions. Unlike the local pheromone update, not all edges will be globally updated but rather only the ones belonging to the best solutions (paths) constructed in that iteration. In T-BAR the global pheromone update is accomplished in several steps. First, the model computes the path trust PT_{ki} for each constructed solution by an ant k_i. The *path trust* [110] is a function of the number of co-rated items $n(I_{xy})$ between two adjacent users x and y and the trust T_{xy} issued by user x towards user y:

$$PT_{ki} = \frac{\sum_{xy \in P_{ki}} \left(n(I_{xy}).T_{xy} \right)}{\sum_{xy \in P_{ki}} n(I_{xy})} \tag{10.7}$$

In T-BAR, P_{ki} refers to the path constructed by ant k_i at the end of an iteration. After calculating the K path trusts, the algorithm adds the paths P_{ki} that satisfy the criteria $PT_{ki} \geq PT_{threshold}$ to the set of best paths P^{best}. The path trust is calculated in this manner in T-BAR based on the belief that the value of trust between two users x and y is strengthened by the number of items that were rated by both users. Note that in this manner, T-BAR manages to maintain a semi-item-based CF feature in the system by

considering the number of co-rated items along the constructed paths. Also note that path trust is a function of the local trust (although it has been already considered in Equation 10.1) to emphasize the importance of local trust over global trust (user reputation). Lastly, after calculating and comparing the path trusts on the constructed solutions, the global pheromone update is applied on the edges belonging to the paths in P^{best}:

$$\tau_{xy} = (1 - \rho).\tau_{xy} + \rho.\Delta\tau_{ki}^{best} \tag{10.8}$$

where $xy \in P_{ki}$, $P_{ki} \in P^{best}$ and:

$$\Delta\tau_{ki}^{best} = PT_{ki} \tag{10.9}$$

Note that the global pheromone update contributes to the pheromone build-up on *good* paths and thus helps the ants in subsequent iterations to ultimately converge to these paths. Also, T-BAR does not use a similarity measure, which is an important step in most RS using CF techniques, but rather uses the number of co-rated items as a semi-similarity measure. Another advantage of T-BAR over other known algorithms is that both local and global trust values are considered in the model; the local trust values influence the probability of the ants choosing a certain path while global trust values, used in the form of a user's reputation, are used as the influence parameter for the local trust.

10.3.3 Pheromone Initialization Mechanism

Since the way T-BAR is modeled coincides with the way the TSP is represented, and since the local pheromone initialization technique (presented in Chapter 8) proved its feasibility when used to solve the TSP, the presented initialization approach was adopted in T-BAR in this research to properly initialize the pheromone level on edges in the system.

In T-BAR, the pheromone level on edges is initialized locally right before an ant encounters it (regardless whether it is traversed or not) using the trust information in the network. Before an ant k_i computes the probability p^{ki}_{xy} of crossing one of the adjacent edges xy, it checks whether they have been initialized or not. If not then the adjacent edges are initialized using the inverse of the sum of their assigned trust values T_{xy} from node x:

$$\tau^0_{xy} = \frac{1}{\sum_{z \in N^{ki}_x} T_{xz}} \tag{10.10}$$

Therefore, the edges in the system will not necessarily have the same initial pheromone level.

10.4 PREDICTING THE RATING FOR THE TARGET ITEM

As previously mentioned, the K ants in T-BAR keep a record of the *good* users that they come across while constructing the paths in the last iteration, where *good* users are considered to be users u that have a rating for the target item i, denoted by r_i^u.

The reason behind keeping the record is that in T-BAR at the end of the last iteration the system should have converged into the best solutions (paths) and that is when the set of good users becomes useful. T-BAR averages the ratings r_i^u obtained from good users y found on these paths to calculate the target item's predicted rating for the active user x:

$$r_i^x = \frac{\sum_{u \in P^{best}} r_i^u}{n(u)} \tag{10.11}$$

where $n(u)$ is the number of good users u. This research believes that since the paths were labeled as being good ones and since they have a high path trust value, we might as well use all the ratings encountered along them as opposed to only using the item rating given

by the last user reached on each path, which is the case in both *TidalTrust* [51] and *MoleTrust* [110]. Considering all the item ratings along the constructed paths rather than stopping the search at the first node encountered with an item rating is another trivial contribution in this dissertation since the ants continue constructing their solutions until the depth d is reached. Also, most of the mentioned proposed algorithms use a trust threshold that limits the paths that can be traversed, however this research strongly believes that every trust value (good or bad) contributes to the rating prediction and thus does not impose such a constraint on the individual trust between users but rather uses the threshold to filter the constructed paths according to the level of their path trust.

10.5 T-BAR ALGORITHM

The following is a high-level pseudocode of T-BAR:

- Initialize the number of ants K, the number of iterations t, the search depth d, and the path trust threshold $PT_{threshold}$
- For each iteration t
    - Initialize P^{best}
    - For each ant k_i
        1. Initialize all neighboring edges, if they have not been initialized yet, using Equation 10.10.
        2. Compute the probability p^{ki}_{xy} of all possible moves from the current node x to all neighboring nodes $y \in N^{ki}_{x}$ using Equation 10.1.
        3. Move to node y that yielded the highest probability p^{ki}_{xy}.

4. Add the edge xy to the constructed path so far P_{ki} and locally update the pheromone level on it using Equation 10.6.
5. If the depth d is reached or if there are no more edges to be crossed, stop.

- Compute the path trust PT_{ki} for each constructed path P_{ki} using Equation 10.7.
- Add the paths P_{ki} having $PT_{ki} \geq PT_{threshold}$ to P^{best}
- Globally update the edges that are part of the paths in P^{best} using Equation 10.8.

- Calculate the predicted rating r^{i}_{x} using the ratings r_i^u that appeared on the paths P_{ki} in the last P^{best} using Equation 10.11.

10.6 EXPERIMENTAL EVALUATION SETUP

This section provides the details of the experiments that were conducted to validate the performance of T-BAR. T-BAR's results are compared to the ones obtained by Massa *et al.* in [94] since it is one of the major techniques that were applied to TBRS [1][8][13][21]. Namely, the results are compared to a basic CF algorithm that uses the *Pearson Similarity* measure for computing the similarity between the users and to Massa's proposed *MoleTrust* algorithm, which replaces the similarities in CF with the explicit trust values in the network. The results are analyzed across different views of the dataset to further understand T-BAR's pros and cons.

10.6.1 The Epinions Dataset

The Epinions dataset was used in this dissertation for the empirical evaluation. The reason behind the dataset choice lies in the fact that there is a scarcity in the availability of datasets that contain both item ratings along with *explicitly* issued trust values among the users. Another reason is that it was the dataset of choice in [94] so for the sake of comparison the same dataset was chosen. The Epinions dataset is composed of 49,290 users that rated 139,738 unique items at least once. The ratings range between 1 and 5 with 5 being the best rating. In addition there are 487,181 explicitly issued trust statements among the users. In this dataset there is no range for the issued trust values; if a user trusts another then that is expressed with a trust value of 1. More than half the users in the dataset rated less than 5 items each. Such users are referred to as *cold start users*; i.e. users who provided only a few ratings for the items in the dataset. Cold start users typically make it harder for RS to predict new item ratings for them due to lack of information in their rating profiles. When it comes to the ratings 45% of them are 5 and 29% of them are 4, which means that more than half the ratings in the dataset are good ones.

The dataset can be further classified into different categories, or *views*. These views are [94]

1. *cold start users*, users who rated less than 5 items;
2. *heavy raters*, users who rated more than 10 items;
3. *opinionated users*, users who rated 5 or more items and whose standard deviation is greater than 1.5;
4. *black sheep*, users who rated 5 or more items and the average distance of their

rating for item *i* with respect to the mean rating of item *i* is greater than 1;

5. *niche items*, items that received less than 5 ratings; and
6. *controversial items*, items with ratings whose standard deviation is greater than 1.5.

10.6.2 The Evaluation Metrics

In order to be capable of comparing T-BAR's results to the ones obtained in Massa *et al.'s* work [94], the same evaluation metrics referenced in their work were used.

Massa *et al.* used the *Mean Absolute Error* (MAE), which is the average of the absolute difference between the predicted rating and the hidden rating of an item. However, they point out that a major drawback with this metric is that it does not weigh the prediction error by the user's number of ratings and therefore the MAE for heavy raters weighs as much as the one for cold start users (more than half the users). This results in the error for heavy raters shadowing the one for cold start users. To overcome this problem Massa *et al.* presented the *Mean Absolute User Error* (MAUE), which can be computed by first finding the MAE for each user independently then by averaging the MAE across the users. In this manner, all the users would have the same weight.

The *ratings coverage* (RC) [66] was also used to assess a RS's ability to generate a prediction for the hidden rating, regardless of its accuracy. The RC refers to the fraction of ratings that were generated by the RS, while using the leave-one-out technique, against the actual number of ratings in the dataset. In other words, it is a measure of a RS's ability to predict a rating for a target item. But just like the MAE, RC suffers from not properly weighing the ratings. Therefore Massa *et al*. used the *users coverage* (UC) as the fraction of users for which the RS was able to provide at least one prediction.

CHAPTER 11

T-BAR'S DETAILED EXPERIMENTS AND THEIR EVALUATIONS

11.1 INTRODUCTION

This chapter covers the details of the experiments conducted to validate the effectiveness of T-BAR in increasing the accuracy of predicted item ratings in trust-based recommender systems. The chapter analyzes the results of applying the basic T-BAR and two variations of it to the Epinions dataset along with the empirical evaluation and comparison of the results against some known algorithms. The results are also inspected with respect to different views of the dataset in which the users and the items are further classified to better understand T-BAR's performance in different situations. In general, the empirical results show that T-BAR and its variations can improve the accuracy of some predictions while always providing a significantly better coverage of the dataset when compared to the other algorithms, regardless of the prediction accuracy. This property can be useful in systems in which we are interested in achieving a higher quantity of predictions over the quality of these predictions.

11.2 T-BAR'S PARAMETERS

The *leave-one-out* technique was applied to test T-BAR's ability to correctly predict the items' ratings in a TBRS. Just like in any ACO algorithm, the different parameters in T-BAR had to be experimented with in order to determine the best set to be used. For this

specific purpose, a total of 22 experiments were carried varying the number of ants K (5, 10, 20, 30, 40, 50), the number of iterations t (5, 10, 20, 30, 40, 50), the search depth d (10, 20, 30, 40, 50), and the path trust threshold $PT_{threshold}$ (0.1, 0.3, 0.5, 0.7, 0.9). The baseline when all these variations were tested was 10 ants, 10 iterations, a search depth of 30, and a path trust threshold of 0.5. It turned out that there was not a significant improvement in the *Mean Absolute Error* (MAE) or the *ratings coverage* (RC) when the parameters were varied, thus the baseline settings were used for the experiments while varying the search depth d (10, 30, 50) for comparing T-BAR with the results presented in [93].

11.3 COLLABORATIVE FILTERING ON EPINIONS DATASET

The results of the experiments in this dissertation are compared to several algorithms including the basic CF technique. The problem with CF is that although it is simple and straightforward but there are different ways it can be applied and the set thresholds and restrictions can greatly affect the results such as [115]:

- Only considering the target item ratings provided by the top k similar users.
- Calculating the similarity with users who have rated the target item and have at least n co-rated items with the target user.
- Considering users that have a similarity ≥ a similarity threshold.

Most CF implementations use one or a combination of the mentioned restrictions. The only restriction that all implementations conform to is to select users that have a positive similarity value because there is no sense in considering ratings from dissimilar users.

Table 11.1: MAE of different CF implementations on Epinions dataset.

Views	Algorithm			
	Basic CF	**Massa's CF**	**Victor et al.'s CF**	**Pham et al.'s CF**
All	0.636	0.843	0.84	0.96
Cold Start Users	1.669	1.094	-	-
Heavy Raters	0.554	0.850	-	-
Controversial Items	1.487	1.515	1.34	-
Niche Items	0.525	0.822	-	-
Opinionated Users	0.829	1.2	-	-
Black Sheep	0.782	1.235	-	-

Table 11.2: RC of different CF implementations on Epinions dataset.

Views	Algorithm			
	Basic CF	**Massa's CF**	**Victor et al.'s CF**	**Pham et al.'s CF**
All	75%	51%	79%	56%
Cold Start Users	44%	3%	-	-
Heavy Raters	73%	58%	-	-
Controversial Items	68%	45%	81%	-
Niche Items	56%	12%	-	-
Opinionated Users	70%	50%	-	-
Black Sheep	78%	56%	-	-

Table 11.1 compares the results reported from different applications of the basic CF algorithm using Pearson Similarity to the Epinions dataset in the literature, in addition to this dissertation's implementation of the technique (*Basic CF*) using the leave-one-out technique. The Pearson Similarity formula used in *Basic CF*'s implementation is:

$$Pearson(x,y) = \frac{\sum_{i \in I_{xy}} (r_i^x - \overline{r^x})(r_i^y - \overline{r^y})}{\sqrt{\sum_{i \in I_{xy}} (r_i^x - \overline{r^x})^2 \sum_{i \in I_{xy}} (r_i^y - \overline{r^y})^2}} \quad (11.1)$$

where r^x_i refers to the user x's rating of item i, $\overline{r^x}$ refers to user x's average item rating, and I_{xy} is the set of items co-rated by users x and y. The detailed steps of *Basic CF* are as follows:

1. For each user, hide the rating for the target item and calculate the Pearson Similarity with all other users in the dataset (using Equation 11.1) as long as:
    a. They have a rating for the hidden target item.
    b. They have rated *at least* two items in common with the target user.
2. After calculating the similarities, discard the users that yielded a negative similarity value (i.e. non-similar users).
3. Average the target item ratings across the remaining users to generate the predicted item rating.
4. Compare the predicted rating with the actual rating to estimate the MAE.
5. Repeat steps 1 – 4 for all items rated by each user.
6. Average the MAE across all users/items.

Both Massa et *al.*'s implementation of CF [94] and Victor et *al.*'s approach [148] take into consideration the similarity weight in the item rating prediction process, and thus predict the rating for the hidden item as:

$$r_i^x = \overline{r^x} + \frac{\sum_{y \in R^+} Pearson(x,y)(r_i^y - \overline{r^y})}{\sum_{y \in R^+} Pearson(x,y)} \tag{11.2}$$

where R^+ refers to the set of users that have rated item i and have a positive similarity with user x. However, although the two approaches claim to be using the same technique for predicting item ratings, but the reported results in Table 11.1 show that some other

parameters or additional restrictions were applied which caused the differences in the reported results.

In [115], Pham et *al.*'s application of the CF technique on the Epinions dataset has an additional restriction for selecting the candidate similar users which requires user *y* to have rated at least four items in common with the target user *x*.

11.4 EXPERIMENTAL RESULTS

In this dissertation, T-BAR's results are compared to two different algorithms: *CF* which is a CF algorithm implemented by Massa [92][93] using the *Pearson Similarity* and the *MoleTrust* algorithm (*MT*) [94] but with three different propagation horizons (1, 2, and 3) referred to as MT1, MT2, and MT3.

Tables 11.3 and 11.4 show the results obtained by T-BAR against the two algorithms. T-BAR10, T-BAR30, and T-BAR50 refer to the three different search depths used. In order to highlight T-BAR's strengths, this research compares the worst of the three T-BAR algorithms with the best of the MT algorithms. Note that it would not make sense to use the same depths used in MT when testing T-BAR because the former follows a breadth-wise search while the latter searches for a solution in a depth-wise manner. Doing so would cause the number of nodes/edges that can be traversed by T-BAR to greatly decrease compared to the ones reached by MT.

A quick glance at the first row in Tables 11.3 and 11.4 will show that T-BAR drastically increases the overall accuracy and coverage of the recommendations over the whole dataset. But when the results are weighed by the number of users it can be seen from the first row in Table 11.5 that T-BAR does not perform as well as CF or MT.

Table 11.3: MAE of the basic algorithms on different views.

Views	Algorithm						
	Massa's CF	**MT1**	**MT2**	**MT3**	**T-BAR 10**	**T-BAR 30**	**T-BAR 50**
All	0.843	0.832	0.846	0.829	0.298	0.315	0.304
Cold Start Users	1.094	0.674	0.833	0.854	1.459	1.4	1.426
Heavy Raters	0.850	0.873	0.869	0.846	0.212	0.22	0.22
Controversial Items	1.515	1.425	1.618	1.687	1.995	1.976	1.913
Niche Items	0.822	0.734	0.806	0.828	0.572	0.582	0.534
Opinionated Users	1.2	1.02	1.102	1.096	1.308	1.56	1.319
Black Sheep	1.235	1.152	1.238	1.242	1.973	1.813	1.915

Table 11.4: RC of the basic algorithms on different views.

Views	Algorithm						
	Massa's CF	**MT1**	**MT2**	**MT3**	**T-BAR 10**	**T-BAR 30**	**T-BAR 50**
All	51%	28%	61%	74%	93%	97%	97%
Cold Start Users	3%	11%	25%	42%	91%	95%	96%
Heavy Raters	58%	31%	65%	78%	93%	97%	97%
Controversial Items	45%	25%	61%	81%	59%	77%	81%
Niche Items	12%	8%	24%	20%	48%	66%	69%
Opinionated Users	50%	23%	57%	74%	94%	99%	99%
Black Sheep	56%	24%	59%	76%	77%	93%	92%

Yet, coverage-wise it is extremely obvious from the first row in Table 11.6 that T-BAR still outperforms the other techniques by increasing the ability to provide a recommendation to a random user by at least 30%.

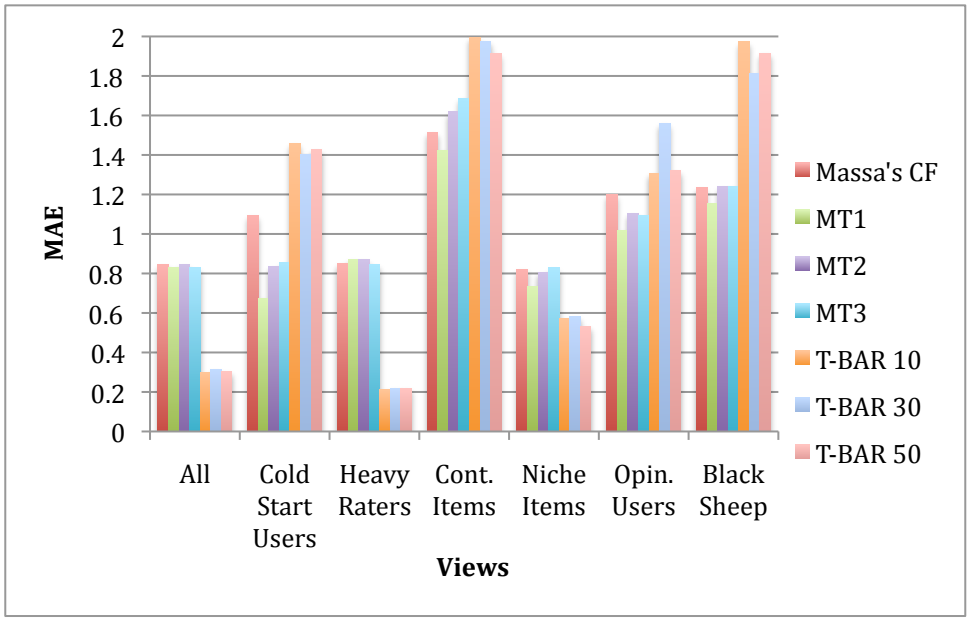


Figure 11.1: MAE of the basic algorithms across different views.

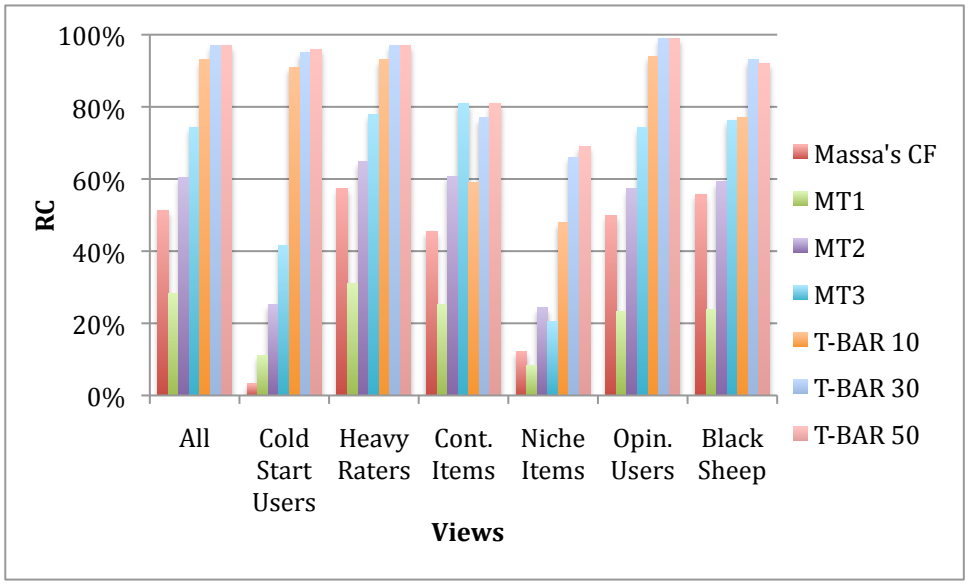


Figure 11.2: RC of the basic algorithms across different views.

Table 11.5: MAUE of the basic algorithms on different views.

Views	Algorithm						
	Massa's CF	**MT1**	**MT2**	**MT3**	**T-BAR 10**	**T-BAR 30**	**T-BAR 50**
All	0.938	0.790	0.856	0.844	1.203	1.2	1.202
Cold Start Users	1.173	0.674	0.820	0.854	1.581	1.561	1.563
Heavy Raters	0.903	0.834	0.861	0.834	0.282	0.298	0.293
Controversial Items	1.503	1.326	1.571	1.650	1.967	2.064	1.971
Niche Items	0.854	0.671	0.808	0.843	0.896	0.874	0.851
Opinionated Users	1.316	0.938	1.090	1.092	1.262	1.307	1.294
Black Sheep	1.407	1.075	1.258	1.285	1.973	1.923	1.984

Table 11.6: UC of the basic algorithms on different views.

Views	Algorithm						
	Massa's CF	**MT1**	**MT2**	**MT3**	**T-BAR 10**	**T-BAR 30**	**T-BAR 50**
All	41%	47%	60%	66%	96%	98%	99%
Cold Start Users	3%	18%	31%	43%	97%	99%	99%
Heavy Raters	86%	80%	88%	89%	93%	97%	98%
Controversial Items	16%	12%	22%	28%	92%	95%	97%
Niche Items	11%	10%	21%	33%	74%	83%	85%
Opinionated Users	61%	61%	77%	80%	94%	99%	99%
Black Sheep	68%	61%	75%	78%	81%	94%	94%

Recall that more than half the users in the Epinions dataset are classified as cold start users, which usually poses a challenge for RS, and although T-BAR does not achieve a better accuracy than the ones reached by MT or CF (as can be seen in Figures 11.1 and 11.3), T-BAR still manages to generate a relatively low error rate in general while drastically improving the ratings coverage as illustrated in Figures 11.2 and 11.4.

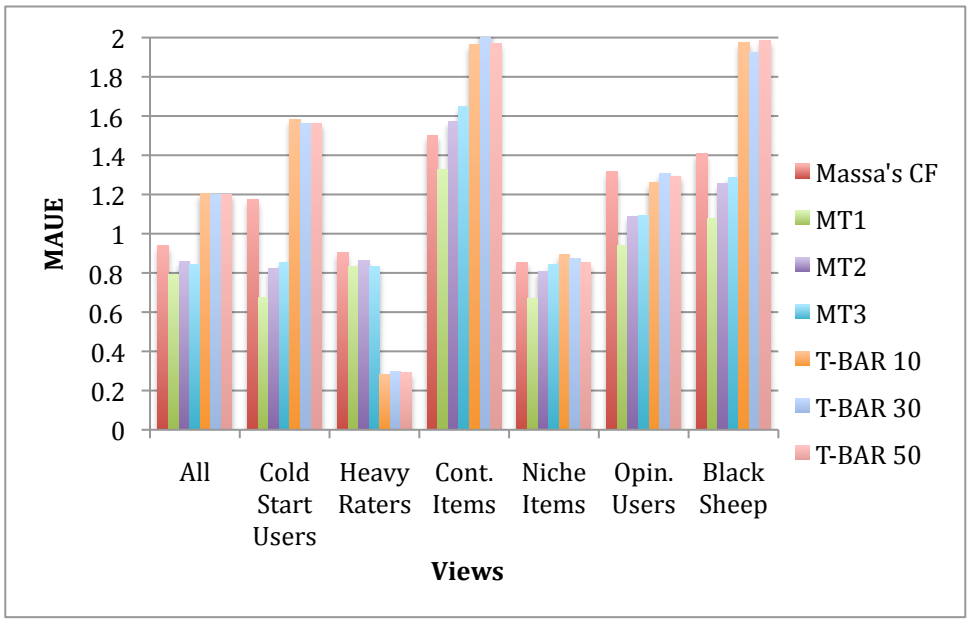


Figure 11.3: MAUE of the basic algorithms across different views.

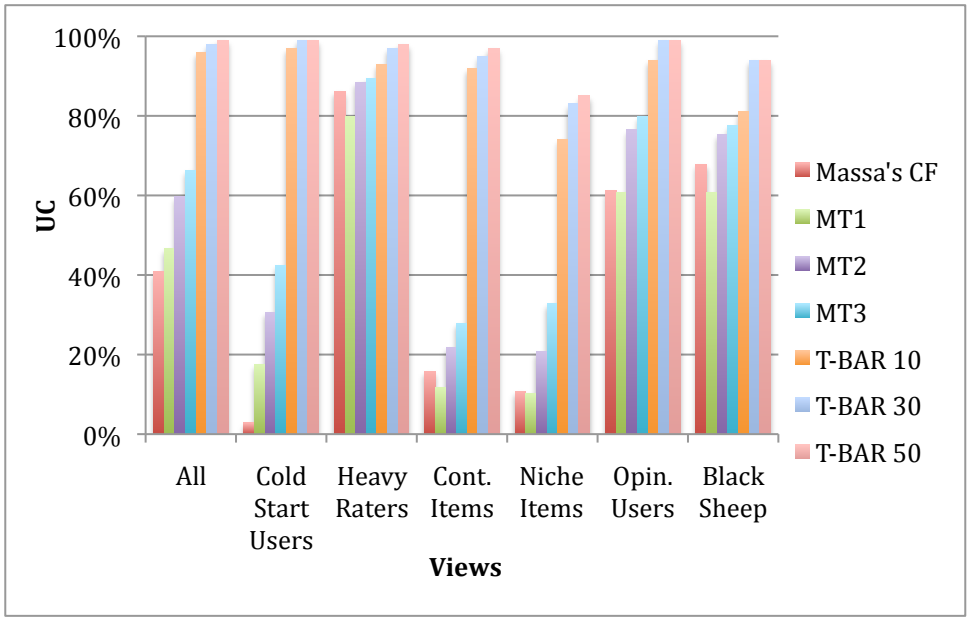


Figure 11.4: UC of the basic algorithms across different views.

On the other hand when considering heavy raters and niche items, T-BAR greatly outperforms the other algorithms with respect to both the prediction accuracy and the

coverage. For heavy raters T-BAR had a MAUE of ~0.3 compared to MT1 and MT3's MAUE of 0.846. The reason behind T-BAR's superior performance for heavy raters can be attributed to the way that T-BAR works. Recall that at the end of each iteration the number of co-rated items between adjacent users along P_{ki} plays a major role in increasing the path trust and increasing the pheromone level on that path, which will also increase the probability of having a greater number of co-rated items between adjacent users. In addition these paths were selected by the ants while constructing their solution partly because of their high trust along their edges. These two reasons contributed to the increase in the predictions' accuracy for heavy raters in T-BAR. The latter argument can be also used to validate T-BAR's high accuracy and coverage for niche items. Niche items can be thought of as the unpopular items since they received less than 5 ratings. By examining Tables 11.3, 11.4, 11.5, and 11.6 it can be seen how both CF and MT both miserably fail to provide an accurate rating for such items, that is if they were able to generate one in the first place because of their low RC and UC for those items. Table 11.4 shows how MT at its best can only predict a rating for only ~24% of niche items. T-BAR's ability to incorporate all the trusted *good* users' ratings for such items along the paths over several iterations plays a major role in its enhanced performance over the others.

However, MT outperformed T-BAR with respect to controversial items, opinionated users, and black sheep users. But, for opinionated users and black sheep users it is still obvious that T-BAR can provide a better rating coverage even after weighing the results by the number of users as seen in Figure 11.4. In general though, Massa's CF technique outperformed both MT and T-BAR with respect to the prediction accuracy of

controversial items. This can be explained by the fact that when an item receives a wide range of ratings it is better to consider the opinion of like-minded users since their taste would be similar rather than relying on users' trust in one another, which may not necessarily indicate that they have the same taste.

11.5 Additional Experiments and their Results

In an attempt to further improve the performance and the coverage of T-BAR, different variations of the algorithm were tested in this dissertation by altering the way some values are calculated. One variation was to dispatch the ants from users in the source user x's WOT_x rather than from the source user x in hopes of improving the ratings' accuracy for cold start users. The results did not show a significant improvement in either the accuracy or the RC and therefore the results were discarded.

In other attempts, the criteria for selecting the best paths at the end of each iteration was changed. Instead of calculating the path trust PT, the average trust of *good* users along the constructed paths was used. A path would be then added to PT^{best} if the average trust (popularity) of such users is greater than a trust threshold. The average was weighed in one variation by the length of the path and in another variation by the number of *good* users on that path. This dissertation refers to the former version as T-BAR^{PL} (PL referring to *path length*) and to the latter as T-BAR^{GU} (where GU stands for *good users*). Tables 11.7 and 11.8 display the results of testing these two variations using different thresholds and depths while Tables 11.9 and 11.10 show the results averaged by users.

Table 11.7: MAE of the new algorithms on different views.

Views	Algorithm									
	Massa's CF	**MT1**	**MT2**	**MT3**	**T-BAR^{PL} 10**	**T-BAR^{PL} 30**	**T-BAR^{PL} 50**	**T-BAR^{GU} 10**	**T-BAR^{GU} 30**	**T-BAR^{GU} 50**
All	0.843	0.832	0.846	0.829	0.952	0. 774	0.675	0.475	0.416	0.383
Cold Start Users	1.094	0.674	0.833	0.854	2.63	2.367	2.44	1.661	1.256	1.378
Heavy Raters	0.850	0.873	0.869	0.846	0.947	0.77	0.671	0.472	0.406	0.377
Controversial Items	1.515	1.425	1.618	1.687	2.907	2. 657	2.616	2.131	2.123	1.946
Niche Items	0.822	0.734	0.806	0.828	3.23	2.998	2.856	2.273	1.988	1.826
Opinionated Users	1.2	1.02	1.102	1.096	3.667	3.667	3.667	3.667	2.037	3.667
Black Sheep	1.235	1.152	1.238	1.242	2.973	2.728	2.765	2.453	2.338	2.338

Table 11.8: RC of the new algorithms on different views.

Views	Algorithm					
	Massa's CF	**MT1**	**MT2**	**MT3**	**T-BAR^{PL} (d = 10, 30, 50)**	**T-BAR^{GU} (d = 10, 30, 50)**
All	51%	28%	61%	74%	100%	100%
Cold Start Users	3%	11%	25%	42%	100%	100%
Heavy Raters	58%	31%	65%	78%	100%	100%
Controversial Items	45%	25%	61%	81%	100%	100%
Niche Items	12%	8%	24%	20%	100%	100%
Opinionated Users	50%	23%	57%	74%	100%	100%
Black Sheep	56%	24%	59%	76%	100%	100%

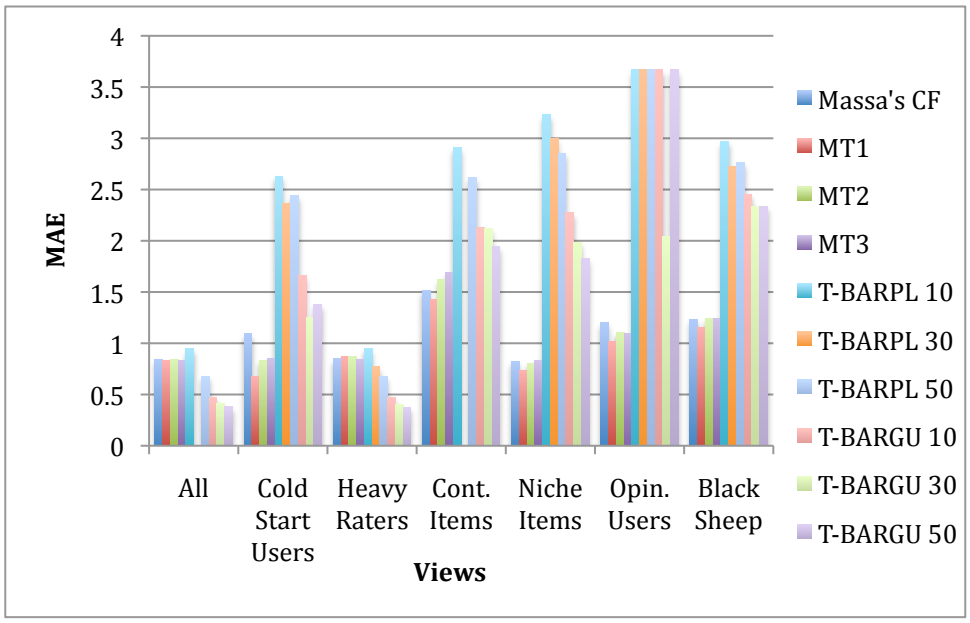


Figure 11.5: MAE of the new algorithms across different views.

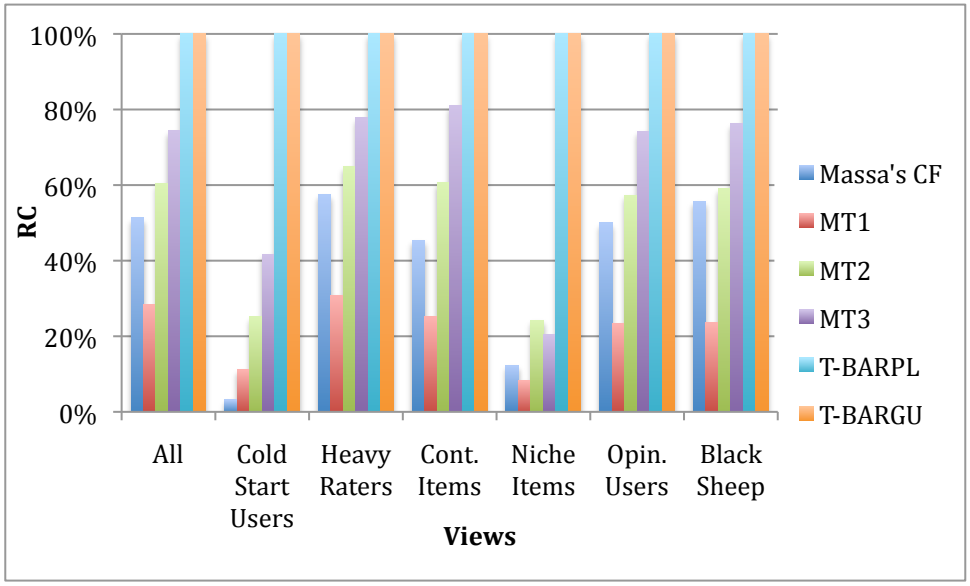


Figure 11.6: RC of the new algorithms across different views.

Table 11.9: MAUE of the new algorithms on different views.

Views	Algorithm									
	Massa's CF	**MT1**	**MT2**	**MT3**	**T-BAR^{PL} 10**	**T-BAR^{PL} 30**	**T-BAR^{PL} 50**	**T-BAR^{GU} 10**	**T-BAR^{GU} 30**	**T-BAR^{GU} 50**
All	0.938	0.790	0.856	0.844	1.167	0.954	0.888	0.619	0.665	0.587
Cold Start Users	1.173	0.674	0.820	0.854	2.813	2.371	2.624	1.725	1.385	1.495
Heavy Raters	0.903	0.834	0.861	0.834	0.941	0.766	0.664	0.466	0.42	0.382
Controversial Items	1.503	1.326	1.571	1.650	2.995	2.779	2.704	2.106	2.207	1.946
Niche Items	0.854	0.671	0.808	0.843	3.303	3.099	2.864	2.253	1.984	1.837
Opinionated Users	1.316	0.938	1.090	1.092	3.667	3.667	3.667	3.667	1.766	3.667
Black Sheep	1.407	1.075	1.258	1.285	1.603	1.603	1.603	2.358	2.338	2.338

Table 11.10: UC of the new algorithms on different views.

Views	Algorithm					
	Massa's CF	**MT1**	**MT2**	**MT3**	**T-BAR^{PL} (d = 10, 30, 50)**	**T-BAR^{GU} (d = 10, 30, 50)**
All	41%	47%	60%	66%	100%	100%
Cold Start Users	3%	18%	31%	43%	100%	100%
Heavy Raters	86%	80%	88%	89%	100%	100%
Controversial Items	16%	12%	22%	28%	100%	100%
Niche Items	11%	10%	21%	33%	100%	100%
Opinionated Users	61%	61%	77%	80%	100%	100%
Black Sheep	68%	61%	75%	78%	100%	100%

The results in general show that T-BAR^{GU} outperforms T-BAR^{PL}, which makes sense since the number of good users along a path would contribute to increasing the accuracy of the prediction better than the length of the path (i.e. quality vs. quantity). Comparing Figures 11.5 and 11.7 to Figure 11.1 shows that T-BAR^{GU} still managed to achieve a MAE and MAUE below or around the 1.5 margin for cold start users, a MAE and MAUE below 0.5 for heavy raters, and a MAE and MAUE of ~ 2 for controversial items, which are all really close to T-BAR's performance. Yet looking at Figures 11.6 and 11.8 will show how T-BAR^{GU} has a *drastic* improvement in coverage compared to T-BAR's coverage. T-BAR^{GU} is considered to have a perfect coverage of 100% for all users and ratings, which means that T-BAR^{GU} was able to provide at least one rating for *all* users and *all* items, and therefore the system will never fail to provide an item rating for any random user (regardless of the rating's accuracy). These reasons make T-BAR^{GU} a better option as an algorithm than T-BAR for the three mentioned categories of users. Just like T-BAR though, black sheep users and opinionated users posed a challenge for both T-BAR^{PL} and T-BAR^{GU}. The only downside of T-BAR^{GU} when compared to T-BAR is that it does not perform well for niche items, producing a MAE and MAUE of ~2 while T-BAR achieves a MAE and MAUE of ~0.6 for such items.

Overall if we were to compare T-BAR, T-BAR^{PL}, and T-BAR^{GU} we can conclude that all three algorithms perform generally well by providing a good balance between prediction accuracy and coverage compared to the results achieved by Massa *et al.* in [94]. Both T-BAR and T-BAR^{GU} can achieve a better prediction accuracy but T-BAR^{GU} is capable of achieving a perfect coverage for all users and all items but at the expense of the prediction accuracy.

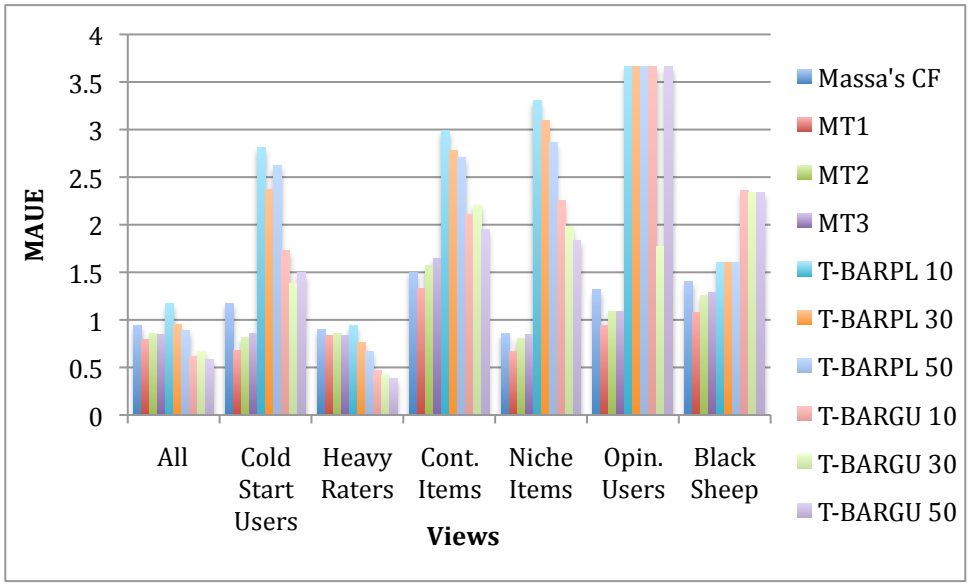


Figure 11.7: MAUE of the new algorithms across different views.

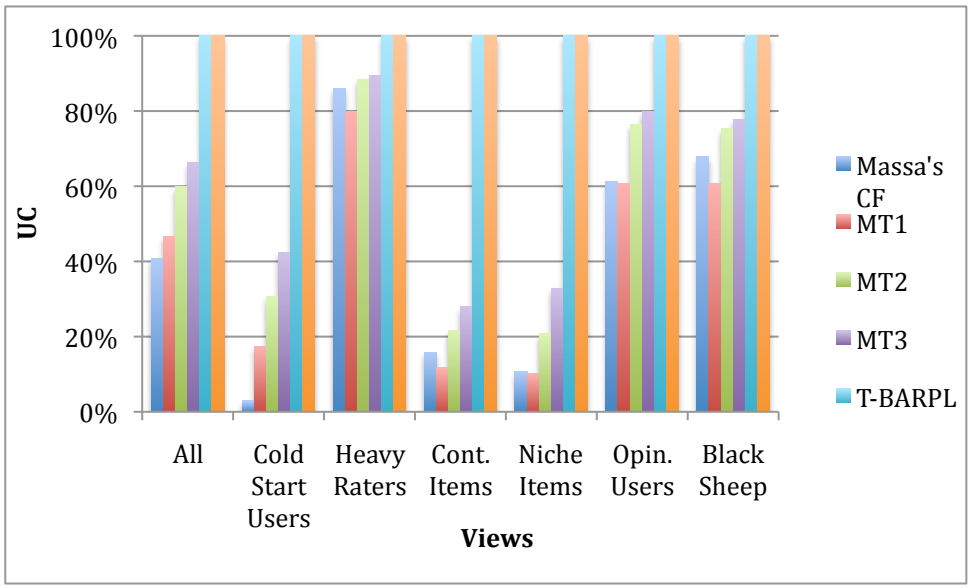


Figure 11.8: UC of the new algorithms across different views.

11.6 SUMMARY OF RESULTS

The empirical evaluation of this dissertation's *Trust-Based Ant Recommender*, T-BAR, on the Epinions dataset proved that the model can drastically improve the prediction accuracy as opposed to the accuracy achieved by traditional CF techniques or by Massa *et al*.'s proposed *MoleTrust*. The novelty of T-BAR and its two variations, especially T-BAR^{GU}, proved to be very useful for heavy raters as they managed to achieve a MAE as low as ~ 0.2 for such users compared to MT and CF algorithms' MAE of ~ 0.8. On the other hand, MT and classic CF techniques outperformed T-BAR with respect to controversial items. But coverage wise, T-BAR^{PL} and T-BAR^{GU} achieved a perfect RC of 100% for all users (including cold start users) regardless of the prediction accuracy. This can be useful in situations where the system can tolerate bad predictions early on for cold start users until they become heavy raters such as in Netflix.com where users are encouraged to rate as many movies as they can so that the system can provide better movie ratings in the future for unseen movies. T-BAR outperformed its two variations, CF, and MT for niche items by achieving a low MAE of ~ 0.6 and a coverage of at least 50%; the other algorithms failed in that manner. T-BAR would be useful in situations where the dataset is composed of many items receiving less than 5 ratings. Netflix.com is a good example for this case since not all movies are expected to be highly rated or even receiving a decent number of ratings such as unknown indie movies that gain popularity (and accumulate ratings) over a long period of time. If CF or MT techniques were to be applied in such scenarios the system would be able to provide a movie rating only for ~ 3% of cold start users in the case of CF algorithms, and for ~ 42% of such users in MT algorithm (MT3). If the generation of ratings for all users is

critical in a system, regardless of the prediction accuracy, then T-BAR^{GU} should be the algorithm of choice since it can achieve a MAE close to the ones obtained by T-BAR but with a better coverage rate. In critical systems though where the accuracy of the prediction outweighs the importance of the coverage, CF techniques would definitely be the better option to be applied rather than MT or T-BAR.

However, the success of T-BAR's application to TBRS inspired the work in this research to explore other ways that T-BAR can be modified in order to further enhance the performance and possibly solve the problem of cold start users.

CHAPTER 12

LOCALIZED T-BAR MODELS

12.1 INTRODUCTION

T-BAR proved its ability to enhance the performance of TBRS in terms of accuracy and coverage especially for heavy raters compared to Massa's CF and *MoleTrust*. However, T-BAR is considered the basic model presented in this dissertation and its success encouraged additional investigation to determine areas in which the model can be improved. Since the local pheromone initialization technique presented in Chapter 8 is considered as one of the main contributions of this research and since it is the first successful attempt in the literature to alter the way pheromone is initialized in ACO algorithms, it has been determined in this dissertation that the area of pheromone initialization still has room for improvement and hence can be further explored and enhanced by presenting localized models of T-BAR that reflect the differences in trust level between edges in the pheromone initialization process.

12.2 RATIONALE BEHIND LOCALIZED T-BAR MODELS

Trust plays a major role in any algorithm applied to TBRS and one of the reasons behind T-BAR's success is the fact that it greatly incorporates trust in its recommendation process by interleaving the trust values with popularities and aspects of similarities between users in the search process, which allowed the system to eventually find *good* users with *good* quality. In other words, the quality of good users reached

through T-BAR is high due to its ability to find users that have many items in common with the active user (source) and that have a high trust level.

T-BAR performs extremely well for heavy raters, which would make sense when considering that, in addition to the reasons discussed above, heavy raters have rated many items and therefore reinforces the quality of *good* users found (i.e. it would be easier to find co-rated items for such users). However, the same cannot be said for cold start users and that would explain T-BAR's inability to perform well for them.

In order to overcome this problem in the localized models, trust is given a bigger role in guiding the ants in their search process without impacting their ability to perform well for other users. This research saw a window of opportunity in the way the initial pheromone level is calculated since T-BAR initializes all edges within a WOT_x using the same value without taking into consideration the individual differences between the edges in terms of trust. Therefore, trust needs to be reinforced whenever an ant needs to initialize the pheromone level on edges in order to reflect the differences in trust (and importance) between the edges to compensate for the lack of item ratings and co-rated items for cold start users. Thus, the new localized T-BAR models differ from T-BAR in that an ant will still use the trust values within a neighborhood to determine the initial pheromone level on edges however in the localized models the trust level on each edge is further incorporated to reflect its importance compared to others in that neighborhood.

12.3 Pheromone Initialization Mechanism in Localized Models

The localized T-BAR models follow the same methodology applied in T-BAR to predict ratings for unseen items for the active user in TBRS. The influence of trust between users in these models is increased in the pheromone initialization step, which

would allow the changes to be reflected in other aspects of the system as well.

In the localized models, each edge *xy* within WOT_x is assigned a different initial pheromone level τ^0_{xy} that would reflect its associated trust level T_{xy} when compared to other edges in the neighborhood. The initialization task is assigned to the individual ants where each ant k_i initializes the pheromone level on edges within a WOT_x upon their first encounter in the system. Before an ant k_i calculates the probability of crossing an edge *xy* it has to check whether that edge has been initialized or not. If not, then the ant would utilize the locally available information within WOT_x to calculate τ^0_{xy}.

Two localized models are presented in this dissertation: Simple Localized T-BAR (SLT-BAR) and Averaged Localized T-BAR (ALT-BAR). SLT-BAR calculates the initial pheromone level on edges *xy* as follows:

$$\tau^0_{xy} = \frac{T_{xy}}{\sum_{z \in WOT_x} T_{xz}} \quad (12.1)$$

which, when compared to the local pheromone initialization technique (Equation 8.5), increases the initial pheromone level on edges. On the other hand, ALT-BAR calculates the initial values using:

$$\tau^0_{xy} = \frac{T_{xy}}{n(u_x).\sum_{z \in WOT_x} T_{xz}} \quad (12.2)$$

where $n(u_x)$ refers to the number of users in WOT_x. $n(u_x)$ is used to average the initial pheromone levels within each WOT_x by its number of users and thus to decrease the values used for initialization while still maintaining the differences in importance between edges.

The effect of the introduced changes in the pheromone initialization step in both models will consequently impact:

1. The probability p^{ki}_{xy} (Equation 10.1) because the initial pheromone level on an edge *xy* determines the initial probability of crossing that edge and since the edges within WOT_x will not have the same initial pheromone value then we expect edges with higher trust levels to have higher initial pheromone levels and thus a higher probability of being crossed.
2. The local pheromone update of τ_{xy} (Equation 10.6) because it involves the initial pheromone level to determine the amount of pheromone to be deposited. If all edges within WOT_x have the same initial pheromone level then it would be expected for the pheromone to increase at the same rate on those edges. But if the initial pheromone levels reflect their associated trust then pheromone will accumulate faster on edges with higher initial pheromone levels.

Figure 12.1 is an example that depicts how pheromone is initialized in the two models.

12.4 EXPERIMENTAL RESULTS

The localized T-BAR models were tested on the Epinions dataset against a basic CF algorithm [92][93], Massa's *MoleTrust* algorithm (MT) [94], and T-BAR since it is the basic trust-based ant recommender. Since varying the search depth *d* in previous experiments did not show any significant change in the results, the experiments for the localized T-BAR models used the search depth of 10 for all compared algorithms (which is equivalent to *MoleTrust*'s MT1).

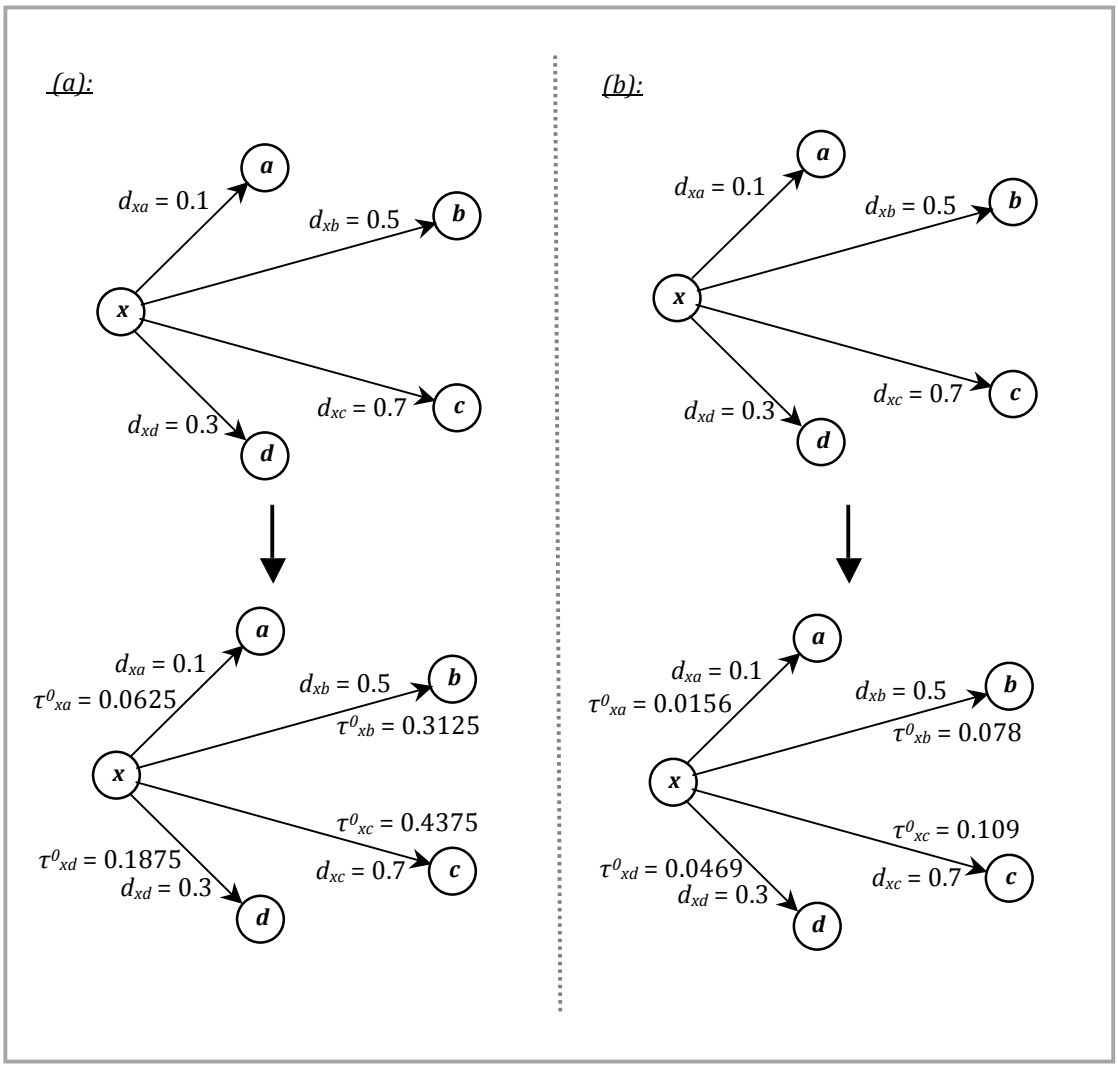


Figure 12.1: Example of pheromone initialization in localized T-BAR models.
(a) Simple Localized T-BAR initialization mechanism.
(b) Averaged Localized T-BAR initialization mechanism.

The overall results show that ALT-BAR has better accuracy and coverage than SLT-BAR, which is contributed to using $n(u_x)$ in ALT-BAR to average the initial pheromone levels within each WOT_x by its number of users and thus to avoid system fluctuations due to using high initial pheromone levels which prevented SLT-BAR from converging to the optimal solutions.

Table 12.1: MAE of localized T-BAR models against the basic algorithms.

Views	Algorithm				
	Massa's CF	**MT**	**T-BAR**	**SLT-BAR**	**ALT-BAR**
All	0.843	0.832	0.298	0.985	0.57
Cold Start Users	1.094	0.674	1.459	1.824	0.502
Heavy Raters	0.850	0.873	0.212	0.912	0.62
Controversial Items	1.515	1.425	1.995	2.1	1.27
Niche Items	0.822	0.734	0.572	0.78	0.56
Opinionated Users	1.2	1.02	1.308	1.13	0.889
Black Sheep	1.235	1.152	1.973	1.44	0.935

Table 12.2: RC of localized T-BAR models against the basic algorithms.

Views	Algorithm				
	Massa's CF	**MT**	**T-BAR**	**SLT-BAR**	**ALT-BAR**
All	51%	28%	93%	81%	90%
Cold Start Users	3%	11%	91%	12%	53%
Heavy Raters	58%	31%	93%	85%	92%
Controversial Items	45%	25%	59%	23%	48%
Niche Items	12%	8%	48%	49%	83%
Opinionated Users	50%	23%	94%	62%	54%
Black Sheep	56%	24%	77%	63%	54%

When it comes to comparing ALT-BAR to the other algorithms, it can be seen by comparing the MAE of the different algorithms (Table 12.1) alongside their RC (Table 12.2) that T-BAR outperforms the other algorithms in terms of overall ratings accuracy and coverage. However since the results obtained for heavy raters can heavily affect the overall MAE of the algorithms, a quick glance at Tables 12.3 and 12.4 would give a better insight on how those algorithms rank against each other regardless of the

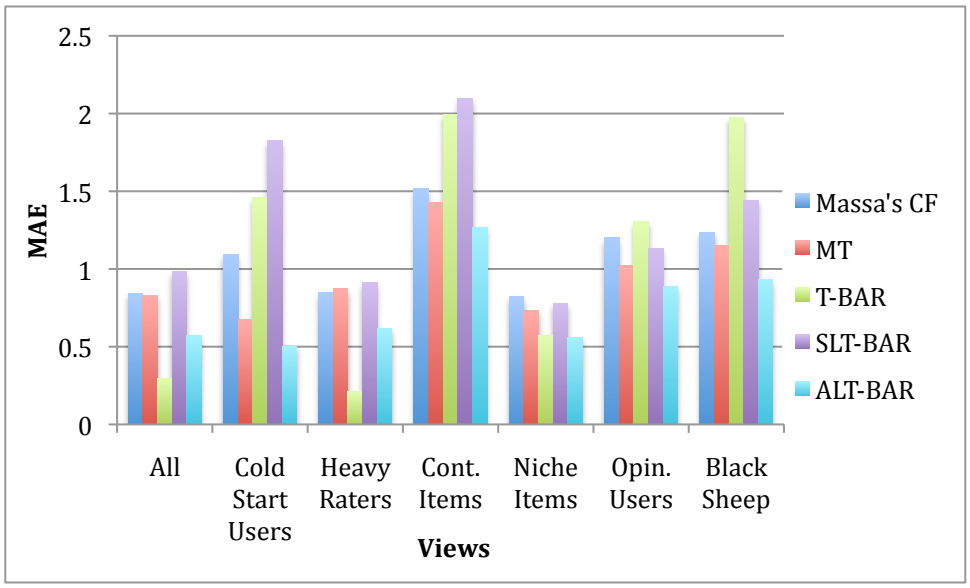


Figure 12.2: MAE of localized T-BAR models against the basic algorithms.

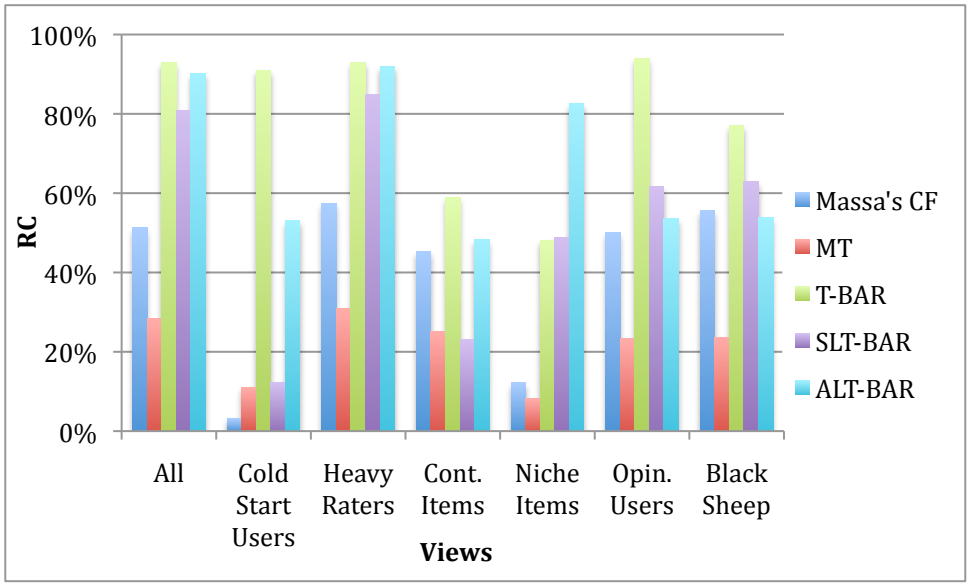


Figure 12.3: RC of localized T-BAR models against the basic algorithms.

number of users within each category. The MAUE and UC show that ALT-BAR achieves a good overall balance by reaching a MAUE ~ 0.6 and a UC of 71% for all users.

Table 12.3: MAUE of localized T-BAR models against the basic algorithms.

Views	Algorithm				
	Massa's CF	**MT**	**T-BAR**	**SLT-BAR**	**ALT-BAR**
All	0.938	0.790	1.203	0.983	0.592
Cold Start Users	1.173	0.674	1.581	1.88	0.43
Heavy Raters	0.903	0.834	0.282	0.87	0.683
Controversial Items	1.503	1.326	1.967	1.97	1.307
Niche Items	0.854	0.671	0.896	0.83	0.91
Opinionated Users	1.316	0.938	1.262	1.325	0.755
Black Sheep	1.407	1.075	1.973	1.303	1.102

Table 12.4: UC of localized T-BAR models against the basic algorithms.

Views	Algorithm				
	Massa's CF	**MT**	**T-BAR**	**SLT-BAR**	**ALT-BAR**
All	41%	47%	96%	78%	71%
Cold Start Users	3%	18%	97%	17%	56%
Heavy Raters	86%	80%	93%	92%	92%
Controversial Items	16%	12%	92%	13%	82%
Niche Items	11%	10%	74%	74%	90%
Opinionated Users	61%	61%	94%	72%	68%
Black Sheep	68%	61%	81%	77%	69%

A major advantage of ALT-BAR over all algorithms is evident in its superior performance for cold start users as can be seen in terms of both MAE and MAUE. Recall that more than half the users in the Epinions dataset are classified as cold start users and they usually pose a challenge for RS. ALT-BAR achieved the best accuracy for cold start users by reaching a MAE of 0.5 and a MAUE of 0.4. Although MT achieved a MAE of

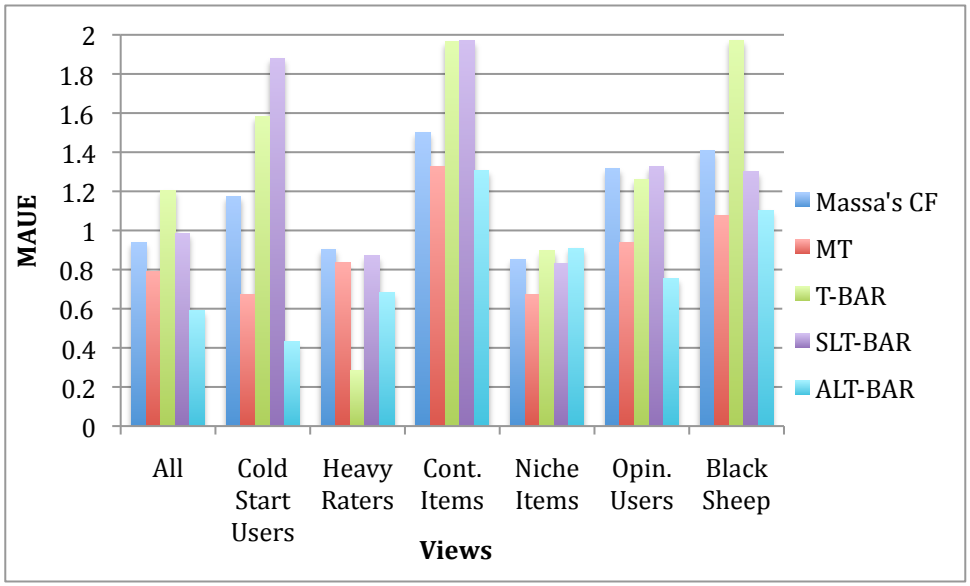


Figure 12.4: MAUE of localized T-BAR models against the basic algorithms.

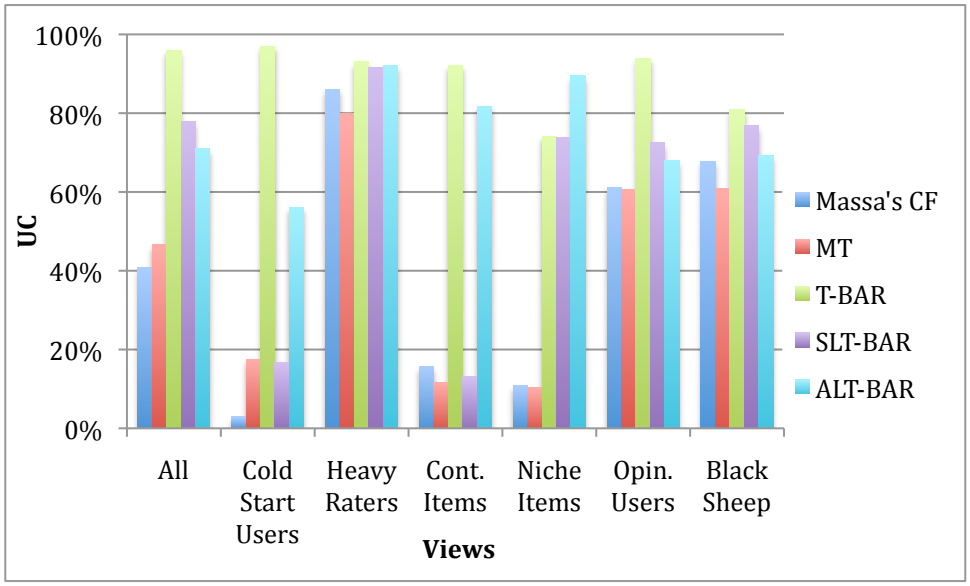


Figure 12.5: UC of localized T-BAR models against the basic algorithms.

~ 0.7 for cold start users but ALT-BAR's RC of 53% is much higher than MT's 11% which still proves that ALT-BAR can handle such users very well.

In general, T-BAR is always the better choice if the dataset is composed mostly of heavy raters since it provides an amazing performance that surpasses the ones achieved by all other algorithms (MAE ~ 0.2 and 93% RC). However, ALT-BAR would be the perfect choice for any dataset that greatly suffers from cold start users. Opinionated users are another category of users that can be a challenge to deal with in different algorithms, as is obvious in the MAUE achieved by CF, MT, and T-BAR in Table 12.3. However, ALT-BAR performs well for those users by dropping the MAUE to ~ 0.8 with a 68% UC.

12.5 Summary of Results

This chapter presented two localized T-BAR models that reinforce the importance of trust in TBRS. The Averaged Localized T-BAR model provided better results in general than the Simple Localized T-BAR model because it reduces the increase in the initial pheromone level (due to using trust) by averaging the values by the number of users within each neighborhood thus still maintaining the differences in importance between the edges.

Since cold start users lack the availability of item ratings, ALT-BAR relies heavily on the trust issued between users to guide the ants in their exploration of the solution space. ALT-BAR achieves that by assigning each edge an initial pheromone level that reflects the edge's associated importance (trust). Given the significance of the initial pheromone level in ACO algorithms in determining the system's convergence to the optimal solution, ALT-BAR's pheromone initialization approach proved to be feasible in terms of allowing the ants to expand their exploration scope for cold start users while managing to

exploit the good discovered paths for heavy raters which results in finding *good* users with a rating for the target item in general.

Although ALT-BAR's expanded exploration of the other paths guided by the trust between users impacted its performance for heavy raters but despite this setback when compared to the other algorithms, ALT-BAR managed to balance the overall trade-off between prediction accuracy and coverage for both cold start users and heavy raters.

CHAPTER 13

DYNAMIC T-BAR MODELS

13.1 INTRODUCTION

Both T-BAR and ALT-BAR demonstrated their abilities to handle heavy raters and cold start users respectively by altering the way pheromone is initialized on edges, which greatly affects the system's ability to properly converge to the optimal solutions in a timely manner. The success of the two models and the observation of the effect of ALT-BAR's initialization mechanism on the way ants explore the solution space inspired the work in this dissertation to shift the focus to the way ants behave and communicate to explore new areas that could enhance the performance of the system.

So far, the communication between the ants is at a minimal level sharing only information about which neighborhoods have been explored in the solution space. Thus, the work in this research is expanded to include the possibility of increasing the level of communication between the decentralized agents and studying the effect on the prediction accuracy in TBRS. The changes in the communication mechanism will also affect the pheromone initialization in the presented dynamic T-BAR models.

13.2 RATIONALE BEHIND DYNAMIC T-BAR MODELS

T-BAR and ALT-BAR's local pheromone initialization approaches, in which the artificial ants initialize all edges within a feasible neighborhood upon their first encounter, result in the ants sharing only information about which neighborhoods have

been encountered and thus initialized so far. The information is useful for the ants in the two models because it informs them about the trust level in the neighborhood as a whole in T-BAR and about the individual trust on edges in ALT-BAR, which eventually dictates which paths to follow and which areas to explore in the solution space. Although the ants collaborate in a decentralized manner, this does not conflict with the possibility of them sharing additional information that can be useful in the exploration of the solution space without affecting the general guidelines of artificial ants' behavior in ACO algorithms. The information sharing among the agents is taken a step further in the presented dynamic models by allowing the ants to pass a message to the other ants about which edges have been crossed and exploited so far.

There are different approaches to describe how an ant can pass this information in the system to subsequent agents, but in the presented models the information sharing must comply with three goals:

1. Allow the artificial ants to share more information among them about the paths that have been explored and thus support the exploration of the other undiscovered paths.
2. Maintain the role of trust in the pheromone initialization process that has been achieved by the localized T-BAR models so that the initial pheromone levels on edges within WOT_x would reflect the different trust levels on those edges.
3. The ants must utilize the information shared in a way that can contribute to enhancing the performance of the system.

To attain the first goal in the new models, the ants will commit the initial pheromone level only on the edge that will be crossed while discarding the other initializations. This

is closely related to the second goal though because if the initial pheromone level is calculated in a manner similar to the way it is done T-BAR (Equation 8.5), then the probability of crossing undiscovered edges would still be relatively unaffected (and possibly low) especially if the crossed edges keep accumulating pheromone on them which would discourage other ants from exploring new paths. Thus, the initial pheromone level in the new models is calculated in a way similar to how it is done in the localized T-BAR models to reflect the trust assigned to each edge within a dynamic WOT_x ($DWOT_x$).

However, sharing the information about which edges have been crossed alone is not useful for the ants unless it affects the probability of selecting edges in the path construction process. If the ants keep initializing the remaining edges within a neighborhood using the same information upon each encounter then the models will not be any different from the ones presented so far. Therefore, the third goal is achieved in the new models by *dynamically* updating the information within a neighborhood to *exclude* trust information from the initialized edges and thus only use the uninitialized edges' information whenever an ant needs to calculate the initial pheromone level for those uncrossed edges.

13.3 Pheromone Initialization Mechanism in Dynamic Models

The models presented so far in this dissertation initialize the pheromone level of edges within a neighborhood upon their first encounter. The dynamic T-BAR models' major difference from T-BAR and the localized T-BAR models is evident in the pheromone initialization step; instead of calculating the initial pheromone level τ^0_{xy} for

all edges xy within WOT_x in a single step (when the edges are encountered for the first time), the presented dynamic T-BAR models allow an ant k_i to commit the pheromone initialization only on the edge that yields the highest probability p^{ki}_{xy} of being crossed. In this manner, a maximum of one pheromone initialization is permitted by an ant k_i in each encountered neighborhood and a maximum of d pheromone initializations is allowed per ant per iteration.

Committing the initialization only on the crossed edges serves as a message to subsequent ants about which edges have been explored and thus after each initialization the neighborhood information used in the pheromone initialization process is dynamically updated in $DWOT_x$ to exclude the trust information associated with the recently initialized and crossed edge. The dynamically updated neighborhood $DWOT_x$ is only used in the pheromone initialization process; the calculation of p^{ki}_{xy} considers the whole WOT_x to determine the path to be followed by an ant k_i.

Two dynamic T-BAR models are presented in this research: Dynamic Localized T-BAR (DLT-BAR) which applies the same pheromone initialization technique used in SLT-BAR (Equation 12.1), and Dynamic Averaged Localized T-BAR (DALT-BAR) that uses the averaged pheromone initialization approach applied in ALT-BAR (Equation 12.2). Figure 13.1 is an example that demonstrates the DLT-BAR's pheromone initialization process. Just like in the localized T-BAR models, the changes in the pheromone initialization process in the dynamic T-BAR models will affect the probability p^{ki}_{xy} and the local pheromone update of τ_{xy}, which will eventually affect the exploration of the solution space.

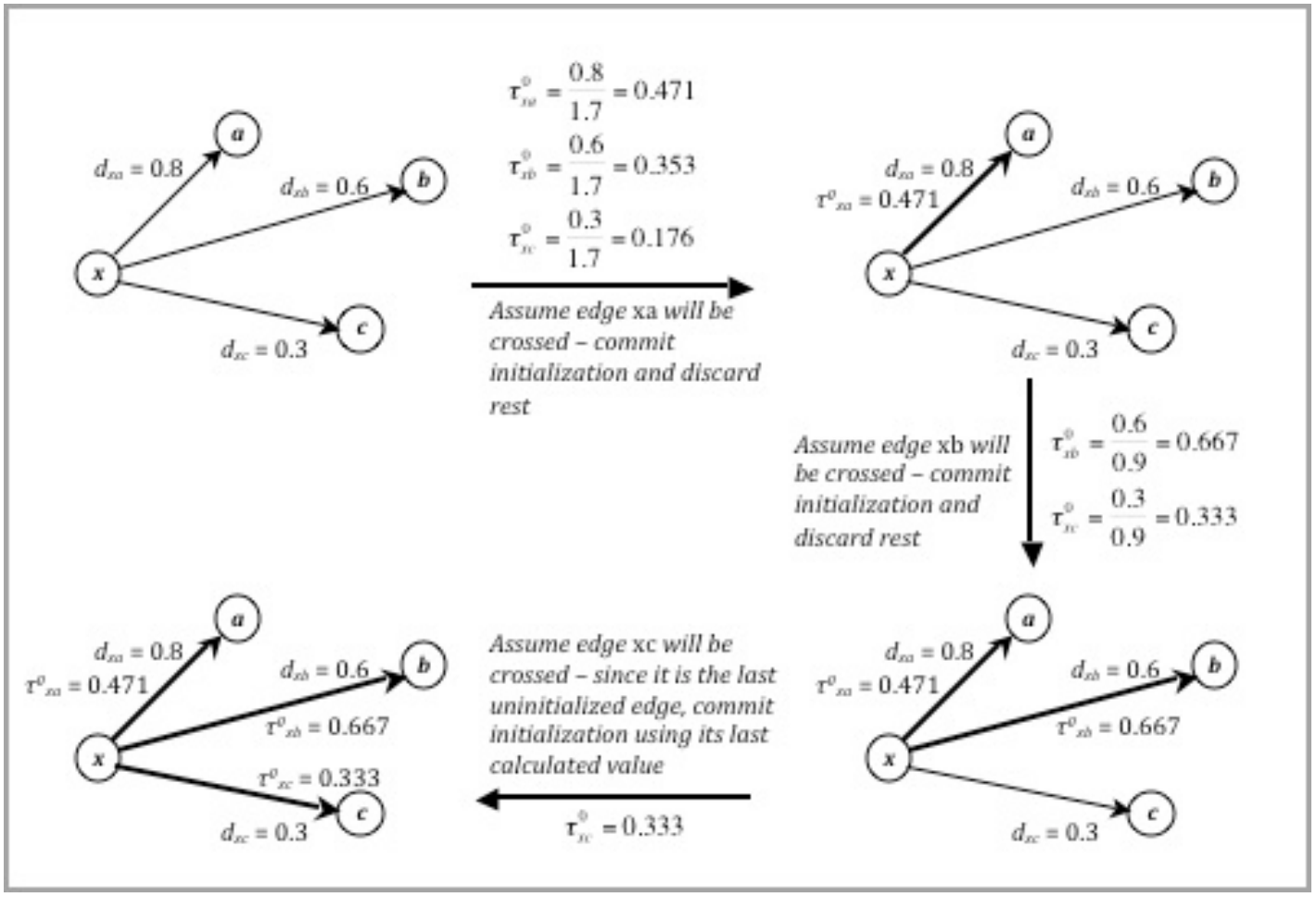


Figure 13.1: Example of pheromone initialization in Dynamic Localized T-BAR (DLT-BAR) model.

13.4 EXPERIMENTAL RESULTS

The dynamic T-BAR models were tested on the Epinions dataset against Massa's CF algorithm [92][93], the *MoleTrust* algorithm (MT) [94], and the trust-based ant recommender (T-BAR). All algorithms were tested using a search depth of 10, which is equivalent to *MoleTrust*'s search depth of 1 since it follows a breadth-wise search approach.

Considering how the MAE is calculated, Table 13.1 and 13.2 show how the overall MAE does not reflect the results for the majority of users in the dataset (i.e. cold start users) but is rather affected by the ones for heavy raters due to the big difference in the accuracy between the two user categories. However, the DLT-BAR model does not suffer from this problem because it achieves almost similar MAE for both cold start users and heavy raters, which results in the overall MAE not being misleading as it is in the case of the other algorithms, including T-BAR. Table 13.1 shows that DLT-BAR achieves a MAE of ~ 0.7 for cold start users, heavy raters and all users in the dataset in general, which indicates that DLT-BAR has a consistent performance for all major user categories. However, DALT-BAR does not achieve a similar consistency across the user categories. Of course the results in Table 13.3 support this observation even further since the results are weighed by the number of users in each user/item category.

Despite DLT-BAR's consistent performance, its recommendation accuracy is not as good as T-BAR's or ALT-BAR's. T-BAR's success for heavy raters is credited to the model's approach of depending on both trust values and rated items among the users resulting in better *exploitation* of explored paths, while ALT-BAR's superior performance for cold start users is attributed to increasing the role of trust in the model to

Table 13.1: MAE of dynamic T-BAR models against the basic algorithms.

Views	Algorithm				
	Massa's CF	**MT**	**T-BAR**	**DLT-BAR**	**DALT-BAR**
All	0.843	0.832	0.298	0.723	0.683
Cold Start Users	1.094	0.674	1.459	0.714	0.864
Heavy Raters	0.850	0.873	0.212	0.778	0.618
Controversial Items	1.515	1.425	1.995	1.629	1.97
Niche Items	0.822	0.734	0.572	0.222	0.643
Opinionated Users	1.2	1.02	1.308	0.411	0.948
Black Sheep	1.235	1.152	1.973	0.812	1.103

Table 13.2: RC of dynamic T-BAR models against the basic algorithms.

Views	Algorithm				
	Massa's CF	**MT**	**T-BAR**	**DLT-BAR**	**DALT-BAR**
All	51%	28%	93%	84%	90%
Cold Start Users	3%	11%	91%	55%	37%
Heavy Raters	58%	31%	93%	87%	91%
Controversial Items	45%	25%	59%	39%	53%
Niche Items	12%	8%	48%	84%	60%
Opinionated Users	50%	23%	94%	34%	22%
Black Sheep	56%	24%	77%	37%	43%

compensate for the lack of item ratings and thus allowing for better *exploration* of the solution space. However, DLT-BAR's approach to exploring the solution space caused a constant increase in the initial pheromone level on unexplored edges, which prevented the ants from properly *exploiting* the discovered paths in addition to not allowing proper *exploration* of the solution space due to the rapid increases in the initial pheromone levels. In other words, DLT-BAR did not explore the solution space well enough to reach

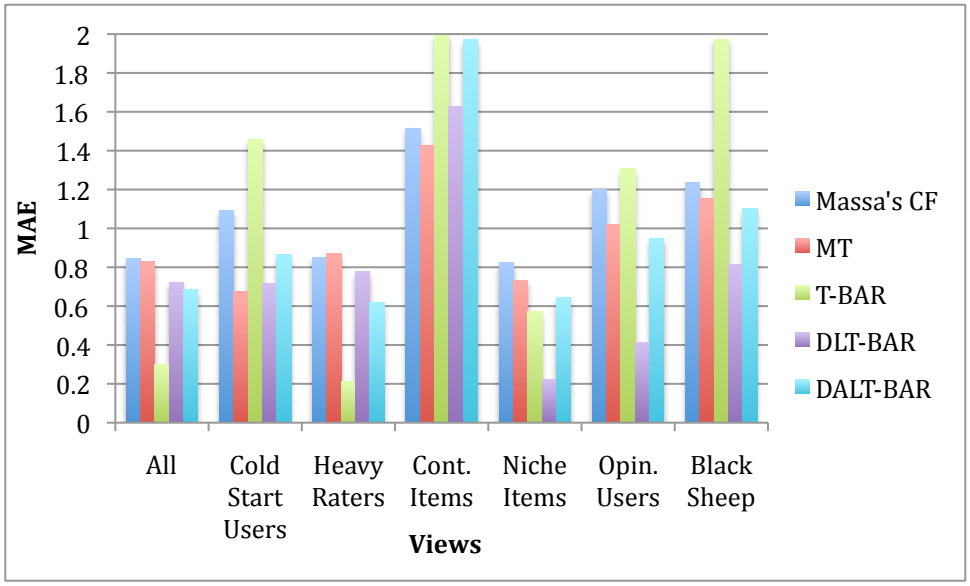


Figure 13.2: MAE of dynamic T-BAR models against the basic algorithms.

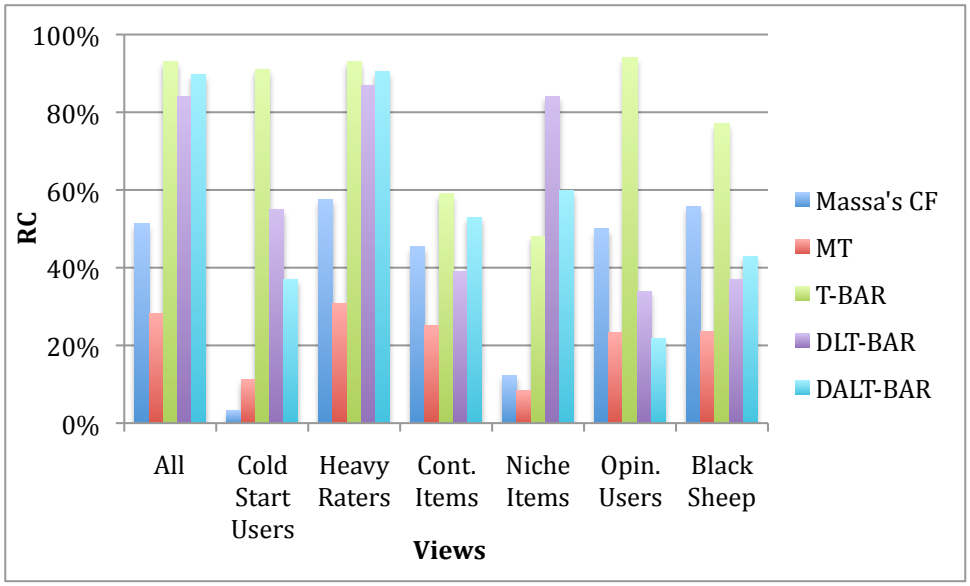


Figure 13.3: RC of dynamic T-BAR models against the basic algorithms.

ALT-BAR's performance for cold start users, and it did not adequately exploit the explored paths to attain T-BAR's results for heavy raters. DLT-BAR has a major

Table 13.3: MAUE of dynamic T-BAR models against the basic algorithms.

Views	Algorithm				
	Massa's CF	**MT**	**T-BAR**	**DLT-BAR**	**DALT-BAR**
All	0.938	0.790	1.203	0.79	0.783
Cold Start Users	1.173	0.674	1.581	0.784	0.874
Heavy Raters	0.903	0.834	0.282	0.806	0.73
Controversial Items	1.503	1.326	1.967	1.75	1.86
Niche Items	0.854	0.671	0.896	0.323	0.78
Opinionated Users	1.316	0.938	1.262	0.498	0.948
Black Sheep	1.407	1.075	1.973	0.895	1.39

Table 13.4: UC of dynamic T-BAR models against the basic algorithms.

Views	Algorithm				
	Massa's CF	**MT**	**T-BAR**	**DLT-BAR**	**DALT-BAR**
All	41%	47%	96%	68%	90%
Cold Start Users	3%	18%	97%	50%	71%
Heavy Raters	86%	80%	93%	90%	93%
Controversial Items	16%	12%	92%	73%	89%
Niche Items	11%	10%	74%	85%	61%
Opinionated Users	61%	61%	94%	39%	59%
Black Sheep	68%	61%	81%	43%	32%

advantage over all other algorithms when it comes to niche items for that it dropped the MAE and the MAUE to 0.22 and 0.32 respectively. Niche items did not attract a lot of attention in the literature but they can be as challenging to deal with in RS just like cold start users due to the scarcity of ratings available for those items.

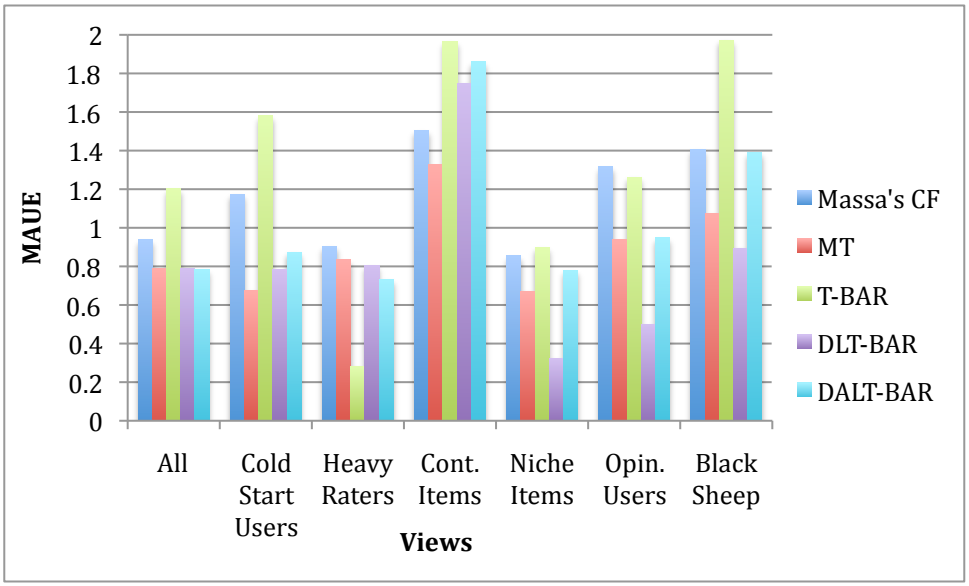


Figure 13.4: MAUE of dynamic T-BAR models against the basic algorithms.

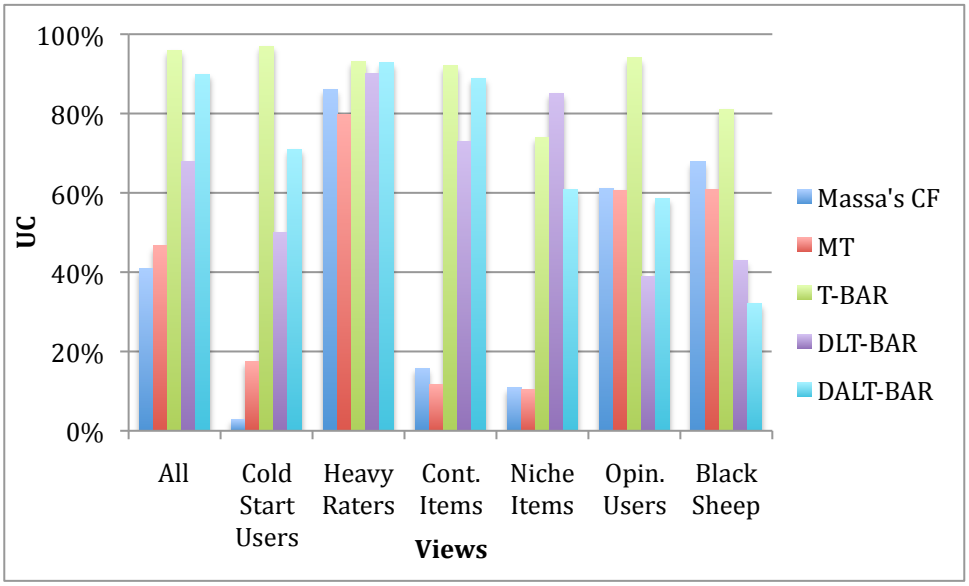


Figure 13.5: UC of dynamic T-BAR models against the basic algorithms.

Although the model did not achieve good results for *users* with a limited number of rated items (cold start users) but it performs well for *items* with few available ratings (niche items).

Despite the fact that T-BAR and ALT-BAR are the obvious algorithm choices for certain situations, DLT-BAR may be considered as a more suitable option when the distribution of user categories in the dataset is unknown since it delivers an acceptable consistent performance across the different discussed views. Another case where DLT-BAR should be considered is when a RS consists of a substantial number of niche items whose accuracy of predictions could affect a user's confidence in the system's performance.

13.5 SUMMARY OF RESULTS

In addition to the models presented in this dissertation, two dynamic models are presented as part of this research to demonstrate the effect of increased information sharing among the ants on the prediction accuracy in TBRS. One of the presented dynamic T-BAR models, DLT-BAR, can achieve a consistent performance for the two major user categories in RS: cold start users and heavy raters but at the expense of a lower prediction accuracy when compared to ALT-BAR and T-BAR respectively. Proposed algorithms in the literature can only deliver good results for one user category at the expense of the other, but DLT-BAR manages to balance the performance with a consistent acceptable accuracy levels across the two categories. This is achieved by allowing the artificial ants to share information about the explored edges and by initializing the pheromone level on edges to values proportional to their corresponding trust level. The models permit each ant to commit the initialization of at most one edge upon each neighborhood encounter, which is the edge with the highest probability to be crossed in that neighborhood. Since subsequent ants use the dynamically updated

neighborhood information to calculate the initial pheromone level on uninitialized edges, the initial pheromone level on those edges tends to rapidly increase upon each neighborhood update, which could increase their probability of being crossed. This behavior explains DLT-BAR's prediction accuracy levels for the two major user categories since it does not allow the artificial ants to properly *explore* the solution space or adequately *exploit* the explored solutions.

CHAPTER 14

CONCLUSION

The goal of this dissertation was to show that the application of algorithms borrowed from the family of swarm intelligence to trust-based recommender systems can enhance the accuracy of recommendations by improving the *exploration* and *exploitation* of the solution space.

The major contributions of this research are:

1. Formalizing a bio-inspired model to enhance the accuracy of predictions in trust-based recommender systems by improving the search criteria in the solution space. The success of the presented model can be attributed to considering:
    - All encountered ratings for the target item along a path
    - Not neglecting the importance of global trust (popularity)
    - Enforcing a threshold on the overall trust on a solution path rather than on trust between users.
2. Presenting a set of enhanced models, based on the formalized model, with the goal of enhancing the recommendation accuracy for a certain group of users, mainly cold start users and heavy raters.
3. Presenting a pioneer local pheromone initialization approach that can be applicable to any ant colony algorithm.

This dissertation has been presented in a way that shows the detailed flow of how ideas were analyzed and formed during the work on this research, which resulted in the presented models and algorithms.

The dissertation presented an ant colony-inspired algorithm that belongs to the family of Ant Colony Optimization algorithms. The algorithm was applied to trust-based recommender systems and was named T-BAR, Trust-Based Ant Recommender. The goal was to increase the accuracy of the ratings' prediction and system coverage. One of the major advantages of T-BAR is that unlike other algorithms it considers all the item ratings that it encounters along the path rather than just using the final item rating that is reached. In addition, T-BAR considers both local trust and global trust in its search process based on the belief that the popularity of users can strengthen the confidence in their item ratings. Also, most recommender systems chose to enforce a threshold on trust between users while building the solution path which could result in the exclusion of *good* users, however T-BAR avoids that by imposing a threshold on the overall trust on the constructed path.

T-BAR was tested on a real-world dataset and the empirical evaluation and comparison of results against some known algorithms in the literature showed T-BAR's ability in greatly improving the results in terms of both accuracy and coverage especially for heavy raters. Based on T-BAR's success, several other algorithms were presented where each aims to solve a specific problem.

The localized T-BAR models focused on reinforcing the importance of trust between users to compensate for the lack of item ratings for cold start users. The localized models incorporated the differences in trust between users within a neighborhood in the

pheromone initialization process. In this manner, the models were able to deal with cold start users by expanding the scope of the explored solution space as a result of reflecting the individual trust on the edges' initial pheromone levels. The presented models' approach impacted the performance for heavy raters (compared to T-BAR) however in comparison with other algorithms, the localized models achieved good results for both cold start users and heavy raters in terms of prediction accuracy and coverage. In general though, the Averaged Localized T-BAR model (ALT-BAR) had better results than the Simple Localized T-BAR model (SLT-BAR) since ALT-BAR averages the initial pheromone levels by the number of users within a neighborhood to avoid possible system fluctuations due to substantial differences in trust levels.

The dissertation presented a set of dynamic T-BAR models to illustrate the effect of increasing the level of information sharing among the artificial ants on the accuracy of results in trust-based recommendations. To achieve that, the dynamic models only committed the pheromone initialization on the edges to be crossed and updated the local information available for subsequent ants by excluding information about the recently crossed edge. This approach caused the initial pheromone level on the remaining edges to rapidly increase (until committed), which increased their overall probability of being crossed while building the solution paths. The dynamic models showed consistent performance for cold start users and heavy raters, which can be useful in situations where the distribution of users in the dataset is unknown. Although the models did not perform well for cold start users or heavy raters when compared to ALT-BAR and T-BAR respectively, but their results matched the average performance of the other compared algorithms.

The novelty of the different T-BAR models lies in the fact that it is the first successful application of an algorithm from the family of swarm intelligence to the area of trust-based recommender systems. The presented results prove that employing different agents to explore the solution space can enhance the prediction accuracy even for problematic users such as cold start users. The differences between the presented models highlight different ways for the agents to behave while constructing their solutions. In general, the presented T-BAR models always provided results that balance the tradeoff between accuracy and coverage when compared to other popular algorithms in the literature.

CHAPTER 15

FUTURE WORK

There are many ways in which the work presented in this dissertation can be extended. For example, the promising results of the presented local pheromone initialization approach in enhancing the performance of ACO algorithms is just the first step in the path of exploring other possible ways to perform local pheromone initialization especially since the initial pheromone level has a major influence on the performance of ACO algorithms in general.

Another idea that can be experimented with is the effect of stigmergy between ants on enhancing the performance of the system. Although the presented dynamic models are based on enhancing information sharing during the pheromone initialization step but additional experiments can be performed to study information sharing at different steps in the algorithm such as during the solution construction step or at the end of each iteration.

Also, the choice of dataset was restricted by the availability of one that allows access to both item ratings and trust between users. However, the presented T-BAR models can be tested on other datasets that do not necessarily consider trust between users. To experiment with such datasets, an extra preprocessing step has to be added in which a value has to be calculated to replace trust in the models. This value can be a function of different aspects of user profiles, such as similarity, item ratings, number of co-rated items, etc.

Another major extension to this dissertation would be the study of the application of other swarm intelligence algorithms to recommender systems. Some of the algorithms that are currently being considered and can be envisioned to be applicable to TBRS include the Firefly algorithm, Cuckoo search, and Bee algorithms.

Also it has been noticed that there is a lot of attention on the application of matrix factorization in RS such as Zhang *et al*.'s work in [157] and Ning's *et al.*'s approach in [108] where they treat the system as a regularized optimization problem that can be solved using a factorization model. Since ACO algorithms proved their success in the literature in solving optimization problems, a deeper understanding of the application of matrix factorization can lead to exploring factorization models and its possible incorporation into T-BAR.

REFERENCES

1. ABDUL-RAHMAN, A. AND HAILES, S. 1997. A distributed trust model. *New Security Paradigms Workshop*. Cumbria, United Kingdom: 48–60.
2. ABDUL-RAHMAN, A. AND HAILES, S. 2000. Supporting trust in virtual communities. *Proceedings of the 33rd Hawaii International Conference on System Sciences*. Maui, Hawaii, USA.
3. ABERER, K., AND DESPOTOVIC, Z. 2001. Managing trust in a peer-2-peer information system. *In proceedings of the 10th international conference on information and knowledge management.* 310-317. New York, NY, USA.
4. ADOMAVICIUS, G., SANKARANARAYANAN, R., SEN, S., TUZHILIN, A. 2005. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Trans. Inf. Syst*. 23(1): 103–145.
5. AGRAWAL, R., AND SRIKANT, R. 1994. Fast algorithms for mining association rules in large databases. *In Proceedings of the 20th International Conference on Very Large Data Bases*.
6. AMATRIAIN, X., LATHIA, N., PUJOL, J. M., KWAK, H., AND OLIVER, N. 2009. The wisdom of the few: A collaborative filtering approach based on expert opinions from the web. *In Proc. of SIGIR '09*.
7. ANDERSON, M., BALL, M., BOLEY, H., GREENE, S., HOWSE, N., LEMIRE, D., AND MCGRATH S. 2003. Racofi: A rule-applying collaborative filtering system. *In Proc. IEEE/WIC COLA'03*.
8. ANSPER, A., BULDAS, A., ROOS, M. AND WILLEMSON, J. 2001. Efficient long-term validation of digital signatures. *Advances in Cryptology - PKC 2001*.
9. AVESANI, P., MASSA, P. 2005. Moleskiing.it: A Trust-aware recommender system for ski mountaineering, *International Journal for Infonomics*, 1-10.
10. AXELROD, R. 1984. *The Evolution of Cooperation*. New York: Basic Books.
11. BAETS, B. D. 2003. Growing decision trees in an ordinal setting. *International Journal of Intelligent Systems*.
12. BAILEY, R. A. 2008. *Design of comparative experiments*. Cambridge University Press, Cambridge.
13. BARNES, J. A. 1972. *Social networks*. Reading, MA: Addison-Wesley.
14. BASU, C., HIRSH, H., AND COHEN, W. 1998. Recommendation as classification: Using social and content-based information in recommendation. *In Proceedings of the Fifteenth National Conference on Artificial Intelligence*, 714–720. AAAI Press.
15. BEDI, P., AND SHARMA, R. 2012. Trust based recommender system using ant colony for trust computation. *Expert Systems with Applications*, 39(1): 1183-1190, Tarrytown, NY, USA.
16. BETH, T., BORCHERDING, M. AND KLEIN, B.. 1994. Valuation of trust in open networks. *Proceedings of ESORICS 94*. Brighton, UK, November 1994.
17. BOUTILIER, C., BRAFMAN, R. I., DOMSHLAK, C., HOOS, H. H., AND POOLE, D. 2003. CPnets: A Tool for Representing and Reasoning with Conditional Ceteris Paribus Preference Statements. *Journal of Artificial Intelligence Research (JAIR),*

2003.
18. BOUZA, A., REIF, G., BERNSTEIN, A., AND GALL, H. 2008. Semtree: ontology-based decision tree algorithm for recommender systems. *In International Semantic Web Conference*.
19. BRICKLEY, D. AND MILLER, L. 2010. FOAF Vocabulary Specification, Namespace Document, August 9, 2010, http://xmlns.com/foaf/spec/.
20. BRIDGE, D., GÖKER, M., MCGINTY, L., AND SMYTH, B. 2006. Case-based recommender systems. *The Knowledge Engineering review*. 20(3): 315–320.
21. BURKE, R. 2007. Hybrid web recommender systems. *The Adaptive Web*, Springer. Berlin, Heidelberg: 377–408.
22. BUSKENS, V. 2002. *Social Networks and Trust*. Dordrecht, The Netherlands: Kluwer Academic Publishers.
23. CATTELL, V. 2001. "Poor people, poor places, and poor health: the mediating role of social networks and social capital." *Social Science and Medicine* 52(10):1501-1516.
24. CHENG, W., HÜHN, J., AND HÜLLERMEIER, E. 2009. Decision tree and instance-based learning for label ranking. *In ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning*, 161–168, New York, NY, USA.
25. CHO, Y., KIM, J., AND KIM, S. 2002. A personalized recommender system based on web usage mining and decision tree induction. *Expert Systems with Applications*.
26. COHEN, W. 1995. Fast effective rule induction. *In Machine Learning: Proceedings of the 12th International Conference*.
27. CONNOR, M., AND HERLOCKER, J. 2001. Clustering items for collaborative filtering. *In SIGIR Workshop on Recommender Systems*.
28. COOK, K. (e.d.). 2001. *Trust in Society*, New York: Russell Sage Foundation.
29. COSMIDES, L. AND TOOBY, J. 1992. "Cognitive Adaptations for Social Exchange," In *The Adapted Mind: Evolutionary Psychology and the Generation of Culture*, 163-228. New York: Oxford University Press.
30. COVER, T. AND HART, P. 1967. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27.
31. DASGUPTA, P. 2000. Trust as a Commodity. In *Trust: Making and Breaking Cooperative Relations*, edited by Diego Gambetta. Electronic edition, Department of Sociology, University of Oxford.
32. DENEUBOURG, J. L., 1990. The self-organizing exploratory patter of the Argentine ant. *Journal of Insect Behavior,* 3, 159-168.
33. DEUTSCH, M. 1962. "Cooperation and Trust. Some Theoretical Notes." in Jones, M.R. (ed) *Nebraska Symposium on Motivation*. Nebraska University Press.
34. DEUTSCH, M. 1973. *The Resolution of Conflict*. New Haven and London: Yale University Press.
35. DORIGO, M. 1992. *Optimization Learning and Natural Algorithms*. PhD thesis, Politecnico di Milano, Milan.
36. DORIGO, M. 2001. Ant algorithms solve difficult optimization problems, *Proceedings of the Sixth European Conference on Artificial Life*, vol. 2159 of *Lecture Notes in Artificial Intelligence*, 11-22. Springer-Verlag, Berlin.
37. DORIGO, M., BONABEAU, E., AND THERAULAZ, G., 2000. Ant Algorithms and

Stigmergy. *Future Generation Computer Systems*, 16(8), 851-871.
38. DORIGO, M., DI CARO, G., AND GAMBARDELLA, L. M. 1999. Ant algorithms for discrete optimization. *Artifial Life*, 5(2): 137-172.
39. DORIGO, M., AND GAMBARDELLA, L. M. 1997. Ant Colony System: A cooperative learning approach to the travelling salesman problem. *IEEE Transaction on Evolutionary Computation*. 1(1): 53-66.
40. DORIGO, M., MANIEZZO, V., AND COLORNI, A. 1991. The Ant System: An autocatalytic optimizing process. Technical Report 91-016, Politecnico di Milano, Milan.
41. DORIGO, M., MANIEZZO, V., AND COLORNI, A. 1996. Ant System: Optimization by a colony of cooperating agents. *IEEE Transaction on Systems, Man, and Cybernetics*, 26(1): 29-41.
42. DORIGO, M. AND STÜTZLE, T. 2004. *Ant Colony Optimization*. MIT Press.
43. DORIGO, M. AND STÜTZLE, T. 2002. The ant colony optimization metaheuristic: Algorithms, applications and advances. *Handbook of Metaheuristics*, *International Series in Operations Research and Management Science.* 251-285. Norwell, MA.
44. DORIGO, M., STÜTZLE, T., AND DI CARO, G., 2000. Special issue on "Ant Algorithms". *Future Generation Computer Systems*, 16, 851-956.
45. DUMBILL, E. 2002. Finding friends with XML and RDF. IBM's XML Watch.
46. FREY, B. J., AND DUECK, D. 2007. Clustering by passing messages between data points. *Science*, 307.
47. FUKUYAMA, F. 1996. Trust: The Social Virtues and The Creation of Prosperity. New York: Free Press.
48. GAMBETTA, D. (1990). Can We Trust? In *Trust: Making and Breaking Cooperative Relations*, edited by Diego Gambetta. Electronic edition, Department of Sociology, University of Oxford. http://www.sociology.ox.ac.uk/papers/trustbook.html
49. GARTON, L., HAYTHORNTHWAITE, C., WELLMAN, B. 1997. Studying Online Social Networks. *Journal of Computer Mediated Communication* 3(1).
50. GIRVAN, M., AND NEWMAN, M. 2002. Community Structure in Social and Biological Networks, *Proceedings of the National Academy of Sciences*, USA.
51. GOLBECK, J. 2005. Computing and Applying Trust in Web-Based Social Networks. PhD thesis, Department of Computer Science, University of Maryland, College Park.
52. GOLBECK, J. 2002. Evolving Strategies for the Prisoner's Dilemma, *Advances in Intelligent Systems, Fuzzy Systems, and Evolutionary Computation*. February 2002: 299-306.
53. GOLBECK, J. 2006. Generating predictive movie recommendations from trust in social networks, *In: Trust Management, 4th International Conference, iTrust 2006*, Pisa, Italy, May 16-19, 93–104.
54. GOLDBERG, K., ROEDER, T., GUPTA, D., AND PERKINS, C. 2001. Eigentaste: A constant time collaborative filtering algorithm. *Journal Information Retrieval*, 4(2):133–151.
55. GOLEMBIEWSKI, R. T. AND MCCONKIE, M. 1975. The Centrality of Interpersonal Trust in Group Processes In *Theories of Group Processes*, edited by Cary Cooper.

Hoboken, NJ: Wiley.
56. GOLUB, G., AND REINSCH, C. 1970. Singular value decomposition and least squares solutions. *Numerische Mathematik*, 14(5):403–420.
57. GORI, M. AND WITTEN, I. 2005. The Bubble of Web Visibility. *Communications of ACM*. 48(3): 115-117.
58. GOSE, E., JOHNSONBAUGH, R., AND JOST, S. 1996. *Pattern Recognition and Image Analysis*. Prentice Hall.
59. GRAY, E., SEIGNEUR, J.M., CHEN, Y., AND JENSEN, C. 2003. Trust Propagation in Small Worlds. *Proceedings of the First International Conference on Trust Management*. LNCS 2692, Springer-Verlag.
60. GRISHCHENKO, V. S., 2004. Redefining Web-of-Trust: reputation, recommendations, responsibility and trust among peers. *Proceedings of the First Workshop on Friend of a Friend, Social Networking*, and the Semantic Web. Galway, Ireland.
61. GUHA, R., KUMAR, R., RAGHAVAN, P., AND TOMKINS, A. 2003. Propagation of Trust and Distrust. *Proceedings of the 13th Annual International World Wide Web Conference*, New York, NY.
62. GUHA, S., RASTOGI, R., AND SHIM, K. 1999. Rock: a robust clustering algorithm for categorical attributes. *In Proc. of the 15th Intl Conf. On Data Eng*.
63. HARDIN, R. 2002. *Trust & Trustworthiness*. New York: Russell Sage Foundation.
64. HARTIGAN, J. A. 1975. *Clustering Algorithms (Probability & Mathematical Statistics)*. John Wiley & Sons Inc.
65. HAYES, C., CUNNINGHAM, P. 2001. Smartradio-community based music radio. *Knowledge Based Systems*. 14(3-4): 197–201.
66. HERLOCKER, J., KONSTAN, J., TERVEEN, L. G., AND RIEDL, J. 2004. Evaluating collaborative filtering recommender systems, *ACM Transactions on Information Systems*, vol. 22, 5-53.
67. HERLOCKER, J., KONSTAN, J., AND RIEDL, J. 2000. Explaining collaborative filtering recommendations. *Proceedings of ACM 2000 Conference on Computer Supported Cooperative Work*, 241–250.
68. ISAKSSON, A., WALLMAN, M., GÖRANSSON, H. AND GUSTAFSSO, M. G. 2008. Cross-validation and bootstrapping are unreliable in small sample classification. *Pattern Recognition Letters* 29:1960–1965
69. JOHNSON, D. S. AND MCGEOCH, L. A. 1997. The travelling salesman problem: A case study in local optimization. *Local Search in Combinatorial Optimization*, 215-310, Chichester, UK.
70. JOLLIFFE, I. T. 2002. *Principal Component Analysis*. Springer.
71. JONES, J. H. AND HANDCOCK, M. S. 2003. Sexual contacts and epidemic thresholds. *Nature* 423:605-606.
72. JONKER, C. AND TREUR, J. Formal analysis of models for the dynamics of trust based on experiences. 1999. *Multi-Agent System Engineering, Proceedings of the 9th European Workshop on Modeling Autonomous Agents in a Multi-Agent World, MAAMAW'99*. Lecture Notes in Artificial Intelligence 1647.
73. JØSANG, A. The Right Type of Trust for Distributed Systems. 1996. *Proceedings of the 1996 New Security Paradigms Workshop*.
74. JØSANG, A., GRAY, E., KINATEDER, M. 2003. Analysing Topologies of Transitive

Trust, *Proceedings of the First International Workshop on Formal Aspects in Security & Trust* (FAST2003).

75. JUNG, Y. AND LEE, A. 2000. Design of a Social Interaction Environment for Electronic Marketplaces. *Proceedings of Designing Interactive Systems: Processes, Practices, Methods, & Techniques 2000*. 129-136.
76. KAMVAR, S. D., SCHLOSSER, M. T., GARCIA-MOLINA, H. 2003. The EigenTrust Algorithm for Reputation Management in P2P Networks. *Proceedings of the 12th International World Wide Web Conference*, May 20-24, 2003, Budapest, Hungary.
77. KANADE, P. M. AND HALL, L. O. 2003. Fuzzy Ants as a Clustering Concept. *In Proc. of the 22nd Int. Conf. of the North American Fuzzy Information Processing Soc.*, 227–232.
78. KAUTZ, H., SELMAN, B., AND SHAH, M. 1997. Combining Social Networks and Collaborative Filtering. *Communications of the ACM* 40(3): 63-65.
79. KEENEY, R. AND RAIFFA, H. 1976. Decisions with Multiple Objectives: Preferences and Value Tradeoffs. Cambridge, UK: Cambridge University Press.
80. KENT, S. AND ATKINSON, R. 1998. Security Architecture for the Internet Protocol. RFC 2401.
81. KUMAR**,** S. AND SINGH, M. 2012. Adaptive and dynamic load balancing in grid using ant colony optimization. *International Journal of Engineering and Technology*. 4(4): 167-174.
82. LATHIA, N., HAILES, S., AND CAPRA, L. 2008. The effect of correlation coefficients on communities of recommenders. In *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing*, New York, NY, USA.
83. LAWLER, E. L., LENSTRA, J. K., RINNOOY KAN, A. H. G., AND SHMOYS, D. B. 1985. *The Travelling Salesman Problem*, Chichester, UK.
84. LEVIN, R. 2003. *Advogato Trust Metric.* PhD thesis, UC Berkeley, USA.
85. LEVIN, R. AND AIKEN, A. 1998. Attack resistant trust metrics for public key certification. *7th USENIX Security Symposium*. January 1998, San Antonio, Texas.
86. LIN, W. AND ALVAREZ, S. 2004. Efficient adaptive-support association rule mining for recommender systems. *Data Mining and Knowledge Discovery Journal*, 6(1).
87. MAES, P. AND KOZIEROK, R. 1994. Agents that reduce work and information overload. *Communications of the ACM*. 37(7): 30-40.
88. MAHMOOD, T., RICCI, F. 2009. Improving recommender systems with adaptive conversational strategies. *Proceedings of the 20th ACM Conference on Hypertext and Hypermedia*, 73–82. Torino, Italy.
89. MAHMOOD, T., RICCI, F. 2007. Towards learning user-adaptive state models in a conversational recommender system. *In A. HINNEBURG (ed.) LWA 2007: Lernen - Wissen - Adaption, Halle, September 2007, Workshop Proceedings*, 373–378. Martin-Luther-University Halle-Wittenberg.
90. MARSH, S. 1992. "Trust and Reliance in Multi-Agent Systems: A Preliminary Report" *4th European Workshop on Modeling Autonomous Agents in a Multi-Agent World*. Lecture Notes in Computer Science 830.

91. MARSH, S. 1994. Formalising Trust as a Computational Concept. PhD thesis, Department of Mathematics and Computer Science, University of Stirling.
92. MASSA, P. 2006. A survey of trust use and modeling in current real systems, *Trust in E-Services: Technologies, Practices, and Challenges*, Idea Group, Inc.
93. MASSA, P. AND AVESANI, P. 2004. Trust-aware collaborative filtering for recommender systems Metrics. In *Proc. of Federated Int. Conference on the Move to Meaningful Internet: CoopIS, DOA, ODBASE*. 492-508.
94. MASSA, P. AND AVESANI, P. 2007. Trust-aware recommender systems. *Proceedings of the 2007 ACM Conference on Recommender Systems*. 17-24. Minneapolis, MN, USA.
95. MASSA, P. AND AVESANI, P. 2007. Trust Metrics on Controversial Users: Balancing between Tyranny of the Majority and Echo Chambers. *International Journal on Semantic Web and Information Systems*. 3(1).
96. MAURER, U. 1996. Modelling a public-key infrastructure. *Proceedings of Computer Security - ESORICS'96*. Springer-Verlag.
97. MCCABE, K. A., RIGDON, M. L., AND SMITH, V. L. 2003. Positive Reciprocity and Intentions in Trust Games. *Journal of Economic Behavior and Organization*.
98. MENDES, S. AND HUITEMA, C. 1995. A new approach to the X.509 framework: Allowing a global authentication infrastructure without a global trust model. *Proceedings of the 1995 Internet Society Symposium on Network and Distributed System Security*.
99. MILGRAM, S. 1967. The small world problem. *Psychology Today* 2, 60–67.
100. MOBASHER, B., DAI, H., LUO, T., AND NAKAGAWA, M. 2001. Effective personalization based on association rule discovery from web usage data. *In Workshop On Web Information And Data Management, WIDM '01*.
101. MONTANER, M., LÓPEZ, B., DE LA ROSA, J. L. 2003. "A taxonomy of recommender agents on the internet." *Artificial Intelligence Review* 19(4): 285–330.
102. MONTGOMERY, J. 1991. "Social Networks and Labor-Market Outcomes: Toward an Economic Analysis." *American Economic Review* 81(5): 1407-1418.
103. MUI, L., MOHTASHEMI, M., AND HALBERSTADT, A. 2002. A Computational Model of Trust and Reputation. *Proceedings of the 35th Hawaii International Conference on System Sciences 2002*.
104. NADI, S., SARAEE, M. H., JAZI, M. D., AND BAGHERI, A. 2011. FARS: Fuzzy Ant based Recommender System for Web Users, *International Journal of Computer Science Issues*, 8(1).
105. NEJDL, W., OLMEDILLA, D., AND WINSLETT, M. 2004. PeerTrust: Automated Trust Negotiation for Peers on the Semantic Web, *Proceedings of the Workshop on Secure Data Management in a Connected World (SDM'04) in conjunction with 30th International Conference on Very Large Data Bases*, August.-September 2004, Toronto, Canada.
106. NEWMAN, M. E. J. 2002. The spread of epidemic disease on networks. *Physical Review* E 66 (016128).
107. NIKOVSKI, D. AND KULEV, V. 2006. Induction of compact decision trees for personalized recommendation. *In SAC '06: Proceedings of the 2006 ACM symposium on Applied computing*, 575–581, New York, NY, USA.

108. NING, X. AND KARYPIS, G. 2011. SLIM: Sparse Linear Methods for Top-N Recommender Systems, *Proceedings of the 11th IEEE International Conference on Data Mining*, Vancouver, BC, Canada.
109. NOWAK, M. A. AND SIGMUND, K. 2000. Cooperation versus Competition. *Financial Analyst Journal*, July/August:13-22.
110. O'DONOVAN, J. AND SMYTH, B. 2007. Trust in Recommender Systems. *Proceedings of the 10th International Conference on Intelligent User Interfaces*, 167-174. San Diego, CA, USA.
111. PAGE, L., BRIN, S., MOTWANI, R., AND WINOGRAD, T. 1998. The PageRank citation ranking: Bringing order to the web. *Technical Report 1998*, Stanford University, Stanford, CA.
112. PAGEL, M., ERDLY, W., AND BECKER, J. 1987. Social networks: we get by with (and in spite of) a little help from our friends. *Journal of Personality and Social Psychology*, 53(4):793-804.
113. PAN, W. AND CHEN, L., 2013. CoFiSet: Collaborative Filtering via Learning Pairwise Preferences over Item-sets, *Proceedings of SIAM International Conference on Data Mining*, 180-188, Austin, Texas, USA.
114. PAPAGELIS, M., PLEXOUSAKIS, D., AND KUTSURAS, T. 2005. Alleviating the sparsity problem of collaborative filtering using trust inferences, *Proceedings of the Third International Conference on Trust Management*, 224-239, Paris, France.
115. PHAM, M. C., CAO, Y., KLAMMA, R., AND JARKE, M. 2011. A clustering approach for collaborative filtering recommendation using social network analysis. *Journal of Universal Computer Science*, 17(4):583-604.
116. POLLOCK, G. B. AND DUGATKIN, L. A. 1992. Reciprocity and the Evolution of Reputation. *Journal of Theoretical Biology*. 159: 25-37.
117. PREECE, J. 2000. *Online Communities: Designing Usability, Supporting Sociability*. Chichester, UK: John Wiley & Sons.
118. PYLE, D. 1999. *Data Preparation for Data Mining*. Morgan Kaufmann, Second Edition.
119. QUINLAN, J. R. 1986. Induction of decision trees. *Machine Learning*, 1(1):81–106.
120. REINELT, G. 1994. The Travelling Salesman: Computational Solutions for TSP Applications, *Lecture Notes in Computer Science.* Berlin, Springer-Verlag.
121. REITER, M. K. AND STUBBLEBINE, S. G. 1998. Resilient authentication using path independence. *IEEE Transactions on Computers*. 47, 12 (Dec.): 1351–1362.
122. RESNICK, P. AND VARIAN, H.R. 1997. Recommender systems. *Communications of the ACM* 40(3): 56–58.
123. RESNICK, P., ZECKHAUSER, R., FRIEDMAN, E., AND KUWABARA, K. 2000. Reputation Systems. *Communication of the ACM*, 43(12).
124. RICCI, F. 2002. Travel recommender systems. *IEEE Intelligent Systems*.17(6): 55–57.
125. RICCI, F., CAVADA, D., MIRZADEH, N., AND VENTURINI, A. 2006. Case-based travel recommendations. *In: D.R. FESENMAIER, K. WOEBER, H. WERTHNER (eds.) Destination Recommendation Systems: Behavioral Foundations and Applications*, 67–93.

126. RICCI, F., ROKACH, L., SHAPIRA, A. B., AND KANTOR, A. P. B. 2010. *Recommender systems handbook*, Springer-Verlag, New York, NY, USA.
127. RICHARDSON, M., AGRAWAL, R., DOMINGOS, P. 2003. Trust Management for the Semantic Web. *Proceedings of the Second International Semantic Web Conference*. Sanibel Island, Florida.
128. ROKACH, L. AND MAIMON, O. 2008. Data Mining with Decision Trees: Theory and Applications, *World Scientific Publishing*.
129. SARWAR, B. *ET AL*. 2002. Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering. *In Proceedings of the Fifth International Conference on Computer and Information Technology*.
130. SARWAR, B. M., KARYPIS, G., KONSTAN, J. A., AND RIEDL, J. T. 2000. Application of dimensionality reduction in recommender systems a case study. *In ACM WebKDD Workshop*.
131. SCHAFER, J., FRANKOWSKI, D., HERLOCKER, J., AND SEN, S. 2007. *Collaborative filtering recommender systems*. 291–324.
132. SCHWARTZ, B. 2004. *The Paradox of Choice*. ECCO, New York.
133. SHANI, G., HECKERMAN, D., AND BRAFMAN, R.I. 2005. An mdp-based recommender system. *Journal of Machine Learning Research*. 1265–1295.
134. SHANKAR, N. AND ARBAUGH, W. 2002. On Trust for Ubiquitous Computing. *Workshop on Security in Ubiquitous Computing, UBICOMP 2002*, Gteborg Sweden.
135. SHAPIRO, S. 1987. Social Control of Impersonal Trust. *The American Journal of Sociology*, 93(3): 623-658.
136. SHARMA, R., SINGH, M., MAKKAR, R., KAUR, H., AND BEDI, P. 2007. Ant Recommender: Recommender system inspired by ant colony, *in Proceedings of International Conference on Advances in Computer Vision and Information Technology*, 361-369.
137. SHNEIDERMAN, B. 2000. Designing websites to enhance online trust. *Communications of the ACM*. 43(12): 81-83.
138. SINHA, R. R. AND SWEARINGEN, K. 2001. Comparing recommendations made by online systems and friends. *In DELOS Workshop: Personalisation and Recommender Systems in Digital Libraries.*
139. SOBECKI, J. 2008. Colony Metaphor Applied in User Interface Recommendation. *New Generation Computing*. 26(3): 277-293.
140. SPERTUS, E., SAHAMI, M., AND BUYUKKOKTEN, O. 2005. Evaluating similarity measures: A large scale study in the orkut social network. *In Proceedings of the 2005 International Conference on Knowledge Discovery and Data Mining (KDD-05)*.
141. STÜTZLE, T. 1999. *Local Search Algorithms for Combinatorial Problems: Analysis, Improvements, and New Applications*, vol. 220 of *DISKI*. Sankt Augustin, Germany, Infix.
142. STÜTZLE, T. AND HOOS, H. H., 1997, The *MAX-MIN* Ant System and local search for the travelling salesman problem. *Proceedings of the 1997 IEEE International Conference on Evolutionary Computation*, 309-314. Piscataway, NJ.
143. STÜTZLE, T. AND HOOS, H. H., 2000, *MAX-MIN* Ant System. *Future Generation Computer Systems*, 16(8): 889-914.

144. SZTOMPKA, P. 1999, *Trust: A Sociological Theory*, Cambridge: Cambridge University Press.
145. TARAH, A. AND HUITEMA, C. 1992. Associating metrics to certification paths. *Computer Security*. 175–189.
146. USLANER, E. 2002. The Moral Foundations of Trust. Cambridge, UK: Cambridge University Press.
147. VICTOR, P., CORNELIS, C., COCK, M. D., AND TEREDESAI, A. M., 2008. Key figure impact in trust-enhanced recommender systems, *AI Commun.,* vol. 21, 127-143.
148. VICTOR, P., CORNELIS, C., COCK, M. D., AND TEREDESAI, A. M., 2011. Trust- and distrust-based recommendations for controversial reviews, *Intelligent Systems,*48-55.
149. WALLACH, D. S., BALFANZ, D., DEAN, D., AND FELTEN, E. W. 1997. Extensible Security Architectures for Java. *Sixteenth Symposium on Operating Systems Principles*.
150. WANG, B., CHEN, X., AND CHANG, W. 2013. A light-weight trust-based QoS routing algorithm for ad hoc networks. *Science Direct*.
151. WASSERMAN, S. AND FAUST, K. 1994. *Social network analysis*: Methods and applications. Cambridge: Cambridge University Press.
152. WATTS, D. 1999. *Small Worlds: The Dynamics of Networks between Order and Randomness*. Princeton, NJ: Princeton University Press.
153. WELLMAN, B. 1982. Studying personal communities. *Social structure and network analysis*, edited by P. Marsden & N. Lin, 61-80. Beverly Hills, CA: Sage.
154. XUE, G., LIN, R., YANG, C., XI, Q., ZENG, W., YU, J., AND CHEN, Z. 2005. Scalable collaborative filtering using cluster-based smoothing. *In Proceedings of the 2005 SIGIR*.
155. YANG, L., QIN, Z., WANG, C., AND LIU, Y. 2010. A P2P reputation model based on Ant Colony Algorithm. *International Conference on Communications, Circuits and Systems,* 236-240.
156. YANIV, I. AND KLEINBERGER, E. 2000. Advice taking in decision making: Egocentric discounting and reputation formation. *Organizational Behavior and Human Decision Processes*. Nov; 83(2):260-281
157. ZHANG, Y., ZHANG, M., LIU, Y., MA, S., AND FENG, S. 2013. Localized Matrix Factorization for Recommendation based on Matrix Block Diagonal Form, *Proceedings of the 22nd International Conference on World Wide Web*, 1511-1520.
158. ZIEGLER, C. N. 2005. *Towards Decentralized Recommender Systems*. PhD thesis, Albert-Ludwigs-Universität Freiburg, Freiburg i.Br., Germany.
159. ZIEGLER, C. N., LAUSEN, G. 2004. Spreading activation models for trust propagation. *Proceedings of the IEEE International Conference on e-Technology, e- Commerce, and e-Service*, Taipei, Taiwan.