Information Aggregation in Quantized Consensus, Recommender Systems, and Ranking

Shang Shang

A Dissertation
Presented to the Faculty of Princeton University in Candidacy for the Degree of Doctor of Philosophy

Recommended for Acceptance by the Department of Electrical Engineering
Advisers: Sanjeev R. Kulkarni and Paul W. Cuff

June, 2014
Abstract

Information aggregation is the science of collecting and aggregating knowledge from data. With the development of large scale datasets, the amount of information is growing rapidly. In recent years, the problem of information aggregation has received considerable attention, and finds applications in multiple disciplines. This dissertation addresses a variety of problems in information aggregation, including quantized consensus, recommender systems, and ranking.

This dissertation starts with investigating a class of distributed quantized consensus algorithms for arbitrary networks. An upper bound on the convergence time of the algorithms is derived for an arbitrary graph of size $N$. Inspired by this class of gossip consensus algorithms and Google’s PageRank, and motivated by the development of group-based social networks, a privacy preserving recommender system based on groups is proposed. The main idea is to use groups as a natural middleware to preserve users’ privacy. A novel hybrid collaborative filtering model based on random walks is constructed to provide recommendation and prediction to group members. Lastly, the error probability of ranking algorithms equipped with differential privacy is analyzed, and upper bounds on the error rates for arbitrary positional ranking rules are derived.
Acknowledgements

In February 2009, I received the admission letter from Princeton University. In the same month, Julien came to China as an exchange student. Over the next seven months, we met, and fell in love. I finished my undergraduate study in Shanghai, and he received his master’s degree in Lyon. I flew to Princeton, and he followed me here. People say French are romantic. That was the sweetest thing he did for me. First and foremost, I thank my husband, Julien, for his support, company and love.

This dissertation would not exist if without the tremendous support and generous help from both of my advisers, Professor Sanjeev Kulkarni and Professor Paul Cuff. They guided me to explore the beautiful world of science and engineering, and to investigate many interesting topics in information sciences and systems. Their wisdom enlightened every single step of my progress and study. I cannot thank them enough for showing me how to be a good researcher with passion and perfection, and how to be a good person with sincerity and care.

I would like to express my sincere gratitude to Professor Pan (Ben) Hui. He is more than a collaborator, a dissertation reader, and a supervisor; he is also a friend. He provided precious intellectual inputs not only during my two summer internships at Deutsche Telekom Innovation Labs, but ever afterwards. In a conference trip to Taiwan, Ben introduced his brother, Dr. Yuk Hui to me. The collaboration with Yuk on group-based recommender systems was very inspiring and enlightening.

I would like to thank Professor Vincent Poor for serving on my general exam committee, and Professor Peter Ramadge and Professor Prateek Mittal for serving on my thesis committee as non-readers.

Special thanks go to Tiance (Mark) Wang: We joined Princeton the same year, and both of us are co-advised by Professor Kulkarni and Professor Cuff. Chapter 4 of this dissertation focused on differential privacy has been contributed heavily by Mark. This project provided me the privilege to work closely with Mark, and the
day-to-day interaction was extremely productive and pleasant. I would like to thank my group mates, Zhuo Zhang, Marc Reinhardt, Chen (Eva) Song, Curt Schieler, Jieqi Yu, Haipeng Zheng and Guanchun (Arvid) Wang. Through collaboration and conversation with them, I got inspired and motivated in both my research and life. I would also like to acknowledge with much appreciation the staff of the Electrical Engineering Department for their support and assistance: Colleen Conrad, Sarah McGovern, Roelie Abdi, Beth Jarvie, Lisa Lewis and Dorothy Coakley.

I owe my deepest gratitude to my parents. Dengping and Qingge, they are always there to support me and guide me with no conditions. I would also like to thank my elder cousins, William and Jay, by whom I was (gently) bullied, with whom I grew up and whom I deeply love.
To my family.
# Contents

Abstract .................................................. iii
Acknowledgements ....................................... iv
List of Tables ........................................... x
List of Figures ............................................ xi

1 Introduction ........................................... 1
   1.1 Distributed Quantized Consensus ...................... 2
   1.2 Privacy Preserving Recommender Systems ............. 3
   1.3 Differentially Private Rank Aggregation ............... 4

2 Distributed Quantized Consensus ..................... 6
   2.1 Background ........................................... 6
   2.2 Binary Voting and Quantized Consensus ............... 8
      2.2.1 Binary Voting Consensus ....................... 9
      2.2.2 Quantized Consensus .......................... 11
   2.3 Convergence Time Analysis ............................ 12
      2.3.1 Definition and Notation ....................... 13
      2.3.2 Hitting Time and Meeting Time on Weighted Graph 14
      2.3.3 An Upper Bound on Binary Consensus ........... 22
      2.3.4 An Upper Bound on Quantized Consensus .......... 25
   2.4 Simulation Results ................................... 27
5 Conclusion and Future Work 83

5.1 Revisit the Contribution of the Dissertation 83

5.2 Future Research Directions 84

Bibliography 86
List of Tables

3.1  Average percentile results obtained by 5-fold cross-validation for recommendation. .................................................. 63

3.2  Average TOPK results obtained by 5-fold cross-validation for recommendation ...................................................... 64
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Sensor nodes deployed in decision-making.</td>
<td>7</td>
</tr>
<tr>
<td>2.2</td>
<td>Update rules for distributed binary vote [6]. The figure shows update principles: when opposite “strong opinion”s meet, they both turn into “weak opinion”s; “strong opinion” affects “weak opinion”; and swap principle.</td>
<td>10</td>
</tr>
<tr>
<td>2.3</td>
<td>A simple example of effective resistance.</td>
<td>16</td>
</tr>
<tr>
<td>2.4</td>
<td>A complete graph.</td>
<td>27</td>
</tr>
<tr>
<td>2.5</td>
<td>A star network.</td>
<td>28</td>
</tr>
<tr>
<td>2.6</td>
<td>A line graph.</td>
<td>30</td>
</tr>
<tr>
<td>2.7</td>
<td>A lollipop graph.</td>
<td>31</td>
</tr>
<tr>
<td>2.8</td>
<td>Simulation results on average convergence time (squares) of binary consensus and quantized consensus versus the size of networks.</td>
<td>33</td>
</tr>
<tr>
<td>2.8</td>
<td>Simulation results on average convergence time (squares) of binary consensus and quantized consensus versus the size of networks (continued).</td>
<td>34</td>
</tr>
<tr>
<td>2.8</td>
<td>Simulation results on average convergence time (squares) of binary consensus and quantized consensus versus the size of networks (continued).</td>
<td>35</td>
</tr>
<tr>
<td>2.8</td>
<td>Simulation results on average convergence time (squares) of binary consensus and quantized consensus versus the size of networks (continued).</td>
<td>36</td>
</tr>
<tr>
<td>2.8</td>
<td>Simulation results on average convergence time (squares) of binary consensus and quantized consensus versus the size of networks (continued).</td>
<td>37</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
<td></td>
</tr>
<tr>
<td>2.8</td>
<td>Simulation results on average convergence time (squares) of binary consensus and quantized consensus versus the size of networks (continued)</td>
<td></td>
</tr>
<tr>
<td>3.1</td>
<td>Group based social networks</td>
<td></td>
</tr>
<tr>
<td>3.2</td>
<td>The structure of the group-based privacy preserving recommender system</td>
<td></td>
</tr>
<tr>
<td>3.3</td>
<td>The pairwise comparison graph for $\pi_1 = {1, 2, 3}$, $\pi_2 = {2, 1, 3}$, $\pi_3 = {3, 2, 1}$</td>
<td></td>
</tr>
<tr>
<td>3.4</td>
<td>Example of a recommendation graph for inter-group recommendations</td>
<td></td>
</tr>
<tr>
<td>3.5</td>
<td>The group distribution of MovieLens datasets</td>
<td></td>
</tr>
<tr>
<td>4.1</td>
<td>An example of Petrie polygon (skew orthogonal projections) of three candidates. Three hyperplanes, under Borda count ranking rule, separate the simplex into six polytopes</td>
<td></td>
</tr>
<tr>
<td>4.2</td>
<td>Error rate vs $\epsilon$</td>
<td></td>
</tr>
<tr>
<td>4.3</td>
<td>Error rate vs number of voters</td>
<td></td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Information aggregation is the process of collecting and summarizing data, often from multiple sources. Data are combined for drawing inferences or some other purpose. Information aggregation has a variety of applications. Take the decision-making sensor network as an example: Sensors are deployed to measure if in a certain region an earthquake is happening or not. Those sensors are scattered randomly or deterministically, coordinating among themselves to form a communication network. They are operated under severe resource constraints. The communication between sensors is highly constrained, and the computation capability and energy of sensors are usually limited. It may turn out that all sensors cannot update their observations synchronously. A majority opinion is desired in order to decide if the event is happening, and good distributed aggregation algorithms attempt to overcome the computation and communication restrictions.

Aggregation tasks can be more complex than a binary decision. For example, a metasearch engine collects search results from multiple search engines and returns a single list. Different metasearch engines return different lists, resulting from the quantity of engines that are used and aggregation methods for generating the ranking results.
We live in a world with ever-growing data. Information aggregation involves more than summarizing the data. It is different from statistics and traditional database operations. Sometimes the aggregation task requires compiling and interpreting multidimensional data in order to make a decision or predict a pattern. Information aggregation overlaps with the disciplines of machine learning and information retrieval. A good example of the overlap is found in collaborative filtering (CF) for recommender systems, which is the technique for making predictions of a user's taste by collecting information from other likeminded users as well as the target user.

The aggregation of massive information also raises concerns about privacy issues. Companies such as Google and Facebook are accumulating and recording enormous amount of personal data. The potential risk in sensitive data leakage is disconcerting. Users normally have no choice but to trust the service provider to keep their sensitive personal profile safe. However, it is not always accomplished. In 2006, Netflix launched a data competition, and released 100 million movie ratings from half a million users. Researchers subsequently demonstrated that individual users from this “sanitized” dataset could be identified by comparing to the Internet Movie Database (IMDb). This raises the privacy concerns about sharing honest opinions.

This dissertation is focused on three interesting topics: distributed quantized consensus, privacy preserving recommender systems, and differentially private ranking algorithms. We are interested in both performance and privacy. Overviews for each chapter are given in the following three sections.

1.1 Distributed Quantized Consensus

Quantized consensus is an essential problem in distributed computing. It finds emerging applications in load balancing, sensor networks, wireless communication networks, multi-agent coordination, etc. A consensus algorithm is a procedure that requires a
number of agents to communicate through designed protocols and asymptotically reach an agreement (consensus), given their initial states/values.

In Chapter 2, we analyze a class of quantized consensus algorithms proposed in [6] and in [34]. Nodes randomly and asynchronously update local estimates and exchange information. In [6], the author proposed a binary voting algorithm, where all the nodes in the network vote “yes” or “no”. The algorithm reaches consensus on the initial majority opinion almost surely. A more general distributed integer averaging algorithm was proposed in [34].

In Chapter 2, polynomial upper bounds of $O(N^3 \log N)$ for both binary voting and quantized consensus algorithms are derived for the expected convergence time on an arbitrary graph of size $N$. In the analysis, we use the theory of electric networks, random walks, and couplings of Markov chains. The analysis for arbitrary graphs is extended to a tighter bound for certain network topologies by computing the effective resistance between a pair of nodes on the graph. Simulations on special graphs such as star networks, line graphs, lollipop graphs, and Erdős-Rényi random graphs are performed in order to validate the analysis. The main content of this chapter is also featured in [55].

1.2 Privacy Preserving Recommender Systems

With the rise of e-commerce, recommender systems have been studied intensively in the context of collaborative filtering (CF) techniques. Early generations of recommender system have already been commercialized and have been widely used. Recommender systems serve as an important component of online retail and Video on Demand (VoD) services such as Amazon and Netflix. These systems give customized recommendations to online users on books, movies, and commodities according to their previous preference data. Personalized service is a two-edged sword in
the emerging development of social media. Personalization provides users with con-
veniences while at the same time making them targets of marketing, and creating
potential privacy concerns. For example, a shopping website that an online shopper
has visited once might keep appearing on the advertising block for days when one is
browsing other web pages.

In Chapter 3, we design a framework for using groups as a natural middleware
to recommend products to users. The goal is to protect users from unreliable service
providers, and to mitigate users’ fear of potential intrusions of privacy by keeping a
certain amount of anonymity. A distributed preference exchange algorithm is pro-
posed to ensure anonymity of the data, wherein the effective size of the anonymity set
asymptotically approaches the group size with time. We construct a hybrid collabora-
tive filtering model to provide recommendations and predictions to group members.
The main content of this chapter is also featured in [57].

1.3 Differentially Private Rank Aggregation

With the increasing interest in social networks and the availability of large datasets,
rank aggregation has received great attention in many areas, such as economics,
politics, and computer science. From the NBA’s Most Valuable Player to Netflix’s
recommended movies, from web search to committee decision making, voting and
ranking is ubiquitous. Informally, rank aggregation is the problem of aggregating a set
of full or partial rankings of a fixed set of candidates into a single consensus ranking.
However, aggregated ranking results may reveal a great deal about an individual’s
preferences. That is, some information about personal preference may be inferred
from the aggregated result. For example, a voter may fear voting because the vote
may be discovered if others collude.
In Chapter 4, we incorporate differential privacy in rank aggregation algorithms. Differential privacy is a framework to obscure individual entries in a database. The main idea is to add random noise to the aggregated ranking profile so that a single voter’s preference is beyond inference. Artificial noise, on the other hand, creates possible non-conformity between the true ranking results and the differentially private results. General upper bounds on the ranking error rates are derived for arbitrary positional ranking rules. Moreover, we show that the asymptotic error rates approach zero when the number of voters goes to infinity for any ranking rules with a fixed number of candidates. An example of Borda Count is given to show how to extend the proposed analysis to derive tighter upper bounds on the error rates for specific positional rules. Lastly, simulations are performed to validate the analysis.
Chapter 2

Distributed Quantized Consensus

In this chapter, we first briefly review some of related work on the quantized consensus problem in Section 2.1. We then describe the binary voting and quantized consensus algorithms proposed in [6] and [34] in detail, and formulate the convergence speed problem in Section 2.2. In Section 2.3 we present a polynomial bound for this class of algorithms. Section 2.4 gives examples on how to derive an upper bound on the given topology of the network. Simulation results are provided to justify the analysis.

2.1 Background

Quantized consensus has gathered a great deal of research interest [6, 9, 12, 13, 15, 28, 34, 37]. It models averaging in a network with a limited capacity channel [34]. Distributed algorithms are attractive due to their flexibility, simple deployment and the lack of central control. This problem is of interest in the context of coordination of autonomous agents, estimation, distributed data fusion on sensor networks, peer-to-peer systems, etc. [9,19]. It is especially relevant to remote and extreme environments where communication and computation are limited, for example, in a decision-making sensor network [20], as shown in Fig. 2.1.
In [6] and in [34], a class of quantized consensus algorithms are introduced. Nodes update local estimates and exchange information randomly and asynchronously. In [6], the author proposed a binary voting algorithm where all the nodes in the network vote “yes” or “no”. The algorithm reaches consensus on the majority opinion almost surely. However, the authors did not bound the convergence time. In [46], the authors studied the convergence speed in the special case of regular graphs for a similar distributed binary consensus algorithm. Draief and Vojnovic [19] derived an expected convergence time bound on binary interval consensus depending on the second largest eigenvalue of a doubly stochastic matrix characterizing the algorithm and voting margin, yet no specific bound is provided for an arbitrary graph. An $O(N^4 \log N)$ bound is given in [56] on the binary voting convergence speed, where $N$ is the number of nodes in the network. A more general distributed quantized integer averaging algorithm was proposed in [34]. Unlike the distributed algorithm in [48], where the sum of values in the network is not preserved, Kashyap et al. proposed an algorithm guaranteeing convergence with limited communication, more specifically, only involving quantization levels. This is a realistic approach in a large-scale network where memory is limited, communication between nodes is expensive and no central control is available to the network. Analysis of the convergence time on the complete
graph and line graph is given in the original paper in [34], and an $O(N^5)$ bound was claimed in [70] by creating a random walk model.

In this chapter, unlike the *natural random walk* model analyzed in [70] (an incorrect model for this system), we utilize a *biased lazy random walk model* to analyze the multi-level quantized consensus problem with the use of Lyapunov functions [34]. By using the relation between the commuting time of a random walk and electric networks [3], we derive an upper bound on the hitting time of a biased random walk. Several coupled Markov processes are then constructed to help the analysis. Thus, we improve the state of art bound in [70] from $O(N^5)$ to $O(N^3 \log N)$. In [19], using different methods, the authors introduced a function $\delta(G, \alpha)$ depending on the graph structure and voting margin to provide an upper bound on the convergence time of the binary consensus algorithm, but did not provide a universal upper bound for arbitrary graphs. Unlike the convergence time bound in [19], which depends on the network topology and the eigenvalues of some contact rate matrices, our result provides a universal upper bound on the convergence time of quantized consensus.

### 2.2 Binary Voting and Quantized Consensus

A network is represented by a connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, 2, \ldots, N\}$ is the set of nodes and $\mathcal{E}$ is the set of edges. If nodes $i, j$ can communicate with each other, there is an edge between them, i.e., $(i, j) \in \mathcal{E}$. Denote $\mathcal{N}_i$ as the set of neighbors of node $i$.

Consider a network of $N$ nodes, labeled 1 through $N$. As proposed in [6, 9, 34], each node has a clock which ticks according to a rate 1 Poisson process. By the superposition property of Poisson process, this set-up is equivalent to a single global Poisson clock with a rate $N$ ticking at times $\{Z_k\}_{k \geq 0}$. The communication and update
of states only occur at $\{Z_k\}_{k \geq 0}$. When the clock of node $i$ ticks, $i$ randomly chooses a neighbor $j$ from the set $N_i$. We say edge $(i, j)$ is activated.

In the rest of this section, we will describe the distributed binary voting consensus algorithm [6] and quantized consensus algorithm [34]. We are interested in the performance of this class of algorithms on arbitrary graphs.

### 2.2.1 Binary Voting Consensus

Initially, each node on the connected graph $G$ has a vote, *strong positive* or *strong negative* (or no vote at all). Assuming that a majority opinion exists, the objective for the binary voting consensus problem is to have each node settle on the majority in a distributed manner.

Let $S^{(i)}(t)$ denote the state of node $i$ at time $t$. $S^{(i)}(t) \in \{S^+, S^-, W^+, W^-, W^\pm, S^\pm\}$, representing *strong positive*, *strong negative*, *weak positive*, and *weak negative* respectively, where $S^\pm = \pm 2$ and $W^\pm = \pm 1$. For all $i \in \mathcal{V}$, $S^{(i)}(0)$ is initialized to the corresponding *strong positive* or *strong negative*. If $i$ does not have a vote at $t = 0$, it is randomly initialized to either *weak opinion*: $W^+$ or $W^-$. When two nodes $i$ and $j$ with opposite *strong opinions* exchange information, they both update to *weak opinions*. Further update rules are as follows:

1. If $S^{(i)}(t) = S^{(j)}(t)$,
   $$S^{(i)}(t + 1) = S^{(j)}(t + 1) = S^{(i)}(t);$$

2. If $|S^{(i)}(t)| > |S^{(j)}(t)|$ and $S^{(i)}(t) \cdot S^{(j)}(t) < 0$,
   $$S^{(i)}(t + 1) = -S^{(j)}(t), S^{(j)}(t + 1) = S^{(i)}(t),$$
   and vice versa;

3. If $|S^{(i)}(t)| > |S^{(j)}(t)|$ and $S^{(i)}(t) \cdot S^{(j)}(t) > 0$,
   $$S^{(i)}(t + 1) = S^{(j)}(t), S^{(j)}(t + 1) = S^{(i)}(t),$$
   and vice versa;
Figure 2.2: Update rules for distributed binary vote [6]. The figure shows update principles: when opposite “strong opinion”s meet, they both turn into “weak opinion”s; “strong opinion” affects “weak opinion”; and swap principle.

4. If $S^{(i)}(t) = -S^{(j)}(t)$,
   \[ S^{(i)}(t + 1) = \text{sign} \left( S^{(j)}(t) \right), \]
   \[ S^{(j)}(t + 1) = \text{sign} \left( S^{(i)}(t) \right). \]

The update rules are illustrated in Fig. 2.2. In the binary voting consensus, a trivial meeting occurs when two nodes with identical or consecutive states exchange information, e.g. $(S^+, S^+)$, $(W^-, W^-)$, $(W^+, W^-)$, $(S^+, W^+)$, etc; otherwise we say a non-trivial meeting occurs. Let $|S^+|$ denote the number of the strong positive opinions and $|S^+(t)|$ denote the number of the strong positive opinions at time $t$. Note that this algorithm supposes that there is an odd number of nodes in the network, in order to guarantee convergence regardless of initial votes of nodes.

**Definition 1** (Convergence on Binary Voting Consensus). A binary voting reaches convergence if all states of nodes on the graph are positive or all states are negative.

A quick validation for this algorithm: we notice that the $S^+$ and $S^-$ will only annihilate each other when they meet, otherwise they just take random walks on the graph. So only the majority strong opinions will be left on the graph in the end. That is the reason why randomly assigning weak opinions to nodes with no initial vote does not affect the convergence to the majority opinion. We also notice that strong opinions can influence weak opinions as shown in Fig. 2.2. Eventually all
agents will take the sign of the majority *strong opinions*. Because the graph has finite size, and this Markov chain has finite states, convergence will happen in finite time almost surely.

2.2.2 Quantized Consensus

Without loss of generality, let us assume that all nodes hold integer values and the quantization step size is 1. Let $Q^{(i)}(t)$ denote the integer value of node $i$ at time $t$, with $Q^{(i)}(0)$ denoting the initial values. Define

$$Q_{\text{sum}} = \sum_{i=1}^{N} Q^{(i)}(0). \quad (2.1)$$

Let $Q_{\text{sum}}$ be written as $qN + r$, where $0 \leq r < N$. Then the mean of the initial value in the network $\frac{1}{N}Q_{\text{sum}} \in [q, q + 1)$. Thus either $q$ or $q + 1$ is an acceptable integer value for quantized average consensus (if the quantization level is 1).

**Definition 2** (Convergence on Quantized Consensus). A quantized consensus reaches convergence at time $t$, if for any node $i$ on the graph, $Q^{(i)}(t) \in \{q, q + 1\}$.

There are a few properties that are desired for the quantized consensus algorithm:

- **Sum conservation:**
  $$\sum_{i=1}^{N} Q^{(i)}(t) = \sum_{i=1}^{N} Q^{(i)}(t + 1). \quad (2.2)$$

- **Variation non-increasing:** if two nodes $i, j$ exchange information,
  $$|Q^{(i)}(t + 1) - Q^{(j)}(t + 1)| \leq |Q^{(i)}(t) - Q^{(j)}(t)|. \quad (2.3)$$

When two nodes $i$ and $j$ exchange information, without loss of generality, suppose that $Q^{(i)}(t) \leq Q^{(j)}(t)$. They follow the simple update rules below:
1. If $Q^{(j)}(t) - Q^{(i)}(t) \geq 2$, a non-trivial meeting occurs:

$$Q^{(i)}(t + 1) = Q^{(i)}(t) + 1, Q^{(j)}(t + 1) = Q^{(j)}(t) - 1.$$ 

2. If $Q^{(j)}(t) - Q^{(i)}(t) \leq 1$, a trivial meeting occurs:

$$Q^{(i)}(t + 1) = Q^{(j)}(t), Q^{(j)}(t + 1) = Q^{(i)}(t).$$

Similar to the argument in Section 2.2.1, we can view this random process as a finite state Markov chain. Because the variation decreases whenever there is a non-trivial exchange, convergence will be reached in finite time almost surely.

Remark 1: In this quantized consensus algorithm, we define convergence to be when all nodes reach two consecutive states. However, this definition cannot be used to solve the binary voting problem. If all nodes converge to $W^+$ and $W^-$ states, no conclusion can be made on the majority opinion.

Remark 2: In this section, the update rules allow the node values to change by at most 1. This is relevant to load-balancing systems where only one value can be exchanged in the channel at a time due to the communication limit [34]. Adjustments can be made for this class of quantized consensus algorithms, e.g. when two nodes exchange information, both nodes can update their value to the mean of the two. The analysis on the convergence time remains similar.

### 2.3 Convergence Time Analysis

The main results of this chapter are the following theorems:

**Theorem 1.** For a connected network of $N$ nodes, an upper bound for the expected convergence time of the binary voting consensus algorithm is $O(N^3 \log(N))$. 
Theorem 2. For a connected network of $N$ nodes, an upper bound for the expected convergence time of the quantized consensus algorithm is $O(N^3 \log(N))$.

We use the analogy of electric networks and random walks to derive the upper bound. Since the clock setting and edge selection strategies are the same in the binary voting algorithm and quantized consensus algorithm, their information exchange processes can be coupled. Hence the meeting time (defined below) is equal in both algorithms. Before deriving the bound on the convergence time, we first provide some definitions and notation that we will use and prove some useful lemmas in Section 2.3.1 and Section 2.3.2.

2.3.1 Definition and Notation

Definition 3 (Hitting Time). For a graph $G$ and a specific random walk, and $i, j \in V$, let $\mathcal{H}(i, j)$ denote the expected number of steps a random walk beginning at $i$ must take before reaching $j$. Define the “hitting time” of $G$ by $\mathcal{H}(G) = \max_{i,j} \mathcal{H}(i, j)$.

Definition 4 (Meeting Time). Consider two random walkers placed on $G$, and $i, j \in V$. At each tick of the clock, they move according to some joint probability distribution. Let $\mathcal{M}(i, j)$ denote the expected time for the two walkers starting from $i$ and $j$ respectively to meet at the same node or to cross each other through the same edge (if they move at the same time). Define the “meeting time” of $G$ by $\mathcal{M}(G) = \max_{i,j} \mathcal{M}(i, j)$.

Define a simple random walk on $G$, with transition matrix $P^S = (P_{ij})$ as follows:

- $P^S_{ii} := 0$ for $\forall i \in V$,
- $P^S_{ij} := \frac{1}{|\mathcal{N}_i|}$ for $(i, j) \in E$.

$\mathcal{N}_i$ is the set of neighbors of node $i$ and $|\mathcal{N}_i|$ is the degree of node $i$.

Define a natural random walk with transition matrix $P^N = (P_{ij})$ as follows:

- $P^N_{ii} = 1 - \frac{1}{N}$ for $\forall i \in V$,
• \( P^N_{ij} = \frac{1}{|N_i|} \) for (\( i, j \)) \( \in \mathcal{E} \).

Define a biased random walk with transition matrix \( P^B = (P_{ij}) \) as follows:

• \( P^B_{ii} := 1 - \frac{1}{N} - \sum_{k \in N_i} \frac{1}{|N_k|} \) for \( \forall i \in \mathcal{V} \),

• \( P^B_{ij} := \frac{1}{N} \left( \frac{1}{|N_i|} + \frac{1}{|N_j|} \right) \) for (\( i, j \)) \( \in \mathcal{E} \).

### 2.3.2 Hitting Time and Meeting Time on Weighted Graph

In this class of algorithms, we label the initial observations (states or values) by the nodes as \( \alpha_1, \alpha_2, ..., \alpha_N \). A random walk is a Markov process with random variables \( A_1, A_2, ..., A_t, ... \) such that the next state only depends on the current state. In the system setting, when the node \( i \)'s clock ticks, \( i \) randomly chooses one of its neighbor node \( j \) from the set \( N_i \) to exchange information. We notice that before any two observations \( \alpha_m, \alpha_n \) meet each other, they take random walks on the graph \( G \). Their marginal transition matrices are both \( P^B \). It may be tempting to think that they are taking the natural random walks as stated in [70]. Upon closer inspection, we find that there are two sources stimulating the random walk from \( i \) to \( j \), for all (\( i, j \)) \( \in \mathcal{E} \): one is active, initiated by node \( i \)'s clock, which leads to \( P^1_{ij} = P^N_{ij} \); the other one is passive, initiated by \( i \)'s neighbor \( j \), which leads to \( P^2_{ij} = P^N_{ji} \). Thus \( P_{ij} = P^1_{ij} + P^2_{ij} = P^B_{ij} \); i.e., the transition matrix is actually \( P^B \) instead of \( P^N \). Because of the system settings, two random walks \( \alpha_m, \alpha_n \) can only move at the same time if they are adjacent. Denote this joint random process as \( X \). Suppose \( \alpha_m \) is at node \( x \), and \( \alpha_n \) is at node \( y \).

For \( x \notin N_y \), and \( i \in N_x \), we have

\[
P_{X\text{joint}}(\alpha_m \text{ moves from } x \text{ to } i, \ \alpha_n \text{ does not move}) = P^B_{xi} - P_{X\text{joint}}(\alpha_m \text{ moves from } x \text{ to } i, \ \alpha_n \text{ moves}) = P^B_{xi}. \quad (2.4)
\]
Similar for $P_{X_{\text{joint}}} (\alpha_n \text{ moves from } y \text{ to } j, \alpha_m \text{ does not move})$, where $j \in \mathcal{N}_y$. Also,

$$P_{X_{\text{joint}}} (\alpha_m \text{ does not move, } \alpha_n \text{ does not move}) = 1 - \sum_{i \in \mathcal{N}_x} P^B_{xi} - \sum_{j \in \mathcal{N}_y} P^B_{yj}. \quad (2.5)$$

For $x \in \mathcal{N}_y$ and $i \neq y$ we have,

$$P_{X_{\text{joint}}} (\alpha_m \text{ moves from } x \text{ to } i, \alpha_n \text{ does not move}) = P^B_{xi} - P_{X_{\text{joint}}} (\alpha_m \text{ moves from } x \text{ to } i, \alpha_n \text{ moves}) = P^B_{xi}. \quad (2.6)$$

$$P_{X_{\text{joint}}} (\alpha_m \text{ moves to } y, \alpha_n \text{ moves to } x) = P^B_{xy} \quad (2.7)$$

$$P_{X_{\text{joint}}} (\alpha_m \text{ does not move, } \alpha_n \text{ does not move}) = 1 - \sum_{i \in \mathcal{N}_x} P^B_{xi} - \sum_{j \in \mathcal{N}_y} P^B_{yj} + P^B_{xy}. \quad (2.8)$$

**Lemma 1.** The biased random walk $X$ is a reversible Markov process.

**Proof.** Let $\pi$ be the stationary distribution of the biased random walk $X$. It is easy to verify that

$$\pi_i = \frac{1}{N} \quad (2.9)$$

for all $i \in \mathcal{V}$. Thus by the symmetry of $P^B$,

$$\pi_i P^B_{ij} = \pi_j P^B_{ji}.$$
Lemma 2. In an arbitrary connected graph $G$ with $N$ nodes, the hitting time of the biased random walk $X$ satisfies

$$\mathcal{H}_{P\theta}(G) < 3N^3.$$  

Proof. The biased random walk $X$ defined above is a random walk on a weighted graph with weight

$$w_{ij} := \frac{1}{N} \left( \frac{1}{|N_i|} + \frac{1}{|N_j|} \right) \text{ for } (i, j) \in \mathcal{E}. \quad (2.10)$$

$$w_{ii} := 1 - \sum_{j \in N_i} w_{ij}. \quad (2.11)$$

$$w_i = \sum_{j \in V} w_{ij} = 1, \quad w = \sum_i w_i = N. \quad (2.12)$$

It is well-known that there is an analogy between a weighted graph and an electric network [3]. Let $r_{ij}$ denote the resistance between two adjacent nodes, i.e. an edge $(i, j) \in \mathcal{E}$, and let $r'_{xy}$ denote the effective resistance between any two nodes $x, y$. For example, in Fig. 2.3 $r_{ij} = 2$, $r_{ik} = 1$, $r'_{ij} = 1$ and $r'_{ik} = 0.75$. In an electric circuit, we always have $r'_{ij} \leq r_{ij}$ because of Ohm's Law. For a random walk on a weighted graph, a wire linking $i$ and $j$ has conductance $w_{ij}$, i.e., resistance $r_{ij} = 1/w_{ij}$.
And the commuting time between $x$ and $y$, $H_{P\theta}(x, y) + H_{P\theta}(y, x)$, has the following relationship with the effective resistance $r'_{xy}$:

$$H_{P\theta}(x, y) + H_{P\theta}(y, x) = wr'_{xy}, \quad \text{(2.13)}$$

where $r'_{xy}$ is the effective resistance in the electric network between node $x$ and node $y$ (Chapter 3 Lemma 11 in [3]). Since the degree of any node is at most $N - 1$, for $(i, j) \in \mathcal{E}$,

$$w_{ij} = \frac{1}{N} \left( \frac{1}{|\mathcal{N}_i|} + \frac{1}{|\mathcal{N}_j|} \right)$$

$$> \frac{1}{N \min(|\mathcal{N}_i|, |\mathcal{N}_j|)}$$

$$r_{ij} < N \times \min(|\mathcal{N}_i|, |\mathcal{N}_j|) \quad \text{(2.14)}$$

Consequently, $r'_{ij} \leq r_{ij} < N \times \min(|\mathcal{N}_i|, |\mathcal{N}_j|)$.

For all $x, y \in \mathcal{V}$, let $Q = (q_1 = x, q_2, q_3, ..., q_{l-1}, q_l = y)$ be the shortest path on the graph connecting $x$ and $y$. Now we claim that $\sum_{k=1}^{l} |\mathcal{N}_{q_k}| < 3N$ (Lemma 3 in [49]).

Since any node, say $u$, not lying on the shortest path can only be adjacent to at most three vertices on $Q$ (otherwise $u$ must be on the shortest path), we have

$$\sum_{k=1}^{l} |\mathcal{N}_{q_k}| \leq 2l + 3(N - l) < 3N. \quad \text{(2.15)}$$

By (2.14) and (2.15), we have

$$r'_{xy} \leq N \times \sum_{k=1}^{l} |\mathcal{N}_{q_k}| < 3N^2 \quad \text{(2.16)}$$
By \(2.13\), we have

\[
\mathcal{H}_{PB}(x,y) < \mathcal{H}_{PB}(x,y) + \mathcal{H}_{PB}(y,x) \\
= wr'_{xy} \\
< N \times 3N^2 \\
= 3N^3. \tag{2.17}
\]

This completes the proof. \(\square\)

Note that this is an upper bound for arbitrary connected graphs. A tighter bound can be derived for certain network topologies. Examples will be given in Section 2.4.

**Lemma 3.** \(\mathcal{H}_{PB}(x,y) + \mathcal{H}_{PB}(y,z) + \mathcal{H}_{PB}(z,x) = \mathcal{H}_{PB}(x,z) + \mathcal{H}_{PB}(z,y) + \mathcal{H}_{PB}(y,x)\).

**Proof.** This is direct result from Lemma 2 in Chap 3 of Aldous-Fill’s book [3] since \(\mathcal{X}\) is reversible. \(\square\)

**Definition 5** (Hidden Vertex). A vertex \(t\) in a graph is said to be hidden if for every other point in the graph, \(\mathcal{H}(t,v) \leq \mathcal{H}(v,t)\). A hidden vertex is shown to exist for all reversible Markov chains in Lemma 3 in [18].

**Lemma 4.** The meeting time of any two random walders on the network \(\mathcal{G}\) following the random processes \(\mathcal{X}\) in Section 2.3.2 is less than \(4\mathcal{H}_{PB}(\mathcal{G})\).

**Proof.** In order to prove the lemma, we construct a coupled Markov chain, \(\mathcal{X}'\) to assist the analysis. \(\mathcal{X}'\) has the same joint distribution as \(\mathcal{X}\) except (2.7) and (2.8).

\[
P_{\mathcal{X}' \text{joint}}(\alpha_m, \alpha_n \text{ meet at } x \text{ or } y) = 2P_{xy}^B \tag{2.18}
\]

\[
P_{\mathcal{X}' \text{joint}}(\alpha_m \text{ does not move}, \alpha_n \text{ does not move}) \\
= 1 - \sum_{i \in N_x} P_{xi}^B - \sum_{j \in N_y} P_{yj}^B. \tag{2.19}
\]
The proof is based on the following sequence of claims:

1. The meeting time of two random walkers following $X'$ is less than $2\mathcal{H}_{PB}(\mathcal{G})$.

2. The meeting time of random process $X$ and $X'$ satisfies $\mathcal{M}_X(\mathcal{G}) \leq 2\mathcal{M}_{X'}(\mathcal{G})$.

3. There holds that $\mathcal{M}_X(\mathcal{G}) < 4\mathcal{H}_{PB}(\mathcal{G})$.

The formal proof is as follows:

For convenience, we adopt the following notation. Let $\mathcal{H}(\bar{x}, y)$ denote the weighted average of $\mathcal{H}(u, y)$ over all neighbors $u$ of $x$; Let $\mathcal{M}(\bar{x}, y)$ denote the weighted average of $\mathcal{M}(u, y)$ over all neighbors $u$ of $x$; Let $\phi(\bar{x}, y)$ denote the weighted average of $\phi(u, y)$ over all neighbors $u$ of $x$. Weightings are according to the edge weights. Similar to [18], define a potential function

$$\phi(x, y) := \mathcal{H}_{PB}(x, y) + \mathcal{H}_{PB}(y, t) - \mathcal{H}_{PB}(t, y),$$

where $t$ is a hidden vertex on the graph. By Lemma 3, $\phi(x, y)$ is symmetric, i.e. $\phi(x, y) = \phi(y, x)$. By the definition of meeting time, $\mathcal{M}$ is also symmetric, i.e. $\mathcal{M}(x, y) = \mathcal{M}(y, x)$. Next we use $\phi$ to bound the meeting time.

By the definition of hitting time, for $x \neq y$ we have

$$\mathcal{H}_{PB}(x, y)$$

$$= 1 + P^B_{xx} \mathcal{H}_{PB}(x, y) + \sum_{i \in N_x} P^B_{xi} \mathcal{H}_{PB}(i, y)$$

$$= 1 + w_{xx} \mathcal{H}_{PB}(x, y) + \sum_{i \in N_x} w_{xi} \mathcal{H}_{PB}(i, y),$$

i.e.,

$$\mathcal{H}_{PB}(x, y) = \frac{1}{\sum_{i \in N_x} w_{xi}} + \frac{\sum_{i \in N_x} w_{xi} \mathcal{H}_{PB}(i, y)}{\sum_{i \in N_x} w_{xi}}$$

$$= \frac{1}{\sum_{i \in N_x} w_{xi}} + \mathcal{H}(\bar{x}, y).$$

(2.22)
So for $x \neq y$,
\[ \phi(x, y) = \frac{1}{\sum_{i \in N_x} w_{xi}} + \phi(\bar{x}, y). \]  \hfill (2.23)

\[
\mathcal{M}_{X'}(x, y) = 1 + \left( 1 - \sum_{i \in N_x} P_{xi}^B - \sum_{j \in N_y} P_{yj}^B \right) \mathcal{M}_{X'}(x, y) \\
+ \sum_{i \in N_x} P_{xi}^B \mathcal{M}_{X'}(i, y) \\
+ \sum_{j \in N_y} P_{yj}^B \mathcal{M}_{X'}(x, j). \tag{2.24}
\]

Note that (2.24) also holds for $x \in N_y$. We now have
\[
\left( \sum_{i \in N_x} P_{xi}^B + \sum_{j \in N_y} P_{yj}^B \right) \mathcal{M}_{X'}(x, y) \\
= 1 + \sum_{i \in N_x} P_{xi}^B \mathcal{M}_{X'}(i, y) + \sum_{j \in N_y} P_{yj}^B \mathcal{M}_{X'}(x, j). \tag{2.25}
\]

(2.25) shows that at least one of the two inequalities below holds:
\[
\mathcal{M}_{X'}(x, y) > \frac{\sum_{i \in N_x} P_{xi}^B \mathcal{M}_{X'}(i, y)}{\sum_{i \in N_x} P_{xi}^B} = \mathcal{M}_{X'}(\bar{x}, y) \tag{2.26}
\]
\[
\mathcal{M}_{X'}(x, y) > \frac{\sum_{j \in N_y} P_{yj}^B \mathcal{M}_{X'}(x, j)}{\sum_{j \in N_y} P_{yj}^B} = \mathcal{M}_{X'}(x, \bar{y}) \tag{2.27}
\]

Without loss of generality, suppose that (2.27) holds (otherwise, we can prove the other way around). From (2.25), we have
\[
\sum_{i \in N_x} P_{xi}^B \mathcal{M}_{X'}(x, y) = 1 + \sum_{i \in N_x} P_{xi}^B \mathcal{M}_{X'}(i, y) \\
+ \sum_{j \in N_y} P_{yj}^B \mathcal{M}_{X'}(x, j) - \sum_{j \in N_y} P_{yj}^B \mathcal{M}_{X'}(x, y). \tag{2.28}
\]
i.e.,
\[
\mathcal{M}_{\mathcal{X}'}(x, y) = \frac{1}{\sum_{i \in \mathcal{N}_x} P_{x_i}^B} + \mathcal{M}_{\mathcal{X}'}(\bar{x}, y) + \sum_{j \in \mathcal{N}_y} P_{y_j}^B (\mathcal{M}_{\mathcal{X}'}(x, \bar{y}) - \mathcal{M}_{\mathcal{X}'}(x, y)) + \sum_{i \in \mathcal{N}_x} \frac{P_{x_i}^B}{\sum_{i \in \mathcal{N}_x} P_{x_i}^B} \mathcal{M}_{\mathcal{X}'}(\bar{x}, y) < \frac{1}{\sum_{i \in \mathcal{N}_x} w_{x_i}} + \mathcal{M}_{\mathcal{X}'}(\bar{x}, y). \tag{2.29}
\]

Now we claim that $\mathcal{M}_{\mathcal{X}'}(x, y) \leq \phi(x, y)$. Suppose it is not the case. Let $\beta = \max_{x, y} \{\mathcal{M}_{\mathcal{X}'}(x, y) - \phi(x, y)\}$. Among all the pairs $x, y$ realizing $\beta$, choose any pair. It is clear that $x \neq y$, since $\mathcal{M}_{\mathcal{X}'}(x, x) = \phi(x, x)$. By (2.23) and (2.29),
\[
\mathcal{M}_{\mathcal{X}'}(x, y) = \phi(x, y) + \beta = \frac{1}{\sum_{i \in \mathcal{N}_x} w_{x_i}} + \phi(\bar{x}, y) + \beta \geq \frac{1}{\sum_{i \in \mathcal{N}_x} w_{x_i}} + \mathcal{M}_{\mathcal{X}'}(\bar{x}, y) > \mathcal{M}_{\mathcal{X}'}(x, y). \tag{2.30}
\]
This is a contradiction. From the definition of the potential function in (2.20), we have $\phi(x, y) < 2\mathcal{H}_{PB}(\mathcal{G})$. Thus $\mathcal{M}_{\mathcal{X}'}(\mathcal{G}) \leq \phi(x, y) < 2\mathcal{H}_{PB}(\mathcal{G})$.

Now we are ready to complete the proof of Lemma 4. We couple the Markov chains $\mathcal{X}$ and $\mathcal{X}'$ so that they are equal until the two random walkers become neighbors. Note that half of the time when the walkers in $\mathcal{X}'$ meet, they do not meet in $\mathcal{X}$, but stay in the same position. We claim that $\mathcal{M}_{\mathcal{X}'}(\mathcal{G}) \leq 2\mathcal{M}_{\mathcal{X}'}(\mathcal{G})$.

In the random process $\mathcal{X}'$, when two random walkers $m, n$ meet, instead of finishing the process, we let them exchange positions and continue the random walks according to $P_{\mathcal{X}' \text{joint}}$. The expected length of each exchange is less than or equal
to \( M_{X'}(G) \). At each cross, the random process \( X \) finishes with a probability of 1/2, independently. Thus for any \( x, y \in V \) we have

\[
M_{X}(x, y) \leq \sum_{i=1}^{\infty} \left( \frac{1}{2} \right)^i iM_{X'}(G) = 2M_{X'}(G). \tag{2.31}
\]

This completes the proof. \( \square \)

Now we are ready to prove Theorems 1 and 2 in the next two subsections.

### 2.3.3 An Upper Bound on Binary Consensus

Without loss of generality, let us suppose that in the initial setting more nodes hold strong positive opinions \((S^+)\). As briefly analyzed in Section 2.2.1, the process undergoes two stages: the depletion of \( S^- \) and the depletion of \( W^- \). By our assumption,

\[
|S^+(0)| > |S^-(0)| \tag{2.32}
\]

and

\[
|S^+(0)| + |S^-(0)| = N, \tag{2.33}
\]

where \( N \) is the number of nodes on the graph.

According to the update rules in Section 2.2.1 we have

\[
|S^+(t)| - |S^-(t)| = |S^+(0)| - |S^-(0)|. \tag{2.34}
\]

Let \( T_1 \) and \( T_2 \) denote the maximum expected time it takes for Stage 1 and Stage 2 to finish. In the first stage, two opposite strong opinions annihilate when an edge between them is activated. Otherwise they take biased random walks on the graph \( G \). In the second stage, the remaining \(|S^+(0)| - |S^-(0)|\) strong positives take random walks over graph \( G \), transforming weak negative into weak positive.
Let $CT_G(v)$ denote the expected time for a random walker starting from node $v$ to meet all other random walkers who are also taking random walks on the same graph but starting from different nodes. Define

$$CT(G) = \max_{v \in V} CT_G(v).$$

**Lemma 5.** Let $M_X(G)$ be the meeting time of the biased random walk $X$ defined in Section 2.2.1. Then

$$CT(G) = O(M_X(G) \log N).$$ \hfill (2.35)

**Proof.** Since there are no more than $N$ consecutive meetings with random walkers that it never meets, we can easily get a union bound for $CT(G)$, which is $N M_X(G)$.

In order to obtain a tighter bound for $CT(G)$, we divide the random walk into $\ln N$ periods of length $k M_X(G)$ each, where $k$ is a constant. Let $a$ be the “special” random walker trying to meet all other random walkers. For any period $i$ and any other random walker $v$, by the Markov inequality, we have

$$\Pr(a \text{ does not meet } v \text{ during period } i) \leq \frac{M_X(G)}{k M_X(G)} = \frac{1}{k}$$ \hfill (2.36)

so

$$\Pr(a \text{ does not meet } v \text{ during any period}) \leq \left(\frac{1}{k}\right)^{\ln N} = N^{-\ln k}$$ \hfill (2.37)
If we take the union bound,

\[
\Pr(a \text{ doesn’t meet some walker during any period}) \leq N \cdot N^{-\ln k}.
\] (2.38)

Conditioning on whether or not the walker \(a\) has met all other walkers after all \(k\mathcal{M}_X(G)\ln N\) steps, and using the previous \(N\mathcal{M}_X(G)\) upper bound, we have

\[
CT(G) \leq k\mathcal{M}_X(G)\ln N + N \cdot N^{-\ln k} \cdot N\mathcal{M}_X(G)
= k\mathcal{M}_X(G)\ln N + N^{2-\ln k}\mathcal{M}_X(G)
\] (2.39)

When \(k\) is sufficiently large, say \(k \geq e^6\), the second term is small, so

\[
CT(G) < (k + 1)\mathcal{M}_X(G)\ln N.
\] (2.40)

This completes the proof. \(\square\)

**Proof of Theorem 1.** In order to analyze Stage 1, we can construct a coupled Markov process which follows the same random walk process as \(X\) with a slight modification. When two different strong opinions meet, instead of following the rules to change into weak opinions, they just keep their states and keep moving along the same path they would have as weak opinions. This process is over when every strong opinion has met all other opposite strong opinions at least once, by when Stage 1 must have finished, i.e. before at most \(N^2/4\) first meetings between one random walker and another (about half strong positives and half strong negatives). The rest of the proof just follows from Lemma 5, except we divide the random walks into \(\ln(N^2/4)\) periods of length \(k\mathcal{M}_X(G)\) instead of \(\ln N\). Hence we have \(T_1 \leq 2CT(G)\). Also, \(T_2 \leq CT(G)\) follows from the fact that there are at most \(N - 1\) meetings for a single strong opinion.
to meet all the weak opinions to ensure convergence. By Lemma 2, Lemma 4 and Lemma 5, both $T_1$ and $T_2$ are $O(N^3 \log N)$.

Hence the upper bound of convergence time of binary voting is thus $O(N^3 \log N)$.

Remark: In the proof of Theorem 1, it agrees with Theorem III.1 in [19] that the two phase duration $T_1$ and $T_2$ are bounded by the same order of $n$. However, in [19], proving through different methods, the authors did not provide a universal upper bound on an arbitrary graph, but introduced a function $\delta(G, \alpha)$ depending on the graph structure and voting margin.

2.3.4 An Upper Bound on Quantized Consensus

Recall that a non-trivial exchange in quantized consensus happens when the difference in values at the nodes is greater than 1.

Let $Q(t)$ denote a vector of values all nodes holding at time $t$. Set $\bar{Q} = Q_{\text{sum}}/N$, where $Q_{\text{sum}}$ is defined in (2.1).

We construct a Lyapunov function $L_{\bar{Q}}$ [34,48,70] as:

$$L_{\bar{Q}}(Q(t)) = \sum_{i=1}^{N} (Q_i(t) - \bar{Q})^2.$$  \hfill (2.41)

Let $m = \min_i Q_i(0)$ and $M = \max_i Q_i(0)$. It is easy to see that $L_{\bar{Q}}(Q(0)) \leq \frac{(M-m)^2 N}{4}$. Equality holds when half of the values are $M$ and others are $m$.

Lemma 6. In a non-trivial meeting,

$$L_{\bar{Q}}(Q(t)) \geq L_{\bar{Q}}(Q(t+1)) + 2.$$

Proof. A non-trivial meeting follows the first update rule of quantized consensus algorithm in Section 2.2.2.
Suppose $Q_i(t) = x_1$ and $Q_j(t) = x_2$ have a non-trivial meeting at time $t$, and the rest of the values stay unchanged. Without loss of generality, let $x_1 \leq x_2 - 2$. We have

\[
L_Q(Q(t)) - L_Q(Q(t + 1)) = x_1^2 + x_2^2 - (x_1 + 1)^2 - (x_2 - 1)^2 = 2(x_2 - x_1) - 2 \geq 2. \tag{2.42}
\]

Proof of Theorem 2. Lemma 6 shows that the Lyapunov function is decreasing. The convergence of quantized consensus must be reached after at most $\gamma = \frac{(M - m)^2 N}{8}$ non-trivial meetings. Similar to the analysis for the binary voting algorithm, when every random walker has met each of the other random walkers $\gamma$ times, all the non-trivial meetings must have finished. By Lemma 5, this process finishes in $O(N^3 \log N)$ time.

Lemma 7. Given a network $\mathcal{G}$ of $N$ nodes, an upper bound for the expected convergence time of the binary voting and quantized consensus algorithm is $O(\mathcal{M}_X(\mathcal{G}) \log N)$.

Proof. This is direct result from the proof of Lemma 5, Theorem 1 and Theorem 2.

For example, for a fully connected network $\mathcal{C}$ (i.e. complete graph) with $N$ nodes, at time $t$, the probability of any two random walker $i$, $j$ to meet is $P_B(\mathcal{C}) = \frac{2}{N(N-1)}$. It is easy to get that $\mathcal{M}_{PB}(\mathcal{C}) = \frac{N(N-1)}{2} = O(N^2)$. Thus by Corollary 7, the upper bound for the expected convergence time of the quantized consensus algorithms is $O(N^2 \log N)$. Note that this result agrees with the analysis of convergence time of
complete graph in Section IV.A in [19], where the authors derived an upper bound of $O(N \log N)$, regarding to local clock (See Section 2.2 for definition of local clock and global clock). Since every second, there are number of $N$ clock ticks on average, this is hence equivalent to a $O(N^2 \log N)$ bound regarding to the number of clock ticks in our case.

2.4 Simulation Results

In this section, we give examples of star networks, line graphs, and lollipop graphs in order to show how to use the analysis in Section 2.3 for the particular graphs with known topologies. Simulation results are provided to validate the analysis. We also simulate the distributed process on Erdős-Rényi random graph in order to get some insight on how the algorithm performs on a random graph.

2.4.1 Star Networks

Star networks are a common network topology. A star network $S$ of $N$ nodes has one central hub and $N - 1$ leaf nodes, as shown in Fig. 2.5. Now let us derive an upper bound following the similar analysis in Section 2.3.
Analysis

By (2.10) in Section 2.3.2, suppose that there is a star network of $N$ nodes, with the central hub denoted as $c$. For $\forall i, j \neq c$, we have

$$w_{ic} = w_{jc} = \frac{1}{N} \left(1 + \frac{1}{N - 1}\right) = \frac{1}{N - 1}. \quad (2.43)$$

The equivalent resistance between any two leaf nodes $i$ and $j$ is

$$r'_{ij} = \frac{1}{w_{ic}} + \frac{1}{w_{jc}} = 2N - 2. \quad (2.44)$$

By the symmetry of the star network, it is easy to see that

$$\mathcal{H}_{PB}(i, j) = \mathcal{H}_{PB}(j, i). \quad (2.45)$$

By (2.13),

$$\mathcal{H}_{PB}(S) = \mathcal{H}_{PB}(i, j) = N(N - 1). \quad (2.46)$$

According to Lemma 2 and Corollary 5, we can bound the convergence time of both binary and multi-level quantized consensus algorithms of a star network by $O(N^2 \log N)$. 
Note that this result is coherent with the upper bound derived from a direct approach in [19].

**Simulations**

We simulate the star networks with the number of nodes $N$ ranging from 21 to 481, with intervals of 20, for both binary consensus and quantized consensus algorithms. For binary consensus, initially, there are $\lceil N/2 \rceil$ strong positive and $\lfloor N/2 \rfloor$ strong negative nodes, i.e., $|S^+| - |S^-| = 1$. Those nodes communicate with each other following the protocol in Section 2.2.1. The process finishes when consensus is reached. Simulation results on binary consensus are shown in Fig. 2.8a, and are indeed of order $O(N^2 \log N)$, as analyzed above.

For quantized consensus, we show two different initial settings: (1) $Q^{(j)}(0) = 2$, $Q^{(j)}(0) = 0$, $Q^{(k)}(0) = 1$, for $k \neq i, j$ and $i, j \neq c$, Fig. 2.8b; (2) Initial values of nodes are drawn uniformly from 1 to 100, Fig. 2.8c. Nodes on the graph exchange information according to the update rules in Section 2.2.2.

We notice that quantized consensus algorithm converges faster than the binary consensus in the above setting, because a non-trivial exchange takes place whenever the difference of the values between the selected nodes is greater than one. The Lyapunov value is non-increasing. However, in binary consensus, a strong negative opinion can influence the weak opinions before its annihilation. In the first setting in quantized consensus simulation, there is only one non-trivial exchange before reaching convergence, hence the convergence time is actually the meeting time of the graph. In the second setting, due to the uniform distribution of node values, the non-trivial exchange is more often than the first setting, because the special structure of a star network, central hub can balance the values of leaf nodes quickly.

Convergence time in all cases is the average of 20 rounds of simulations.
2.4.2 Line Graph

A line graph is a simple graph structure. Nodes in a line graph $L$ are connected to one another in a line as shown in Fig. 2.6. Although a line graph is less realistic in applications, it serves as an example here to show that the convergence time can indeed reach $O(N^3 \log N)$, thus to show the tightness of the derived bound.

Analysis

In a line graph of $N$ nodes, for two adjacent nodes $i, j$ (not end points), we have

$$w_{ij} = \frac{1}{N} \left( \frac{1}{2} + \frac{1}{2} \right) = \frac{1}{N}. \quad (2.47)$$

The equivalent resistance is

$$r_{ij} = N. \quad (2.48)$$

For two end points, similarly, both have resistance of $2/3N$ with their neighbors. Thus the effective resistance between two end points $m, n$ of the line graph is $N^2 - \frac{5}{3}N$. By the symmetry of a line graph, the hitting time of the graph is

$$\mathcal{H}_{PB}(L) = \mathcal{H}_{PB}(m, n) = \frac{1}{2}N \left( N^2 - \frac{5}{3}N \right). \quad (2.49)$$

The rest follows the analysis in Section 2.4.1 an upper bound on the convergence time for binary and multi-level quantized consensus is $O(N^3 \log N)$. 

Figure 2.6: A line graph.
Simulations

Experiment settings for line graph are same as in Section 2.4.1. The results are plotted in Fig. 2.8d - Fig. 2.8f.

2.4.3 Lollipop Graph

A lollipop graph is a line graph joined to a clique. Fig. 2.7 shows a lollipop graph $P$ with $N$ nodes, $m$ of which form a clique $K_m$, and the rest of $N - m$ nodes connected to $K_m$ by node $i$ as a line. It is well known that a lollipop graph, when $m = \lceil (2N+1)/3 \rceil$, is the extremal graph for the maximum hitting time $O(N^3)$ of a simple random walk \cite{10,40}. In a simple random walk starting from $i$, the walker is very unlikely to go to $j$, compared with from $j$ to $i$. This results in a latency factor of $N$ for a simple random walk starting from the clique and going to the end of the line on the lollipop. For a natural random walk $X_N$ on $P$, the hitting time is $O(N^4)$ \cite{70}, because of the laziness of $X_N$. However, it is not the case in a biased random walk. Since $P^B_{ij} = P^B_{ji}$, it is equal likely that a random walker moves from $i$ to $j$ and from $j$ to $i$.

For any two nodes $s, t$ on the clique $K_m$, and $m$ is $O(N)$,

$$w_{st} = \frac{1}{N} \left( \frac{1}{m-1} + \frac{1}{m-1} \right) = \frac{2}{N(m-1)}. \quad (2.50)$$
The effective resistance \( r'_{st} \) of a clique is clearly less than \( r_{st} = 1/w_{st} = O(N^2) \), and therefore the hitting time from any node on \( K_m \) to \( i \) is \( O(N^3) \). Furthermore, the line on \( \mathcal{P} \) has hitting time of \( O(N^3) \), as analyzed in Section 2.4.2. We then have

\[
\mathcal{H}_{P_B}(\mathcal{P}) = O(N^3). \tag{2.51}
\]

Similarly, an upper bound of convergence time is \( O(N^3 \log N) \) for lollipop graph according to Lemma 2 and Corollary 5.

Experiment settings for a lollipop graph are the same as in Section 2.4.1. The results are plotted in Fig. 2.8g - Fig. 2.8i.

### 2.4.4 Erdős-Rényi random graph

In an Erdős-Rényi random graph \( \mathcal{R} \), an edge is set between each pair of nodes independently with equal probability \( p \). As one of the properties of Erdős-Rényi random graphs, when \( p > \frac{(1+\epsilon) \log N}{N} \), the graph \( \mathcal{R} \) will almost surely be connected [21].

\[
E(\text{number of edges}) = 0.5N(N - 1)p.
\]

The diameter of Erdős-Rényi random graphs is rather sensitive to small changes in the graph, but the typical distance between two random nodes on the graph is

\[
d = \frac{\log N}{\log(\log N)} \quad [21].
\]

We created Erdős-Rényi random graphs by setting \( p = 5 \log N/N \), where \( N \) ranged from 21 to 481, with an interval of 20. Other settings are the same as in Section 2.4.1. Experiment results of Erdős-Rényi random graphs are shown in Fig. 2.8 - Fig 2.8j. It appears that the expected convergence time of binary consensus is on the order of \( N^2 \log N \), which is lower than the general upper bound of Theorem 1.
(a) Simulation results of average convergence time on binary consensus algorithm of star networks. The solid line indicates $0.63N^2 \log N$.

(b) Simulation results of average convergence time on quantized consensus of star networks in setting 1. The solid line indicates $0.6N^2$, and the dash line indicates $0.7N^2$.

Figure 2.8: Simulation results on average convergence time (squares) of binary consensus and quantized consensus versus the size of networks.
(c) Simulation results of average convergence time on quantized consensus of star networks in setting 2. The solid line indicates $13N \log N$, and the dash line indicates $15N \log N$.

(d) Simulation results of average convergence time on binary consensus algorithm of line graphs. The solid line indicates $0.15N^3 \log N$.

Figure 2.8: Simulation results on average convergence time (squares) of binary consensus and quantized consensus versus the size of networks (continued).
(e) Simulation results of average convergence time on quantized consensus of line graphs in setting 1. The solid line indicates $0.17N^3$.

(f) Simulation results of average convergence time on quantized consensus of line graphs in setting 2. The solid line indicates $0.25N^3$.

Figure 2.8: Simulation results on average convergence time (squares) of binary consensus and quantized consensus versus the size of networks (continued).
(g) Simulation results of average convergence time on binary consensus algorithm of lollipop graphs. The solid line indicates $0.14N^3 \log N$.

(h) Simulation results of average convergence time on quantized consensus of lollipop graphs in setting 1. The solid line indicate $0.15N^3$.

Figure 2.8: Simulation results on average convergence time (squares) of binary consensus and quantized consensus versus the size of networks (continued).
(i) Simulation results of average convergence time on quantized consensus of lollipop graphs in setting 2. The solid line indicates $0.3N^3$.

(j) Simulation results of average convergence time on binary consensus algorithm of Erdős-Rényi random graphs. The solid line indicates $2N^2 \log N$, and the dash line indicates $2.3N^2 \log N$.

Figure 2.8: Simulation results on average convergence time (squares) of binary consensus and quantized consensus versus the size of networks (continued).
(k) Simulation results of average convergence time on quantized consensus of Erdős-Rényi random graphs in setting 1. The solid line indicates $0.5N^2$, and the dash line indicates $0.7N^2$.

(l) Simulation results of average convergence time on quantized consensus of Erdős-Rényi random graphs in setting 2. The solid line indicates $1.1N^2$.

Figure 2.8: Simulation results on average convergence time (squares) of binary consensus and quantized consensus versus the size of networks (continued).
Chapter 3

Privacy Preserving Recommender Systems

A fortuitous byproduct of the distributed consensus algorithms discussed in Chapter 2 is that it preserves a degree of privacy, due to the local nature of the algorithms. During the process, the local estimates of the average value are exchanged instead of broadcasting the initial observation to all nodes. In this chapter, we focus on privacy preservation in recommender systems, in which accuracy and privacy are often in tension with each other, especially with the recent development of social media. We modify the algorithm from Chapter 2 for use as a peer-to-peer preference exchange module of the recommender system. This chapter is organized as follows. We first give an overview on privacy preserving recommender systems and related work in Section 3.1. We then formulate the recommendation problem in Section 3.2. In Section 3.3 we introduce the group-based recommender system. The performance of the proposed framework is evaluated in Section 3.4.
3.1 Background

Recommender systems are systems that recommend customized products to users based on historical data, such as users’ explicit or implicit rating history and/or item content information. Over the last decade, the commercialization of early generations of recommender systems has achieved great success. Recommenders typically provide the target user with a list of customized recommendations through collaborative filtering or content-based filtering. Intensive work has been done to improve the performance of both of these techniques. Collaborative filters use known preferences of users to make related recommendations or predictions. Memory-based collaborative filtering uses the entire user-item database to calculate the similarity value between users or items, and then a weighted sum is used to predict the preference level for a particular user and item. See, for example, GroupLens [52]. Model-based approaches such as Bayesian Belief Net collaborative filtering [60] and regression-based collaborative filtering [65] learn a complex pattern from training data and use the model to predict a user’s preference.

Current approaches to protect privacy in recommender systems mostly address two different privacy concerns: protecting users’ privacy from curious peers or malicious users [42, 44], and protecting against unreliable service providers [2, 14, 47]. In order to make the outcome of a recommendation insensitive to a single input so as to protect users’ private preference data from other users, privacy preserving algorithms from the differential privacy literature [22] have been modified to provide privacy guarantees. McSherry et al. [44] adapted the leading approaches from the Netflix challenge by adding noise to data to provide differentially private recommendations on movies. Machanavajjhala et al. [42] studied recommendations based on a user’s social network with differential privacy constraints. In addition to differential privacy [22], other notions of privacy such as $k$-anonymity [61], $l$-diversity [41], effective size of the anonymity set [50], etc., have also been studied and are used to protect
individual privacy. On the other side, in order to prevent a single party, e.g. the
service provider, from gaining access to every user’s data, cryptographic solutions
based on secure multi-party computation are proposed in [2, 14], and a distributed
hierarchical neighborhood formation was proposed in [7] to reduce the privacy haz-
ard. In [14] cryptography and recommendation are computed by end-users, which is
likely to suffer from the limitation of personal computation devices. A¨ımeur et al. [2]
introduced a semi-trusted third party to share the sensitive information with service
provider, and a two-party computation is then performed for recommendation.

Group-based social networks were originally an alternative for social networks
such as Facebook, twitter, etc. Various examples have emerged such as Douban,
Diaspora, Crabgrass, and Lorea. Notably, Douban, as shown in Fig. 3.1 is a Chinese
group-based social network that has already attracted more than 50 million users.
These alternatives suggest a new interpretation of social networks, and indicate a
fundamental shift away from a purely individualistic approach to social formation
that sees the collective as the ensemble of individuals, to an understanding that
the collective has the same priority as the individual [31]. In this chapter, groups
serve a similar function as an associated milieu that contributes to the preservation
of individual privacy, while still supporting the functioning of the social network.
Instead of adding noise or using cryptography, we find that it is possible to give
reasonably accurate recommendations based on groups while maintaining privacy
from the service provider. Differential privacy [22] essentially captures the risk to
one’s privacy incurred by participating in a database. It was originally defined for
randomized algorithms using noise to obscure the details of individuals. Our approach
has some similarities with differential privacy, in the sense that the group replaces
the role of the randomized mechanism, and the group size is analogous to noise level.
Thus individual data is protected by group-wise preference aggregation.
An example of a group-based social network: douban.com, a Chinese group-based social network focus on building interest groups around books, films, music, etc., with more than 50 million users. On the left of the webpage is shown information of a DIY group, and on the right is shown a list of new-coming group members and associated groups.

(b) Structure of group-based social networks. Two groups are linked if they are associate groups.

Figure 3.1: Group based social networks.

The focus of the system discussed in this chapter is to protect users from unreliable service providers, and to mitigate users’ fear of potential intrusions of privacy by keeping a certain amount of anonymity. We design a simple distributed protocol to preserve users’ privacy through a peer-to-peer preference exchange process. In this process, neither a third party nor cryptography is needed. The service provider only receives mixed preferences for the purpose of preference aggregation. Although data uploaded by individuals might not be $k$-identical as in a $k$-anonymity dataset [61], the origins cannot be identified by the service provider. We evaluate the privacy by the effective size of the anonymity set [50], which is a generalized concept of $k$-anonymity [61]. After group opinion is aggregated, we construct a recommendation graph and use a random walk based method to make recommendations. The stationary distribution resulting from a random walk on the graph is interpreted as a ranking of nodes for the purpose of prediction and recommendation. Personalized recommendation is only performed locally so that no private information is revealed to the service provider. We evaluate the performance of the proposed algorithm using the MovieLens dataset,
and we compare the results with recommendation algorithms designed for individual users.

3.2 Problem Statement

In a typical setting, there is a list of $m$ users $\mathcal{U} = \{u_1, u_2, ..., u_m\}$, and a list of $n$ items $\mathcal{I} = \{i_1, i_2, ..., i_n\}$. Each user $u_j$ has a list of items $I_{u_j}$, which the user has rated or from which user preferences can be inferred. The ratings can either be explicit, for example, on a 1-5 scale as in Netflix, or implicit such as purchases or clicks. This information is stored locally. In a group-based social network, the basic atoms are groups instead of individuals. $\mathcal{G} = \{g_1, g_2, ..., g_k\}$ is a list of $k$ groups. $\mathcal{S} = \{\mathcal{G}, \mathcal{E}_s\}$ is a group-based social network, containing social network information, represented by an undirected or directed graph. $\mathcal{G}$ is a set of nodes and $\mathcal{E}_s$ is a set of edges. For all $u, v$, $(u, v) \in \mathcal{E}_s$ if $v$ is an associated group of $u$. Let $\mathcal{T} = \{t_1, t_2, ..., t_y\}$ be a set of tagging information for the items. For example, for movies, $\mathcal{T}$ can be genre, main actor, release date, etc. $T_i \in \{0, 1\}^y$ denotes the features of item $i$, where $y$ is the total number of tags. We want to make privacy preserving recommendations to users by using groups as natural middleware while no individual preference information is revealed to the central server.

3.3 Group-based Privacy Preserving Recommender System

The structure of the recommender system (Fig.3.2) is as follows:

- **Module 1**: Peer-to-peer preference exchange. Users exchange preference information with other group members in a distributed manner. Only the exchanged
information is then uploaded to the central node, thus the individual preferences are kept private.

- **Module 2**: Intra-group preference aggregation. The central server aggregates group preferences to minimize the disagreement heuristically. The group preference will serve as an input for inter-group recommendation and prediction.

- **Module 3**: Inter-group recommendation. A recommendation graph is constructed. A random walk based algorithm is performed for recommendations.

- **Module 4**: Local recommendation personalization. The top-$k$ recommendations are returned to group members. Items that have been rated by the user are removed from the recommendation list.

In the rest of this section, we describe and analyze the system in detail.

### 3.3.1 Peer-to-peer Preference Exchange

Preference exchange is a process to mix individual preferences so that no full rating profile is collected by the recommendation service provider. Some of the benefits of our preference exchange scheme could be obtained by anonymous communications such as *The Onion Router* [51]. Users could use persistent pseudo-identities and make anonymous ratings, either directly on the central server or by letting a trustful
third party collect this information. However, pseudo-identities still expose users to privacy risks unless the user data is further protected [14] (e.g. Netflix Prize lawsuit due to privacy concerns). Multi-party computation methods were introduced in [2,14], but these either require heavy computation by end users or the introduction of a third party. Our proposed peer-to-peer preference exchange procedure lets users exchange information within the group in a distributed manner. Only the mixed preferences are sent to the central server. In a group based social network, such as Douban, group members are maintained by group masters, thus we assume that users within the group are trustful and uncorrupted. Otherwise, techniques of fake accounts and malicious users detection in social networks can be used [59,69]. Note that the proposed P2P procedure also protects users preference information among peers. Since this is beyond the scope of this work, we do not measure the privacy guarantee among users quantitatively.

In the rest of Section 3.3.1 we describe our peer-to-peer preference exchange scheme in detail and analytically give the privacy guarantee towards the service provider.

**Pairwise Comparison Matrix**

Before sending preference information to the server, group users exchange information with other group members distributedly. Users then upload the mixed information. Suppose every user has a partial ranking on $\mathcal{I}$. Each user keeps an $n \times n$ pairwise comparison matrix $M$ locally. $M^{(u)}_{xy} = 1$ if user $u$ considers $x$ is better than $y$; $M^{(u)}_{xy} = 0$ if otherwise, including when no comparison is made between $x$ and $y$ or they are equally liked.

Very often, recommender systems use $p$-rating to represent users’ rating preferences. In $p$-rating records, users rate products on the scale of 1 to $p$. For example, Netflix uses 5-rating to rate the movie, where 5 represents the most favorable, and
1 represents the least. We can transform $p$-rating history into a partial rank, which naturally normalized the individual ratings. For example, user A who gives ratings 7,8,9 to items a,b,c has the same pairwise comparison matrix to user B who rates a, b, c as 1,2,3 respectively. Let $r^{(u)}_x$ denote the rating of user $u$ on item $x$.

- If $r^{(u)}_x > r^{(u)}_y$, $M^{(u)}_{xy} = 1$, and $M^{(u)}_{yx} = 0$.
- If $r^{(u)}_x = r^{(u)}_y$, $M^{(u)}_{xy} = 0$, and $M^{(u)}_{yx} = 0$.

### Pre-exchange Preparation

Although our focus is to prevent the central server from collecting individual preference, the proposed P2P preference exchange scheme also protects users preference information from other group members. Before the preference exchange starts, each user $u$ randomly chooses $p$ pairwise comparison pairs $x, y$ with $M^{(u)}_{xy} = M^{(u)}_{yx} = 0$, and changes it to $M^{(u)}_{xy} = M^{(u)}_{yx} = 1$, where

$$p = \frac{1}{2} \left( \frac{1}{2} n(n - 1) - \sum_{i,j} 1_{\{M^{(u)}_{ij} + M^{(u)}_{ji} = 1\}} \right), \quad (3.1)$$

i.e. after inserting some 1’s in the pairwise comparison matrix, there are equal number of 0’s and 1’s among all non-diagonal entries in the matrix (diagonal entries of the matrix are always 0).

### Preference Exchange Rules

Although in a group-based social network, a user can belong to multiple groups, in the recommender system, each user only subscribes to one group for recommendations (If assigning users to multiple groups for recommendations, trivial changes are needed, e.g., preference aggregation on the recommendation results from multiple groups). Consider a group $g_i$ of $N$ members. Group members form a network of $N$ nodes, labeled 1 through $N$, as a complete graph. As in some distributed systems [9], each
node has a clock which ticks according to a rate 1 Poisson process. In addition, a synchronized clock is also present at each node.

The preference exchange phase is a process to mix individual preferences so that users do not upload anyone’s full rating profile but the mixed preference of the group. The only requirement for the preference exchange is sum conservation. When a user $u$’s local Poisson clock ticks, $u$ randomly picks another user $v$ in the same group, and randomly picks a non-diagonal entry in the pairwise comparison matrix $M_{xy}$ to exchange with $v$.

This phase ends at synchronized time $t = T_{th}$. All nodes then check all pairwise comparisons: If $M_{xy} = M_{yx} = 1$, then they reset both entries to be 0, i.e. make $M_{xy} = M_{yx} = 0$. They then upload their current preference information to the central server. Because the information uploaded is a mixed preference, individual preference information is not provided and user privacy is protected.

Remark: Note that in the pre-exchange stage, changing pairwise comparison entries from 0 to 1 does not change the individual preference profile, but only helps to protect user’s privacy from being revealed to peers in the preference exchange process.

Anonymity Analysis

Anonymity is the state of being not identifiable within a set of subjects, which is called the anonymity set [50]. One popular measurement of anonymity is the notion of an anonymity set, which was introduced for the dining cryptographers problem [16]. However, a rating does not necessarily arise with equal probability from each of the group members, and so the size of the group is not necessarily a good indicator of anonymity. Instead, we adopt an information theoretic metric for anonymity proposed in [54]:
Definition 6. Define the effective size $\mathcal{A}$ of an anonymity probability distribution as,

$$\mathcal{A} = 2\sum_{u \in s_t} -p_u \log_2 p_u$$

(3.2)

where $p_u$ is the probability that a rating record is from user $u$. Note that the exponent is the entropy of the distribution $p_u$.

In order to find the probability distribution of a certain rating record, we first analyze the random process of preference exchange. Because of the superposition property of the Poisson process, the setup is equivalent to a single global rate $N$ Poisson clock ticking at times $\{Z_k\}_{k \geq 0}$. The communication and exchange of preferences occurs only at $\{Z_k\}_{k \geq 0}$.

Theorem 3. The effective size of the anonymity set of any preference record $\mathcal{A}$ approaches the group size $N$ asymptotically, i.e.,

$$\lim_{t \to \infty} \mathcal{A}(t) = N.$$  

(3.3)

Proof. In this random process, there are two sources stimulating the random walk from $i$ to $j$, $\forall (i,j) \in \mathcal{E}$: one is the clock of the node $i$, $P^1_{ij} = P^N_{ij}$; the other one is the clock of its neighbor $j$, $P^2_{ij} = P^N_{ji}$. Thus $P_{ij} = P^1_{ij} + P^2_{ij}$, i.e., each rating record $\alpha$ in a node takes a biased random walk on a complete graph, with marginal transition matrix $P = (P_{ij})$:

- $P_{ii} := 1 - \frac{2}{N} \frac{1}{n'}$ for $\forall i \in \mathcal{V}$,

- $P_{ij} := \frac{1}{n'} \frac{1}{N} \frac{2}{N-1}$ for $i \neq j$,

where $n'$ is the number of entries exchanged in the pairwise comparison matrix, i.e., $n' = n(n - 1)$, $n$ is the number of items, and $N$ is the number of members in the group.
Hence at time $t$, the probability distribution $P_t(i)$ of a certain rating record $\alpha$ starting from node $i$ is $P_t(i) = P^t \cdot e_i$. $P$ is a symmetric stochastic matrix,

$$P = \begin{pmatrix} 1 - \frac{2}{N} & \frac{1}{N} & \frac{2}{N - 1} & \cdots & \frac{1}{N} & \frac{2}{N - 1} \\ \frac{1}{N} & \frac{1}{N} & \frac{2}{N - 1} & \cdots & \frac{1}{N} & \frac{2}{N - 1} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \frac{1}{N} & \frac{1}{N} & \frac{2}{N - 1} & \cdots & \frac{1}{N} & \frac{2}{N - 1} \end{pmatrix},$$

(3.4)

with eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N$. It is a basic property of eigenvalues that the sum of all eigenvalues, including multiplicities, is equal to the trace of the matrix. It is easy to check that $\lambda_1 = 1$, and $\lambda_2 = \cdots = \lambda_N = 1 - \frac{2}{n'(N-1)}$.

We can express $P$ as $P = \sum_{i=1}^{N} \lambda_i v_i^T v_i$, where the row eigenvectors $v_i$ are unitary and orthogonal. Specifically, $v_1 = (\frac{1}{\sqrt{N}}, \ldots, \frac{1}{\sqrt{N}})$, and $P^t = \sum_{i=1}^{N} \lambda_i^t v_i^T v_i$.

Notice that $\lambda_1 v_1^T v_1 = \lambda_1^t v_1^T v_1 = \frac{1}{N} \mathbf{1}\mathbf{1}^T$. Hence $P = \frac{1}{N} \mathbf{1}\mathbf{1}^T + \sum_{i=2}^{N} \lambda_i^t v_i^T v_i$. We thus have

$$P^t = \frac{1}{N} \mathbf{1}\mathbf{1}^T + \left(1 - \frac{2}{n'(N-1)}\right)^{t-1} \begin{pmatrix} 1 - \frac{2}{N} & \frac{1}{N} & \frac{2}{N - 1} & \cdots & \frac{1}{N} & \frac{2}{N - 1} \\ \frac{1}{N} & \frac{1}{N} & \frac{2}{N - 1} & \cdots & \frac{1}{N} & \frac{2}{N - 1} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \frac{1}{N} & \frac{1}{N} & \frac{2}{N - 1} & \cdots & \frac{1}{N} & \frac{2}{N - 1} \end{pmatrix}.$$  

(3.5)

As $t \to \infty$, each rating record $\alpha$ shows up at each node with equal probability, i.e. $\lim_{t \to \infty} P_t(i) = \frac{1}{N} \mathbf{1}$, for $\forall i \in \{1, 2, \ldots, N\}$. Then the effective size $A$ of the anonymity distribution for $\alpha$ is $A(t) = 2^{-\sum_{u \in G} p_u(t) \log_2(p_u(t))}$, where $p_u(t)$ is the $u^{th}$ element in $P_t(i)$.  

49
Hence \( \lim_{t \to \infty} A(t) = N. \)

### 3.3.2 Intra-group Preference Aggregation

Suppose every member has a preference profile \( \pi_i \) (full ranking or partial ranking). In the recommender system, we focus on the top-\( k \) rank \( \pi^k \), which is a partial rank consisting of the \( k \) most popular alternatives. One way to define top-\( k \) rank is that a partial rank contains \( k \) items which minimizes the disagreement with all individual user’s preferences, as explicitly formulated below:

\[
\text{minimize} \sum_{i=1}^{\lfloor g_j \rfloor} K(\pi^k, \pi_i) \quad (3.6)
\]

\( K(\pi^k, \pi_i) \) is the Kendall tau distance \([35]\), defined by the number of disagreement of pairwise comparisons between two (partial) ranks. More specifically,

\[
K(\pi_1, \pi_2) = |\{(i,j) : i < j, (\pi_1(i) < \pi_1(j) \land \pi_2(i) > \pi_2(j)) \lor (\pi_1(i) > \pi_1(j) \land \pi_2(i) < \pi_2(j))\}| \quad (3.7)
\]

If \( k \) is the size of the items, i.e. \( k = n \) and \( \pi^k \) satisfies \([36]\), \( \pi^k \) is called a Kemeny ranking \([68]\). For example, suppose \( \pi_1 = \{1, 2, 3\} \), \( \pi_2 = \{2, 1, 3\} \), \( \pi_3 = \{3, 2, 1\} \), \( K(\pi_1, \pi_2) = 1 \), \( K(\pi_1, \pi_3) = 2 \), and the Kemeny Ranking is \( \pi^3 = \{1, 2, 3\} \).

In our recommendation system, the mixed preferences are recorded in the form of pairwise comparisons. For a group \( g_j \), let \( M^{(j)} = \sum_{i \in g_j} M^{(i)} \). We can construct a direct weighted graph \( G^{(j)} = \{I, E^{(j)}\} \). \((x, y) \in E^{(j)}\) if \( M^{(j)}_{xy} - M^{(j)}_{yx} > 0 \), and \( w^{(j)}_{xy} = M^{(j)}_{xy} - M^{(j)}_{yx} \) i.e., if more group members in \( g_j \) prefer \( x \) to \( y \). The weight of the edge is the corresponding difference of matrix entries. In order to find the top-\( k \) list \( \pi^k \) satisfying \([36]\), we need to reverse a set of edges, the sum of which is
Figure 3.3: The pairwise comparison graph for $\pi_1 = \{1,2,3\}$, $\pi_2 = \{2,1,3\}$, $\pi_3 = \{3,2,1\}$.

minimal so that we can do the topological sort on the graph for the first $k$ nodes. The pairwise comparison graph for the example above is drawn in Fig. 3.3. Partial rank aggregation is known to be NP-hard [1].

**Heuristic Rank Aggregation**

We now propose an efficient heuristic method for intra-group preference aggregation for top-$k$ items. As mentioned in the last section, if we can do topological sort in the partial rank graph for the first $k$ nodes, we then have the top-$k$ list of the group preference. We modify Tarjan’s *strongly connected components* (SCC) algorithm [62] to find the top-$k$ list in linear time if the size of the top SCC is small compared to the size of item list $I$. Since Tarjan’s algorithm returns SCCs in reverse topological order, we first create the graph $G'$, the transpose graph of $G$. Let $c$ be the counter of nodes contained in the current SCC. Detection for SCCs stops when $c \geq k$. Let $\beta$ denote the maximum size of SCC popped so far. Considering the large number of items in a recommendation system, we set a threshold $\theta_{\text{sc}}$: if $\beta \geq \theta_{\text{sc}}$, a heuristic method is used to find $\pi^k$; otherwise we compute the exact result. $k \ll \theta_{\text{sc}} \ll n$.

In reality, the assumption that all items are equally likely to be rated may not hold. Let us define the popularity of an item $\gamma(i)$ as the percentage of users who
rated item $i$. In order to balance popularity and quality, let $\theta_p$ denote the popularity threshold. An item will not be included in the top-$k$ list if $\gamma(i) < \theta_p$.

A summary of the algorithm is shown in Algorithm 1:

```
G' ← G^T;
{create a graph $G'$, which is a transpose graph of $G$};
c ← 0, β ← 0;
while c < k do
    TarjanSCC;
    {update c and β after every SCC is popped};
end
if β < θ_scc then
    topk ← Kemeny;
else
    topk ← HeuristicKemeny;
end
return topk;
```

**Algorithm 1**: Algorithm sketch for intra-group preference aggregation.

We use a modified version of TarjanSCC from [62] in order to update $c$ and $β$. The modified SCC detection algorithm is summarized in Algorithms 2 and 3.

```
index ← 0;
empty stack $S$;
for $v$ do
    if $v$.index is undefined then
        SCC($v$);
end
```

**Algorithm 2**: SCC detection: TarjanSCC

The function SCC recursively explores the connected nodes in the SCC, as shown in Algorithm 3.
Much work has been done on heuristic methods for computing optimal Kendall tau distance (Kemeny-Young method) [1,26,36,53]. In the experiments in Section 3.4, we use Borda count algorithm for HeuristicKemeny. Borda count is a 5-approximation of the Kemeny-Young method, and is often computational effective in practice [36]. In a rating based system, the Borda count result can be calculated by adding up the rating scores of the item. However, other heuristic methods can also be integrated easily in the proposed framework. We do not discuss these methods further since it is out of the scope of this chapter. It is easy to see that TarjanSCC runs in linear
time as a function of the number of edges and nodes because it is based on depth-first search. Borda count runs in linear time as a function of the number of items, i.e. $O(|V|)$. We assume $k \ll \theta_{scc} \ll n$, and hence the proposed heuristic method runs in linear time in $O(|E| + |V|)$.

### 3.3.3 Inter-group Recommendation

The intra-group preference aggregation described above gathers existing preference information from group members. However, it is desirable to recommend new items that have similar features but have not yet been rated by group members. Thanks to the “homophily principle” [30], a group preference can serve as a natural middleware to help make recommendation decisions while protecting the privacy of users, with the absence of individual preference records.

An intuitive approach for recommendation is collaborative filtering (CF) [4,7,66]. It uses the known preferences of users to make recommendations to a target user. Weighted sum is typically used to make predictions.

In CF, a generally adopted similarity measure is called *Pearson Correlation*. It measures the extent to which two variables linearly relate with each other [52]. For user-based algorithms, the Pearson Correlation between user $u$ and $v$ is

$$w_{u,v} = \frac{\sum_{i \in I} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{i \in I} (r_{v,i} - \bar{r}_v)^2}}, \quad (3.8)$$

where $i \in I$ is an item rated by both users $u$ and $v$, $r_{u,i}$ is the rating of user $u$ on item $i$, and $\bar{r}_u$ is the average rating of user $u$ in the co-rating set $I$. A weighted sum is then taken to predict the rating for target user $u$ on a certain item $i$,

$$R_{u,i} = \bar{r}_u + \frac{\sum_{v \in U} (r_{v,i} - \bar{r}_v) \cdot w_{u,v}}{\sum_{v \in U} |w_{u,v}|}. \quad (3.9)$$
Recommenders based on collaborative filtering then refer to this prediction to provide the top-$k$ recommendations to the user. For our group-based recommendation, we can treat the groups as users in the equations above, and use the aggregated group preference as the rating history. In this way, a group recommendation would be proposed.

However, traditional collaborative filtering methods are challenged by problems such as *cold start* and *data sparsity*. In the case of a group based recommendation system, these problems are inevitable, especially since groups in a social network already form natural clusters. Hence, there may not be many co-rated items between different groups for the Pearson Correlation computation.

In order to overcome the disadvantages of collaborative filtering, we propose a random walk based inter-group recommender system, which is an extension of our previous work in [58]. Our model incorporates content information of items and social information of groups together as group preference information. We create a recommendation graph, as shown in Fig. 3.4 consisting of items, groups, and item genres as nodes. Similar to PageRank [11], the stationary distribution resulting from a random walk on the recommendation graph is interpreted as a ranking of the nodes for the purpose of recommendation and prediction. We describe how to construct this recommendation graph and represent the flow on the graph in the rest of this section.

**Graph settings**

Let $G = \{V, E\}$ be a graph model for a recommender system, where $V := \mathcal{G} \cup \mathcal{I} \cup \mathcal{T}$. The nodes of the graph consist of groups, items and item information. For $v_i, v_j \in V$, $(v_i, v_j) \in E$ if and only if there is an edge from $v_i$ to $v_j$, which is determined as given below. The weights are specified in the next subsection.
Figure 3.4: Example of a recommendation graph for inter-group recommendations.

- For $g \in \mathcal{G}, i \in \mathcal{I}$, $(g, i) \in \mathcal{E}$ and $(i, g) \in \mathcal{E}$ if and only if $i \in \pi^k(g)$. i.e., an item $i$ and a group $u$ are connected with weights $w_{gi}$ and $w_{ig}$ if $i$ is in $g$’s top-$k$ list.

- For $i \in \mathcal{I}, t \in \mathcal{T}$, $(i, t) \in \mathcal{E}$ and $(t, i) \in \mathcal{E}$ if and only if $T_i(t) \neq 0$. i.e., an item $i$ and tag $t$ are connected with weights $w_{it}$ and $w_{ti}$ if $i$ is tagged by $t$.

- For $g_1, g_2 \in \mathcal{G}$, $(g_1, g_2) \in \mathcal{E}$ with weight $w_{g_1g_2}$ if and only if $g_1, g_2$ are associated groups, i.e. $(g_1, g_2) \in \mathcal{E}_s$, as mentioned in Section 3.2.

**Edge weight assignment**

The main part of our rank graph is the collaborative filtering graph, which includes the group nodes, item nodes, and the edges between them. One way to assign weights on the collaborative filtering graph is by setting

$$w_{gi} = w_{ig} = \frac{k + 1 - \pi^k_g(i)}{k} w_{\text{max}},$$

(3.10)

where $\pi^k_g(i)$ is the rank of item $i$ in the top-$k$ item rank list of group $g$, and $w_{\text{max}}$ is the max weight assigned on the graph. Let $\pi^k_g(i) = k + 1$ if $i \notin \pi^k_g$. Note that a larger
edge weight indicates greater chance that the random walk passes through that edge. An item \( i \) with better rank in \( \pi^k_g(i) \) results in larger weights on edges involving \( i \).

For the extended graph, i.e. nodes and edges containing item content, group social network information, etc., we simply assign an edge weight of 1 if an edge is present.

**Rank Score Computation**

For the recommendation graph \( G = \{V, E\} \). Let \( v = |V| \) denote the number of nodes on the graph. \( \theta \) is a \( 1 \times v \) *customized probability vector*.

\[
\theta = e_g, \tag{3.11}
\]

where \( e_1, e_2, ..., e_v \) are the standard basis of row vectors. \( \beta \) is a *damping factor*. With probability \( 1 - \beta \), the random walk is teleported back to node \( g \). The rank score \( s \) satisfies the following equation:

\[
s = \beta sW + (1 - \beta)\theta, \tag{3.12}
\]

where \( W \) is the weighted transition matrix with \( W_{ij} = P_{ji} \).

So we have,

\[
s = s(\beta W + (1 - \beta)\theta 1^T) := sM \tag{3.13}
\]

Hence the rank score is the *principal eigenvector* of \( M \), which can be computed by iterations fast and easily via Algorithm [1].

The rank score \( s \) can be interpreted as the importance of other nodes to the target group \( g \). It is easy to see that we can increase the rank score by shortening the distance, adding more paths, or increasing the weight on the path to \( g \). These are desirable properties in a recommender system. For example, even if item \( i \) is not directly connected with \( g \), but it is in a category to which many of \( g \)'s top-\( k \) items
\[ s^{(0)}_j \leftarrow \frac{1}{v} \text{ for all } j; \]
\[ t = 1; \]
\[ \text{while } |s^{(t)} - s^{(t-1)}| < \epsilon \text{ do} \]
\[ \quad \text{for } j = 1 \text{ to } v \text{ do} \]
\[ \quad \quad s^{(t)}_j = \sum_{i=1}^{v} \beta W_{ij} s^{(t-1)}_i + (1 - \beta) \theta_j; \]
\[ \quad \text{end} \]
\[ t \leftarrow t + 1; \]
\[ \text{end} \]

Algorithm 4: Iterative computation of rank score

belong, then \( i \) is very likely to have a high rank score. Or if group \( g \) and \( g' \) have many overlapping top-\( k \) items, \( g' \) will have high rank, so we can use \( g' \)'s top-\( k \) list to make recommendations and predictions for \( g \).

Recommendations

**Direct Method:** Solving Equation (3.12) iteratively, we obtain a rank score for all nodes of the recommendation graph \( G \). Since the rank score represents the importance to the target group, we can then separate and sort them according to the categories, i.e. groups \( \mathcal{G} \), items \( \mathcal{I} \), tags \( \mathcal{T} \), etc. Sorted items form a recommendation list to the target group \( g \), and we can compute the recommendation for every group.

**User-based Prediction:** For items above the group popularity threshold, we simply take the average rating of group members as the rating prediction. For other items, we can use rank score as an influence measure to make predictions, which is similar to memory-based collaborative filtering, using *Pearson Correlation* [52] as a similarity measure between users and items. Given the rank score of the group set \( \mathcal{G} \), we take the weighted sum of the groups’ ratings on item \( i \) as a prediction for the target group \( g \), as shown below:

\[
\hat{r}_{gi}^{user} = \frac{\sum_{x \in G_i} s_x (\bar{r}_{xi} - \bar{r}_x)}{\sum_{x \in G_i} s_x} + \bar{r}_x. \tag{3.14}
\]

\( G_i \) is the set of groups for which item \( i \) is above the popularity threshold. \( s_x \) is the target group’s personalized rank score of group \( x \).
**Item-based Prediction:** As above, in order to perform an item-based recommendation, we can use the rank score of item set \( \mathcal{I} \) as weight to predict the rating of the item \( i \) for the target group \( g \), if the popularity of the item is below the threshold. Specifically,

\[
\hat{r}_{gi}^{item} = \frac{\sum_{j \in I_g} s_j r_{gj}}{\sum_{j \in I_g} s_j}.
\]  

(3.15)

In Equation (3.15), we use \( u \)'s rating on similar items to predict the rating on \( i \). \( s_j \) is the target group’s personalized rank score of item \( j \).

After a recommendation is made, results are returned to individual users. Items that have been rated by the user, which are stored locally, are then removed from the recommendation list.

### 3.4 Experiments and Evaluation

#### 3.4.1 Dataset

In order to evaluate the performance of the proposed algorithm, we run experiments on the MovieLens dataset, which is a widely used benchmark for recommender systems. The MovieLens dataset consists of 1,682 movies and 943 users. Movies are labeled by 19 genres. User profile information such as age, gender, and occupation is also available. In order to evaluate the group-based recommender system, we take user profile categories provided in the dataset as groups. In the experiments, we group users in three different ways, namely, gender, age, and occupation. Detailed group category distribution is as follows:

- **Gender:** male (71.16%) and female (28.84%).

- **Age:** below 21, 21 to 30, 31 to 40, 41 to 50, above 50, indexed from 1 to 5, respectively, as shown in Fig. 3.5a.
(a) Percentage of the population of 5 different age categories.

(b) Percentage of the population of 21 different occupation categories.

Figure 3.5: The group distribution of MovieLens datasets.

- **Occupation**: administrator, artist, doctor, educator, engineer, entertainment, executive, healthcare, homemaker, lawyer, librarian, marketing, none, other, programmer, retired, salesman, scientist, student, technician and writer, indexed from 1 to 21, respectively, as shown in Fig. 3.5b.

Remarks: We choose MovieLens datasets and take the user personal information to categorize users. It may not be the best way to group users, but the purpose of our experiments is to show that the recommendation provided by aggregated preference
performs reasonably well compared with personalized recommendation, which is a good compromise between privacy and accuracy.

### 3.4.2 Experimental Methodology and Results

We evaluate our results with two popular evaluation metrics for top-\( k \) recommendations: percentile and \( \text{TOPK} \).

**Percentile:** The individual percentile score is simply the average position (in percentage) that an item in the test set occupies in the recommendation list. For example, if four items are ranked 1st, 9th, 10th and 20th in a recommendation list consisting of 100 items, with individual percentile scores of 0.01, 0.09, 0.10 and 0.20. The average percentile of the system is 0.1. A lower percentile indicates a better prediction.

**\( \text{TOPK} \) :** Given a recommendation test, we consider any item in the top-\( k \) recommendations that matches any item in the test set as a “hit”, as in [64].

\[
\text{TOPK}(k) = \frac{\text{#hits of top-}k}{T}, \tag{3.16}
\]

where \( T \) is the size of test set. A higher \( \text{TOPK} \) value indicates a better prediction. Note that \( \text{TOPK} \) is proportional to precision, for fixed data and \( k \). In other words, when comparing different recommender systems on fixed data and fixed \( k \), the one with the larger \( \text{TOPK} \) value also has the larger precision.

In this experiment, all items in the test set \( T \) are rated 5 (highest rating) by users, thus we can consider them as relevant items for recommendation. The recommendation list has a length of 900 items. The top-500 movies in the aggregated group preference list are used to construct the recommendation graph. Note that the popularity threshold of the recommender system can be decided by users, since different groups may have a different requirement for popularity. In our experiment, we set the popularity threshold at 0.01. We compare the proposed method with two
state-of-art personalized recommender systems: L+ [27] and ItemRank [29]. L+ sug-
gested a dissimilarity measure between nodes of a graph, the expected commute time
between two nodes, which the authors applied to recommendation [27]. Specifically,
they constructed a non-directed bipartite graph where users and movies form the
nodes. A link is placed between a user and movie if the user watched that movie.
Movies are then ranked in ascending order according to the average commute time
to the target node. ItemRank built the recommendation graph by only using movies
as nodes. In [29], two nodes are connected if at least one user rated both nodes.
The weight of the edge is set as the number of users who rated both of the nodes.
A random-walk based algorithm is then used to rank items according to the target
user’s preference record. In order to see how much information is lost by grouping
users, we also compare the proposed privacy-preserving recommendation algorithm
with a recommendation graph of similar structure, but with all the individual rating
information, where nodes of the recommendation graph are formed by users, items,
user social profile information (gender, age and occupation). The weight of an edge
between users and items is given by

\[
w_{ui} = w_{iu} = \exp \left( \frac{r_{ui} - \bar{r}_u}{\sqrt{\sum_{i \in I_u} (r_{ui} - \bar{r}_u)^2}} \right), \tag{3.17}
\]

\[
\bar{r}_u := \frac{\sum_{i \in I_u} r_{ui}}{|I_u|}. \tag{3.18}
\]

where \(I_u\) denotes the set of items which user \(u\) has rated. Note that a larger edge
weight indicates more chance that the random walk passes through that edge. If user
\(u\)’s rating on item \(i\) \(r_{ui}\) is lower than the average rating \(\bar{r}_u\), \(w_{ui}\) and \(w_{iu}\) are less than
1; otherwise are greater than 1. The assignment of weights do not depend on the
variance of the user’s ratings.
Table 3.1: Average percentile results obtained by 5-fold cross-validation for recommendation.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Percentile</th>
</tr>
</thead>
<tbody>
<tr>
<td>L+</td>
<td>0.1157</td>
</tr>
<tr>
<td>ItemRank</td>
<td>0.1150</td>
</tr>
<tr>
<td>Personal Recommendation</td>
<td>0.0790</td>
</tr>
<tr>
<td>Group by Gender</td>
<td>0.1110</td>
</tr>
<tr>
<td>Group by Age</td>
<td>0.1066</td>
</tr>
<tr>
<td>Group by Occupation</td>
<td>0.1060</td>
</tr>
<tr>
<td>Random 2 Groups</td>
<td>0.1172</td>
</tr>
<tr>
<td>Random 5 Groups</td>
<td>0.1149</td>
</tr>
<tr>
<td>Random 21 Groups</td>
<td>0.1104</td>
</tr>
</tbody>
</table>

Experimental results of cross-validation on percentile scores of the MovieLens dataset are shown in Table 1. We create five training/testing splits. Although it does not utilize knowledge of individual’s preference information, the proposed group-based privacy preserving recommendation algorithm still has a better performance than L+ and ItemRank, which are two state-of-art personalized recommendation methods. And as expected, due to the absence of personal rating information, the performance of the proposed group method is inferior to personal recommendation, i.e., recommendations with individual rating information. It is also worth noting that among all three different ways of grouping users, grouping by occupation outperforms the other two grouping methods, which shows the promise of group-based recommender system with finer groups. Moreover, in order to evaluate the effectiveness of groups in the dataset, we did contrast experiments on random groups, which are users divided randomly into 2, 5, 21 groups to compare with gender, age and occupation groups. Experimental results show that the natural groups outperform the random groups, as shown in Table 1.

We also perform 5-fold cross-validation experiments for TOPK values, as shown in Table 2. In real settings, a user is unlikely to browse a very long recommendation list. Thus, we only test the top-5 to top-50 TOPK values. As introduced in Section
Table 3.2: Average TOPK results obtained by 5-fold cross-validation for recommendation.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Top-5</th>
<th>Top-10</th>
<th>Top-15</th>
<th>Top-20</th>
<th>Top-25</th>
</tr>
</thead>
<tbody>
<tr>
<td>L+</td>
<td>0.1569</td>
<td>0.2335</td>
<td>0.2776</td>
<td>0.3166</td>
<td>0.3519</td>
</tr>
<tr>
<td>ItemRank</td>
<td>0.1690</td>
<td>0.2330</td>
<td>0.2851</td>
<td>0.3352</td>
<td>0.3788</td>
</tr>
<tr>
<td>Personal Recommendation</td>
<td>0.2187</td>
<td>0.3027</td>
<td>0.3477</td>
<td>0.4162</td>
<td>0.4596</td>
</tr>
<tr>
<td>Group by Gender</td>
<td>0.1038</td>
<td>0.1660</td>
<td>0.2435</td>
<td>0.3130</td>
<td>0.3658</td>
</tr>
<tr>
<td>Group by Age</td>
<td>0.1260</td>
<td>0.2275</td>
<td>0.2861</td>
<td>0.3334</td>
<td>0.3771</td>
</tr>
<tr>
<td>Group by Occupation</td>
<td>0.1490</td>
<td>0.2399</td>
<td>0.3048</td>
<td>0.3479</td>
<td>0.3856</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Methods</th>
<th>Top-30</th>
<th>Top-35</th>
<th>Top-40</th>
<th>Top-45</th>
<th>Top-50</th>
</tr>
</thead>
<tbody>
<tr>
<td>L+</td>
<td>0.3774</td>
<td>0.4118</td>
<td>0.4350</td>
<td>0.4591</td>
<td>0.4814</td>
</tr>
<tr>
<td>ItemRank</td>
<td>0.4076</td>
<td>0.4364</td>
<td>0.4582</td>
<td>0.4837</td>
<td>0.5037</td>
</tr>
<tr>
<td>Personal Recommendation</td>
<td>0.4912</td>
<td>0.5139</td>
<td>0.5459</td>
<td>0.5710</td>
<td>0.5910</td>
</tr>
<tr>
<td>Group by Gender</td>
<td>0.4077</td>
<td>0.4417</td>
<td>0.4702</td>
<td>0.4894</td>
<td>0.5095</td>
</tr>
<tr>
<td>Group by Age</td>
<td>0.4110</td>
<td>0.4422</td>
<td>0.4689</td>
<td>0.4920</td>
<td>0.5140</td>
</tr>
<tr>
<td>Group by Occupation</td>
<td>0.4210</td>
<td>0.4485</td>
<td>0.4729</td>
<td>0.4958</td>
<td>0.5184</td>
</tr>
</tbody>
</table>

3.4.2 a TOPK value of $k$ is the probability that an item in the test set hits the top-$k$ items recommended by the system. A higher TOPK value means a higher chance that items in the test set appear in the top-$k$ list. Since these items all have the highest ratings, a higher TOPK value indicates better performance of the recommendation algorithm. In Table 2, personal recommendation, our proposed algorithm with individual preference information, trading privacy for quality, has the best performance. Otherwise, L+ has better performance on top-5 TOPK, and the recommender system based on occupation groups outperforms gender and age groups, and also has a higher TOPK value than L+ and ItemRank for top-10 to top-50 recommendations.
Chapter 4

Differentially Private Rank Aggregation

In Chapter 3, we used groups as a natural middleware to protect the preference privacy of individual users. In this chapter, we look at differential privacy, a widely used approach to reduce the chance of identifying individual records.

4.1 Background

The potential risk of privacy leakage prevents users from sharing their honest opinions on social platforms. In this chapter, we consider the problem of privacy preservation if the query returns the histogram of rankings. Differential privacy is a framework that aims to obscure individuals’ appearances in the database. It makes no assumptions on the attacker’s background knowledge. Mathematical guarantees are provided in [22] and [23]. Differential privacy has gained popularity in various applications, such as social networks [63], recommendations [45], advertising [39], etc. However, there is a trade-off between the accuracy of the query results and the privacy of the individuals included in the statistics. In [43], the authors showed that good private
social recommendations are achievable only for a small subset of users in the social network.

In this chapter, we apply the framework of differential privacy to rank aggregation. Privacy is protected by adding noise to the query of ranking histograms. The histogram query result may then be used in any rank aggregation rule. In general, more noise guarantees better differential privacy. However, excessive noise reduces the utility of the query results. We measure the utility by the probability that the aggregated ranking is accurate.

The rest of the chapter is organized as follows. We define the problem of rank aggregation, introduce the definition of differential privacy and describe the privacy preserving algorithm in Section 4.2. We then discuss the accuracy of the algorithm and provide analytical upper bounds on the error rate in Section 4.3 followed by simulation results in Section 4.4.

4.2 Differential Privacy in Rank Aggregation

4.2.1 Rank Aggregation: Definitions and Notations

Let $\mathcal{C} = \{1, \ldots, M\}$ be a finite set of $M$ candidates, $M \geq 3$. Denote the set of permutations on $\mathcal{C}$ by $T_M$. Denote the number of voters by $N$. Each ballot $x_i, i = 1, \ldots, N$ is an element of $T_M$, or a strict linear ordering. A rank aggregation algorithm, or a ranking rule is a function $g : T^N_M \rightarrow T_M$. The input $(x_1, \ldots, x_N)$ is called a profile.

A ranking rule $g$ is neutral if it commutes with permutations on $\mathcal{C}$ [32]. Intuitively, a neutral ranking method is not biased in favor of or against any candidate.

A ranking rule $g$ is anonymous if the “names” of the voters do not matter [32], i.e.

$$g(x_1, \ldots, x_N) = g(\pi(x_1, \ldots, x_N)) \quad (4.1)$$
for any permutation \( \pi \) on \( 1, \ldots, N \). For an anonymous ranking method, we use the anonymized profile, a vector \( q \in \mathbb{N}^{M!} \), instead of the complete profile \((x_1, \ldots, x_N)\) as the input. Let \( q \) denote the histogram of rankings: It counts the number of appearances of each ranking in all \( n \) rankings. The rank aggregation function can therefore be rewritten as \( g : \mathbb{N}^{M!} \rightarrow T_M \).

An anonymous ranking rule is scale invariant if the output depends only on the empirical distribution of votes \( v = q/N \), not the number of voters \( N \). That is,

\[
g(q) = g(\alpha q)
\]

for any \( \alpha > 0 \).

There are many different neutral and scale invariant rank aggregation algorithms. Popular ones include plurality, Borda count, instant run-off, the Kemeny-Young method and so on. Each algorithm has its own merits and disadvantages. For example, the Kemeny-Young method satisfies the Condorcet criterion (a candidate preferred to any other candidate by a strict majority of voters must be ranked first) but is computationally expensive. In fact it is NP-Hard even for \( M = 4 \) [17]. This is especially an issue for recommender systems since the number of items to be recommended can be large.

A class of ranking rules, known as the positional rules, has an edge in computational complexity. A positional rule takes complete rankings as input, and assigns a score to each candidate according to their position in a ranking. The candidates are sorted by their total scores summed up from all rankings. The time complexity is only \( O(M N + M \log M) \), where the \( M \log M \) term comes from sorting. All positional rules satisfy anonymity and neutrality but fail the Condorcet criterion [24]. A positional rule with \( M \) candidates has \( M \) parameters: \( s_1 \geq \cdots \geq s_M \), where \( s_i \) is the score assigned to the \( i \)th highest-ranked candidate. We can further normalize
the scores without affecting the ranking rule so that \( s_1 = 1, s_M = 0 \). Borda count, a widely used positional rule, is specified by \( s_i = (M - i)/(M - 1) \). Note that plurality is a positional rule with \( s_i = 0 \) for \( i \geq 2 \). Plurality is popular due to its simplicity. However, it is not ideal as a rank aggregation algorithm because it discards too much information. In this chapter, we specifically focus on positional rules because of their computational efficiency and ease of error rate analysis.

*Remark:* There are no deterministic rules that are neutral and anonymous, because of ties. To maintain neutrality, we break ties randomly when there is a tie. For example, if the score of candidate \( a \) and \( b \) happens to be equal, then we rank \( a \) ahead of \( b \) with probability one half. We only mention tie as a side remark since it does not have an affect on the probability analysis.

### 4.2.2 Differential Privacy

In this chapter, we consider a strong notion of privacy, *differential privacy* [22]. Intuitively, a randomized algorithm has good differential privacy if its output distribution is not sensitive to a single entity’s information. For any dataset \( A \), let \( \mathcal{N}(A) \) denote the set of neighboring datasets, each differing from \( A \) by at most one record, i.e., if \( A' \in \mathcal{N}(A) \), then \( A' \) has exactly one entry more or one entry less than \( A \).

**Definition 7.** [23] A random algorithm \( \mathcal{M} \) satisfies \((\epsilon, \delta)\)-differential privacy if for any neighboring datasets \( A \) and \( A' \), and any subset \( S \) of possible outcomes \( \text{Range}(\mathcal{M}) \),

\[
\Pr[\mathcal{M}(A) \in S] \leq \exp(\epsilon) \times \Pr[\mathcal{M}(A') \in S] + \delta. \tag{4.3}
\]

*Remark:* \((\epsilon, \delta)\)-differential privacy is a slight relaxation from the \( \epsilon \)-differential privacy in that the ratio

\[
\Pr[\mathcal{M}(A) \in S] / \Pr[\mathcal{M}(A') \in S]
\]
need not be bounded if both probabilities are very small. Differential privacy has been widely used in various applications [39, 45].

4.2.3 Privacy Preserving Algorithms

Much work has been done on developing differentially private algorithms [5, 25]. Let \( \mathcal{D} \) denote the set of all datasets.

**Definition 8.** The \( l_2 \)-sensitivity \( \Delta f \) of a function \( f : \mathcal{D} \rightarrow \mathbb{R}^d \) is

\[
\Delta f(A) = \max_{A' \in \mathcal{N}(A)} \| f(A) - f(A') \|_2
\]

for all \( A' \in \mathcal{N}(A) \) differing in at most one element, and \( A, A' \in \mathcal{D} \).

**Theorem 4.** [23] Define \( \mathcal{M}(A) \) to be \( f(A) + \mathcal{N}(0, \sigma^2 I_{d \times d}) \). \( \mathcal{M} \) provides \( (\epsilon, \delta) \)-differential privacy, whenever

\[
\sigma^2 \geq \frac{2 \ln \left( \frac{2}{\delta} \right)}{\epsilon^2} \cdot \max_{A' \in \mathcal{N}(A)} \| f(A) - f(A') \|_2^2, \tag{4.4}
\]

for all \( A' \in \mathcal{N}(A) \) differing in at most one element, and \( A, A' \in \mathcal{D} \).

In our model, \( f(A) \) is the histogram of all rankings, i.e. the input vector \( q \) defined in Section 4.2.1. It is clear that the \( l_2 \) sensitivity of \( f(A) \) is 1, since adding or removing a vote can only affect one element of \( q \) by 1. In the exposition, we will denote the private data and released data by \( x \) and \( \hat{x} \) respectively. When we add noise \( n \) to a variable \( x \), we write \( \hat{x} = x + \text{noise}. \) Thus

\[
\hat{q} = q + \mathcal{N}(0, \sigma^2 I_{M! \times M!}) \tag{4.5}
\]

where \( \sigma^2 = 2 \ln \left( \frac{2}{\delta} \right)/\epsilon^2 \), and \( M \) is the number of candidates. We use Gaussian instead of Laplacian noise which achieves stronger \( \epsilon \)-privacy [22] because Gaussian noise enjoys
the nice property that any linear combination of jointly Gaussian random variables is Gaussian.

Note that there is a positive probability that $\hat{q}_i < 0$ for some index $i$. This does not harm our analysis since positional rules are well defined even if we allow negative vote counts.

Finally, we define the error rate of a privacy preserving rank aggregation algorithm on ranking. The error rate is the probability that the aggregated ranking changes after adding noise. This probability depends on the ranking rule, the noise distribution, and the distribution of profiles.

**Definition 9.** The error rate $P_e^M$ of a privacy preserving rank aggregation algorithm $g$ with $M$ candidates is defined as $\mathbb{E}1_{\{g(q) \neq g(\hat{q})\}}$.

### 4.3 General Error Bounds

In this section, we discuss the error rates in the rank aggregation problem. We give the expression for the general error rate and derive an upper bound on the error rate for all positional ranking rules under the assumption that profiles are uniformly distributed.

#### 4.3.1 Geometric Perspective of Positional Ranking Systems

We normalize the anonymous profile by dividing by the number of voters $N$. The resulting vector $v = q/N$ is the empirical distribution of votes, $v \in [0,1]^M$. All empirical distributions are contained in a unit simplex, called the *rank simplex*:

$$\mathcal{V} = \{v \in \mathbb{R}^M : \sum_{i=1}^{M!} v_i = 1 \text{ and } v_i \geq 0 \text{ for } \forall i\}. \quad (4.6)$$
A rank simplex with \( M \) candidates has a dimension of \( M! - 1 \). We assume that the normalized profile \( v \) is uniformly distributed on the rank simplex \( V \).

Geometrically, a ranking rule is a partition of the rank simplex. For positional ranking rules, the rank simplex is partitioned into \( M! \) congruent polytopes by \( \binom{M}{2} \) hyperplanes. Each polytope represents a ranking, and each hyperplane represents the equality of the score of two candidates. Moreover, each polytope is uniquely defined by \( M - 1 \) hyperplanes and the faces of the rank simplex \( V \). An example of how to define the hyperplane from given ranking rule will be given in Section 4.4.

**Proposition 1.** Let

\[
\hat{v} = v + \omega \tag{4.7}
\]

where \( \omega \) is a \( M! \)-dimensional random variable with distribution

\[
\mathcal{N}(0, \hat{\sigma}^2 I_{M! \times M!}),
\]

where \( \hat{\sigma}^2 = \frac{2\ln(2/\delta)}{\epsilon^2 N^2} \). We have

\[
\mathbb{E}1_{\{g(q) \neq \hat{g}(\hat{q})\}} = \mathbb{E}1_{\{g(v) \neq \hat{g}(\hat{v})\}}.
\]

**Proof.** This follows directly from the scale invariant property of the ranking rules. \( \square \)

**Remark:** Note that \( \hat{v} \) may not be in the probability simplex. The ranking result of \( \hat{v} \) is uniquely defined by the cone formed by \( M - 1 \) hyperplanes representing the equality of scores of two candidates.

### 4.3.2 An Upper Bound on the General Error Rate

Rather than providing different upper bounds for each and every positional rule, we derive a general bound that works for any positional rule. Therefore, the user can
decide which positional rule to apply to the queried noisy histogram, and the system has some guarantee on the error rate given the privacy level.

If noise switches the order of the scores of any two candidates, then the final ranking necessarily changes. Let \( S_i(v), S_j(v) \) denote the score of candidate \( i \) and \( j \) for an arbitrary positional rule given the profile \( v \). As mentioned in Section 4.3.1, there are \( \binom{M}{2} \) hyperplanes separating the simplex into \( M! \) polytopes. The hyperplanes are defined by \( S_i = S_j \) for any pair of candidates \( i, j \), and there are \( \binom{M}{2} \) such pairs. Let \( \beta_{ij} \) denote the unit normal vector of hyperplane \( \mathcal{H}_{ij} : S_i = S_j \). That is,

\[
||\beta_{ij}||_2 = 1 \tag{4.8}
\]

Then \( \beta_{ij} \cdot w \) is the scalar projection of \( \beta_{ij} \) for vector \( w \). Let \( D_{ij}(v) \) be the distance from \( v \) to hyperplane \( \mathcal{H}_{ij} \). Given the uniform distribution of \( v \) over the rank simplex, \( D_{ij}(v) \) is a continuous random variable that takes values on \([-\sqrt{2}, \sqrt{2}] \) (\( \sqrt{2} \) is the edge length of the probability simplex). The sign indicates on which side of the hyperplane \( v \) locates. Let \( p_D \) denote the probability density function of \( D_{ij} \). By the neutrality of positional rules, \( p_D \) is identical for any \( i \neq j \) and \( p_D(l) = p_D(-l) \). By symmetry,

\[
\int_0^{\sqrt{2}} p_D(l)dl = \frac{1}{2}. \tag{4.9}
\]

Geometrically, \( p_D(l) \) is proportional to the \((M! - 2)\)-measure of the cross section of the hyperplane \( \mathcal{H}_{ij}(l) \) with the simplex, where \( \mathcal{H}_{ij}(l) \) is parallel to \( \mathcal{H}_{ij} \) with distance \( l \).

**Lemma 8.** Let \( p_D \) be as defined as above. Then \( p_D \) is maximal at 0 on \([0, \sqrt{2}]\) for any positional rule.

**Proof.** Let \( \mathcal{H} \) be the hyperplane defined by the equality of the score of two candidates for an arbitrary positional rule, and \( \beta \) be the unit normal vector of \( \mathcal{H} \). That is, \( \mathcal{H} = \{ v \in \mathbb{R}^M : \beta v = 0 \} \). Let \( \mathcal{H} + s\beta \) denote the hyperplane \( \beta v = s \). Let \( X_1, \ldots, X_M \)
be i.i.d. random variables with the following density function:

\[ f(x) = \begin{cases} 
  e^{-x} & \text{if } x \geq 0 \\
  0 & \text{otherwise.}
\end{cases} \quad (4.10) \]

That is, \( X_j \)'s are independent exponential random variables with parameter \( \lambda = 1 \).

The density of the random variable \( Y = \sum_{i=1}^{M!} \beta_j X_j \) is [67]

\[ G(s) = \int_{\mathcal{H} + s\beta} \prod_{j=1}^{M!} f(x) dVol_{\mathcal{H}} \quad (4.11) \]

where \( Vol_{\mathcal{H}} \) denotes the Lebesgue measure on \( \mathcal{H} \). It is shown in [67] that

\[ Vol_{M!-2}(\mathcal{H} \cap \mathcal{V}) = \frac{\sqrt{M!}}{\Gamma(M! - 1)} \int_{\mathcal{H}} \prod_{j=1}^{M!} f(x) dVol_{\mathcal{H}} \quad (4.12) \]

where \( Vol_{M!-2} \) denotes \( M! - 2 \)-dimensional volume, \( \mathcal{V} \) is the unit regular \( M! - 1 \)-simplex embedded in \( R^{M!} \), as defined in Equation (4.6). This result is shown in [67] for \( \mathcal{H} \) passing through the origin and centroid, but it holds for any hyperplane, i.e.,

\[ Vol_{M!-2}((\mathcal{H} + s\beta) \cap \mathcal{V}) = \frac{\sqrt{M!}}{\Gamma(M! - 1)} G(s). \quad (4.13) \]

The characteristic function of \( Y \) is

\[ \phi_Y(t) = \prod_{j=1}^{M!} \phi_{X_j}(\beta_j t) = \prod_{j=1}^{M!} (1 + i\beta_j t)^{-1}. \quad (4.14) \]

Note that for any entry \( j \), there is a corresponding entry \( j' \) such that the \( j' \)th ranking is the reversed order of the \( j \)th ranking. By symmetry, \( \beta_j = -\beta_{j'} \), \((1 + i\beta_j t)(1 + i\beta_{j'} t) = \)
\[ 1 + \beta_j^2 t^2. \] Without loss of generality, suppose \( \beta_j > 0 \) for \( 1 \leq j \leq M/2 \), then

\[
\phi_Y(t) = \prod_{j=1}^{M/2} (1 + \beta_j^2 t^2)^{-1}. \tag{4.15}
\]

Since \( \phi_Y(t) \) is always real and positive, by Bochner’s theorem \([8]\), \( G(s) \) is a positive-definite function, i.e.,

\[
|G(s)| \leq G(0).
\]

This is also easy to prove by directly applying the inverse Fourier Transform:

\[
|G(s)| = \left| \frac{1}{2\pi} \int_{-\infty}^{+\infty} \phi_Y(t) e^{-ist} ds \right|
\leq \frac{1}{2\pi} \int_{-\infty}^{+\infty} |\phi_Y(t) e^{-ist}| ds
= \frac{1}{2\pi} \int_{-\infty}^{+\infty} \phi_Y(t) |e^{-ist}| ds
= \frac{1}{2\pi} \int_{-\infty}^{+\infty} \phi_Y(t) ds
= G(0). \tag{4.16}
\]

Thus we have,

\[
\text{Vol}_{M-2}( (\mathcal{H} + s\beta) \cap \mathcal{V}) \leq \text{Vol}_{M-2}(\mathcal{H} \cap \mathcal{V}).
\]

\[
\square
\]

**Lemma 9.** The ranking error rate \( P_e^M \) satisfies

\[
P_e^M \leq \left( \frac{M}{2} \right) \cdot 2 \int_0^\tau p_D(l) Q \left( \frac{l}{\hat{\sigma}} \right) dl + Q \left( \frac{\tau}{\hat{\sigma}} \right), \forall \tau > 0,
\]

74
for all positional ranking aggregation algorithms with \( M \) candidates and \( N \) voters, taking input from the \((\epsilon, \delta)\)-differentially private system defined in Section 4.2.3.

**Proof.** The main idea of the proof is as follows. Divide the rank simplex into two parts: a “high error” region, denoted as \( \mathcal{R}_H \), and a “low error” region, denoted as \( \mathcal{R}_L \), as shown in Figure 4.1. \( \mathcal{R}_H \) consists of the thin slices of the simplex close to the boundary hyperplanes. \( \mathcal{R}_L \) occupies most of the simplex, but \( P(\text{error} \mid v \in \mathcal{R}_L) \) is upper bounded by the error rate at the point closest to the boundary. We choose an appropriate thickness \( \tau \) of \( \mathcal{R}_H \) such that the sum of the error rate of the two parts is minimized. Thus we have,

\[
P_e^M = P_{e \in \mathcal{R}_H} + P_{e \in \mathcal{R}_L} \\
\leq \binom{M}{2} \cdot P(S_i, S_j \text{ switches order in } \mathcal{R}_H) + P_{e \in \mathcal{R}_L} \\
= \binom{M}{2} \cdot 2 \int_0^\tau p_D(l)P(\beta_{ij} \cdot \omega > l)dl + P_{e \in \mathcal{R}_L} \\
= \binom{M}{2} \cdot 2 \int_0^\tau p_D(l)Q\left(\frac{l}{\hat{\sigma}||\beta_{ij}||_2}\right)dl + P_{e \in \mathcal{R}_L} \tag{4.17}
\]

\( Q(\cdot) \) is the tail probability of the standard normal distribution and is decreasing on \([0, +\infty)\). Thus for the “low error” region, we have,

\[
P_{e \in \mathcal{R}_L} < P(v \in \mathcal{R}_L) \cdot Q\left(\frac{\tau}{\hat{\sigma}||\beta_{ij}||_2}\right) \\
< Q\left(\frac{\tau}{\hat{\sigma}||\beta_{ij}||_2}\right) \tag{4.18}
\]

From Equation \(4.8\), \(4.17\), and \(4.18\), we have,

\[
P_e^M \leq \binom{M}{2} \cdot 2 \int_0^\tau p_D(l)Q\left(\frac{l}{\hat{\sigma}}\right)dl + Q\left(\frac{\tau}{\hat{\sigma}}\right). \tag{4.19}
\]

75
Theorem 5. For any positional ranking aggregation algorithm with $M$ candidates and $N$ voters, taking input from the $(\epsilon, \delta)$-differentially private system defined in Section 4.2.3, the ranking error rate $P_e^M(N)$ satisfies

$$P_e^M(N) \leq \left(\frac{M}{2}\right) \frac{M! - 1}{\sqrt{2}} \tau + Q\left(\frac{\epsilon N \tau}{\sqrt{2 \ln(2/\delta)}}\right), \forall \tau > 0.$$ 

Proof. By Lemma 9, we have,

$$P_e^M \leq \left(\frac{M}{2}\right) \cdot 2 \int_0^\tau p_D(l) Q\left(\frac{l}{\sigma}\right) \, dl + Q\left(\frac{\tau}{\sigma}\right)$$

$$\leq \left(\frac{M}{2}\right) \cdot 2 \int_0^\tau p_D(l) Q(0) \, dl + Q\left(\frac{\tau}{\sigma}\right)$$

$$= \left(\frac{M}{2}\right) \cdot \int_0^\tau p_D(l) \, dl + Q\left(\frac{\tau}{\sigma}\right) \quad (4.20)$$

By Lemma 8, for any positional rules, $p_D(l) \leq p_D(0)$. Hence we have,

$$P_e^M \leq \left(\frac{M}{2}\right) \cdot \int_0^\tau p_D(0) \, dl + Q\left(\frac{\tau}{\sigma}\right)$$

$$= \left(\frac{M}{2}\right) \cdot p_D(0) \tau + Q\left(\frac{\tau}{\sigma}\right) \quad (4.21)$$

For positional rules, all hyperplanes $\mathcal{H}_{ij}$ pass through the $(M! - 1)$-simplex centroid for any $i, j \in \{1, \ldots, M\}$ since the profile at the centroid must be a tie for all candidates due to symmetry. From the literature in high dimensional geometry [67], we know that the largest cross section through the centroid of a regular $M! - 1$-simplex is exactly the slice that contains $M! - 2$ of its vertices and the midpoint of the remaining two vertices. The $(M! - 2)$-measure of the cross section is $\sqrt{M!} / (\sqrt{2}(M! - 2)!)$
for the probability simplex. Since the $(M! - 1)$-measure of the probability simplex is $\sqrt{M!/(M! - 1)!}$, we have,

$$p_D(0) \leq \frac{\sqrt{M!}}{\sqrt{2(M! - 2)!}} = \frac{M! - 1}{\sqrt{2}} \quad (4.22)$$

From Equations (4.21) and (4.22), and the fact that $\hat{\sigma}^2 = 2 \ln(\frac{2}{\delta})/\epsilon^2 N^2$, we have

$$P_e^M(N) \leq \binom{M}{2} \frac{M! - 1}{\sqrt{2}} \tau + Q \left( \frac{\tau}{\hat{\sigma}} \right)$$

$$= \binom{M}{2} \frac{M! - 1}{\sqrt{2}} \tau + Q \left( \frac{\epsilon N \tau}{\sqrt{2 \ln(2/\delta)}} \right) \quad (4.23)$$

By taking the derivative with respect to $\tau$, we can show that the right side of Equation (4.23) is minimized when

$$\tau = \frac{\sqrt{2 \ln(2/\delta)}}{\epsilon N} \sqrt{-2 \ln \frac{\sqrt{\pi \ln(2/\delta)} M(M - 1)(M! - 1)}{\sqrt{2} \epsilon N}}. \quad (4.24)$$

**Remark:** To better understand this upper bound, we can use a Q-function approximation to represent the result of Theorem 5. It is known that

$$Q(x) \leq \frac{e^{-x^2/2}}{\sqrt{2\pi x}}, \forall x > 0. \quad (4.25)$$

This is a good approximation when $x$ is large [33]. Thus we can rewrite Equation (4.23) as

$$P_e^M(N) \leq \binom{M}{2} \frac{M! - 1}{\sqrt{2}} \tau + \frac{\sqrt{\ln(2/\delta)}}{2\sqrt{\pi} \epsilon N \tau} e^{-\frac{(\epsilon N \tau)^2}{2\ln(2/\delta)}}, \forall \tau > 0. \quad (4.26)$$
Figure 4.1: An example of Petrie polygon (skew orthogonal projections) of three candidates. Three hyperplanes, under Borda count ranking rule, separate the simplex into six polytopes.

We can further simplify the expression by letting

\[
\tau = 2\sqrt{\ln N \ln(2/\delta)}/(\epsilon N):
\]

\[
P^M_N \leq \frac{1}{N} \left( \binom{M}{2}(M! - 1)\sqrt{2\ln N \ln(2/\delta)} \cdot \frac{1}{\epsilon} + \frac{1}{2\sqrt{\pi \ln N}} \right). \tag{4.27}
\]

It is shown in (4.27) that the error rate goes to 0 at least as fast as \(O(\frac{\sqrt{\ln N}}{N})\) for fixed \(\delta, \epsilon\).

4.3.3 Asymptotic Error Rates

In this section, we analyze the asymptotic error rates for any positional ranking rule.

We start by showing a tighter bound on the general error rate that can be derived from the proof of Theorem 5.

**Lemma 10.** An upper bound for the ranking error rate of any \((\epsilon, \delta)\)-differentially private positional ranking system with \(M\) candidates and \(N\) voters is

\[
\binom{M}{2}\sqrt{2}(M! - 1)Q \left( \frac{\epsilon N \tau}{2\sqrt{2\ln(2/\delta)}} \right)^\tau + \frac{Q}{\epsilon^\tau} \left( \frac{\epsilon N \tau}{\sqrt{2\ln(2/\delta)}} \right)
\]

78
for $\forall \tau > 0$.

**Proof.** Since the Q-function is convex on $[0, +\infty)$, by Jensen’s Inequality, from Lemma 8 and Lemma 9 we have

$$P_{\epsilon}^M(N) \leq \left( \frac{M}{2} \right) \cdot 2 \int_0^\tau p_D(l)Q\left( \frac{l}{\sigma} \right) dl + Q\left( \frac{\tau}{\sigma} \right)$$

$$\leq \left( \frac{M}{2} \right) \cdot 2 \int_0^\tau p_D(0)Q\left( \frac{l}{\sigma} \right) dl + Q\left( \frac{\tau}{\sigma} \right)$$

$$\leq \left( \frac{M}{2} \right) \cdot 2p_D(0)Q\left( \frac{\tau}{2\tilde{\sigma}} \right) + Q\left( \frac{\tau}{\tilde{\sigma}} \right)$$

$$= \left( \frac{M}{2} \right) \sqrt{2}(M! - 1)Q\left( \frac{\epsilon N\tau}{2\sqrt{2\ln(2/\delta)}} \right) \tau$$

$$+ Q\left( \frac{\epsilon N\tau}{\sqrt{2\ln(2/\delta)}} \right). \tag{4.28}$$

Lemma 10 slightly improves the bound in Theorem 5. We use this lemma to assist the proof of the following Theorem.

**Theorem 6.** For any positional ranking aggregation algorithm with $M$ candidates, taking input from the $(\epsilon, \delta)$-differentially private system defined in Section 4.2.3, 

$$\lim_{N \to \infty} P_{\epsilon}^M(N) = 0$$

for any given $\epsilon$ and $\delta$.

**Proof.** This directly follows from Lemma 10 and the Bounded Convergence Theorem. 

\qed
4.4 Simulation Results

In this section, we use Borda count with three candidates as an example. Once the ranking rule is known, we can derive a tighter bound than the general error rate bound in Section 4.3 because we know exactly what the pairwise comparison boundaries are. We will compare all upper bounds with the simulation error rates.

In Borda count, for every vote the candidate ranked first receives 1 point, the second receives 0.5 points, and the bottom candidate receives no points. The aggregated rank is sorted according to the total points each candidate receives. We list $3! = 6$ permutations in the following order, and we will stick to this order for the rest of this chapter: $abc, acb, cab, cba, bca, bac$. Let

$$M = \begin{pmatrix}
1 & 1 & 0.5 & 0 & 0 & 0.5 \\
0.5 & 0 & 0 & 0.5 & 1 & 1 \\
0 & 0.5 & 1 & 1 & 0.5 & 0
\end{pmatrix}. \quad (4.29)$$

Then we have

$$\begin{pmatrix}
S_a \\
S_b \\
S_c
\end{pmatrix} = Mq, \quad (4.30)$$

where $q$ is defined in Section 4.3.1 and $S_a, S_b, S_c$ are the aggregated score of candidates $a, b$ and $c$ respectively. The hyperplane $H_{ab}$ satisfies $S_a = S_b,$

$$2v_1 + 2v_2 + v_3 + v_6 = v_1 + v_4 + 2v_5 + 2v_6 \quad (4.31)$$

i.e.

$$H_{ab} : v_1 + 2v_2 + v_3 - v_4 - 2v_5 - v_6 = 0 \quad (4.32)$$
Similarly, we have

\[ H_{bc} : v_1 - v_2 - 2v_3 - v_4 + v_5 + 2v_6 = 0 \]  

(4.33)

\[ H_{ac} : 2v_1 + v_2 - v_3 - 2v_4 - v_5 + v_6 = 0 \]  

(4.34)

With Equations (4.32), (4.33) and (4.34), we can compute the volume of the cross section made by the hyperplane cutting through the probability simplex (4.6), using methods proposed in [38]. Then an upper bound specifically for Borda count can be derived with a similar approach as Theorem 5 or Lemma 10.

Figure 4.2 shows the simulation results of Borda count with 3 candidates and 2,000 voters, repeated 100,000 times. We set \( \delta = 5 \times 10^{-4} \) (which is 0.1 divided by the number of voters), and plot the graph of error rate with \( \epsilon \) taking values between 0.05 and 0.24. We compare the simulation results with the general upper bound derived in Theorem 3 and the improved upper bound in Lemma 10, as well as the ranking rule-specific upper bound described above.

Figure 4.3 shows the simulation results for Borda count with 3 candidates with fixed \( \epsilon \), repeated 20,000 times. We set \( \epsilon = 0.1 \) and \( \delta = 0.1/N \), where \( N \) is the number of voters. The number of voters varies from 1,000 to 100,000. The error vanishes fast with a growing number of voters, even if we set \( \delta \) to be inversely proportional to the number of voters. We also compare the simulation results with the general upper bound derived in Theorem 3 and the improved upper bound in Lemma 10, as well as the ranking rule-specific upper bound described above.
Figure 4.2: Error rate vs $\epsilon$.

Figure 4.3: Error rate vs number of voters.
Chapter 5

Conclusion and Future Work

The thriving development of big data holds great promise for many applications, but also leads to potential risk on privacy issues. Information aggregation is an important field in big data. This dissertation has focused on three problems in information aggregation: quantized consensus, recommender systems, and ranking. A common underlying thread in this dissertation is privacy preservation. We hope that this work will contribute to a better understanding of the topics covered in this dissertation and stimulate further progress.

5.1 Revisit the Contribution of the Dissertation

In Chapter 2, a class of gossip quantized consensus algorithms proposed in [6] and [34] are studied. In the communication process, little information of the initial observation by agents is revealed, and only agents’ current estimates of the global average are exchanged. In the analysis, the theories of electric networks, random walks, and couplings of Markov chains are used to derive a polynomial bound on the convergence time with respect to the size of the network. This analysis can be extended to a tighter bound for certain network topologies using the effective resistance analogy.
In Chapter 3, a framework for group-based privacy preserving recommender systems is designed. The main idea is to use groups as a natural protective mechanism to preserve individual users' private preference data from the central service provider. A distributed peer-to-peer preference exchange process is designed to provide anonymity of group members. Moreover, a hybrid recommendation model based on random walks is proposed. It incorporates item content and group social information to make recommendations for groups. Personalized recommendations are made locally to group members, so that no user preference profile information is leaked to the service provider. Experiments show that by aggregating group preferences and then making recommendations, a reasonable compromise between privacy and accuracy can be achieved.

Lastly, in Chapter 4, the framework of differential privacy is applied to rank aggregation by adding noise to the ranking profile. The probability that the aggregated ranking becomes inaccurate is analyzed, and upper bounds on the error rate for all positional ranking rules are derived. The bound can be tightened by using techniques in high dimensional polytope volume computation given a specific ranking rule. The results provide insights into the trade-offs between privacy and accuracy in rank aggregation.

5.2 Future Research Directions

This dissertation has discussed some interesting topics in information aggregation. It also opens up further investigation and research.

The work in Chapter 2 proposed to use the effective resistance of the network to obtain a tighter bound than the general bound on the convergence time of quantized consensus algorithms. It would be interesting to look at the results for random geometric graphs, and compare the results with the work in [19]. It would also
be worth looking at how to extend this analysis to other gossip quantized consensus algorithms.

The experiments in Chapter 3 used movieLens datasets to evaluate the proposed group-based recommender system. Users were grouped by the characteristics of their profile information. It would be interesting to do experiments on datasets with both group and rating information, such as Yelp, a business review site. The friendship information can be used to infer group information.

Chapter 4 opened a new direction of applying differential privacy in rank aggregation. The utility was measured by the probability that the aggregated complete ranking is accurate. When the number of the candidates is large, it is often interesting and useful to look at the Top-$k$ ranking, which consists of the first $k$ candidates of the aggregated ranking. It would be worth investigating the error bound for differentially private Top-$k$ ranking.
Bibliography


87


