# Preference Elicitation for Group Decisions Using Voting Theory

Thesis submitted in partial fulfillment
of the requirements for the degree of
"DOCTOR OF PHILOSOPHY"

by

Lihi Dery

Submitted to the Senate of
Ben-Gurion University of the Negev

July 2014

Beer-Sheva

This work was carried out under the supervision of

Dr. Meir Kalech

Prof. Lior Rokach

Prof. Bracha Shapira

In the Department of Information Systems Engineering

Faculty of Engineering Sciences

# Acknowledgments

I began working with Prof. Lior Rokach when I was at my last year of undergraduate studies. This was enough to intrigue me to stay. Throughout the years I was often astounded by his quick thinking and never-ending scientific optimism which often made me leave his office muttering to myself: why didn't I think about it?". Prof. Bracha Shapira, the head of the Information Systems Department has been my feminine role model, showing me that research and motherhood do not necessarily contradict. Dr. Meir Kalech was always there when I needed him. Through long periods of this research it was only his encouragement, support and advice that kept me going. I especially enjoyed our discussions around paper deadlines.

Prof. Daniel Berry, from the University of Waterloo, changed my perspective and motivated me with his presentation: "Advice for finishing that Damn PhD". I thank him for caring enough to write and lecture on this important subject.

For some reason, students are not usually mentioned in acknowledgments. However, for me teaching served as a constant reminder of the lively side of higher education. I especially thank my project students: Inon Golan, Eli Ponyatovski and Aviad Carmeli for their contribution of ideas and data.

Friends in grad school shared this journey with me, were and still are always there for support, reminding me that I am not alone on this path.

I could not have managed without my parents who changed their plans and gave up activities so that I could meet (yet another) deadline. Their dedication to my kids and therefore to my PhD is one of the reasons I can now write these acknowledgments.

My beloved husband and partner in life kept me focused on what's important. My kids, born into this PhD, are the true scientists: they invent research questions, build hypothesis, experiment with the world and arrive at amazing conclusions. They inspire me. It will take some years before they learn English, or even learn to read and write and I hope that when they will finally read this it will make them proud of their mom.

Advisers, colleagues, students, friends and family: I thank you all. You are all part of this work. I could not have done it without you.

In memory of my grandma

Ruth Naamani (1923 - 2012)

# Abstract

This study addresses the issue of preference elicitation for group decision making using voting rules. We propose a general, domain-free framework for preference management, where the goal is to minimize the communication cost with the users. We introduce novel heuristics and show how they can operate under rating and ranking voting protocols, specifically under the Range and the Borda protocols. We suggest an interactive incremental framework where at each step one user is queried for either her rating for one item or for her ranking order of two items. We propose two approaches for heuristics that determine what query to select next (i.e., whom to query regarding what item or items). One heuristic computes the information gain of each potential query. The other heuristic uses the probability distribution of the voters' preferences to select the candidate most likely to win and the voter that is expected to maximize the score of that item. Both heuristics rely on probabilistic rating distributions. We show how these distributions can be estimated. The rating distributions are updated iteratively, allowing their accuracy to increase over time.

Although outputting a definite winner is the most accurate result, we also examine possible effort-accuracy tradeoffs and aggregation strategies in preference elicitation. First, we suggest terminating preference elicitation sooner by returning $k$ alternatives to the group members where one of the alternatives is the winner, rather than returning just one winner item. Second, we suggest computing approximate winner or winners with some confidence level. On one hand, receiving an approximate winner item is less accurate; on the other hand, it further reduces the communication cost. Finally, we suggest considering the aggregation strategy when combining the user preferences. We show that the aggregation strategy affects the communication cost required of the preference elicitation and we compare two state-of-the-art aggregation strategies: the Majority based strategy and the Least Misery strategy. We demonstrate the effectiveness of our framework by evaluating the different heuristics on four real-world datasets. In order to examine the possible effect of different data characteristics we also use simulated datasets were we can play with the data parameters.

Keywords: Group recommender systems, preference elicitation, multi-agent voting

# Table of Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Joint decisions are often required in common daily scenarios. For example, a group who wishes to dine together may appreciate recommendations on relevant restaurants that fit their joint preferences (Berkovsky and Freyne2010). Other examples include: a group of users wishing to engage in a joint activity such as watching a movie (O’connor et al. 2002) , a TV show (Yu et al. 2006; Masthoff2004; Senot et al. 2011), listening to music (McCarthy and Anagnost1998), and travelling (McCarthy et al. 2006).  In a different context, an acceptance committee is required to reach a joint decision; the members need to decide which candidates to choose for a graduate program out of many applicants (Xia and Conitzer2011).

When the group members’ preferences regarding the items are known, preference elicitation is not required and some aggregation strategy (Masthoff2011) is used to compute the item that best suites the group. However, often full preferences are not readily available for different reasons. First, due to privacy concerns, even when available, full preference revelation should be treated with caution. Second, it is sometimes impractical to collect complete preferences due to the communication burden (Conitzer and Sandholm2005; Konczak and Lang2005) or to voter limitations. Consider, for example a meeting scheduling application whose purpose is to set a time for a conference (Kalech et al. 2011), or an application that recommends movies, such as Netflix (www.netflix.com). It is impractical to expect the voters to provide their preferences on all available options as there might be hundreds available. In this study we focus on a preference elicitation model for a group of users.

## 1.1 Motivation

The motivation for this research is drawn from two domains: social choice (Brandt et al. 2013) and recommender systems (Resnick and Varian1997). Both domains assist users in reaching a joint decision. Social choice originates in economics and political science and is concerned with different aspects of aggregating the preferences of users, usually termed *agents* or *voters*. In this study we use voters and users *interchangeably*. Recommender systems seek to predict items to users. We are interested specifically in group recommender systems where items are predicted to a group of users.

Preference elicitation is addressed by both of these fields. In social choice, preference elicitation is essential when not enough information is available in order to reach a verdict (i.e., a winning item according to some voting protocol). However, in recommender systems, preference elicitation is viewed as a measure for improving the prediction accuracy.

To illustrate a relevant social choice scenario, we return to the scenario of the faculty acceptance committee[1] (Xia and Conitzer2011). A committee assembles in order to select a fixed number of candidates out of a field of hundreds for a Ph.D. program. Ideally, a group decision requires each committee member to express his or her opinion about each of the applicants; then a joint decision is reached based on all opinions. As their time is limited, the committee members do not have the resources for such a process. Instead, each member reads a portion of the total requests and then grades her assigned applicant files on a scale from 1 to 5. Applicants in the high and low range of scores do not require any special care as they are immediately accepted or declined; it is those who receive an average rating which present a problem. Their files need to be reread by other members in order to receive additional opinions. The question is then, which committee member should be assigned which file to read so that a decision will be reached in optimal time. Assuming we know the member's previous grading pattern (e.g., members who tend to grade high/low as opposed to members who give all candidates average scores), we can define the member's grading probability distribution. Then *voter-item query pairs* can be selected: the voter being the committee member, the item being the applicant assigned to her for grading and the query being a request for the voters rating for this item. By carefully selecting specific voter-item query pairs, the interaction with the users can be reduced. The goal is to

[1] This example is taken from a private discussion with Vincet Conitzer in 2011

reach a decision with partial information (i.e., to reduce the number of interactions with the voters needed in order to reach a verdict).

To illustrate the relevance of our research to group recommendation, consider the following: a group recommender system (GRS) can process $N$ items, where N might be very large. For example, in the Netflix dataset $N \cong 16{,}000$ movie items. The GRS ranks the items according to their predicted relevance to the group and then returns the top-ranked items. Note that the items returned by the GRS are *predicted* items, and their relevance to the group depends on the accuracy of the GRS. We propose to add a *preference elicitation* procedure that follows the recommendation process and uses the predicted items as input. This enables us to enhance the recommendation system since we are then able to return an item or items that *certainly* fit the group's preferences. Preference elicitation is the process of collecting the preferences from the users. It would be impractical to suggest an elicitation process on large numbers of items (e.g., when $N = 16{,}000$). Studies have shown that too much choice can be demotivating. Users are more satisfied when presented with a single-digit number of options to choose from (Iyengar and Lepper2000). Thus we suggest to narrow down the ranked list of $N$ items provided by a recommender system, and to apply a preference elicitation procedure on the top-ranked items only.

Preference elicitation requires time and effort so the goal is to stop the elicitation as soon as possible. In the worst case for most voting protocols all the preferences are needed in order to determine a winning item (i.e., an item that most certainly suits the group's joint preferences) (Conitzer and Sandholm2005). Nevertheless, in this research we show that in practice the required information can be cut by more than 50%. Given partial preferences, it is possible to define the set of the necessary winners (Konczak and Lang2005), i.e., items which must necessarily win, as well as the set of possible winners, i.e., items which can still possibly win. These definitions enable the elicitor to determine whether there is need of more information concerning the voters' preferences.

## 1.2 Overview

In this study we suggest a framework for efficient preference elicitation for joint group decisions. We propose novel heuristics that aim at minimizing the preference elicitation costs (i.e., the cost of communicating with the users). In addition, we investigate approaches for managing the tradeoff between accuracy and elicitation reduction. Finally, we examine two strategies for aggregating the users' preferences in order to reach a verdict. To evaluate our ideas, we have conducted experiments with real-world datasets as well as with simulated data. The following sections provide an overview of the study and our underlying assumptions.

### 1.2.1 Heuristics for Minimizing Preference Elicitation Costs

We introduce heuristics for preference elicitation. Computing the optimal minimal set of queries required to determine a winner is computationally intractable due to the combinatorial space of queries' orders (Walsh2008). Therefore, we propose two heuristic approaches for determining a winner. As we illustrate, each approach is preferred under different circumstances.

Both approaches proceed iteratively; at each step one user is selected and queried for her preferences. The queries are performed under one of the following protocols: the Range voting protocol and the Borda voting protocol. The Range voting protocol allows users to rate an item within a fixed range of scores. A selected user is queried for her rating for a specific item, forming a *voter-item* query. The Borda voting protocol assumes users have a predefined order of preferences over the items, and can be related to pairwise comparisons. A selected user is queried for her preference between two items, forming a *voter-item-item* query.

To determine a query, the first approach heuristically computes the information gain of each potential query based on the entropy of the item's probability to win. The query that maximizes the information gain is selected. The second approach uses the probability distribution of the voters' preferences to select the candidate most likely to win and the voter that is expected to maximize the score of that item. In both algorithms, probability distributions of the users' preferences are computed and updated as new information is revealed. The heuristics output a necessary winner item (Konczak and Lang2005), i.e., a definite winner.

Previous studies have reported the theoretical upper and lower bounds of the required communication with the voters (Conitzer and Sandholm2005). However, to the best of our knowledge, only two studies propose practical algorithms for preference elicitation (Kalech et al. 2011; Lu and Boutilier 2011). The first assumes each voter holds a predefined decreasing order of the preferences where the voters are requested to submit their highest preferences in an iterative process. However, the requirement to predefine preferences may be inconvenient to the users. Therefore, we require voters to respond and rate a specific item only when necessary.

Furthermore, the procedure suggested by Kalech et al. (Kalech et al. 2011) does not necessarily reduce communication significantly since the authors request the rating of one item from all the users, thus requiring the voting center to contact each of the users at each step. In the second study (Lu and Boutilier 2011) a practical elicitation process is proposed for the Borda voting protocol. This algorithm is confidential; the authors did not provide details about their algorithm which prohibited its expansion to the Range voting protocol or to other aggregation strategies.

### 1.2.2 Approximations for Further Reducing Preference Elicitation

Preference elicitation costs can be further reduced by trading off the accuracy and communication costs. We define one unit of communication as one elicitation request. First, a tradeoff exists between the *amount of items* outputted to the group and the *amount of queries* required. Less elicitation effort is required for outputting top-$k$ items than for outputting one necessary winner (i.e., $k = 1$). Although outputting one definite winner is the most accurate result, there are advantages to outputting $k$ items where one of them is the winner. Not only issues such as communication costs and privacy are reduced, it may actually be preferred to present a *few* alternatives to the user since, if one of the alternatives is unavailable, we can quickly switch to another alternative without requiring more elicitation (Brandt et al. 2013).

Another tradeoff is the one that exists between the *accuracy* of the proposed winner item and the *amount of queries* required. We suggest outputting an item that approximately suits the group with some confidence level $1 - \alpha$ rather than outputting an item that definitely satisfies the group (where $\alpha = 0$). As we later show, the confidence level is based on the items' winning probabilities. To reduce the elicitation costs even further, the two methods can be combined and top-$k$ approximate items can be offered to the group. For example, consider a group that wishes to watch a movie together and needs to choose one from the current list available in theaters. The members of the group define the amount of options they wish to receive ($k$) and the level of confidence in the options ($0 \leq \alpha \leq 1$). Thus, we wish to define new preference elicitation termination conditions. The first is *multi-winner termination*; terminate elicitation once a few items are found where one of them is the winner. The second is *approximate multi-winner termination*; terminate elicitation when a few items are found and one of them is approximated to be the winner item.

### 1.2.3 Aggregation Strategies

In order to compute a winning item, or, as discussed in the previous section, a few winning items or a winning items within top-$k$ items, the user preferences need to be aggregated using a fair aggregation strategy. In his well-known work, Arrow shows that there is no perfect aggregation system (Arrow 1951). One of the major differences between aggregation strategies is the social environment in which they are used; in particular, the perspective in which fairness is viewed. The emphasis can be either on the individual user or towards the majority of the group (Jameson and Smyth2007). Two aggregation strategies that differ in their emphasis are the *Majority Based Strategy* and the *Least Misery Strategy*. In the *Majority Based Strategy* the users' ratings of the different items are aggregated and the item with the highest total value is the winner. In the *Least Misery Strategy* the chosen item cannot be the least preferred by any of the users. The idea is that a group is as happy as its least happy member (Masthoff2011). One of the contributions of this research is in proposing an iterative preference elicitation algorithm which fits these strategies. In the social choice literature, these strategies are known as the utilitarian and the egalitarian settings.

### 1.2.4 Evaluation

We evaluate our methods on four real-world domains: Netflix data (http://www.netflixprize.com), Sushi dataset (Kamishima et al. 2005), Pubs dataset and Restaurants dataset[2]. We also perform evaluation on simulated data which allows us to manipulate the data and thus further study the different parameters.

Experiments highlight that the suggested model reduces the communication cost in the elicitation process while guaranteeing that a winning candidate is found. For example, in the real-world Netflix contest dataset, we show that the communication cost can be reduced by more than 50%. Furthermore, the analysis of the heuristics under different settings provides interesting insights. For example, DIG heuristic excels when the data is noisy and the voters do not vote according to a certain pattern, while ES runs faster and performs better when there is some known pattern to the voter preferences (e.g., if it is known from previous analysis that the voters generally prefer to watch the newest movie in the cinema). We show that selecting the suitable

[2] The Pubs dataset and Restaurants dataset are taken from "Lets Do It" recommender systems, developed by Eli Ponyatovski and Aviad Carmeli, 4th year students in the Information Systems Department, under the supervision of: Lihi Dery, Ofrit Lesser and Meir Kalech, Ben Gurion University 2014.

aggregation strategy and relaxing the termination condition can reduce communication up to 90%.

### 1.2.5 Assumptions

In this study we follow the underlying assumptions:

1. Unknown preferences estimation - uncertainty regarding the voter preferences is facilitated by creating and updating voter-item probability distributions. For example, in the case of a group of users wishing to watch a movie together, the distribution can be inferred from rankings of these movies by similar users using collaborative filtering methods (Koren and Sill2011). We present and demonstrate a method for calculating probability distributions (see Chapter 4 section 2).
2. User response - we assume that users' preferences are unknown in advance, but can be acquired during the process (i.e., a user who is asked about her preference on an item, responds to the request). Note that the user is not required to decide on all of her preferences beforehand.
3. Sincere communication with the user - we assume that a user submits her true preferences. Therefore, in this research we do not consider manipulation (Gibbard1973, Satterthwaite 1975).
4. Equal communication cost – we assume that the cost of asking a user for her preferences is equal for all users and for all items.
5. Consistency – we assume consistency in voting preferences. If this assumption is not held, intransitive preferences might occur, and then no solution can be found. To the best of our knowledge, this is an assumption kept throughout the field of voting and social choice. Although Tversky (1969) has shown that preferences might be intransitive, there have been other that have shown that in most cases, voters are in fact transitive (Regenwetter et al. 2011). Another form of inconsistency is when manipulation is performed. However this is out of scope of this research.

## 1.3 Main Contribution and Dissertation Overview

The overall structure of the study takes the form of seven chapters, including this introductory chapter. We lay out the related work in Chapter 2. We present the problem description and a general preference elicitation model, with the goal of reducing the communication cost in the preference elicitation process in Chapter 3. We implement the framework on ranking (Range) and non-ranking (Borda) voting protocols. The main contributions of this research are:

1. **A preference elicitation model for the Range voting protocol (Chapter 4):** We present an elicitation model and novel heuristics for preference elicitation for the Range protocol. The Range protocol is very common in Recommender systems settings as it requires users to submit a score. Since computing the optimal minimal set of queries that are required to determine a winner is computationally intractable due to the combinatorial space of queries' orders, we propose two novel heuristic approaches for determining a winner. We show that each approach has an advantage under different circumstances. Both approaches proceed iteratively, selecting a voter-item pair and querying the selected voter for her score on the item. To determine a voter-item query pair, the first algorithm heuristically computes the information gain of each potential query based on the entropy of the item's probability to win. The query that maximizes the information gain is selected. The second algorithm uses the probability distribution of the voters' preferences to select the candidate most likely to win and the voter that is expected to maximize the score of that item. In both algorithms, voter-item probability distributions are computed and updated as new information is revealed. This is achieved by computing a nonparametric probability distribution for each voter's preferences of items, i.e., for voter-item pairs. The algorithms return a definite winner item.
2. **A preference elicitation model for the Borda voting protocol (Chapter 5):** We present an elicitation model and novel heuristics for preference elicitation for the Borda protocol. The Borda voting protocol requires users to rank their preferences so that no two items can receive the same score. Pairwise comparison queries can be related to the Borda voting protocol, since users are already required to have a fixed ranked list of preferences for items. Studies have shown it is easier for users to state opinions when the queries are pairwise (Balakrishnan and Chopra2012). The Borda protocol is different from the Range protocol in the methods for computing the necessary winner, the item winning probability, and the distribution model.

3. **Tradeoffs and Aggregation Strategies in Preference Elicitation (Chapter 6):** To the best of our knowledge, the following ideas have not been studied for the Range voting protocol.
    a. **Selection:** we suggest terminating preference elicitation sooner by returning $k$ alternatives to the group members where one of them is a necessary winner rather than returning just one item. This issue has recently been studied by (Lu and Boutilier2013) for the Borda voting protocol. However, the algorithm suggested by the authors remains confidential and we cannot attempt to adapt it to the Range voting protocol.
    b. **Approximation:** we suggest computing approximate winner or winners with some confidence level. For example: item $a$ is the winner with a 95% confidence level. This also shall reduce the communication cost.
    c. **Aggregation:** we suggest considering the aggregation strategy when combining the user preferences. Previous work has always implemented the Majority based strategy, however, we show that the aggregation strategy affects the effort required in the preference elicitation and we evaluate two state-of-the-art aggregation strategies.

Finally, the conclusions chapter (Chapter 7) gives a brief summary, critique of the findings' and further research areas. A schematic overview of the structure of the key chapters (Chapters 3-6) is provided in the following table.

Table 1: Schematic overview

| | | |
|---|---|---|
| General framework of preference elicitation | **Chapter 3** **The Problem Description** | |
| Preference elicitation for non-ranking and ranking voting rules | **Chapter 4** **Iterative Voting Under the Range Protocol** | **Chapter 5** **Iterative Voting Under the Borda Protocol** |
| The necessary winner | **Section 4.1** Possible maximum and possible minimum under the Range protocol | **Section 5.1** Possible maximum and possible minimum under the Borda protocol |
| Probabilistic preference distribution model | **Section 4.2** Voter-item distribution model | **Section 5.2** Voter permutations distribution model |
| Item winning probability | **Section 4.3** Dynamic programming algorithm | **Section 5.3** Monte Carlo sampling approach |
| Query selection heuristics for finding a definite winner | **Sections 4.4 - 4.5** | **Sections 5.4 - 5.5** |
| | **Chapter 6** **Tradeoffs and Aggregation Strategies in Preference Elicitation** | |
| Aggregation strategies | **Section 6.1** | Future work |
| Query selection heuristics for winner approximation | **Section 6.2.1** Selection among Top-*k* | Future work |
| | **Section 6.2.2** Approximation using a confidence interval | Future work |

## 1.4 Publications

The following papers have been published or submitted for publication as a result of the research described in this dissertation:

Chapter 4: Iterative voting under uncertainty using the Range protocol:

Naamani-Dery[3], L., Kalech, M., Rokach, L., and Shapira, B. 2010. Iterative Voting under Uncertainty for Group Recommender Systems. In *Proceedings of the Fourth ACM Conference on Recommender Systems*. ACM, New York, NY, 265-268.

Naamani-Dery, L., Kalech, M., Rokach, L., and Shapira, B. 2014. Reaching a Joint Decision with Minimal Elicitation of Voter Preferences. *Information Sciences.* Vol.278, 466-487. http://dx.doi.org/10.1016/j.ins.2014.03.065

Chapter 5: Iterative voting under uncertainty using the Borda protocol

Naamani-Dery, L., Kalech, M. and Rokach, L. 2014. Preference Elicitation for Group Decisions. *In proceedings of the Group Decisions and Negotiation conference (GDN),* Toulouse, June 2014. 193-201.

Naamani-Dery, L., Kalech, M. and Rokach, L. 2015. Preference Elicitation for Group Decisions. *Accepted to Group Decisions and Negotiation Journal.* http://dx.doi.org/10.1007/s10726-015-9427-9

Chapter 6: Tradeoffs and Aggregation Strategies in Preference Elicitation

Naamani-Dery, L., Kalech, M., Rokach, L., and Shapira, B. 2014. Preference Elicitation for Narrowing the Recommended List for Groups. *In Proceedings of the 8th ACM Conference on Recommender systems, 333-336. ACM.*

Naamani-Dery, L., Kalech, M., Rokach, L., and Shapira, B. 2014. Reducing Preference Elicitation in Group Decision Making. *Under review in Transactions on Intelligent Systems and Technology*

[3] Naamani is my maiden name (Lihi Dery)

# Chapter 2

# Related works

This chapter surveys previous work on preference elicitation and preference aggregation for group decision making. Preference elicitation is studied in two domains of interest to us: recommender systems (section 2.1 and social choice theory (section 2.2). Preference aggregation strategies are surveyed in section 2.3. A discussion on drawbacks of previous research is presented in section 2.4

## 2.1 Preference Elicitation in Recommender Systems

Recommender systems are designed to predict which items users are expected to like. We first survey personal recommendation systems and then group recommender systems.

### 2.1.1 Personal Recommender Systems

In personal recommendation systems the goal is to output specific recommendations for one user (Resnick and Varian1997). The recommendations can either be computed using available information or by constructing and matching user profiles or item profiles. When available information is insufficient, preference elicitation can be performed.

Guidelines for preference elicitation for personal recommendation systems were established by (Pu and Chen2009). In their study, the authors consider an incremental user system interaction process. The authors include the following four recommendations: how many items to display to the user, what items to display to the user, interfaces for tweaking the final decision, and interfaces for explaining the recommendations. Similarly, the model we propose in this study is incremental and interactive. At each round we elicit preferences from one user. Pending on the voting protocol, the user is either required to submit her rating for one item

(Range voting protocol) or to submit her preferred item between two items (Borda voting protocol). One of this study's focuses is determining what preference to elicit, or in Chen and Pu's terms, what items to display to the users. According to Pu and Chen's guidelines, the items suggested to the users for preference elicitation must be diverse (2nd guideline) and have a high likelihood of optimality (3rd guideline). Item diversity can be detected only if the item features are known. In our study we assume no item features are available. We follow the 3rd guideline to some extent, since we search for items that are optimal to the user. The other guidelines are less relevant for this study. For example, the results do not need an explanation interface since they are not predictions but rather approximate or necessary winners, according to preset properties.

Some studies that perform preference elicitation for personal recommendation employ active learning that assists to depict what preferences to elicit. The decision is based on the goal of the recommender system: accuracy of the recommendation, system profit, user satisfaction, and the user's ability to rate the required items (Rubens et al. 2011). Our focus is on *minimizing* preference elicitation with queries geared towards finding approximate or necessary winner items for a group of users. Thus, employing active learning might increase the accuracy of our user profiles, but opposes our goal of minimizing the preference elicitation process.

Content based recommender systems make use of the attributes of the items. For example, in a recent study by Freyne et al. (Freyne et al. 2013), an active learning algorithm is suggested, that learns the bias users have towards attributes of an item. These are reflected in the users rating patterns; thus, the algorithm can estimate the rating users will give to certain items, and use this information when deciding which preferences to elicit. One major drawback is that the authors limit their analysis to food recommendation and focus exclusively on meal recipes. We propose a domain-free model that does not require any knowledge about the items' features.

To conclude, the studies surveyed in this section are relevant to systems in which a single voter exists, although we remain interested in preference management for a group of users. While we share a few common ideas, such as the guideline of eliciting items that are optimal to the user, most earlier research assume that item attributes are readily available, an assumption that we do not hold. More importantly, we did not find studies that share the same goal of minimizing the preference elicitation process. In the next section we discuss preference elicitation that is specific to groups.

### 2.1.2 Group Recommender Systems

In this section we first introduce the features of group recommender systems in general, and illustrate the unique features of our model within each category. We then discuss preference elicitation for group recommender systems and lastly present preference estimation.

#### 2.1.2.1 Features of Group Recommender Systems

Group recommendation systems output recommendations to a group of users rather than to one user. A classification of the state-of-the-art group recommendation systems can be found in the paper of Garcia et al. (Garcia et al. 2011). The systems are classified according to the following six independent features that influence their design:

1. Information source – our proposed model is based on collaborative filtering techniques. We assume that no knowledge is available about the users (e.g., demographic knowledge is unavailable) or about the items (e.g., content-based attributes are unavailable).
2. User-system interaction - our proposed model falls under what the author's term a *passive user-system interaction*. In a passive user-system, the final goal is to provide items to the group without further interaction with the users or the system.
3. Domain – most existing systems are domain specific. Our model offers a domain-free platform.
4. Outcome – the systems are classified according to the recommendation outcome: a single recommendation or a ranked list of items. Our model does not fit either category. We propose to go one step further and offer necessary or approximate winner items, which defiantly fit the group.
5. Group size – the systems are classified according to group size -- any group size or small groups of 2-4 users. Our system does not fall under either category. Our model is designed for groups of up to 30 users, and up to 30 items.
6. Aggregation approach – the systems can either compute a group recommendation based on: (a) an aggregation of the recommendations to personal user profiles, or (b) build a group profile that is treated as a single user to which a recommendation is given. In our model we aggregate *user preferences*, not *user recommendations*.

Table 2 presents a summary of our system in comparison with the group recommender categories mentioned above. The grey cells in the table indicate the places where our model does not fit the available categories.

Table 2: Our model according to the group recommender systems categories proposed by Garcia et.al. (2011)

| | **Available categories** | **Our model** |
|---|---|---|
| **Information source** | Content based, collaborative filtering, knowledge based, hybrid approaches | Collaborative filtering |
| **User-system interaction** | Passive members, active members | Passive members |
| **Domain** | Domain specific, generalist | Generalist (domain-free) |
| **Outcome** | Single recommendation, ranked list of recommendations | Definite or approximate winner items |
| **Group size** | Any group size, small groups | Medium size |
| **Aggregation approach** | Aggregation of the recommendations to personal user profiles, recommend to a group profile | Aggregation of user preferences |

In a study by Garcia et al. (2011) the authors propose a domain-free group recommendation system. The user and item profiles are built on the data available. However, their approach assumes that all the user opinions are *known* and they focus on investigating the best way to aggregate all preferences. Conversely, we assume that the user preferences are *unknown* and need to be elicited. In our view, preference elicitation is another category that can be used to classify group recommender systems. This category is not mentioned in (Garcia et al. 2011).

### 2.1.2.2 Preference Elicitation in Group Recommender Systems

A growing body of literature is investigating preference elicitation for group recommenders. In the critique model approach (Chen and Pu2012), case-based reasoning is applied in order to elicit voter preferences on items. Such systems require analysis and maintenance of item features which is not always feasible. In this study the critique model is irrelevant since we consider dataset where item features are unknown. Braziunas and Boutilier provide (Braziunas and Boutilier2009) several utility elicitation techniques from research fields such as artificial intelligence, operations research and conjoint analysis. The common base of the techniques is that users are requested to provide unambiguous simple answers to queries. In this study, we follow these researchers' method and focus on explicit preference elicitation where the users' answers are unambiguous. Our model does not allow flexibility in stating the preferences; the user is required to submit preferences from a predefined discrete scale of values, or to choose between two items. Rodriguez et al. (Rodríguez et al. 2013) present an algorithm for eliciting complex linguistic expressions, as part of a group decision-making process. The focus of their study is in allowing flexibility in the expression of the preferences. Critique-based reasoning and linguistic expressions may be a more natural way to interact with users. However, this approach requires the users to spend an ample amount of time on rating their preferences and the system does not enable the modeling of users' preferences unambiguous on a numbered scale.

### 2.1.2.3 Preference Estimation

Instead of eliciting preferences, some studies estimate the unknown user preferences using fuzzy preference relations based on the additive consistency measure (Chen et al. 2014; Herrera-Viedma et al. 2007). Another approach estimates the voter preferences using Bayesian Networks and computes an estimated recommended item (de Campos et al. 2009). Our model does not attempt to estimate the unknown preferences but rather computes nonparametric probability distributions for the unknown preferences.

A probabilistic algorithm that accounts for uncertainty in a single voter's preferences was developed by Yu et al. (Yu et al. 2004). While their model assumes a normal distribution of voting preferences, we do not make such an assumption. Instead, we use a non-parametric voting distribution. Popescu and Pu present, a probabilistic music playlist group recommendation system (Popescu and Pu2013). The probability distributions are defined as the probability for each item to be the winner item. However, estimations for missing items are not considered.

Koren and Sill (2011) developed a framework for finding probability distributions that is calculated according to the predicted rating. We, on the other hand, show how to extract the probability distribution without calculating the predicted rating. We then use this distribution to select the next query during the preference elicitation process. The system is constantly updated with new information, so the probability distributions' accuracy increases as more preferences are available.

To conclude, Table 3 illustrates two categories we propose to add to the existing categories in Garcia et al (2011) for classifying group recommender systems:

1. Statement of preferences – the users can state their preferences as: (a) linguistic expressions, (b) using a critique based model or (c) as unambiguous preferences. Our model focuses on unambiguous preferences.
2. Estimation of missing preferences - the missing preferences can either be estimated using a fuzzy or Bayesian model (point estimation), or a probability distribution can be created for each missing preference. Our model creates a non-parametric probability distribution for each missing preference.

Table 3: Proposal of additional classification categories

| | **Existing** | **Our model** |
|---|---|---|
| **Preference elicitation** | None<br>Unambiguous<br>Critique based<br>Linguistic expressions | Unambiguous |
| **Estimation of missing preferences** | Point estimation<br>Probability distributions | Probability distributions |

## 2.2 Preference Elicitation Using Voting Theory

Voting protocols determine how user preferences are treated (Rossi et al. 2011). In social choice (Suzumura, Arrow, and Sen 2010), preference elicitation is termed as voting. The theoretical basis for addressing voting with partial information (i.e., where users do not set the preferences for all items), can be found in (Conitzer and Sandholm2005; Konczak and Lang2005). Conitzer and Sandholm (2005) analyze the communication complexity of various voting protocols and determine upper and lower bounds for communication. In general, they show that for most voting protocols, in the worst case voters should send their entire set of preferences. Konczak and Lang (2005) demonstrate how to compute the sets of possible winners and a set of necessary winners. These sets determine which items no longer have a chance of winning and which will certainly win. We adopt their approach to propose a systematic preference aggregation protocol where the agents do not need to send their entire set of preferences.

Theoretical bounds for the computation of necessary winners have been previously addressed (Walsh2007; Betzler et al. 2009; Pini et al. 2009). Others considered settings where preferences may be unspecified, focusing on soft constraint problems (Gelain et al. 2007) or on sequential majority voting (Lang et al. 2007). They do not provide empirical evaluation nor do they focus on minimizing the preference elicitation process.

In a recent paper, the subject of unavailable candidates is addressed (Boutilier et al. 2014). In our research we assume that all candidate items are available. We do offer a model for returning top-$k$ items where one of them is the winner item, which can be used when some of the items might not be available. The hardness of winner determination under various multi-winner voting rules was discussed in (Procaccia et al. 2008; Skowron et al. 2013; Betzler et al. 2014). The authors study systems were a proportional representation of users' wishes is required. Multi-winner voting rules are discussed in a recent work (Elkind et al. 2014). The authors address some multi-winner voting rules and discuss their properties under different settings. The setting relevant to our study is defined by the authors as "Shortlisting": situations where $k$ out of $N$ items are shortlisted as the most appropriate. However vote elicitation is not discussed by the researchers.

Predefined probability distribution of the votes is assumed by Hazon et al. (Hazon et al. 2008). The winning probability of each candidate is evaluated in Plurality, Borda, and Copeland protocols. They show theoretical bounds for the ability to calculate the probability of an outcome. Bachrach et al. (Bachrach et al. 2010) provide an algorithm for computing the probability of a candidate to win, assuming a polynomial time computable voting rule (such as Range voting) and assuming a uniform random distribution of voters' choice of candidates. However, while both of these papers focus on calculating the winning probability for each candidate, we focus on *practical vote elicitation,* specifically on finding the winner using a minimal amount of queries.

As opposed to our goal of minimizing the number of queries in general, Nisgav and Patt-Shamir (Nisgav and Patt-Shamir2011) propose theoretical bounds for minimizing the number of queries sent to each voter. Their goal is not to find a winner but to retrieve enough information in order to fill a voter's preference vector (defined as the voter's preferences for different items), so that the information can be used for collaborative filtering.

#### 2.2.1.1 Practical Preference Elicitation

Practical vote elicitation has been addressed recently. Pfeiffer et al. (Pfeiffer et al. 2012) the goal is to predict the ranking of $n$ items, by querying voters using pairwise comparison of items. However, the authors do not explicitly aim to reduce the number of queries. Furthermore, they assume each voter can be approached only once and that there is no prior knowledge on the voters. As a result, voter-item distributions cannot be computed. Their method is therefore suitable when a large amount of voters is available and the task is to determine some hidden truth (also known as the wisdom of the crowds). We, on the other hand, wish to reach a joint decision for a specific group of voters. In Ding and Lin (Ding and Lin2013), a candidate winning set is defined as the set of queries needed in order to determine whether the candidate is a necessary winner. The authors show that for rules other than the plurality voting, computing this set is NP-Hard. This theorem further supports our claim that heuristics are needed for preference elicitation.

An attempt to reduce the number of queries is made by Kalech et al. (Kalech et al. 2011). The authors assume that each user holds a predefined decreasing order of the preferences. In an iterative process, the voters are requested to submit their highest preferences; the request is for the rating of a single item from all the users. One major disadvantage of this approach is that requiring the users to predefine their preferences can be inconvenient to the users. While these authors do not consider the probability distribution of the voters, our work illustrates how the probability distribution of the voters can be used to decrease the number of queries. Lu and Boutilier (2011) offer yet another practical elicitation process is proposed for the Borda voting protocol using the minmax regret concept. The output is a definite winner or an approximate winner, but the authors do not state the approximation confidence level. The authors recently extended their method to return multiple winners, again using the Borda protocol and minmax regret (Lu and Boutilier2013).

## 2.3 Aggregation Strategies

In the previous sections we surveyed preference elicitation. In this section we survey research on aggregating the user preferences.

In his well-known work, Arrow (Arrow 1951) shows that there is no perfect aggregation system. A summary of 11 different aggregation strategies can be found in a paper by Masthoff (Masthoff 2011) and their classification into three main categories can be found in a research be Senot et al. (Senot et al. 2011). A new strategy based on the Nash equilibrium is proposed by Carvalho et al. (Carvalho et al. 2013). Both Masthoff and Senot et al. study how different strategies affect group members. However, the aggregation strategies have not been studied in the context of *preference elicitation*.

Masthoff (Masthoff2004) studies how humans prefer to integrate personal recommendations. She concludes that users use the Majority based strategy, the Least Misery based strategy and Majority without Misery strategy. The Majority based Strategy and the Least Misery based strategy were also chosen by Baltrunal et al. (Baltrunas et al. 2010), in a study focusing on evaluation of the effectiveness of GRS obtained by aggregating user preferences. In (Garcia et al. 2011) the authors note that the Least Misery and the Majority strategies are the most common strategies used which motivated us to focus our research on these two strategies.

In the *Majority Based Strategy* the users' ratings of the different items are aggregated; the item with the highest total value is the winner. Note that the result is similar to taking the item with the highest average, thus this strategy is sometimes referred to as the Average Strategy or the Additive Strategy. The Majority Based strategy is used in numerous applications including the MusicFX system when the square of the individual preferences are summed (McCarthy and Anagnost1998) and the Travel Decision Forum that assists in planning a holiday (Jameson2004). Berkovsky and Freyne compare weighted and un-weighted additive strategies when recommending a recipe to a group (Berkovsky and Freyne2010). Another example is of TV programs recommendation for a group (Yu et al. 2006; Masthoff2004). A disadvantage of this strategy is that it can be unfair towards users with the minority view. In fact, Yu et al. (Yu et al. 2006) state that their system works well for a homogenous group but when the group is heterogeneous, dissatisfaction of the minority group occurs. Endriss and Grandi (Endriss and Grandi2013) use the Majority based strategy is used for a different purpose: to find the most representative user. The authors assume the votes are binary. The most representative user's opinion is presented to the group. Contrary, in our study the Majority based strategy is performed on the items and not on the users.

The *Least Misery Strategy* defines that the chosen item cannot be the least preferred by any of the users. In the Polylens system the Least Misery strategy is used to recommend movies to small groups (O'connor et al. 2002). Their survey shows that 77% of the users found the group recommendation more helpful than the personal one. The disadvantage is that the minority opinion can dictate the group – if all users but one really want some item to win, this item will not be chosen (Masthoff2011). Another system with a Least Misery approach is the CATS system, a collaborative based travel recommendation system (McCarthy et al. 2006). A model for group recommendations for households sharing a movie rental account is proposed in Gorla et al. (Gorla et al. 2013). They offer a novel probabilistic framework for combining the relevance of items to users with combining the relevance of the items to the group as a whole. The aggregation strategy they choose to use is the Least Misery strategy.

To conclude, the Majority and the Least Misery strategies are commonly used in the literature. One of the contributions of this study is considering the Least Misery strategy, which, to our best knowledge, has not been studied in the context of *preference elicitation*.

## 2.4 Drawbacks of previous research

In the previous sections we surveyed preference elicitation and preference aggregation in recommender systems and in social choice; we have shown where our research relates and where it differs from these domains. Table 4 offers a summary comparison of the differences between the two fields. The greyed cells indicate the way our proposed model operates:

1. Set of items - in group recommender systems the recommendation is usually based on the available user preferences. If preference elicitation is performed, it is usually done on a set of other items (i.e. on the voters rating history and not on the set of items in question). In our model the preference elicitation is performed on the available items.
2. Output - the items presented to the group are recommendations, while in social choice the items are necessary winners. In our model the outputs are necessary winners.
3. Termination condition - the focus of studies in group recommender systems is usually on the recommendations accuracy, recommendations fairness or on the user satisfaction. Therefore preference elicitation terminates once the system reaches an

adequate level of one of these conditions. In social choice, preference elicitation terminates once a necessary winner is found. In this study we expand the termination condition and terminate the process when an approximate winner item is found within a list of top-$k$ items. We have not encountered any study that approximates a winner or $k$ alternative winner with a confidence level.

4. Aggregation strategy - we introduce algorithms which can use the Majority based or the Least Misery strategy in order to output one or top-$k$ definite winner items or approximate winner items within some confidence level. Current research in social choice is limited to the Majority based strategy for preference aggregation. To the best of our knowledge the issue of preference elicitation and returning one or more items under the Least Misery strategy has not yet been investigated.

We have not seen any attempt of research in the field of social choice to connect to the current research in group recommender systems. Our model can be used as a second step in existing group recommender systems. First, the group recommender system computes the list of top-$N$ recommended items.Then our model can be used in order to output approximate or definite winner items to the group.

While preference elicitation and preference aggregation is addressed in the recommendation literature, the issue of minimizing the preference elicitation is ignored. To the best of our knowledge, in the social choice field, only the two attempts (Lu and Boutilier 2011 ; Kalech et al. 2011) have been made at vote elicitation in order to try to minimize the amount of queries. The advantage of our approaches is that users are not required to predefine their preference as in (Kalech et al. 2011) and are not necessarily required to hold a strict set of ordered untied preferences as in ( Lu and Boutilier 2011).

We propose a general, domain-free framework for preference management. The model estimates probability distributions for the unknown preferences. We introduce novel heuristics and show how they can operate under ranking and non-ranking voting protocols. The heuristics determine which preferences to elicit, in an incremental process. One heuristic computes the information gain of each potential query based on the entropy of the item's probability to win. The rationale behind this proposal is that reducing the entropy as fast as possible will direct us towards the winner as fast as possible. The other heuristic uses the probability distribution of the voters' preferences to select the candidate most likely to win and the voter that is expected to maximize the score of that item. The rationale behind this is that maximizing the item with the current maximum score will lead us towards an item whose minimum score is higher than the maximum of all others (Konczak and Lang2005) and in turn, towards the winner item. We present our model in detail in the following chapter.

Table 4: Preference elicitation in recommender systems and social choice

| | Recommender systems | Social choice |
|---|---|---|
| **Set of items on which preference elicitation is conducted** | *other items* | *available items* |
| **Output (item or list of items) to the group of users** | *predicted/recommended* | *definite* item/s according to some voting rule |
| **Termination condition** | Once the output is viewed as adequate: *accurate*, *fair*, or when the users are *satisfied* | *Necessary winners* with some confidence |
| **Aggregation strategy** | Any aggregation strategy, typically *Majority or Least Misery* | *Majority* |

# Chapter 3

# Problem Formulation

We introduce a general model for reaching joint decisions with minimal elicitation of voter preferences. Our approach has two components: a rating distribution estimation component and a voting center component. The rating distribution estimation component (referred to hereafter as the distribution component) computes and holds a probabilistic voter preference distribution model, which is specific for every voting protocol. It is presented for Range Voting in section 4.2 and for Borda Voting in section 5.2. The voting center component is responsible for collecting voter preferences and returning a recommendation, given predefined termination conditions. A simple termination condition is, for example, the discovery of one definite winner item, which is recommended to the group as their best option. More termination conditions are discussed in section 6.2.

The model is illustrated in Figure 1 and proceeds as follows: The distribution component holds a database of historical item ratings given by voters and uses them to estimate nonparametric probability distributions for all unknown ratings. The voting center component receives the probability distributions and utilizes them for selecting whom to query about which items. The heuristic generates a query that is sent to the appropriate voter, and if the response causes the termination condition to be reached, the voter center ends the process. If not, the voter's response is sent to the distribution center, which updates the probability distributions, and the process is repeated. The goal is to meet the termination condition using a minimal number of queries, thus incurring minimal cost.

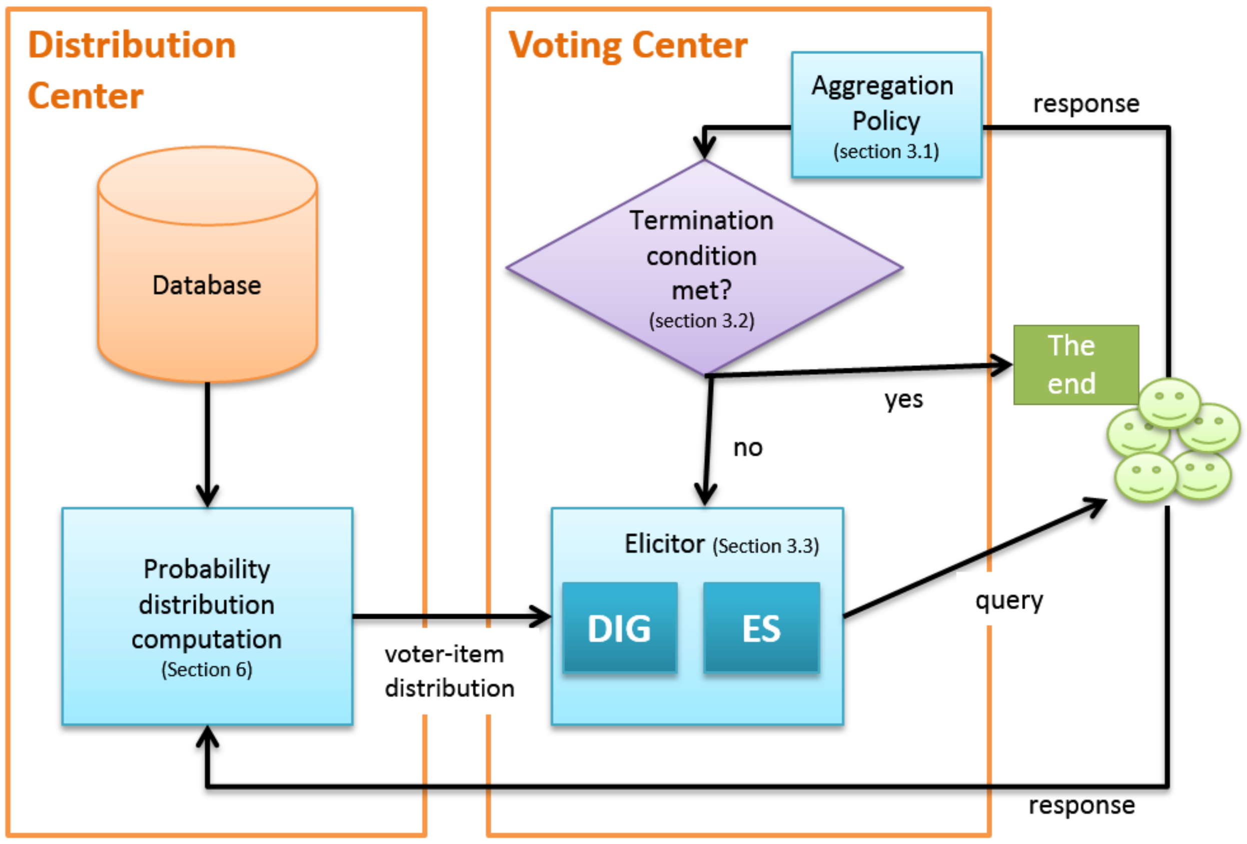


Figure 1: Model description

## 3.1 The Minimal Cost Problem

Let us define a set of users (voters) as $V = \{v_1, v_2, \dots, v_m\}$ and a set of candidate items as $C = \{c_1, c_2, \dots, c_n\}$. We define a request for specific information from the voter as a query $q$. The type of query depends on the voting protocol employed. Under the Range voting protocol, a query is a request for the rating of one voter $v_i$ for one item $c_j$. This is a *voter-item* query, denoted $q_j^i$. The rating is determined by the user from a set of ordered values, such as 1,…,5. Under the Borda voting protocol, a query is a request for the voter's preference between two items $c_j$ and $c_k$. This is a *voter-item-item* query, denoted $q_{j,k}^i$.

A query has a cost, e.g., the cost of communicating with the voter, or the cost of interfering the voter's regular activities. We assume that the cost is equal for all queries.

*Definition 1.(Cost): Given a query q, the cost function* $cost: q \rightarrow \mathbb{R}$ *returns the communication cost of the query.*

Throughout this research we assume that the cost is equal for all queries and that the cost is constant throughout the elicitation process. It is possible to determine the winner from the partial voters' ratings (Konczak and Lang2005; Walsh2007). We adopt an iterative method (Kalech et al. 2011; Lu and Boutilier 2011) which proceeds in rounds. On each round one voter is queried for her rating for one item. Consequently, we aim to determine the next query, such that the total expected cost is minimized.

We assume that a user always responds to a query and that the response is sincere. Let $O^i$ represent the set of voter $v_i$'s responses to the queries. Note that this set does not necessarily contain all the items. $\mathcal{O}^A = \{O^1, \dots, O^m\}$ is a set of $O^i$ sets. At the end of each round, one response to a query is added to $\mathcal{O}^A$. Figure 2 illustrates one query execution round according to the: (a) Range protocol and (b) the Borda protocol. For the Range protocol, a request to rate an item is presented to the user. The user rates the requested item out of a predefined discrete domain of values. For the Borda protocol, the user is requested to decide between two items. In both protocols, the user's response is added to $O^i$ and to the set of known preferences $\mathcal{O}^A$.

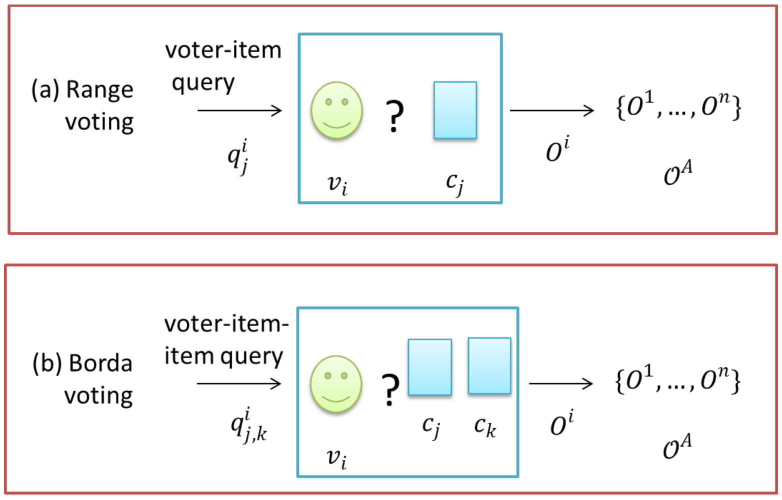


Figure 2: One query execution round

The process continues iteratively, in rounds, until the *termination condition* is met. Consider for example the termination condition: one *necessary winner* item, i.e., the process should terminate when one item is discovered, to be presented to the group as their best option. To determine the necessary winner we compute the *possible maximum* and *possible minimum* of each candidate item. The possible maximum of an item represents the possible highest score for that item based on the known preferences in $O^A$, namely, by completing the unknown preferences with the highest score. Respectively, the possible minimum of an item represents the lowest score possible for that item based on the known preferences in $O^A$, namely, by completing the unknown preferences with the lowest score. If item $c_j's$ possible minimum is bigger than the possible maximum of all other items then $c_j$ is a *necessary winner* (Konczak and Lang 2005).

The goal is to guarantee the requested termination condition with minimal cost. For example, for the termination condition of one necessary winner item, the goal is to execute a minimal number of queries in order to find an item which is a necessary winner.

### 3.1.1 MDP Formulation

The challenge stated at the end of the former paragraph challenge can be represented as a Markovian Decision Process (MDP). An MDP is a tuple of states, actions, a transition function from state to state by an action and a reward function. In our case, the states are the possible combinations of the users' ratings for the items. Every user can assign $|D|$ possible values to item $c_j$. If a user has not yet assigned any value, the current value of the item is unknown. Thus, the combination space is $(|D|+1)^{n \cdot m}$ , where $m$ is the number of users and $n$ is the number of items. The actions are the possible queries. A query is a request to a specific user to either rate an item (in the Range voting protocol) or to state a preference between two items (in the Borda voting protocol). Thus, the queries space is $n \cdot m$ or ${n \cdot m}/_2$ respectively. The transition function between two states is affected by the probability distribution of the ratings of the item about which the user was queried. Finally, the reward is the negative cost of the queries. The goal is to determine which query to choose on each round so that the communication with the user is at a minimum.

We can compute the optimal query's vector by finding the optimal policy by applying dynamic programming methods such as Value Iteration or Policy Iteration (Bellman1962). These methods grow polynomially in the number of states and actions. However, in our case the state space itself is exponential in the number of voters and items and dynamic programming is not suitable for such large settings. Thus, we present heuristic approaches that use greedy moves to compute the next query.

The heuristics and the computation of the possible minimum and possible maximum score are specific for each voting protocol. The heuristics for the Range voting protocol are presented chapter 4 and for the Borda voting protocol in Chapter 5. In the next section we introduce the two protocols.

## 3.2 Voting Protocols

In this study we focus on two representative voting protocols; the Range protocol represents the class of non-ranking rules, and the Borda protocol represents the class of ranked based rules (Rossi et al. 2011). The Range voting protocol requires users to assign scores from a predefined range. The Borda voting protocol requires users to rank their preferences so that no two items can receive the same score. Pairwise comparison queries can be related to the Borda voting protocol, since according to the Borda protocol users are required to have a fixed ranked list of preferences for items. As each protocol has its advantages and disadvantages, the final decision is in the hands of the system administrator.

The Range voting protocol has a few advantages. First, perhaps the main advantage of the Range voting protocol is that it is relevant to many already existing applications, where voters are asked to rate items on a specified scale and users are familiar with the requests for ratings. For example, on the Netflix website (www.netflix.com) users are asked to rate a movie on a scale of 1-5. Amazon is another example (www.amazon.com). Secondly, in the worst case, Range voting requires only $n$ queries to rate the items, while pairwise comparisons (using the Borda protocol) usually require more queries: in the worst case it has been shown that O(nlogn) pairwise comparisons are required in order to restore arbitrary preferences of one user over $n$ items (Conitzer2009). Lastly, the Range voting protocol satisfies the Independence of Irrelevant Alternatives (IAA) criterion, meaning that if candidate item $c_i$ is preferred over $c_j$, then by changing the preference of a third candidate $c_k$, $c_j$ must not be preferred over $c_i$ (Arrow 1951; Smith2001). This is important in iterative voting since the users are not required to pre-define their preferences. The IAA criterion is not satisfied under the Borda protocol.

The Borda voting protocol does offer some advantages. First, studies have shown that it is easier for users to state opinions when the queries are pairwise (Balakrishnan and Chopra2012). Consider, for example an application trying to find a sushi type most preferred by a group (Kamishima et al. 2005). A user might find it easier to answer a question such as:

"Which of these two sushi types do you prefer?" as opposed to "On a scale of 1 to 5, how would you rate this sushi?" Secondly, users are more accurate when making relative indirect judgments than when they directly rank items using a linear scale (Stillwell et al. 1982). Lastly, the Borda protocol is computationally hard to manipulate (Betzler et al. 2011; Davies et al. 2011).

Our research can be extended to other voting protocols such as the non-ranking protocols: Approval, Cumulative, and the ranking protocols: Copeland and Maximin. The differences between the protocols are in the way the possible minimum and possible maximum are calculated, as that is protocol specific (Kalech et al. 2011). The rest of the framework is affected by whether the protocol is a ranking or a non-ranking protocol. To summarize, the iterative voting framework is affected by the voting protocol in several ways:

a) The definitions of the possible maximum and possible minimum (sections 4.1 and 5.1).
b) The model of the distribution of preferences – for the Range voting protocol a model of voter-item distribution is built. For the Borda voting protocol a model of ranked item preferences distribution is built (sections 4.2 for Range and 5.2 for Borda).
c) The computation of item winning probability. For the Range protocol we propose a dynamic programming algorithm (section 4.3). For the Borda protocol we propose a Monte Carlo sampling approach (section 5.3).
d) The query selection heuristics (sections 4.4 and 4.5 and 5.4 and 5.5)

## 3.3 The Evaluation Procedure

The evaluation procedure is identical throughout the study for the different protocols. In this section we describe the evaluation metrics and the datasets used in the experimental sections in chapters 4, 5, and 6.

### 3.3.1 Metrics

We evaluate the heuristics in terms of:

(1) Communication cost – lower communication costs are desirable. We used two methods to compute the communication cost:

   i. Number of queries – the number of queries required for finding the necessary winner

ii. Percentage of the dataset queried – the upper bound to the queries amount is the amount of queries a naïve voting center would have asked $(n \times m)$. The percentage of the dataset queried is: $1 - \frac{(number\ of\ queries)}{n \times m}$

(2) Runtime

(3) Probability the winner is in the top-$k$ – this metric is used to measure the accuracy of the output, when the termination condition is that the winner is within the top-$k$ with some confidence level. A higher probability is more desirable.

The voting distributions were examined under different settings:

(1) Sensitivity – the communication cost under different voting distributions. We show that the voting distribution within a dataset affects the communication cost.

(2) Updates – the effect of iterative updates to the voting distributions. We argue that updating the voting distributions as more data is revealed leads to a decrease in the communication cost.

(3) Size – the effect of generating the voting distributions from larger or smaller datasets.

### 3.3.2 Statistical Test

In order to conclude which algorithm performs best over multiple datasets, we follow a robust non-parametric procedure proposed by García et al. (García et al. 2010): we first used the Friedman Aligned Ranks test in order to reject the null hypothesis that all heuristics perform the same; this test was followed by the Bonferroni-Dunn test to find whether one of the heuristics perform significantly better than other heuristics.

### 3.3.3 Datasets

This evaluation was performed in four domains. The first is a simulated meeting scenario where voters are required to vote for their preferred time slot for a meeting. Simulating the data allows us to investigate different distribution settings. The second domain is the Netflix prize dataset[4], a real world dataset containing the ratings that voters assigned to the movies. The third domain is the Sushi dataset (Kamishima et al. 2005). The fourth domain is taken from a user-study on a new recommendation system: “Lets Do It”. The last three domains allow us to examine performance in real world scenarios. The second and third domains were recently added to the new PrefLib library (Mattei and Walsh 2013). Specific adjustments of the datasets to the protocols, when required, are detailed in the evaluation sections of chapters 4-6.

In all domains we explore scenarios that include a group of up to 30 users that are required to choose from up to 30 items. We assume these 30 items are the top ranked items returned by a group recommender system. These group sizes are rational for real world scenarios. It is uncommon that groups of hundreds of people wish to receive a recommendation for a joint activity. In the case where there are many available items (for example, 16,000 possible movies), a group recommendation system can be used to narrow down the options to a magnitude of 10-30 top items. Our proposed model then operates on these items, and requests users for their votes for these top items. It is unrealistic to request users to vote for their preferences between thousands of items.

#### 3.3.3.1 Simulated Datasets

We simulated the scenario of a group of 5-30 users who wish to schedule a joint meeting on one out of four available timeslots (items). We assume that when a user is queried regarding her preference for a specific timeslot, she replies with a score on a 1-4 scale.

To examine the algorithm's sensitivity to different rating distributions, we manipulated the voter-item distribution settings so that the voter-item rating distribution is *skewed* in different ways. *Skewness* is a measure of the asymmetry of a distribution. A higher absolute skewness level indicates a higher asymmetry. A negative skew indicates that the distribution is concentrated on high vote values, while a positive skew indicates the distribution is concentrated on low vote values. We manually controlled the skewness level by creating rating distributions with different skewness levels (see Table 5). In our experiments, we manually controlled the skewness level of one of the items and set the skewness of the other items in one of the following ways:

[4] Netflix prize: http://www.netflixprize.com

(1) UNIFORM – all items except one receive a uniform skew (skew “0” in Table 5).

(2) LOTTERY – all items except one receive skewness out of the available options in Table 5. The skewness option is determined in a lottery.

Having set a rating distribution for every voter-item pair, we cast lots to set the voter-item rating based on the distribution of the voter-item. To account for the randomness, each experiment was repeated 20 times.

Table 5: Skewness levels

| Skewness level | $d_1 = 1$ | $d_2 = 2$ | $d_3 = 3$ | $d_4 = 4$ |
|---|---|---|---|---|
| -6 | 0.011 | 0.011 | 0.147 | 0.832 |
| -5 | 0.014 | 0.014 | 0.193 | 0.778 |
| -4 | 0.018 | 0.018 | 0.263 | 0.7 |
| -3 | 0.039 | 0.084 | 0.243 | 0.634 |
| -2 | 0.053 | 0.165 | 0.225 | 0.557 |
| -1 | 0.183 | 0.183 | 0.183 | 0.45 |
| 0 | 0.25 | 0.25 | 0.25 | 0.25 |
| 1 | 0.45 | 0.183 | 0.183 | 0.183 |
| 2 | 0.557 | 0.225 | 0.165 | 0.053 |
| 3 | 0.634 | 0.243 | 0.084 | 0.039 |
| 4 | 0.7 | 0.263 | 0.018 | 0.018 |
| 5 | 0.778 | 0.193 | 0.014 | 0.014 |
| 6 | 0.832 | 0.147 | 0.011 | 0.011 |

#### 3.3.3.2 The Netflix Dataset

We examined a scenario of a group of friends wishing to watch a movie together. We assume that there exists an incomplete history of previous movie ratings, i.e., some of the friends and/or some other users have rated some of the movies in question and/or other movies.

To explore this scenario we used the real world Netflix prize dataset. The original dataset contains over 17,000 movie items and over 400,000 voters. Understandably, the dataset is sparse. In order to evaluate our heuristics, we require a dataset where all ratings are known, so that we can simulate the worst case scenario, where every user is queried about every movie before a decision is reached. To evaluate our algorithms, we used a subset of the Netflix dataset containing 1000 voters and 1000 movies $U \times I$. This subset is relatively dense, with 75% sparsity. This subset was used for estimating the rating distribution. We further found a subset of 111 voters with over 116 items, which is completely full, i.e., all ratings are known for all items and voters. We created smaller non-overlapping test sets $V \times C$ in varied sizes of $10\times 10$ up to $30 \times 30$. All of these matrices are sub-matrices of the $1000 \times 1000$ subset: $V \times C \subset U \times I$. In these matrices, all of the voter-item ratings are known. However, the algorithms start with no knowledge of these ratings. The amount of $V \times C$ matrices on which the experiments are run is described in Table 6.

Table 6: File amount for each experiment size

| $V \times C$ | Amount |
|---|---|
| $10 \times 10$ | 10 |
| $15 \times 15$ | 7 |
| $20 \times 20$ | 5 |
| $25 \times 25$ | 4 |
| $30 \times 30$ | 6 |

#### 3.3.3.3 The Sushi Dataset

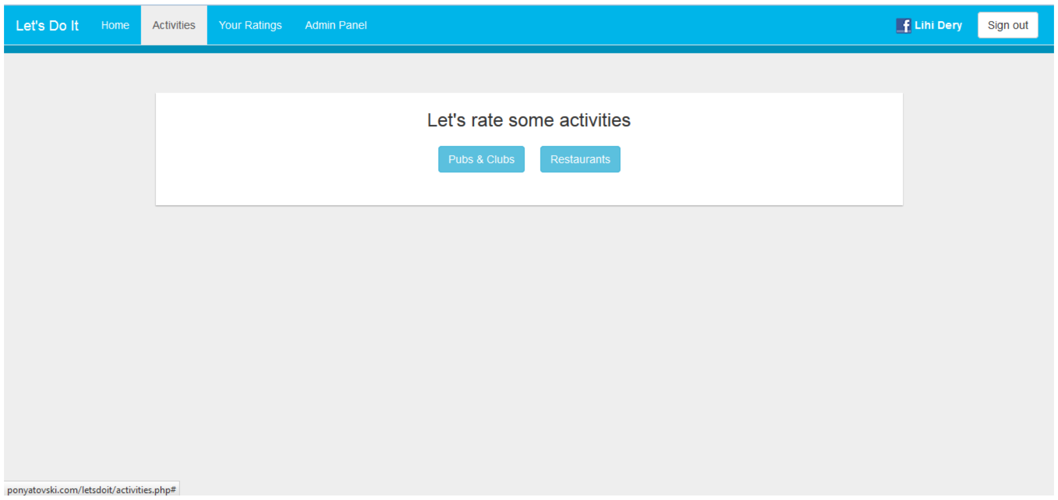
We examined a scenario of users who are required to decide between ten types of sushi. The Sushi dataset (Kamishima et al. 2005) contains 5000 preference rankings over 10 kinds of sushi. We derived 10 different random matrices of size $10x6$. In order to create an initial permutation probability distribution, we aggregated the number of appearances of each permutation in the training set and divided it by the total number of voters. Thus the initial permutation distribution is equal for all voters. As more queries are answered, the distributions are updated for each voter. Over time a unique permutation distribution pattern emerges for each user.

#### 3.3.3.4 User Study

We created our own set of real data and examined two scenarios of a group that wishes to: (a) select a restaurant or (b) select a pub or club. The data was collected using a group recommendation system, named “Lets Do It”[5].

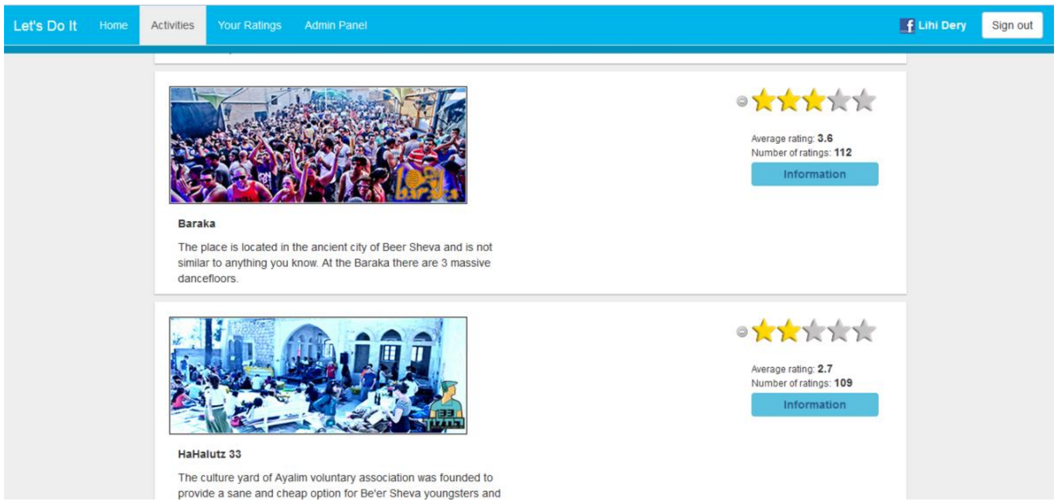
The system obtained a full set of ratings from 90 students in Ben Gurion University, for two different domains: (a) restaurants (16 items) and (b) pubs and clubs (23 items). Figure 3 presents the opening screen. The students were instructed to rate each item on a 1 to 5 scale, according to their satisfaction from past visits, or in case they were unfamiliar with a place , according to how appealing it was for them to visit it. Each item had a picture and a short description, as shown in Figure 4. The students could view the items they rated, the items left for them to rate. They could also change the ratings. This is demonstrated in Figure 5.  Rating distributions were derived in the same manner as for the Netflix dataset (section 3.3.3).

[5] The credit for building the system goes to Eli Ponyatovski and Aviad Carmeli, 4th year students in the Information Systems Department 2014 at Ben Gurion University, under the supervision of: Lihi Dery, Ofrit Lesser and Meir Kalech. The recommendation system is designated to study group recommendation with social networks (the study is in an initial phase).



Let's rate some activities

Figure 3: The student rate pubs&clubs and restaurants



Average rating: 3.6
Number of ratings: 112

Baraka

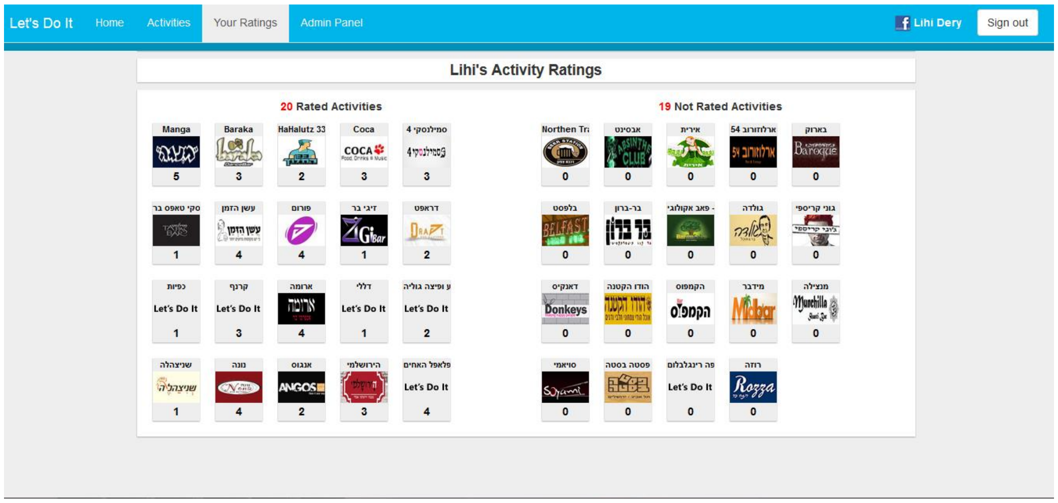
The place is located in the ancient city of Beer Sheva and is not similar to anything you know. At the Baraka there are 3 massive dancefloors.

Average rating: 2.7
Number of ratings: 109

HaHalutz 33

The culture yard of Ayalim voluntary association was founded to provide a sane and cheap option for Be'er Sheva youngsters and

Figure 4: Rating for two clubs



Lihi's Activity Ratings

20 Rated Activities

| | | | | |
|---|---|---|---|---|
| Manga 5 | Baraka 3 | HaHalutz 33 2 | Coca 3 | סמילנסקי 4 3 |
| סקי טאפס בר 1 | עשן הזמן 4 | פורום 4 | זיגי בר 1 | דראפט 2 |
| כפיות 1 | קרנף 3 | ארומה 4 | דללי 1 | ע ופיצה גוליה 2 |
| שניצהלה 1 | נונה 4 | אנגוס 2 | הירושלמי 3 | פלאפל האחים 4 |

19 Not Rated Activities

| | | | | |
|---|---|---|---|---|
| Northen Tr 0 | אבסינט 0 | אירית 0 | ארלוזורוב 54 0 | בארוק 0 |
| בלפסט 0 | בר-ברון 0 | - פאב אקולוגי 0 | גולדה 0 | גוני קריספי 0 |
| דאנקיס 0 | הודו הקטנה 0 | הקמפוס 0 | מידבר 0 | מנצילה 0 |
| סויאמי 0 | פסטה בסטה 0 | פה רינגלבלום 0 | רוזה 0 | |

Figure 5: The student can see what places need to be rated

# Chapter 4

# Preference Elicitation Using the Range Voting Protocol

In this chapter, we propose heuristics for query selection for determining a necessary winner using the Range voting protocol (Smith2001). As explained in section 3.2 the Range voting protocol is widely used in existing applications. We address Range voting with incomplete information. At the beginning of the process the voter-item preferences are unknown. When voter-item pairs are queried, their ratings are revealed. The algorithms we propose for vote elicitation are iterative. In each round, the algorithm selects one voter-item pair so that the rating of one voter for one item is revealed. The algorithm continues until a *necessary winner* is found. When queried, the voters assign ratings to the items from a discrete domain of values $D$ where $d_{min}$ and $d_{max}$ are the lowest and highest values, respectively. The score $s \in \{m \cdot d_{min}, \dots, m \cdot d_{max}\}$ is the aggregated rating an item received. The winner is the item with the highest aggregated score: $max_j \sum_i q_j^i$.

We first define the necessary winner under the Range voting protocol (section 4.1), and present a method for computing voter-item distributions (section 4.2) We then present a novel dynamic programming algorithm for computing item winning probabilities (section 4.3). Next, we suggest two heuristics for query selection (section 4.4 and section 4.5): the Dynamic Information Gain (DIG) heuristic computes the information gain of each potential query based on the entropy of the item's probability to win. The query that maximizes the information gain is selected. The second heuristic, Expected Score (ES), uses the probability distribution of the voters' preferences to select the candidate most likely to win and the voter that is expected to maximize the score of that item. In both algorithms, voter-item probability distributions are computed and updated as new information is revealed. The algorithms output a necessary winner item (Konczak and Lang2005). The heuristics are evaluated under different settings (section 4.6. Finally, we discuss the findings (section 4.7

## 4.1 The Necessary Winner

We now define the necessary winner under the Range voting protocol. In Range voting, the pessimistic value (possible minimum) and the optimistic value (possible maximum) of an item are the lowest bound and the highest bound of the range respectively. Formally, let $O^i = \{q_p^i, \dots, q_t^i\}$ represent the set of voter $v_i$ responses to queries. Note that this set does not necessarily contain all the items. $\mathcal{O}^A = \{O^1, \dots, O^n\}$ is a set of $O^i$ sets. The function $pmax^A(c_{j,}\mathcal{O}^A)$ computes the possible maximum rating for item $c_j$, given the preference values of the voters.

*Definition 2.(Range voting Possible Maximum):*

$$pmax^A(c_j, \mathcal{O}^A) =$$

$$\sum_i pmax^i(c_j, O^i), where\ pmax^i(c_j, O^i) = \begin{cases} d_g & if\ \exists q_p^i = d_g \in O^i \\ d_{max} & otherwise \end{cases}$$

Similarly, the function of the possible minimum rating of item $c_j$: $pmin^A(c_{j,}O^A)$ is:

*Definition 3.(Range voting Possible Minimum):*

$$pmin^A(c_j, \mathcal{O}^A) =$$

$$\sum_i pmin^i(c_j, O^i), where\ pmin^i(c_j, O^i) = \begin{cases} d_g & if\ \exists q_p^i = d_g \in O^i \\ d_{min} & otherwise \end{cases}$$

A necessary winner $NW$ is an item whose minimum aggregated rating is greater than the maximum aggregated rating of all the others. Formally:

*Definition 4.(Necessary Winner):*

$$NW(c_i) = \{c_i | pmin^A(c_i, \mathcal{O}^A) > pmax^A(c_j, \mathcal{O}^A)\ \forall c_j \in C \backslash ci\}$$

## 4.2 Probabilistic Voter Ratings Distribution Model

In this section we present a method for computing voter-item distributions. When given a set of voters $V$ and a set of items $C$, the goal is to determine a querying policy which minimizes the cost and determines a necessary winner. While each voter has a unique rating for each item $q_j^i$, it is not necessarily known to the voting center. We assume that there exists an approximate rating distribution of the voter-item preferences (i.e., an approximate distribution of each voter's preferences for each item). In an iterative process, voter-item pairs are queried and the ratings are revealed. The Rating distribution is then updated.

*Definition 5.(Rating Distribution): the voting center considers $q_j^i$ as a discrete random variable distributed according to some rating distribution $vd_j^i$, such that $vd_j^i[d_g] \equiv P_r(q_j^i = d_g)$.*

The example presented in Table 7 shows the rating distribution of three voters for two items in the domain $D = \{1,2,3\}$. For example, the probability that $v_1$ will assign a rating of 1 to item $c_1$ is 0.2. The probabilities for each item sum to 1. For example for item $c_1$: $0.2 + 0.2 + 0.6 = 1$.

Table 7: Rating distribution of the voters in the set $V = \{v_1, v_2, v_3\}$

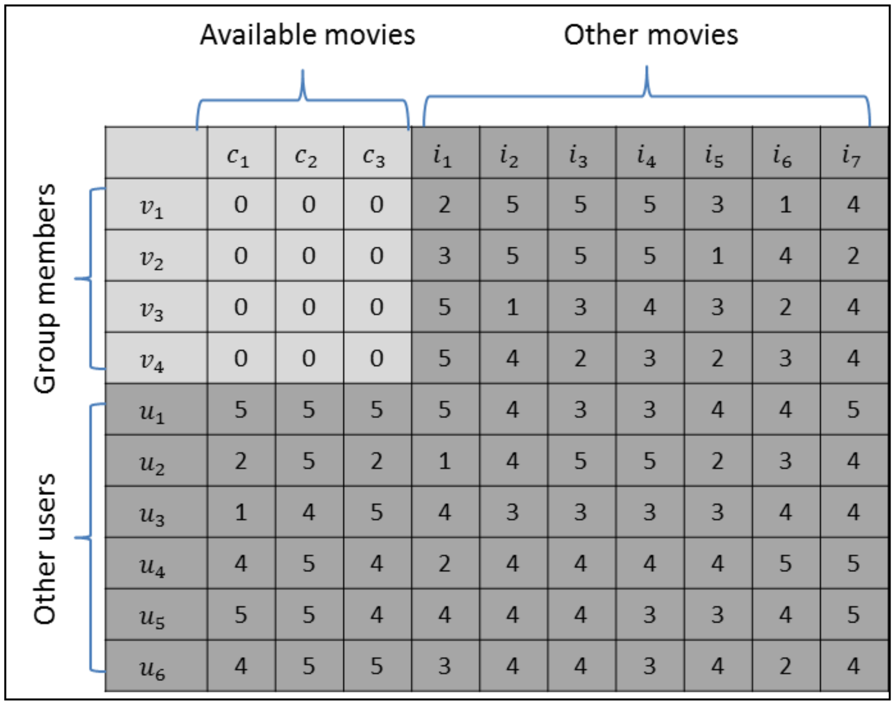
|  | $v_1$ |  | $v_2$ |  | $v_3$ |  |
|---|---|---|---|---|---|---|
|  | $c_1$ | $c_2$ | $c_1$ | $c_2$ | $c_1$ | $c_2$ |
| $d_1 = 1$ | 0.2 | 0.2 | 0.4 | 0.5 | 0.3 | 0.7 |
| $d_2 = 2$ | 0.2 | 0.2 | 0.3 | 0.2 | 0.3 | 0.1 |
| $d_3 = 3$ | 0.6 | 0.6 | 0.3 | 0.3 | 0.4 | 0.2 |

In order to show that the assumption of the existence of a-priori rating probabilities is a realistic assumption, we present a method for approximating voter-item rating distributions. Furthermore, in order to demonstrate the methods feasibility, a detailed example of how these ratings are calculated is demonstrated.

We assume independence between the probability distributions. While the independence assumption is naive, it can be used for approximating the actual probability. An attempt to address dependency will yield probabilities that are too complex for a system to realistically hold. When facing the tradeoff between the model's accuracy and practicality, we chose to model a practical system. However, note that the precise probability value is not required if the queries are still sorted correctly according to the value of the information they hold (their informativeness). In the closely related domain of machine learning, a similar naive assumption is known to provide accurate classification, though the independence assumption is not always true (Domingos and Pazzani1997). We therefore argue the system's loss of accuracy, if it exists at all, is insignificant.

The method uses *historical ratings* data to examine the correlation between voters and thus predicts the rating distribution of the voters. For example, consider a voting center whose task is to decide which movie to recommend to a group of members, out of a few available movies. The center has some *historical voter ratings* (i.e., some of the voters have rated movies which are not current candidates). The center also has *historical item ratings*(i.e., some of the candidate movies have been previously rated by voters who are not part of the group of current members). This is illustrated in Figure 6 that follows. The numbers represent known ratings, $v_1..v_4$ represent the group members who wish to see a movie together, and $c_1..c_3$ represent the available movies. We use a collaborative filtering (CF) method (Goldberg et al. 1992) to examine the correlation between voters, based on their ratings. CF examines voter rating patterns and is usually used to predict ratings. We extended this method and used the rating predicted by the CF method to approximate a rating distribution. Specifically, our goal is to approximate voter-item voting distributions $vd_j^i$ for voter-item pairs whose ratings are unknown. The benefit of this algorithm, beyond the computation of the a-priori distribution, is that it can be used to compute posterior probabilities after receiving the response of each query.

Note that the voting center examines a set of voters $V$ and a set of items $C$; the center's goal is to output an item from $C$ that fits the preferences of the voters in $V$. To represent the voters and items beyond $V$ and $C$ we define a new set of voters $U = \{u_1..u_q\}$ and a new set of items $I = \{i_1..i_p\}$ so that $V \subset U$ and $C \subset I$. We assume that we have a history of ratings, that is, the ratings are known for at least some of the voter-item pairs of the sets $U$ and $I$, respectively. The reader is reminded that we denoted the set of known voter ratings $O^A$, where $O^i \in O^A$ is the set of known ratings of voter $v_i$. In the same manner, let $O_j = \{q_j^p, \ldots, q_j^t\}$ represent the known ratings of item $c_j$ and $\mathcal{O}^B$ signify the set of sets so that $O_j \in \mathcal{O}^B$.


Available movies
Other movies
Group members
Other users

| | $c_1$ | $c_2$ | $c_3$ | $i_1$ | $i_2$ | $i_3$ | $i_4$ | $i_5$ | $i_6$ | $i_7$ |
|---|---|---|---|---|---|---|---|---|---|---|
| $v_1$ | 0 | 0 | 0 | 2 | 5 | 5 | 5 | 3 | 1 | 4 |
| $v_2$ | 0 | 0 | 0 | 3 | 5 | 5 | 5 | 1 | 4 | 2 |
| $v_3$ | 0 | 0 | 0 | 5 | 1 | 3 | 4 | 3 | 2 | 4 |
| $v_4$ | 0 | 0 | 0 | 5 | 4 | 2 | 3 | 2 | 3 | 4 |
| $u_1$ | 5 | 5 | 5 | 5 | 4 | 3 | 3 | 4 | 4 | 5 |
| $u_2$ | 2 | 5 | 2 | 1 | 4 | 5 | 5 | 2 | 3 | 4 |
| $u_3$ | 1 | 4 | 5 | 4 | 3 | 3 | 3 | 3 | 4 | 4 |
| $u_4$ | 4 | 5 | 4 | 2 | 4 | 4 | 4 | 4 | 5 | 5 |
| $u_5$ | 5 | 5 | 4 | 4 | 4 | 4 | 3 | 3 | 4 | 5 |
| $u_6$ | 4 | 5 | 5 | 3 | 4 | 4 | 3 | 4 | 2 | 4 |

Figure 6: A movie scenario example

We assume that other than their ratings, no information about the voters is known. The correlation between the voters is therefore based only on their ratings. However, observed ratings might have a bias due to voter and item effects. A voter bias occurs when a voter tends to rate higher or lower than average. For example, a voter who usually rates “1” or “2” on a scale of 1 to 5 has a negative bias. Similarly, an item bias occurs when an item tends to receive higher or lower rates than the average. Typical data can contain a large amount of voter and item bias (Koren and Bell2011). Therefore, examining the correlation between voters on the given ratings will give a skewed result which does not reflect the real probability distribution.

As proposed by Koren and Bell (2011), to cope with possible existing bias in item and voter data, we choose to correct the given rating of a voter-item pair $q_j^i$ . To compute the bias for voters and items we look at the deviation from the average. Suppose *a* is the average rating of all items by all voters and $b^i$, $b_j$ are the bias of voter $v_i$ and item $c_j$, respectively. For example, a voter has a positive bias if she tends to give candidates a higher rating than average. An item has a positive bias if it is rated higher than average. Equally, a negative bias exists for voters rating lower than average and items rated lower than average. The bias of an item is:

$$(4.1)\ b_j = \sum\nolimits_{i \in O_j} \frac{q_j^i - a}{|O_j|}$$

To avoid a double calculation of the bias, the voter deviation considers the item bias:

$$(4.2)\ b^i = \sum\nolimits_{j \in O^i} \frac{q_j^i - b_j - a}{|O^i|}$$

The baseline prediction of a voter-item pair is denoted as $b_j^i$ and can be computed from the average rating and the voter and item bias:

$$(4.3)\ b_j^i = a + b^i + b_j$$

For example, suppose we want a baseline $b_1^1$. Suppose $v_1$ tends to rate higher than the average voter and the $c_1$ is a popular item which receives ratings higher than average. This will result in a high baseline prediction. In order to find the probability distribution the proposed algorithm relies on the Cosine similarity equation, which is used in many collaborative filtering recommender systems for computing similarity between items or voters (Breese et al. 1998; Koren and Bell2011). However, we do not apply the equation directly on the ratings since they might be biased. We are interested in a bias-free correlation between the voters.

Subsequently, we propose to compute the baseline prediction (eq.4.3) and subtract it from the voters given rating. The obtained delta expresses the actual bias-free voter behavior:

$$(4.4)\ \Delta_j^i = q_j^i - b_j^i$$

We propose computing the similarity between a voter $v_i$ and a voter $v_k$ on the bias-free rating (eq.4.4). Note that the similarity is calculated based on ratings provided by $v_i$ and $v_k$ on mutually-rated items.

$$(4.5)\ sim^{i,k} = \frac{\sum_{j \in O^i} \Delta_j^i \cdot \Delta_j^k}{\sqrt{\sum_{j \in O^i} (\Delta_j^i)^2 \cdot \sum_{j \in O^i} (\Delta_j^k)^2}}$$

We can now predict a rating for $v_i$ and item $c_j$ based on the degree of similarity between voters $(v_i, v_k)$ and the voter bias:

$$(4.6)\ \hat{q}_j^{i,k} = b_j^i + sim^{i,k} \cdot \Delta_j^k$$

```
Input:
        $O^A$ - user rating sets
        $O_B$ - rated item sets
        $V$ - users
        $C$ - items
     Output:
        VD - a rating distribution
1. Initialize: $\forall_{i,j}\ vd_j^i[d_g] \equiv P_r(q_j^i = d_g) \leftarrow 0$
2. For each $v_i$ and $c_j$ do:
3.    Compute $\Delta_j^i$ (eq.4.4)
4. For each $v_i \in V$ do:
5.    For each $v_k \in O_B$ calculate the similarity $sim^{i,k}$ (eq.4.2)
6. For each user-item pair do:
7.   For each user $v_k$ do:
8.     Compute the predicted rating $\hat{q}_j^{i,k}$ (eq.4.6)
9.     Round $\hat{q}_j^{i,k}$ to the nearest rating $d_g$
10.For each user-item pair do:
11.  For each rating $d_g$ do:
12.      If $(rounded\ \hat{q}_j^{i,k}) = d_g$ then $pr(q_j^i = d_g) \leftarrow pr(q_j^i = d_g) + sim^{i,k}$
13.       Normalize $VD$
Return VD
```

Figure 7: Pseudo code for computing the initial probability distribution

We now extend this method and use the rating predicted by the CF method to approximate an initial rating distribution. The pseudo code for computing the initial rating distribution is presented in Figure 7. The algorithm receives the sets of historical ratings as inputs, voter ratings sets, and rated items sets. First, an empty rating distribution is initialized (line 1) and the bias-free voter behavior is calculated (lines 2-3). Next, the similarity of voter pairs is calculated (lines 4-5). Then, for every voter-item pair, the predicted rating according to neighbor voter $v_k$ is determined according to eq.5.6 (line 8). The result is rounded to the closest rating (line 9). Next, the similarity results are aggregated into buckets according to ratings (lines

11-12). The buckets are normalized (line 13). The normalization is done for each voter-item pair by dividing the value of each bucket in the total aggregated values for this pair. Finally, a rating distribution is returned.

We calculate the initial rating distribution before the heuristics are applied. Both heuristics iteratively reveal one new rating at a time. This allows the update of the distribution every time a new rating is added. The accuracy is expected to grow with the number of ratings acquired. Note that the proposed algorithm can be used when no history of ratings is given (this is known as cold start). In such a case, the returned distribution will be uniform, updated as ratings are acquired. A full example can be found in the appendix.

## 4.3 Item Winning Probability Using Dynamic Programming

We present a dynamic programming algorithm for computing the item winning probability under the Range voting protocol. First, let us define the probability that an item has a certain score and the probability of an item to win:

*Definition 6. (Item Score Probability): the probability that the score of item $c_i$ equals $s$ is $Pr(c_j = s)$, when $s \in \{n \cdot d_{min}, \dots, n \cdot d_{max}\}$ is the score of the aggregated ratings an item received.*

*Definition 7. (Item Winning Probability): Under the independence of probabilities assumption, the probability that item $c_j$ is a winner is the aggregation of $c_i$'s probabilities to win over the possible ratings s: $Pr(NW = c_j) = \sum_{s=n\cdot d_{min}}^{n\cdot d_{max}} Pr(c_j = s \wedge \forall\, i \neq j\; c_i < s) = \sum_{s=n\cdot d_{min}}^{n\cdot d_{max}} Pr(c_j = s) \cdot \prod_{\forall i \neq j} Pr(c_i < s)$*

To compute the probability that an item will receive the score $s$ and to compute the probability that an item will receive a score of at most $s$, we use:

(4.7) $Pr(c_j = s | v_1 .. v_m) = \sum_{x=d_{min}}^{d_{max}} \left( Pr(c_j = s - x | v_1 .. v_{m-1}) \cdot Pr(q_m^j = x) \right)$

where $Pr(c_j = s | v_i) = Pr(q_i^j = s)$

(4.8) $Pr(c_j < s) = \sum_{k=n\cdot d_{min}}^{s-1} Pr(c_j = k)$

## 4.4 Information Gain Heuristic for Range Voting

The Dynamic Information Gain Heuristic (DIG) heuristic focuses on selecting queries that will at each stage maximize the available information in terms of entropy (Shannon2001). The heuristic computes the item winning probability using the dynamic programming algorithm presented in section 4.3. DIG is an iterative algorithm. It uses a greedy calculation in order to select a query out of the possible $m \times n$ queries. The chosen query is the one that maximizes the information gain. The information gain of a specific query is the difference between the prior and the posterior probability of the candidates to win given the possible responses to the query.

The algorithm steps are presented in Figure 8. The algorithm continues until a necessary winner is found. In order to select a query, the heuristic calculates the information gained from each one of the optional queries and then selects the one that maximizes it. To compute the information gain, the winning probability of each item is dynamically calculated (lines 1-6), as shown in section 4.3. Next, the heuristic calculates the information gain of the $m \times n$ possible queries (lines 7-10). The information gain of a query is the difference between the prior entropy (line 7) and the posterior entropy given the possible responses to the query:

(4.9) $H(NW) = -\sum_{j=1}^{m} Pr(NW = c_j) \cdot \log\left(Pr(NW = c_j)\right)$

*Definition 8. (Information Gain): The Information Gain (IG) of a query is:* $IG(NW|q_j^i) = H(NW) - \sum_{g=min}^{max} H(NW|q_j^i = d_g) \cdot Pr(q_j^i = d_g)$ *where* $H(NW|q_j^i = d_g)$ *represents the entropy of NW given the possible values by querying voter $v_i$ about item $c_j$.*

The query that maximizes information gain is selected: $argmaxIG_{i,j}(NW|q_j^i)$. The query selection process continues until a necessary winner is found.

```
Input:
          $V$ - users
          $C$ - items
          $S$ - scores
       Output:
          A query
 1. For each item $c_j$:
 2.    For each score $s_t$:
 3.         Calculate the probability that $c_j$ will receive $s_t$
            (eq. 4.7)
 4.         Calculate the probability that $c_j$ will receive at
            most $s_t$ (eq.4.8)
 5. For each item $c_j$:
 6.    Calculate the probability that $c_j$ will win: Pr($NW = c_j$)
 7. Calculate the entropy of  NW (eq. 4.9)
 8. For each possible query $q_i^j$:
 9.    Calculate the entropy of NW given $q_i^j$ (H(WC|$q_i^j$))
 10.   Calculate the information gain achieved by $q_i^j$
```

Figure 8: Algorithm 1 - Dynamic Information Gain heuristic

We return to the example in section 4.3. In order to calculate the information gain ($IG$) of a certain query, we calculate the entropy reduction of $NW$ that is achieved by that query. Table 8 shows the entropy $H(NW)$ for our example.

The entropy of $NW$ for each possible query response $\left(q_j^i = d_g\right)$ is denoted as $H\left(NW \middle| q_j^i = d_g\right)$. This entropy is now calculated. In our example, in Table 9, $NW = c_1|q_1^1 = 1$ is 0.997, $NW = c_1|q_1^1 = 2$ is 0.92 and $NW = c_1|q_1^1 = 3$ is 0.646. To calculate the weighted average of the entropy we multiply the entropy by the probability of the random variable $PR\left(q_j^i = d_g\right)$ (the reduced side in the Information Gain equation in definition 8). For instance, the weighted average of query $q_1^1$ is 0.771. Consequently, the information gain for query $q_1^1$ is:

$IG(NW|q_1^1) = H(NW) - (H(NW|q_1^1 = 1) \cdot 0.2 + H(NW|q_1^1 = 2) \cdot 0.2 + H(NW|q_1^1 = 3) \cdot 0.6) = 0.844 - 0.771 = 0.073.$

Finally, we select the query that maximizes the information gain. In our example, querying voter $v_2$ about $c_2$ generates the maximum information gain (0.084, Table 9, row 5, last column). The algorithm iterates until a necessary winner is found. Note that the information gain is calculated only for the unknown ratings.

The complexity of this algorithm is affected by the dynamic programming algorithm that computes the probability that $c_j = s$ $(\forall s \in \{n \cdot d_{min}, \dots, n \cdot d_{max}\})$. We calculate this probability for all items ($m$) and ratings ($n$*|$D$|) for every voter ($n$). This is done by scanning the possible ratings:

$$\sum_{k=d_{min}}^{d_{max}} Pr(rating\ of\ c_i\ from\ users\ v_1 \dots v_{n-1} = s) \cdot Pr\left(q_n^i = k\right) (|D|).$$

This dynamic algorithm is implemented for every possible query of the voters over the items $(m \cdot n|D|)$. Thus, the worst case complexity is $O(m^2 n^3 |D|^3)$.

Table 8: The Entropy Function H(NW)

| | Entropy |
|---|---|
| $c_1$ | 0.332 |
| $c_2$ | 0.511 |
| sum | 0.844 |

Table 9: Information Gain

| **Item** | **Voter** | $d = 1$ | $d = 2$ | $d = 3$ | Weighted average | IG (Information Gain) |
|---|---|---|---|---|---|---|
| $c_1$ | $v_1$ | 0.997 | 0.92 | 0.646 | 0.771 | 0.073 |
| | $v_2$ | 0.982 | 0.8 | 0.489 | 0.779 | 0.065 |
| | $v_3$ | 0.996 | 0.853 | 0.545 | 0.773 | 0.071 |
| $c_2$ | $v_1$ | 0.415 | 0.717 | 0.943 | 0.793 | 0.051 |
| | $v_2$ | 0.572 | 0.872 | 1 | 0.76 | 0.084 |
| | $v_3$ | 0.68 | 0.939 | 0.99 | 0.768 | 0.076 |

## 4.5 Highest Expected Score Heuristic for Range Voting

The highest expected heuristic (ES) score is based on the exploration vs. exploitation tradeoff. As mentioned earlier, a necessary winner is an item whose possible minimum is greater than the possible maximum of the other items. The possible maximum of an item decreases while its possible minimum increases as more information about voter preferences is revealed. Thus, an item for which no voter has yet submitted a rating has the highest possible maximum and must be considered as a possible winner. On the other hand, the same item has the lowest possible minimum and cannot yet be a necessary winner. Therefore, for more information, we may want to explore the voters' preferences for the items in order to determine their potential of being a necessary winner. Once we have enough information about the items' rating, we can exploit this information to further inquire about the items that are more likely to win given that the item in question is not the winner

We propose a heuristic which chooses its next query by considering the *item that has the possible maximu*m and the *voter expected to maximize the rating of that item*. The expected rating of $q_j^i$ based on the rating distribution $vd_i^j$ is:

(4.10) $ES\left(vd_j^i\right) = \sum_{g=min}^{max} \Pr(q_j^i = d_g) \cdot d_g$

```
Input:
    $V$ - users
    $C$ - items
Output:
    A query
1. Initialize: $ps \leftarrow 0$ (possible maximum) $max \leftarrow 0$
   (maximum) $index \leftarrow 0$
2. For each item $c_j$:
3.    $ps \leftarrow$ calculate the possible maximum of $c_j$ according
      to definition 2
4.    If $ps > max$ then $max \leftarrow$ ps and $index \leftarrow j$
5. For each user $v_i \in O_{index}$:
6.    Calculate the expected rating according to eq.4.10
Return the query that maximizes the expected rating:
$argmax_i ES\left(vd_{index}^i\right)$
```

Figure 9: Algorithm 2 – Highest Expected Score heuristic

For item $c_j$, we choose the voter that maximizes the expected rating: $argmax_i ES\left(vd_j^i\right)$. Using this approach, we encourage a broad exploration of the items since the less information we have about an item's rating, the higher possible maximum it has. In addition, we exploit the preferences revealed in order: (1) to refrain from querying about items that have been proven as impossible winners (since their possible maximum is less than a minimum of another item) and (2) to further examine an item that has the highest possible maximum and might be a necessary winner. The pseudo code for the ES algorithm is presented in Figure 9.

In the first step of the above algorithm, the target item of the query is chosen (lines 2-4). This is done by calculating the possible maximum of each item according to definition 3 in section 4.1 (line 3). Next, we choose the voter who is to be queried about that item (lines 5-6). We choose the voter who is expected to maximize the item's rating by computing the expected rating using the rating distribution of that item. This process is repeated until a necessary winner is found. Ties are broken according to the item positions according to an increasing order of all items.

The following is an illustration of the algorithm using the example used in the previous section. To begin with, we have only probabilistic knowledge of voter preferences. Since no voter has submitted any preference yet, in the first round the possible maximum of each item is 9 (since there are 3 voters and the maximum rating that can be assigned is 3). The first item $c_1$ is selected for a query according to the tie breaking policy. According to the distribution in Table 7, the expected ratings of the voters over $c_1$ are:

$$ES(vd_1^1) = 0.2 \cdot 1 + 0.2 \cdot 2 + 0.6 \cdot 3 = 2.4$$

$$ES(vd) = 0.4 \cdot 1 + 0.3 \cdot 2 + 0.3 \cdot 3 = 1.9$$

$$ES(vd_3^1) = 0.3 \cdot 1 + 0.3 \cdot 2 + 0.4 \cdot 3 = 2.3$$

Thus, the voter-item query pair is $q_1^1$. Assuming the response is $q_1^1 = 2$, in the next iteration the possible maximum of $c_1$ is 8 and of $c_2$ is 9. Therefore in the next round, $c_2$ is selected as the item in the voter-item query pair. The algorithm iterates until a necessary winner is found.

The complexity of this algorithm is polynomial in the number of voters, items, and domain size. In order to select the item that is to be queried, we compute the possible maximum of each item, which is $O(mn)$. To select which voter to query we compute the expected rating of the voters about the specific item, which is $O(n|D|)$. Thus, the total complexity is $O\big(n(m + |D|)\big)$.

## 4.6 Evaluation

In sections 4.4 and 4.5 we proposed two novel heuristics, DIG and ES, which determine a necessary winner. In this section, we investigate the performance of DIG and ES, both with and without updating the rating distribution (section 4.2. The most similar scenario (although not identical) to ours is found in (Kalech et al. 2011). Hence we have also compared our methods to their sequential-Top method. We refer to this method as SEQTOP. To the best of our knowledge, there are no other algorithms that attempt to find a necessary winner by eliciting voter preferences and minimizing cost on the Range voting protocol, therefore the baseline for measuring the effectiveness of our methods is a random procedure (RANDOM), which randomly selects the next query. To account for the randomness of the RANDOM algorithm, we repeated each experiment 20 times. An overview of the evaluation procedure was presented in chapter 3.

### 4.6.1 Simulated Data

A comparison of the four algorithms, DIG, ES, SEQTOP and RANDOM, using the UNIFORM technique for DIG and ES probability distribution, is presented in Figure 10. Axis x represents the different skewness levels assigned to one specific item. According to the UNIFORM technique, the other items have a uniform skewness (level "0" is Table 5). Axis $y$ represents percentage of the dataset queried.

The graph presents an experimental run on 15 voters and 20 items. Results illustrate that the DIG and ES perform almost equally and better than RANDOM. All three methods improve as the skewness negativity increases (i.e., when an item has a high probability of receiving a high rating and being the winning item). That is to say, when the winner item is more distinct, all three methods can easily identify the winner. Therefore, all three methods seem to discover this item quicker than in a uniform or positive skewness setting, where any of the items have an equal chance of being the winner. The Friedman Aligned Ranks test with a confidence level of 95% rejected the null-hypothesis that all heuristics perform equally. The Bonferroni-Dunn test concluded that ES and DIG significantly outperform RANDOM and SEQTOP at a 95% confidence level. However, DIG and ES are not significantly different. We obtained similar results for other sizes of $V \times C$ matrices. Note that in our experiments SEQTOP is not significantly different from the RANDOM baseline. This result repeats throughout the experiments and is discussed in section 4.7.

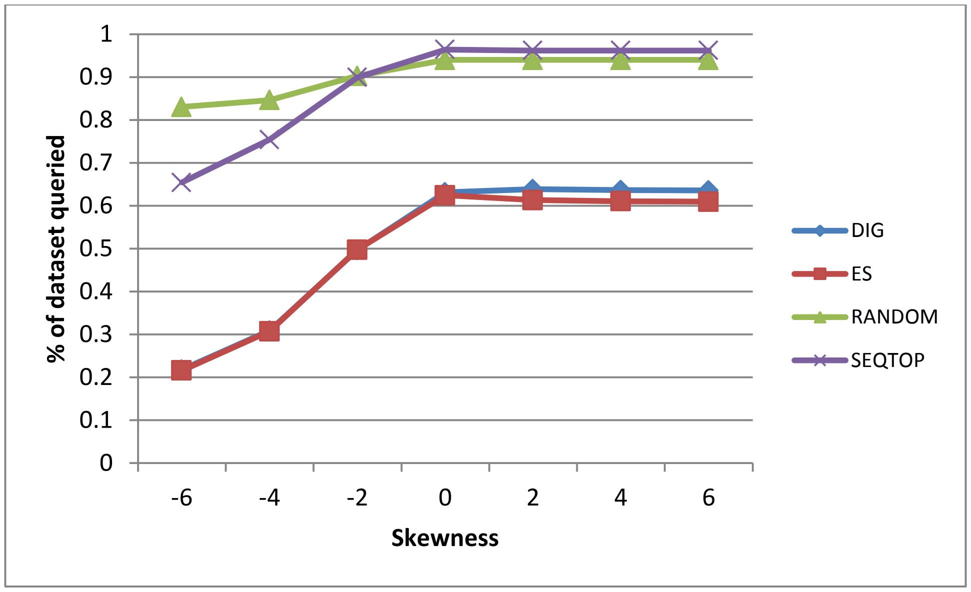


Figure 10: 15x20 dataset with one skewed item and UNIFORM skew for the rest of the items

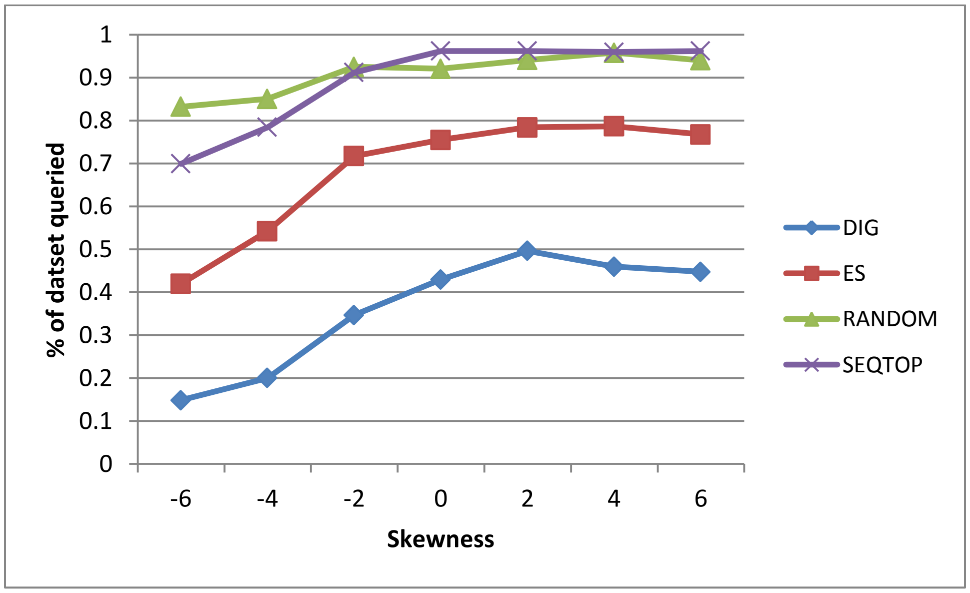
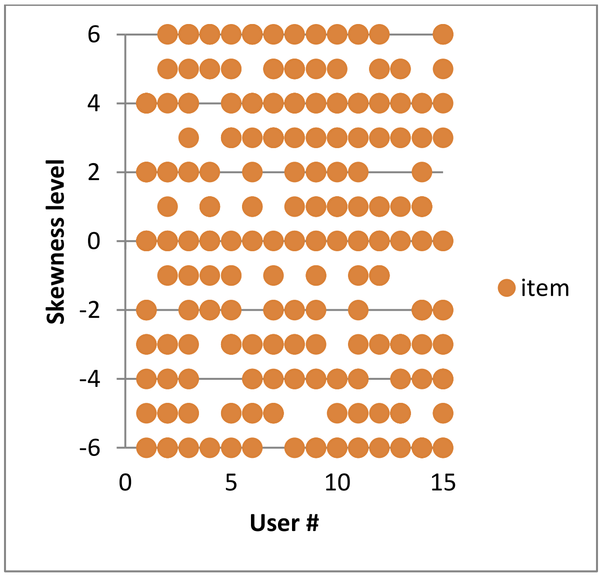
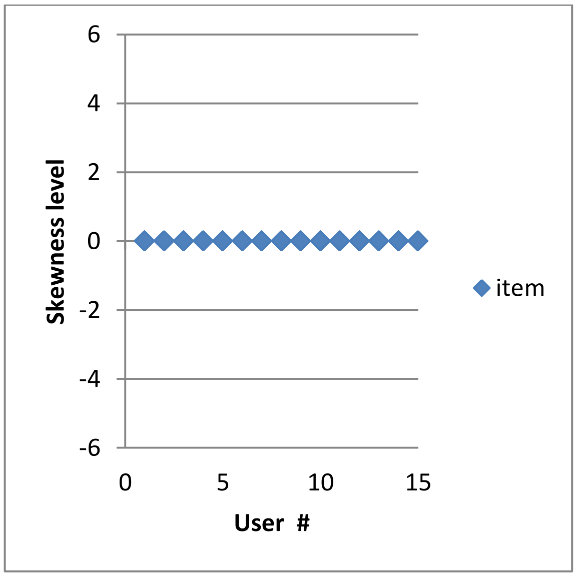


Figure 11: 15x20 dataset with one skewed item and LOTTERY skew for the rest of the items

Using the LOTTERY technique to set skewness for all items but one, we obtained different results (see Figure 11). Axis x represents the dataset size and axis y is the percentage of the dataset queried. The methods perform similarly on different dataset sizes; the graph illustrates performance on the 15x20 dataset. The Friedman Aligned Ranks test with a confidence level of 95% rejected the null-hypothesis that all heuristics perform the same. The Bonferroni-Dunn test concluded that DIG significantly outperforms RANDOM and SEQTOP at a 95% confidence level. However, ES outperforms only SEQTOP.

It is interesting to see that DIG outperforms ES when the LOTTERY technique is applied. We argue that the reason for this is due to the skewness of all voter-item pairs. To illustrate, Figure 12 displays the skewness of 15 voters and 20 items, when one item is set with uniform skewness and the other items have a skewness determined by (a) using the LOTTERY technique or (b) using the UNIFORM technique. Axis *x* represents the voters (from voter #1 to voter #15) and axis *y* represents the skewness level. The dots on the graph are the items. For example, the bottom-most dot in Figure 12(a) is an item whose pair is voter #1, which has a skewness level of -6. As illustrated, using the LOTTERY technique, the voter-item pairs have a scattered skewness with no distinct pattern. In this setting, DIG has an advantage as it considers all rating probabilities using an entropy function. Therefore, DIG is superior for noisy data and when no assumption can be made on the voter-item probable rating. ES focuses on the item that is most likely to win; when there is no such item, ES loses its advantage.

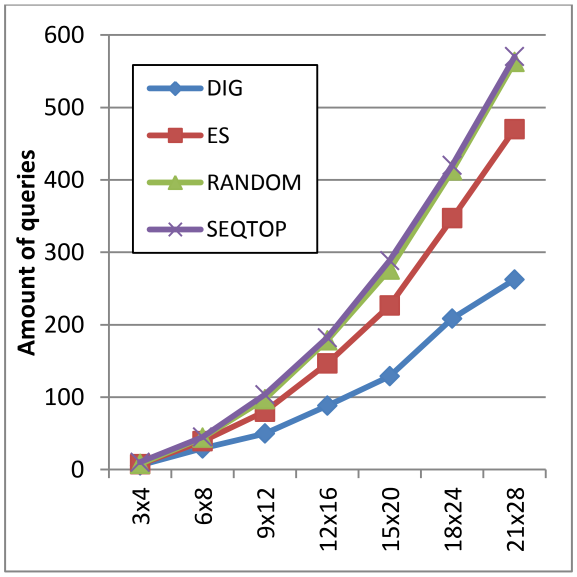


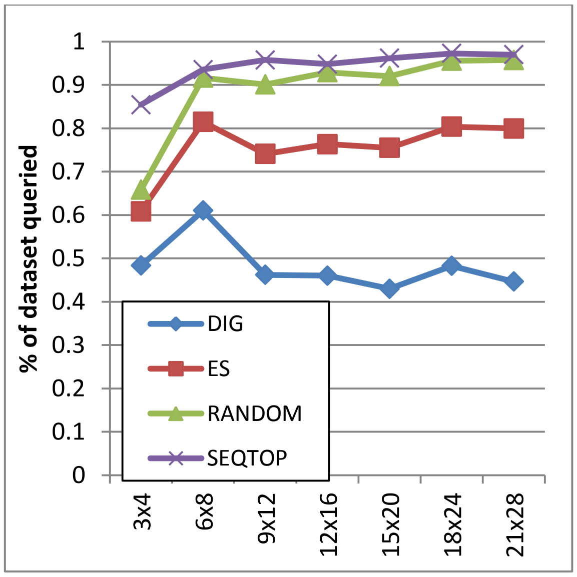


(a) (b)

Figure 12: The skewness of 15x20 using (a) LOTTERY and (b) UNIFORM

In the next set of experiments we varied the number of voters from 3 to 21 and the number of items from 4 to 28 in order to examine our algorithms under small and large settings. In Figure 13 we illustrate a comparison between the three algorithms when the dataset size increases and the skewness is determined by LOTTERY. Axis *x* is the dataset size and axis *y* is (a) the amount of queries or (b) the percentage of the dataset queried. As can be observed on graph (a) the number of queries required to identify the winning item grows with the size of the dataset. Graph (b) shows us that the percentage of the dataset queried is stable. Yet again, the Friedman Aligned Ranks test with a confidence level of 95% rejected the null-hypothesis that all heuristics perform the same. The Bonferroni-Dunn test concluded that DIG significantly outperforms RANDOM and SEQTOP at a 95% confidence level. However, ES outperforms only SEQTOP.





(a) (b)

Figure 13: One item with uniform skewness. For the rest of the items skewness is set by LOTTERY

Runtime results are presented in Figure 14. Axis *x* presents the datasets size and axis *y* shows the runtime per query in milliseconds. As observed, while DIG runs in polynomial time ES, SEQTOP and RANDOM run in linear time. This coincides with the runtime complexities we presented in sections 4.4 and 4.5.

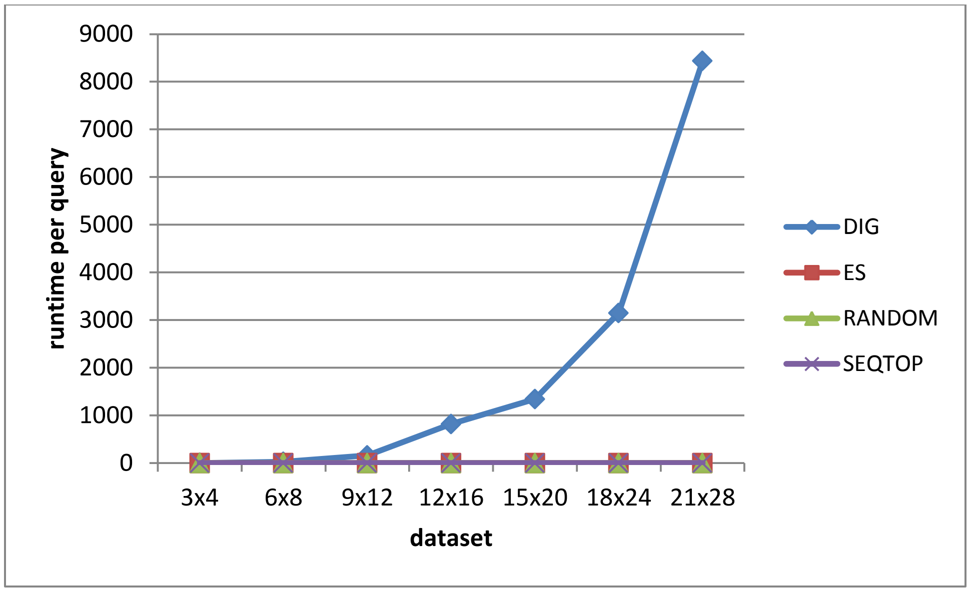


Figure 14: Heuristics runtimes

We further examined our methods, using the LOTTERY distribution settings under the following conditions: when the number of voters increase but the number of items remains the same, and when the number of items increases but the number of voters remains the same. This is illustrated in Figure 15 and Figure 16, respectively. In both cases, DIG performs in the best manner, followed by ES and SEQTOP. RANDOM is the worst performer. When the number of voters increases, all methods query a larger percentage of the dataset. When the number of items increases, the performance of all methods improves. Since finding a winner requires querying each one of the voters, this conclusion adheres to the necessary winner protocol. This task becomes more difficult as the number of voters increases. For the data in Figure 15, the Friedman Aligned Ranks test with a confidence level of 95% rejected the null-hypothesis that all heuristics perform the same. The Bonferroni-Dunn test concluded that DIG significantly outperforms RANDOM and SEQTOP at a 95% confidence level. However, ES outperforms only SEQTOP. In Figure 16, the Friedman Aligned Ranks test with a confidence level of 95% rejected the null-hypothesis that all heuristics perform the same. The Bonferroni-Dunn test concluded that DIG and ES significantly outperforms RANDOM and SEQTOP at a 95% confidence level.

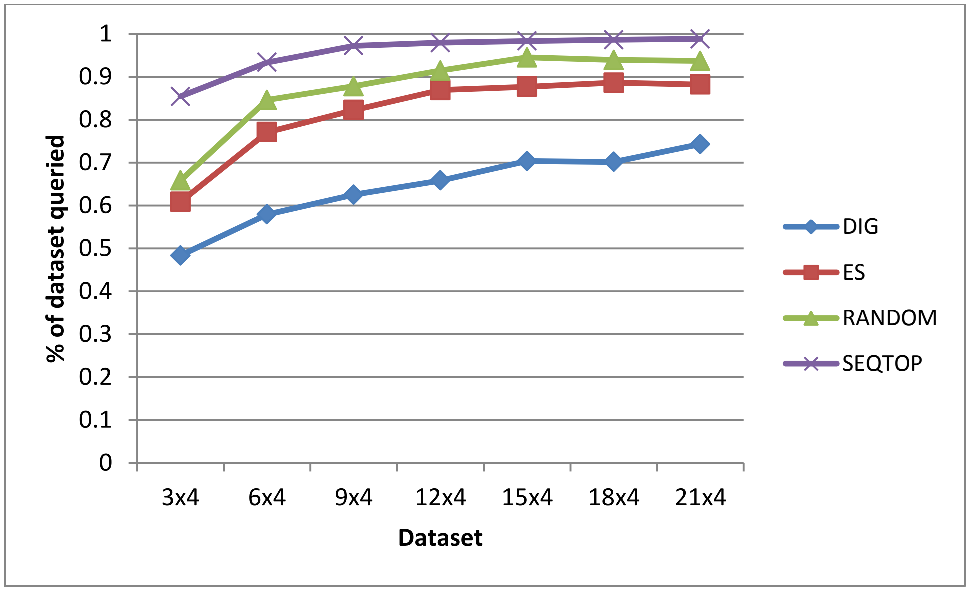


Figure 15: Increasing voter amount with UNIFORM skewness for one specific item, LOTTERY skewness for the rest of the items

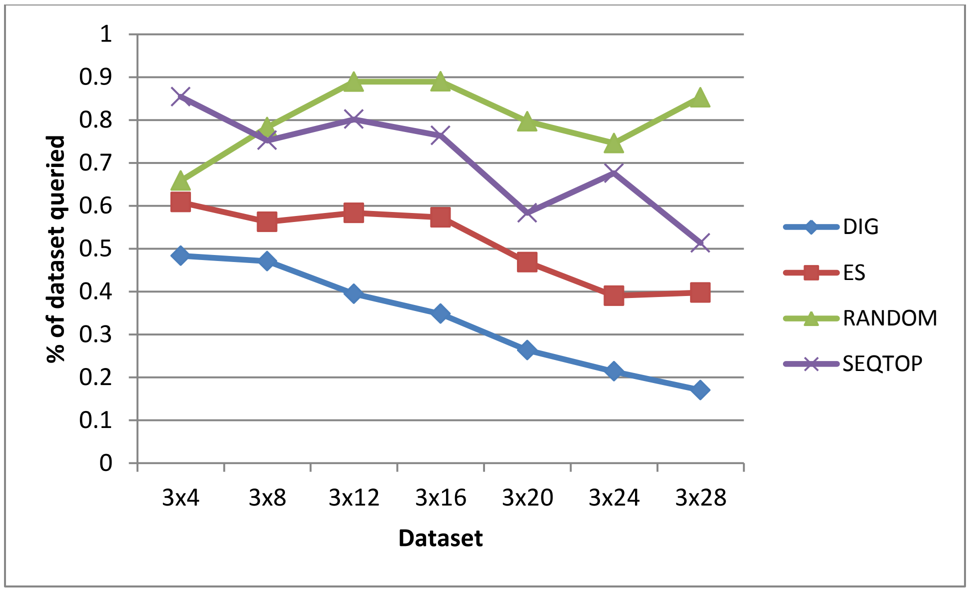


Figure 16: Increasing item amount with UNIFORM skewness for one specific item, LOTTERY skewness for the rest of the items

### 4.6.2 The Netflix prize dataset

In this section we examine the performance of DIG and ES on the real world Netflix dataset and analyze their sensitivity to different settings.

Note that in every iteration the rating of one voter-item pair is revealed and the rating distributions are updated with this new information. Our hypothesis states that updating the rating distribution would lead each algorithm to a faster solution. In order to examine this assumption, we generated two more settings:

(a) "UPDATE" – when the algorithm's rating distribution is updated as new voter-item ratings are revealed.

(b) "NO UPDATE" – when the algorithm's rating distribution is not updated.

In Netflix there is a huge amount of (sparse) historical rating data. Prior to the algorithm's execution, we generated the rating distributions of the voter-item pairs using the algorithm described in section 5. Both DIG and ES rely on the rating distributions. To compare rating distribution generated from a larger dataset to a rating distribution generated from a smaller dataset, we generated two settings:

(c) "BIG" – the rating distribution is generated from a matrix of size 1000 (i.e., $U \times I = 1000 \times 1000$). This matrix is more than 30 times bigger than the biggest evaluated dataset ($30 \times 30$).

(d) "SMALL" – the rating distribution is generated from a matrix of size 100 (i.e., $U \times I = 100 \times 100$). This matrix is about three times bigger than the biggest evaluated dataset.

Figure 17 presents DIG's performance on the different datasets using the SMALL matrix size for approximating and updating the distribution. Axis $x$ is the dataset size and axis $y$ is the percentage of queries from the amount of queries a naïve voting center would have asked ($n \times m$). As hypothesized, DIG algorithm performs significantly better, with a 95% confidence level, when the distribution is updated following each iteration. Interestingly, while there is no significant difference between the normalized amount of queries for different dataset sizes under NO UPDATE, once the distribution is updated, performance improves with an increase in the dataset size. The best results are obtained for the $30 \times 30$ dataset; the communication cost is cut to 48%.

This result is closely linked with the results presented in Figure 18 where DIG is presented with UPDATE on the BIG and SMALL matrices. Axis $x$ illustrates the different datasets and axis $y$ is the percentage of the dataset queried. DIG performs significantly better under SMALL. This can be explained by the fact that DIG is very sensitive to distribution updates (see Figure 17) and the distribution updates are more meaningful on the smaller dataset

(SMALL) since the ratio between each newly acquired rating and the dataset size is bigger for small datasets and thus has more impact on the distribution.

Although achieving the best results, ES behaves quite differently from DIG. ES generally seems to be less sensitive to changes in the settings than DIG. Figure 19 and Figure 20 demonstrate ES's performance under UPDATE and NO UPDATE and under BIG and SMALL respectively. Axis *x* shows the dataset size and axis *y* shows the percentage of the dataset queried until a necessary winner is found.

As observed in Figure 19  ES performs better under UPDATE. The result was found significant at a 95% confidence level.  Yet, another trend is noticed: ES generally performs better as the dataset size increases, regardless of the performance of an update. This is unlike DIG that presents the same trend for UPDATE. This might explain the somewhat baffling results presented in Figure 20 where no significant difference in ES's performance under settings BIG or SMALL is observed. While DIG inherently relies on the distribution by finding the query with the highest information gain, ES focuses on the expected score of a certain item. This makes ES less sensitive to the accuracy of the distribution and therefore, less sensitive to the matrix size (BIG or SMALL) from which the distribution is derived.

To understand the reason for the superiority of ES over DIG for the Netflix dataset, we examined a sample dataset of size 10x10 and checked the skewness of the rating distributions. Figure 21 demonstrates the skewness when the rating distributions are created using the (a) SMALL and (b) BIG techniques, respectively. Axis *x* illustrates the voter number (voters 1-10) and axis *y* illustrates the skewness level. Each dot on the graph accounts for one item. In graph (b) most of the voter-item pairs have a skewness clustered around the "0" level skewness. In graph (a) the skewness of the voter-item pairs is a bit more scattered, but still  clustered around the "0"  skewness level and again around the "10" skewness level. Note that some of the items overlap in their skewness so that only one dot is visible.

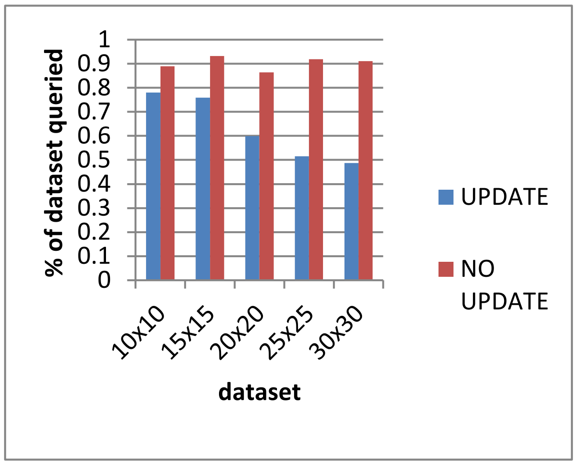


Figure 17: DIG algorithm on the “SMALL” rating distribution, with and without updates

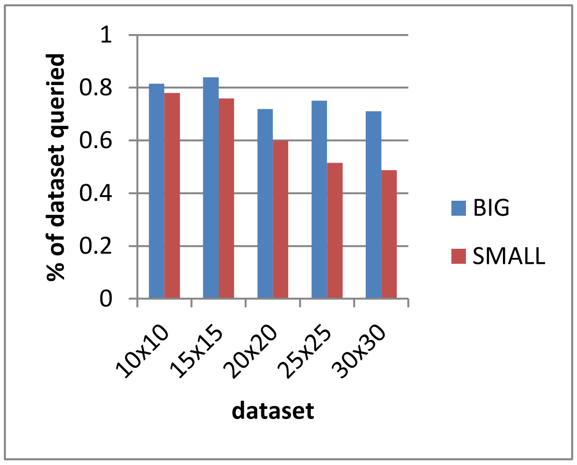


Figure 18: DIG algorithm with UPDATE on BIG and SMALL

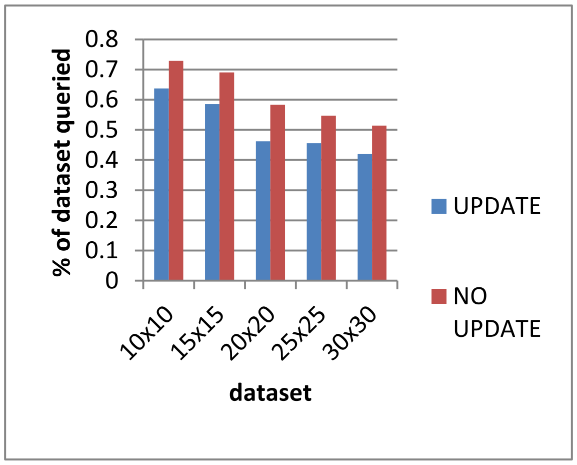


Figure 19: ES algorithm on the “SMALL” rating distribution, with and without updates

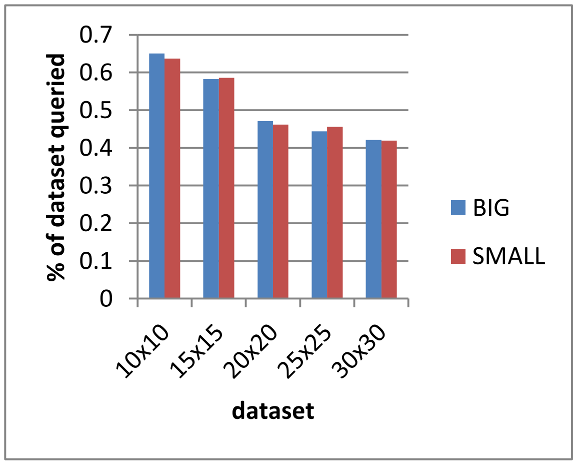
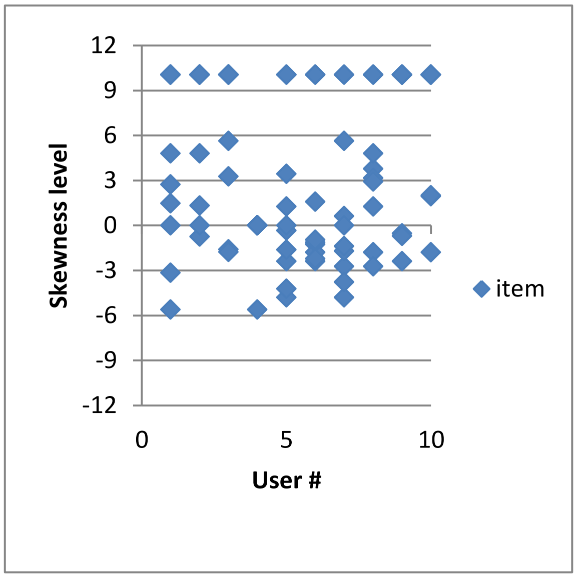
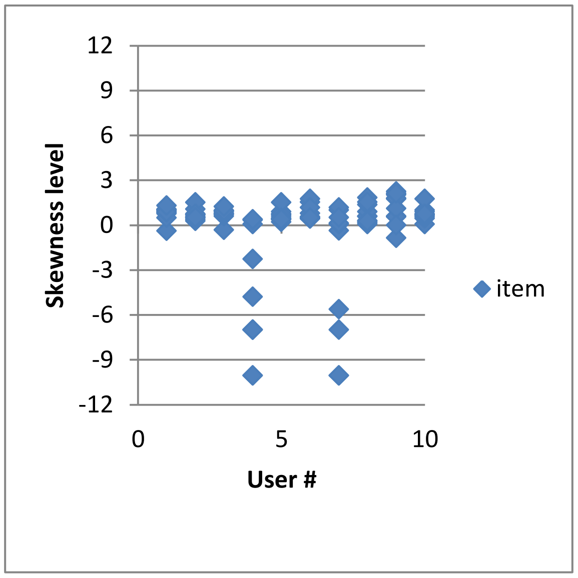


Figure 20: ES algorithm with UPDATE on BIG and SMALL

Recall that in our experiments on simulated data we showed that DIG has an advantage when there is no typical skewness pattern and ES is advantageous when there is some pattern to the skewness. In Figure 21 the distributions are clustered enough for ES to depict a probable winner, therefore making ES more attractive.





(a) (b)

Figure 21: Skewness when the distributions are created in the
(a) SMALL technique or (b) BIG technique

A comparison of the three algorithms DIG, ES, and RANDOM on all aspects measured is presented in Table 10. In this table we measured the percentage of queries out of the amount of queries a naïve voting center would have asked ($n \times m$). We averaged the percentages. Both algorithms show that updating the distribution based on the revealed ratings significantly reduces the amount of communication. The best result is achieved by ES; that is able to cut the communication load up to 51%. Note that RANDOM and SEQTOP are presented separately in the last column since they perform equally for all settings. The Friedman Aligned Ranks test with a confidence level of 95% rejected the null-hypothesis that all heuristics perform the same. The Bonferroni-Dunn test concluded that ES significantly outperforms DIG, RANDOM and SEQTOP at a 95% confidence level.

Table 10: Average percentage of dataset exhaustion under different settings

| | **(a) BIG** | | **(b) SMALL** | | **SMALL/BIG** | **SMALL/BIG** |
|---|---|---|---|---|---|---|
| | **DIG** | **ES** | **DIG** | **ES** | **RANDOM** | **SEQTOP** |
| **(c) UPDATE** | 0.77 | 0.51 | 0.63 | 0.51 | 0.95 | 0.91 |
| **(d) NO UPDATE** | 0.82 | 0.6 | 0.9 | 0.61 | 0.95 | 0.91 |

### 4.6.3 The Sushi dataset

The results of experiments on different dataset sizes are illustrated in Figure 22. Axis *x* is the dataset size and Axis *y* is the percentage of the dataset that was queried. In this dataset there are no more items beyond the ten items in question. Therefore, the DIG and ES heuristics begin with a uniform voter-item probability distribution for all voters and items. The results are similar to the results obtained in section 4.6.1 on simulated data with uniform distribution, namely there is no significant difference between DIG and ES. As explained in the previous sections, this result can be expected when the voter-item distributions are uniform.

The Friedman Aligned Ranks test with a confidence level of 95% rejected the null-hypothesis that all heuristics perform the same. The Bonferroni-Dunn test concluded that ES and DIG significantly outperform SEQTOP and ES also outperforms RANDOM at a 95% confidence level. As mentioned, there is no significant difference between DIG and ES.

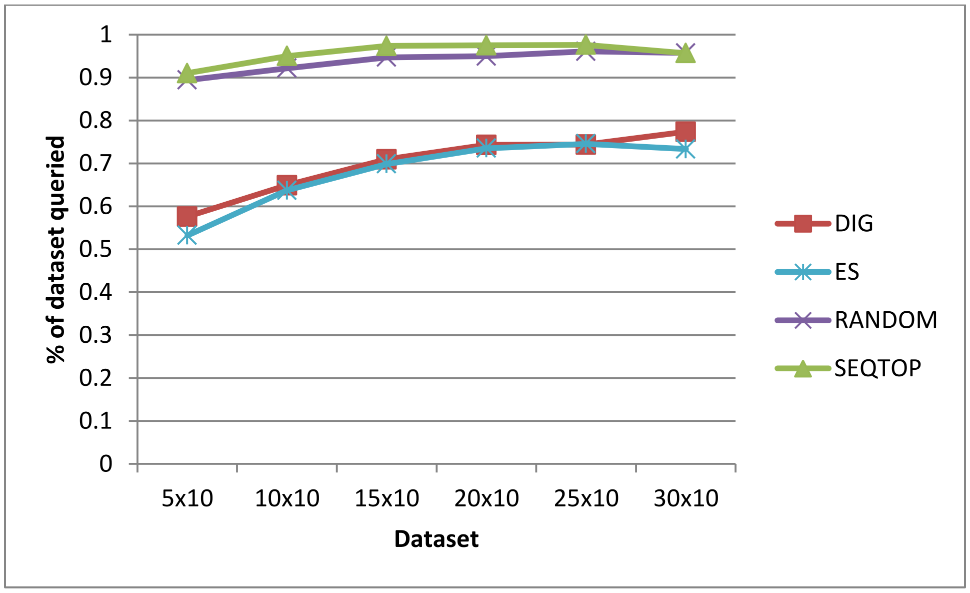


Figure 22: Comparison of algorithms on the Sushi dataset

### 4.6.4 The User Study Datasets

The results of the experiment on two datasets that we collected as part of a user study are displayed in Figure 23 for the pubs dataset and in Figure 24 for the Restaurants dataset. Axis x is the dataset size: from 5 to 30 users over 10 items. Axis y is the percentage of the dataset queried. In both dataset, we observe that ES requires fewer queries in order to find a winner.

For the Pubs dataset, the Friedman Aligned Ranks test with a confidence level of 95% rejected the null-hypothesis that all heuristics perform the same. The Bonferroni-Dunn test concluded that ES outperforms DIG and RANDOM and that DIG outperforms RANDOM at a 95% confidence level. For the Restaurants dataset, the Friedman Aligned Ranks test with a confidence level of 95% rejected the null-hypothesis that all heuristics perform the same. The Bonferroni-Dunn test concluded that ES significantly outperforms the rest of the heuristics at a 95% confidence level.

In order to explain these results, we examined the datasets skewness distributions (see Figure 25). In both cases, the skewness is clustered so that ES depicts a pattern and outperforms DIG as explained in section 4.6.1.

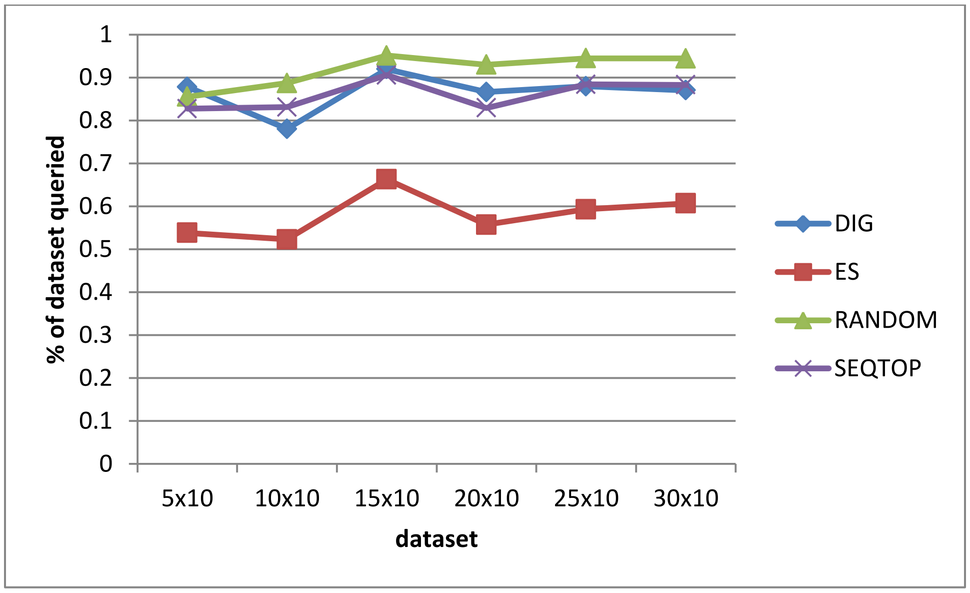


Figure 23: Comparison of algorithms on the Pubs dataset

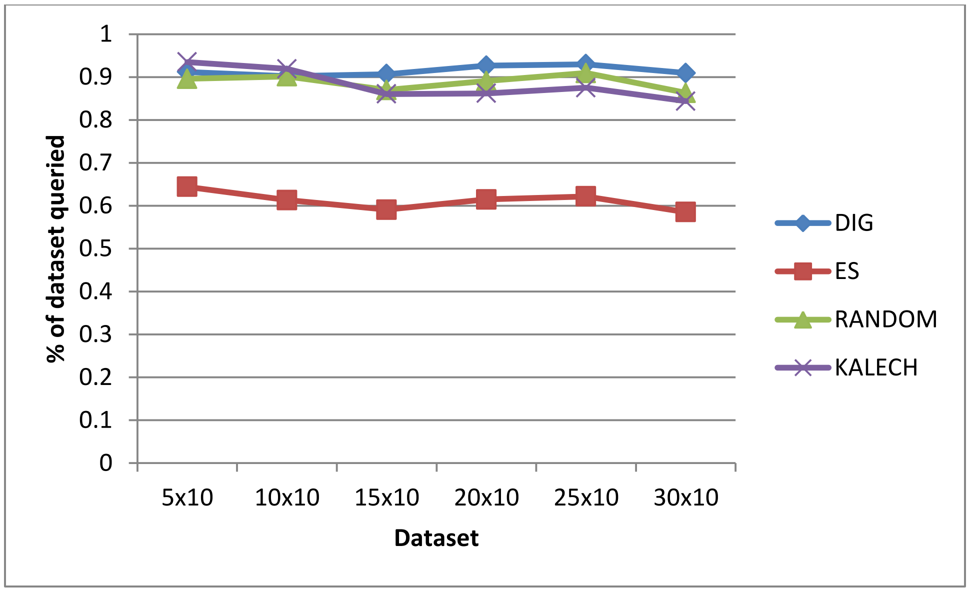
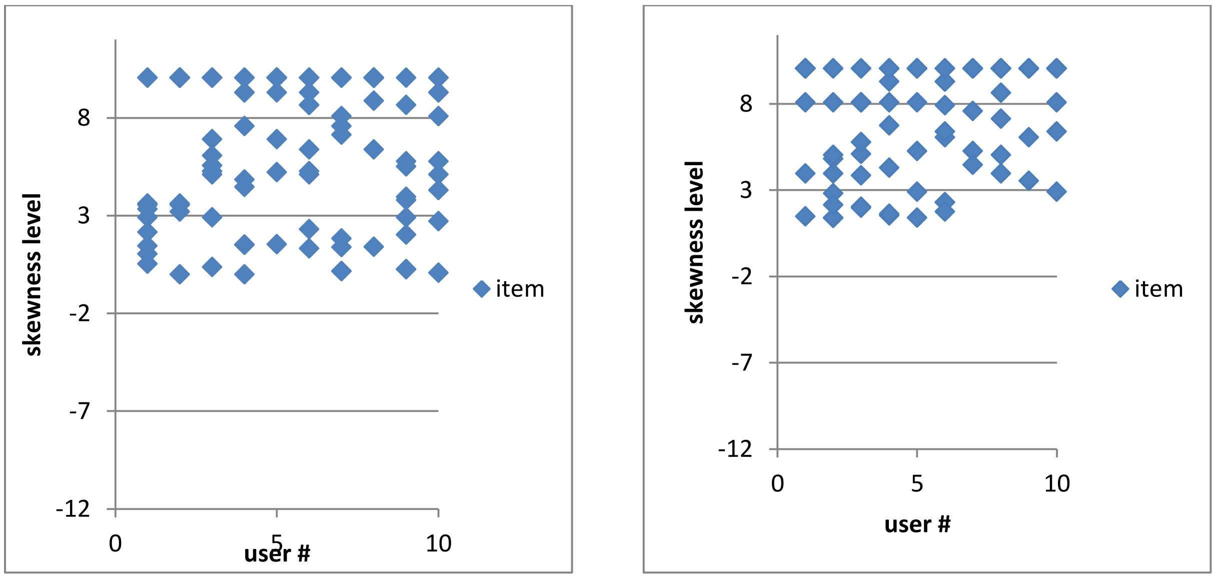


Figure 24: Comparison of algorithms on the Restaurants dataset



(a) (b)

Figure 25: Skewness of the distributions for datasets: (a) Pubs and (b) Restaurants

## 4.7 Discussion

In this chapter we presented algorithms that employ voting mechanisms and aim to find a winning item with minimal communication between the voting center and the voters. We assume that the voters' preferences are unknown in advance, but some historical ratings for voter-item pairs do exist. We proposed an algorithm for approximating the voter-item rating distribution.

Moreover, we offered two novel heuristics, DIG and ES, which iteratively choose a voter to query about a certain item's value until a winning item is found. DIG uses an entropy function to calculate the information gained from each possible voter-item pair and chooses to query the voter-item that maximizes the information gain. ES focuses on the most probable item to win. We have compared DIG and ES to a random baseline and to the sequentialTop method suggested in (Kalech et al. 2011), termed SEQTOP. Our experiments show that our proposed algorithms can reduce the communication load between the voting center and the voters by more than 50%. We have analyzed our algorithms in different settings both on simulated data and on real-world datasets and have identified the preferred settings for each of the algorithms.

The SEQTOP method performs poorly in our experiments, due to the fact that it was not designed for the purpose of single voter-item queries. Furthermore, SEQTOP does not consider voter-item probabilities. Therefore SEQTOP cannot operate optimally in the scenario we wish to evaluate in this study. In our scenarios, there is an intelligent voting center that computes what is the best next query, whereas SEQTOP was originally designed for scenarios where an intelligent voting center did not exist and therefore asks the voters for their top-rated item in a batch of queries. In (Kalech et al. 2011) SEQTOP is compared to a method where voters submit random items, not their top-rated items, and reveals that SEQTOP is significantly better.

Using simulated data, we showed that DIG is holds the upper hand when the data is noisy and there is no clear winning item. ES outperforms DIG when there is some skewness pattern in the data. From examining different dataset sizes on the percentage of queries, out of the amount of possible queries a naïve voting center would have asked ($n \times m$), we conclude that as expected, an increase in the number of the voters in the dataset leads to an increase in the percentage of queries. However, interestingly, an increase in the number of items leads to a decrease in the percentage of the required queries. This can be explained by the fact that each voter needs to be queried at least once in order for the voting center to determine a necessary winner, but not all items must be queried. With regards to runtime, for a fast convergence, ES is preferable since it runs in linear time while the DIG runs in polynomial time.

We wished to examine DIG and ES's performance in a real world setting. Therefore we used the Sushi dataset and the Netflix, Pubs and Restaurants datasets where our method for approximating voter-item rating distributions was applied. On the Sushi dataset, we did not find a significant difference between DIG and ES although both heuristics significantly outperformed

RANDOM and SEQTOP (as they did in all experiments). In this dataset the voter-item probability does not play a major role. This difference was found on the Pubs dataset and the Restaurants dataset.

On the Netflix dataset, we examined the algorithms with and without updating the voter-item distributions when new ratings were revealed. We found that both DIG and ES perform better when updates are enabled. DIG was found to be more sensitive than ES to the distribution updates. We also examined a scenario where the distribution is approximated from a small 100x100 dataset of historical ratings as opposed to a scenario where the distribution is approximated from a larger dataset of 1000x1000 historical ratings (we could not perform this evaluation for the Pubs and Restaurants datasets since they contain 90 users only). ES is indifferent to the size of the dataset, but DIG performs significantly better on the smaller 100x100 dataset. The differences between DIG and ES can be explained by the differences in the way they are constructed. ES identifies the item that is most probable to win and focuses on extracting ratings tied to this item. DIG looks at all of the voter-item pairs and chooses to query the pair with the highest information gain. Therefore, the more accurate the rating distributions are, the better DIG performs. Since the rating distributions are more accurate when updated and on a smaller dataset, these are settings in which DIG will excel. ES, being less sensitive to accuracy in the distributions, is also less affected by distribution updates and distribution accuracy. To conclude, we recommend using DIG when the data is noisy with no clear trend and when runtime is not of significance. ES should be used when runtime is an issue or when there is a high probability that a certain item or items are more favored.

# Chapter 5

# Preference Elicitation using the Borda Voting Protocol

In this chapter, we propose heuristics for query selection using the Borda voting protocol. The Borda protocol assumes every voter has a total order of ranked preferences on $n$ items. The voting center translates the preferences into an ordered sequence of values with a decreasing value of 1: $\{n-1, n-2\ldots 0\}$. Each value is uniquely assigned to one item only. The winning item is the item with the highest aggregated score: $max_j \sum_i q_j^i$. We address Borda voting with incomplete information. At the beginning of the process the voter-item-item preferences are unknown.

In an incremental elicitation model, the voting center queries for voter $v_i$'s pairwise preferences. A pairwise query $q_{j,k}^i$ for user $v_i$'s preference between candidates $c_j$ and $c_k$ has two possible responses: $q_j^i \prec q_k^i$ or $q_k^i \prec q_j^i$ meaning candidate item $c_k$ is either preferred over candidate item $c_j$ or vice versa. The goal of the elicitation process is to minimize the overall number of queries. Determining the next optimal query recursively depends on the order of the rest of the queries. There are an exponential number of such orders $(O(m \cdot n \cdot (n-1)^2)!)$ so finding the optimal minimal set of queries is intractable. Therefore, we propose a myopic approach for selecting the next user-item-item query trio.

We first define the necessary winner under the Borda voting protocol (section 5.1), and present a preferences distribution model for the Borda voting protocol (section 5.2). We then present a method for computing item winning probabilities (section 5.3) under the Borda protocol. Next, we suggest two heuristics for query selection (section 5.4 and section 5.5). The heuristics are examined under different settings (section 5.6). Lastly, a discussion of the analysis is provided (section 5.7).

## 5.1 The Necessary Winner

We now define the necessary winner under the Borda voting protocol. The *Borda possible maximum* of an item represents the possible highest score for an item based on the known preferences. When no preferences are known, the Borda possible maximum of item $c_j$ is the maximum score $s_n = n-1$ that any item can receive multiplied by the total number of items: $(n-1)\cdot m$. This score will be achieved if all voters will rank $c_j$ as their most preferred item. The Borda possible maximum of $c_j$ decreases by 1 for every voter that states some other item is preferred over $c_j$: $q_j^i \prec q_k^i$. Formally:

*Definition 9. (Borda Possible Maximum):*

$$pmax\left(c_j, O\right) = m \cdot (n-1) - \sum_{\forall\ q_{j,k}^i \in O} pmax\left(q_{j,k}^i\right),$$

$$where\ pmax(q_{j,k}^i) = \begin{cases} 1 & if\ q_j^i \prec q_k^i \\ 0 & otherwise \end{cases}$$

Similarly, when no preferences are known, the Borda possible minimum of an item $c_j$ is the minimum score $s_1 = 0$ multiplied by the total number of items: $0 \cdot m$. The Borda possible minimum of $c_j$ increases by 1 for every voter that states $c_j$ is preferred over some other item $c_k$: $q_k^i \prec q_j^i$. Formally:

*Definition 10. (Borda Possible Minimum):*

$$pmin\left(c_j, O\right) = \sum_{\forall\ q_{j,k}^i \in O} pmin(q_{j,k}^i)\ ,\ where\ pmin(q_{j,k}^i) = \begin{cases} 1 & if\ q_k^i \prec q_j^i \\ 0 & otherwise \end{cases}$$

The necessary winner is defined as:

*Definition 11. (Necessary Winner):*

$$NW = \left\{c_j \middle| pmin\left(c_j, O\right) > pmax(c_i, O), \forall c_i \in C\right\}$$

## 5.2 Probabilistic Voter Permutations Distribution Model

In this section we present a preferences distribution model for the Borda voting protocol. At the beginning of the elicitation process the voters preferences for items are unknown (i.e., the voting center does not know the response to any pairwise query $q^i_{j,k}$). The methods we suggest for elicitation require the voting center to hold probabilistic information as to each voter's preference between each pair of items. The pairwise preference probability is noted as: $p\left(q^i_j \prec q^i_k\right)$.

According to the preference probability, the voting center determines which query to execute (as will be shown in the next section). One option is to hold the preference probability of each voter for each $n(n-1)/2$ pairs of items. The advantage of this model is that the state space of the number of possible pairs per voter is polynomial and the model can easily cope with a large amount of candidates. However, this option ignores the dependency between pairwise preferences: according to the Borda protocol: $p\left(q^i_j \prec q^i_k \mid q^i_k \prec q^i_l\right) \neq p\left(q^i_j \prec q^i_k \mid q^i_l \prec q^i_k\right)$.

Hazon et al. (2012) consider these dependencies and to hold a full list of probabilities for all order permutations. An example of a permutation distribution for 3 voters and 3 items is given in Table 11. The pairwise preference probability of $c_j \prec c_k$ can be extracted by aggregating all the permutation probabilities where $c^i_j \prec c^i_k$. However, since $n!$ is the amount of permutations, this model cannot cope with a large amount of candidates. Therefore, one must choose whether to trade off model complexity with model accuracy. We follow Hazon et al. (2012) and hold a complete set of permutation probabilities.

Table 11: Voter permutation distribution for 3 voters and 3 items

| voters | $c_3 \prec c_2 \prec c_1$ | $c_3 \prec c_1 \prec c_2$ | $c_2 \prec c_3 \prec c_1$ | $c_2 \prec c_1 \prec c_3$ | $c_1 \prec c_3 \prec c_2$ | $c_1 \prec c_2 \prec c_3$ |
|---|---|---|---|---|---|---|
| $v_1$ | 0.1 | 0.1 | 0.2 | 0.3 | 0.2 | 0.1 |
| $v_2$ | 0.2 | 0.2 | 0.2 | 0.2 | 0.1 | 0.1 |
| $v_3$ | 0.3 | 0.3 | 0.1 | 0.1 | 0.1 | 0.1 |

Formally, the permutation set is defined as $VC = \{vc_1, \dots, vc_{n!}\}$. Voter $v_i$ ‘s permutation distribution, denoted by $vc^i$, is a discrete random variable, taking the values in $VC$. In the above example, $Pr\left(vc^3 = (c_3 \prec c_2 \prec c_1)\right) = 0.1$. We assume transitive closure exists.

The model can be derived from the voters’ history of preferences or from other voters’ preferences on the items in question. Deriving the permutation distribution is data specific; in section 5.6 we describe how the permutation distribution for the experiments data is derived.

## 5.3 Item Winning Probability Using Monte Carlo Sampling

The computation of the item winning probabilities under the Borda voting protocol differs from the computation under the Range voting protocol. For the Range voting protocol we demonstrated a dynamic programming algorithm for computing the item winning probability. The algorithm assumes independence of voter preferences. This assumption does not hold under the Borda protocol since: $p\left(q_j^i \prec q_k^i \,|q_k^i \prec q_l^i\right) \neq p\left(q_j^i \prec q_k^i \,|q_l^i \prec q_k^i\right)$. We therefore turn to an alternative method and approximate the item winning probabilities using a Monte Carlo algorithm (Fishman 1996) that uses sampling to estimate the winner. The Item Winning Probability algorithm proceeds as follows: for each user $v_i$, one permutation is sampled out of all possible user permutations $VC$. Once the permutations of all users are collected the winner is determined using the Borda protocol. These two steps are repeated $\gamma$ times. Finally, the winning probability of each item is calculated as the number of times the winner was found is divided by the sample size $\gamma$.

```
Input:
      $V$ - the set of voters
      $C$ - the set of candidate items
      $VC$ - the set of possible permutations
Output:  winning probabilities array $PrWin[n]$ for all $j's$
Initialize  $winnerArray[n]$ ← 0
Initialize $voterArray[m]$ ← 0
Repeat $\gamma$ times:
    For each voter $v_i \in V$
           $voterArray[i] \leftarrow$ sample a permutation from $vc^i$
    $c_{localwinner} \leftarrow$ winner in $voterArray$
 $winnerArray[localWinner] \leftarrow winnerArray[localWinner] + 1$
    For each item $c_j \in C$
       Compute $PrWin[j] \leftarrow winnerArray[j]/\gamma$
```

Figure 26: Pseudo code for Item Winning Probability Algorithm in the Borda protocol

## 5.4 Information Gain Heuristic for Borda Voting

Similarly to the Information Gain Heuristic (DIG) for Range Voting discussed in section 4.4, the heuristic described here focuses on selecting queries that will maximize the available information in terms of entropy (Shannon2001) at each stage. However, some adjustments are needed due to the differences in protocol. This heuristic is named Information Gain for Borda (IGB).

First, the information gain of each possible query is calculated. The information gain of a specific query is the difference between the prior and the posterior entropy of the probability to win of the item candidates given the possible responses to the query. The chosen query is the user-item-item query trio that maximizes the weighted information gain. The heuristic continues until a necessary winner is found. Ties in weighted information gain are broken according to the item positions in an increasing order of all items.

The entropy function is used in order to compute the query information gain. Given the item winning probabilities array $PrWin$, the entropy function is:

(5.1) $E(\mathrm{PrWin}) = -\sum_{j=1}^{m} PrWin[j] log(PrWin[j])$

The posterior entropy is calculated for the probability winner vector that has been computed for the two possible outcomes of a query $q_{j,k}^{i}$ We use $E\left(\mathrm{PrWin}\middle|q_{c_j \succ c_k}^{i}\right)$ to denote the entropy given user i prefers $c_j \succ c_k$. The information gain (IG) is the difference between the prior entropy of the local winner and the posterior entropy given that the response of an executed query $q_{j,k}^{i}$ is $c_j \succ c_k$:

(5.2) $IG\left(q_{c_j \succ c_k}^{i}\right) = \left(E(\mathrm{PrWin}) - E\left(\mathrm{PrWin}|q_{c_j \succ c_k}^{i}\right)\right)$

The probability that user $v_i$ prefers $c_j$ over $c_k$ $\mathrm{p}\left(q_{c_j \succ c_k}^{i}\right)$ can be calculated based on the prior permutation distribution. Thus we can compute the weighted information gain (WIG):

(5.3) $WIG\left(q_{j,k}^{i}\right) = IG\left(q_{c_j \succ c_k}^{i}\right) \cdot \mathrm{p}\left(q_{c_j \succ c_k}^{i}\right) + IG\left(q_{c_j \prec c_k}^{i}\right) \cdot \mathrm{p}\left(q_{c_j \prec c_k}^{i}\right)$

The chosen query is the query that maximizes the weighted information gain.

## 5.5 Highest Expected Score Heuristic for Borda Voting

The highest expected score heuristic for Borda (ESB) is based on the idea that it is better to select voter-item-item trios where one of the items is expected to win. This argument is supported by the idea that queries on the winner item will increase its possible minimum and finally identify it as a necessary winner. The next theorem considers the correlation between the winner and the number of queries.

*Theorem 1: The minimum amount of queries O necessary to determine the winner $c_j$ is $O = \frac{(n-1)^2 \cdot m}{n}$, on condition that all queries in O contain $c_j$ and that $c_j$ always wins when queried.*

Proof: Initially before the query process begins the Borda possible minimum and possible maximum are: $(c_j, O) = 0$ $and$ $\forall_{k \neq j}$ $pmax(c_k, O) = m \cdot (n-1)$. $c_j$ will be declared as a necessary winner once $pmin(c_j, O) > pmax(c_k, O)\ \forall_{k \neq j}$ (Definition 3). Let $|O|$ be the total amount of queries needed for verifying that $c_j$ is the winner. With each query $q_{j,k}^{l}$ the possible minimum of $c_j$ increases in 1 if $c_j \succ c_k$, else if $c_j \prec c_k$ the possible maximum of $c_k$ is decreased in 1. Thus, in the case that $c_j$ always wins and all queries contain $c_j$ we reach the necessary winner with the minimum amount of queries. The minimum score of $c_j$ will be in this case: $pmin(c_j, O) = |O|$. Each new query subtracts 1 from the possible maximum of some item $c_{k \neq j}$. To bring the items to the same possible maximum the queries should be distributed equally between the items. So every item (except $c_j$) is queried $|O|/(n-1)$ times, so that $pmax(c_k, O) = m \cdot (n-1) - |O|/(n-1)$. According to Definition 3 a winner is found when after $|O|$ queries the minimum is bigger than the maximum. Thus, $c_j$ will be a necessary winner once: $|O| > m \cdot (n-1) - |O|/(n-1)$ . Extracting $|O|$ reveals: $|O| = (n-1)^2 \cdot m/n$.□

Note that in the above extreme case, where in all queries the winner always wins, we see that solely querying the winner reduces the amount of queries to the minimum. The queries are distributed equally among the non-winning candidates. In less extreme cases where the winner does not win in all queries, it is still guaranteed that the winner will win in more queries than the other candidates. Therefore, we support a strategy that queries the item with the highest winning probability and thus increases the possible minimum of the expected winner rapidly.

The *Expected Score for Borda* (*ESB*) heuristic focuses on selecting queries that maximize the probability of an item to win. Given the item winning probabilities array $PrWin$, the highest probability is: $max(PrWin)$. The expected maximum ($EM$) represents the highest probability of the winning probabilities array given voter $v_i$ prefers $c_j$ over $c_k$ $\left(q_{c_j \succ c_k}^{i}\right)$:

(5.4) $EM\left(q_{c_j \succ c_k}^{i}\right) = max\left(PrWin | q_{c_j \succ c_k}^{i}\right)$

Since a query has two possible outcomes, the weighted expected maximum is:

$$(5.5)\quad WEM(q^{i}_{j,k}) = EM\left(q^{i}_{c_j \succ c_k}\right) \cdot \mathrm{p}\left(q^{i}_{c_j \succ c_k}\right) + EM\left(q^{i}_{c_j \prec c_k}\right) \cdot \mathrm{p}\left(q^{i}_{c_j \prec c_k}\right)$$

The chosen query is the query that maximizes the weighted expected maximum.

## 5.6 Evaluation

In sections 5.4 and 5.5 we proposed two novel heuristics, IGB and ESB, which determine a necessary winner under the Borda voting protocol. In this section, we evaluate the heuristics performance. The heuristics were compared to a baseline RANDOM method that selects queries at random. Since IGB and ESB use sampling, to accommodate for randomness each experiment was run 25 times. The $\gamma$ parameter in the Item Sampling algorithm (the algorithm that sets the item winning probabilities, described in section 5.3) was set to 300, as above this number we did not detect a noticeable difference in results.

In order to adjust Netflix to the probabilistic permutation distribution, as required in Borda protocol (section 5.2), we first derived a probability distribution of scores for each voter and item (as explained in section 4.2). Next, the probability distribution of scores was translated into a permutation distribution by aggregating the probabilities of each score for each possible permutation. Netflix contains ratings and not rankings and a voter is not limited to ordering items, and may give a few or all of the movies the same score. In cases where two items received an equal score from the user, we chose the items with the highest lexicographical order.

Figure 27 and Figure 28 display a comparison between the heuristics on the Netflix and Sushi datasets respectively. Axis x presents a varying size of voters and items. Axis y presents the percentage of the dataset queried (as explained in section 4.6.2). ESB is ~15% better than IGB and RANDOM with a 95% confidence interval according to a t-test. IGB does not significantly differ from RANDOM. ESB reduces communication up to 60% in total.

Runtime on the Sushi and Netflix datasets is presented in Figure 29 and Figure 30 respectively. Axis x presents a varying size of voters and axis y presents the runtime in seconds. RANDOM has a constant runtime while ESB and IGB have an exponential runtime. There is no significant difference between the runtime of ESB and IGB. Note that the runtime on the Sushi dataset is longer since it runs on 6 items and only on 5 items in the Netflix dataset.

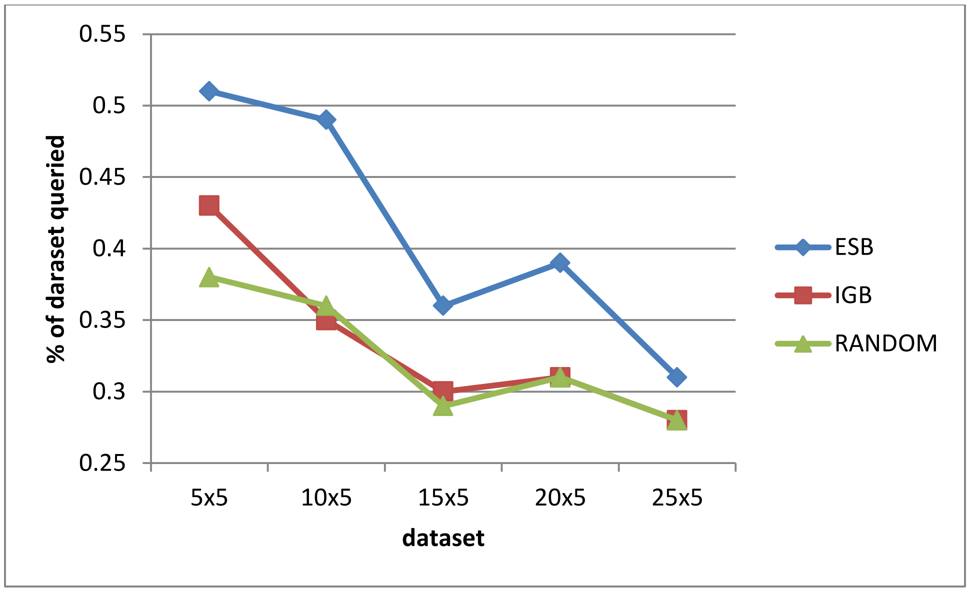


Figure 27: Comparison of algorithms on the Netflix dataset

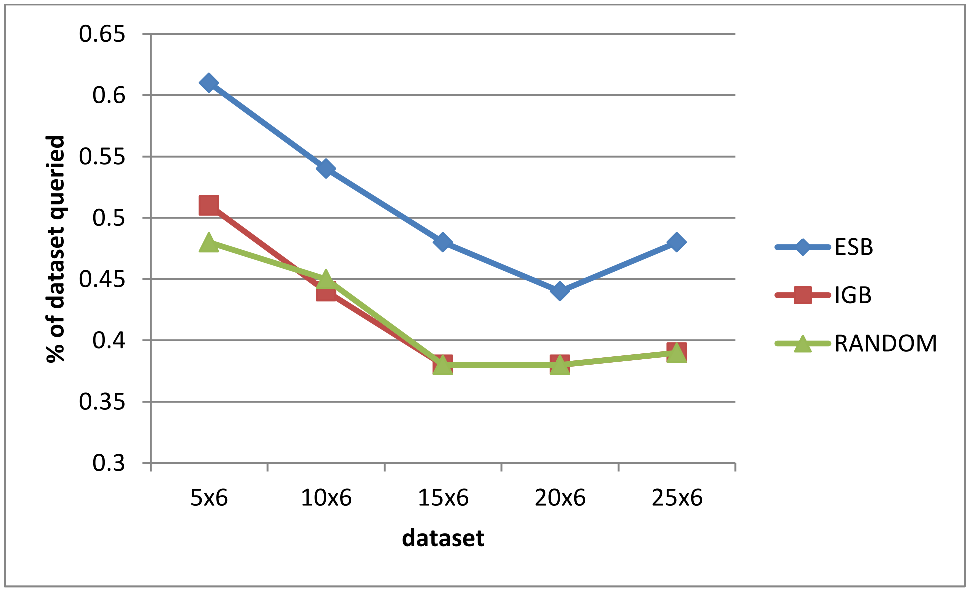


Figure 28: Comparison of algorithms on the Sushi dataset

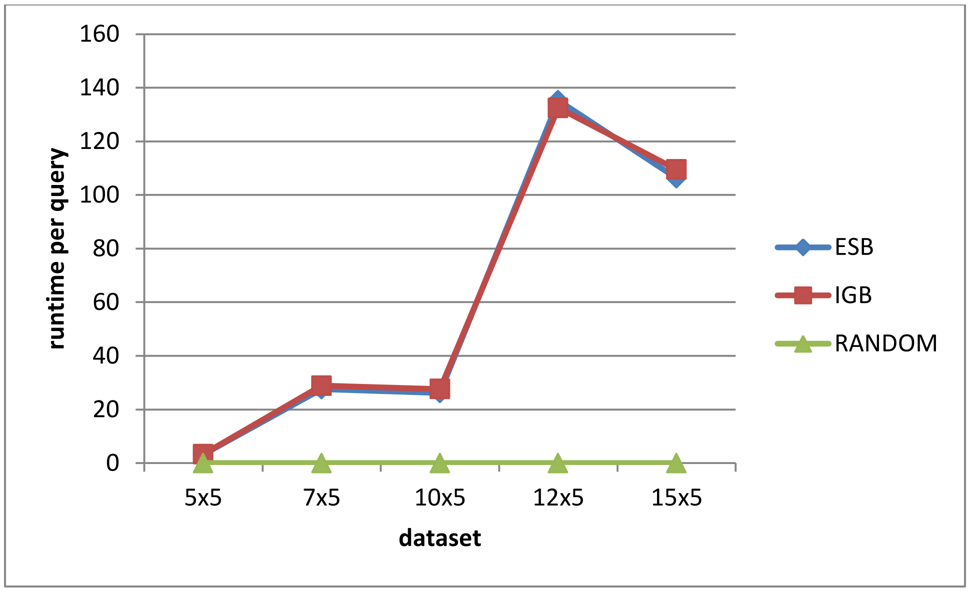


Figure 29: Runtime on the Netflix dataset

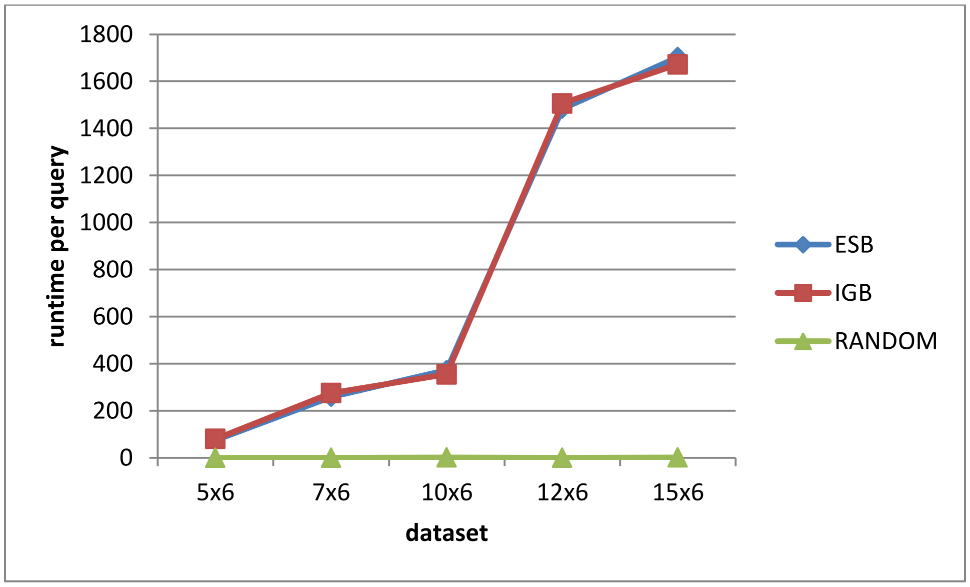


Figure 30: Runtime on the Sushi dataset

## 5.7 Discussion

In this chapter we presented heuristics that attempt to minimize the overall amount of queries needed for reaching a joint decision under the Borda protocol. In an iterative elicitation process, voters are queried for their preferences between two items. The process continues until a necessary winner item is found. The heuristics use probabilistic information of the voters' preferences. Usually the permutation distribution for datasets does not readily exist. However, we demonstrated a realistic method for easily learning this distribution on the Sushi dataset. Experiments on two real world datasets illustrate the superiority of the ESB heuristic over other possible heuristics. ESB manages to cut the communication (i.e., the queries) up to 60%. IGB, which seemed like a reasonable heuristic for this problem setting, failed in its performance.

Our findings imply that a reduction in the entropy does not necessarily bring us closer to finding a winning item. This seemingly surprising result can be explained by the fact that IGB heuristic focuses on reducing the overall uncertainty, while the focus should be on finding a winner. In detail, using the notion of entropy to select the next query seems like a reasonable idea for the discussed problem set. Based on this idea, IGB algorithm attempts to reduce the entropy to its minimum value (of 0). The minimum entropy is reached when the candidate winning probability vector ($PrWin$) becomes an indicator vector (in which the probability of exactly one item is a 1 and the others are 0). In this situation the necessary winner is revealed. Thus the entropy looks like a good proxy for our goal of finding the necessary winner. However, while the minimum entropy is equivalent to finding a winner, a reduction in the entropy value does not necessarily indicate that we are closer to finding the winner. Consider the following example: for 6 items the winning probability vector is: (0.9,0.02,0.02,0.02,0.02,0.02). After executing a certain query we receive the following probability vector: (0.88,0.11,0.025,0.025,0.025,0.025). In this case, the query has improved the entropy (the entropy dropped from -0.70119 to -0.59902). However, the query does not seem to bring us closer to finding the winner since there is also a drop in the probability that item number 1 is the winner. This brings us to the idea on which ESB is based: instead of calculating the information gain, simply select the query which potentially provides the highest increase in the maximum entry in the probability vector. In the last example we should prefer a query that brings us to the probability vector: (0.91,0.018,0.018,0.018,0.018,0.018). Although this vector provides a smaller drop in the entropy (-0.64544), it provides a positive improvement in the maximum probability of some item to win and thus brings us closer to finding a necessary winner.

Perhaps the main disadvantage of the presented framework is its lack of scalability, due to the need to hold a probabilistic model of all order permutations of items. Therefore while the number of voters can be increased, the number of items cannot.

# Chapter 6

# Tradeoffs and Aggregation Strategies in Preference Elicitation

The previous chapters dealt with returning one definite, necessary winner to a group of users. In this chapter we explore aggregation strategies and tradeoffs that reduce the required preference elicitation and thus reduce the communication costs. First, we offer to consider different preference aggregation strategies. These strategies differ in their emphasis: towards the individual users or towards the majority of the group (section 6.1). Second, rather than offering a single winner, we propose to offer the group top-$k$ best alternatives. This can be beneficial if a certain item suddenly becomes unavailable, or if the group wishes to choose manually from a few selected items (section 6.2.1). Finally, rather than offering a definite winning item, we suggest to approximate the item or the top-$k$ items that best suit the group, according to a predefined confidence level. We study the tradeoff between the accuracy of the proposed winner item and the amount of preference elicitation required (section 6.2.2). We evaluate these three challenges and show how they contribute to the minimization of the preference elicitation (section 6.3).

## 6.1 Aggregation Strategies

The item's score depends on the strategy used. We denote the employed aggregation strategy $str$. We now define the aggregation strategies: Majority and Least Misery. As mentioned, in the Majority strategy the emphasis is towards the majority of the group.

*Definition 12.(Majority Based Strategy): given the users' preferences, the Majority Based Strategy computes the score of item* $c_j$ *as follows:* $s_{majority}(c_j) = \sum_{i\in=\{1,\ldots,m\}} q_j^i$

In the Least Misery strategy, the chosen item cannot be the least preferred by any of the users.

*Definition 13.(Least Misery Strategy): given the users' preferences, the Least Misery Strategy computes the score of item* $c_j$ *as follows:* $s_{least}(c_i) = \min_{i\in=\{1,\ldots,m\}} q_j^i$

Each of the two strategies has its pros and cons. The choice of the strategy might impact the outputted result. Consider the example in Table 12, showing the preferences of 3 users for 3 items. According to the Majority Based strategy, the winning item is $c_1$, with a total score of 11, followed by items $c_2$ and $c_3$. According to the Least Misery strategy, the winning item is $c_2$, with a score of 3, followed by items $c_1$ and $c_3$. Therefore: $SV_{majority} = (c_1, c_2 c_3)$ and $SV_{least} = (c_2, c_1 c_3)$.

Table 12: Three users and their preferences for three items

| Group Members (Users) | Candidate Items | | |
|---|---|---|---|
| | $c_1$ | $c_2$ | $c_3$ |
| $v_1$ | 5 | 3 | 1 |
| $v_2$ | 4 | 3 | 5 |
| $v_3$ | 2 | 3 | 4 |
| Majority Based Score | 11 | 9 | 10 |
| Least Misery Score | 2 | 3 | 1 |

## 6.2 Termination Conditions

During the preference elicitation process, the preferences are submitted to the voting center that aggregates the preferences. This process continues until a termination condition is reached. The termination condition is pre-set by the system administrator according to the group's request. The termination condition is one of the following: a definite winning item, an approximate winning item with some confidence level, top-$k$ items where one of them is the winner, or approximate top-$k$ items where one of the items is the winner with some confidence level.

Given a set of responses to queries and a termination condition, the goal is to determine whether the iterative process can be terminated. Let $O^i = \{q_p^i, \dots, q_t^i\}$ represents the set of voter $v_i$'s responses to queries. Note that this set does not necessarily contain all the items. $\mathcal{O}^A = \{O^1, \dots, O^n\}$ is a set of $O^i$ sets. The function $pmax^A(c_{j,}\mathcal{O}^A)$ computes the possible maximum rating for item $c_j$, given the known preference values of the voters.

*Definition 14.(Possible Maximum): given the set of responses $\mathcal{O}^A$ and an aggregation strategy str, the possible maximum score of candidate $c_j$, denoted $pmax^A(c_j, \mathcal{O}^A)$, is computed as follows:*

$$pmax^A(c_j, \mathcal{O}^A, str) = \begin{cases} \sum_{i \in 1..m} pmax^i(c_j, O^i) & str = majority \\ min_i\left(pmax^i(c_j, O^i)\right) & str = least\ misery \end{cases}$$

$$where\ pmax^i(c_j, O^i) = \begin{cases} d_g & if\ \exists q_p^i = d_g \in O^i \\ d_{max} & otherwise \end{cases}$$

Similarly, the function of the possible minimum rating of item $c_j$: $pmin^A(c_{j,}O^A)$ is:

*Definition 15.(Possible Minimum): given the set of responses $\mathcal{O}^A$ and an aggregation strategy str, the possible minimum score of candidate $c_j$, denoted $pmax^A(c_j, \mathcal{O}^A)$ is computed as follows:*

$$pmin^A(c_j, \mathcal{O}^A, str) = \begin{cases} \sum_{i \in 1..m} pmin^i(c_j, O^i) & str = majority \\ min_i\left(pmin^i(c_j, O^i)\right) & str = least\ misery \end{cases}$$

$$where\ pmin^i(c_j, O^i) = \begin{cases} d_g & if\ \exists q_p^i = d_g \in O^i \\ d_{min} & otherwise \end{cases}$$

Consider the example given in Table 12 but assume that the rating of $c_1$ for $v_3$ is unknown: thus, $q_1^1 = 5, q_1^2 = 4, q_1^3 = ?$. Using definitions 3 and 4 we can compute the possible maximum and minimum under each aggregation strategy. For example, the possible maximum of $c_1$ in the Majority strategy is $pmax^1(c_1, O^1, majority) = 5 + 4 + 5$, since $q_1^3 = d_{max} = 5$. For the Least Misery strategy the possible maximum is: $pmax^1(c_1, O^1, least\ misery) = min(5,4,5)$ since $q_1^3 = d_{max} = 5$. The possible minimum for the two strategies is $pmin^1(c_1, O^1, majority) = 5 + 4 + 1$ and $pmin^1(c_1, O^1, least\ misery) = min(5,4,1)$ since $q_1^3 = d_{min} = 1$.

### 6.2.1 Selection Among top-$k$ Alternatives

As mentioned in previous chapters, one possible termination condition is to stop the preference elicitation process once one necessary winner is found. We follow Kalech et al. (2011) and define a necessary winner $NW$ as a set of items whose possible minimum aggregated rating is equal or greater than the possible maximum aggregated rating of all the others. This definition is equivalent to the definition provided by Konzak and Lang (2005), as proven in proposition 2 in Konzak and Lang (2005). Formally the necessary winner item set is:

*Definition 16.(Necessary Winners set):*

$$NW = \{c_i | pmin^A(c_i, O^A) \geq pmax^A(c_j, O^A)\ \forall c_j \in C \backslash ci\}$$

It is possible to receive more than one necessary winner. Although the necessary winner set may contain more than one item, we assume that there is only one necessary item. In the case of more than winning items, the first item is selected lexicographically.

In some cases, the group members can be satisfied with a shorter preference elicitation process. They may agree to trade the result accuracy with less elicitation cycles. In other words, instead of terminating the preference elicitation once a necessary winner is found, the group may agree to terminate the preference elicitation once a set of top-$k$ items is presented to them. One of these items is the necessary winner, but without further elicitation it is not possible to determine the winner. To accurately define the top-$k$ items, let us define the possible winner group. The possible winners are all the items whose possible maximum aggregated rating is greater than or equal to the possible minimum rating of all the other items.

*Definition 17.(Possible Winners Set):*

$$PW = \{c_i | pmax^A(c_i, O^A) \geq pmin^A(c_j, O^A)\ \forall c_j \in C \backslash ci\}$$

Note that the possible winning group subsumes the necessary winners: $NW \subset PW$. After each query, the necessary winner set and the possible winner set need to be recalculated. To begin with, when none of the preferences are known, the possible winner set contains all items: $|PW| = |C|$ and the necessary winner's set is empty: $|NW| = \emptyset$. The process is terminated once the size of the set of possible winners is reduced to $k$. We denote the possible winner set of size k $PW^k$. Thus, the set contains the top-$k$ possible winners, where, by definition, these top-k are guaranteed to include the necessary winners. The group of users is left with the task of selecting one among the top-$k$ items.

### 6.2.2 Winner Approximation

Another possible trade-off is the accuracy-elicitation tradeoff. The preference elicitation process can be reduced, but the accuracy of the output is affected, the returned items are estimated to contain the winning item at some confidence level with an error rate $\alpha$. To compute a necessary winner with some confidence level we will first define the score space of the aggregation. The score $s$ that the candidate can achieve after aggregating the preferences of the voters depends on the strategy:

$$S = \begin{cases} \{n \cdot d_{min}, n \cdot d_{min} + 1 \dots, n \cdot d_{max}\} & if\ str = majority \\ \{d_{min}, d_{min} + 1 \dots, d_{max}\} & if\ str = least \end{cases}$$

Let us begin with examining the probability that one item has a certain score: $Pr(c_j = s)$. The probability of any item to be the necessary winners is:

*Definition 18.(Item Winning Probability): Under the independence of probabilities assumption, the probability that item $c_j$ is the necessary winner is the aggregation of $c_j$'s probabilities to win over the possible ratings s:*

$$Pr(c_j = NW) = \sum_{s \in S, \forall\ i \neq j} Pr(c_j = s | v_1, \dots, v_m) \wedge \mathrm{Pr}(c_i < s)$$

$$= \sum_{s \in S \forall\ i \neq j} Pr(c_j = s | v_1, \dots, v_m) \cdot \prod_{\forall i \neq j} Pr(c_i < s)$$

The probability that given $m$ voters an item will receive the score $s$ $Pr\left(c_j = s|v_1, \dots, v_m\right)$ can be computed recursively. This probability depends on the aggregation strategy. For the Majority strategy we use:

$$\text{(6.1)} \quad Pr\left(c_j = s|v_1, \dots, v_m\right) = \sum_{x=d_{min}}^{d_{max}} \left(Pr\left(c_j = s - x|v_1, \dots, v_{m-1}\right) \cdot Pr\left(q_m^j = x\right)\right)$$

where $Pr\left(c_j = s|v_i\right) = Pr\left(q_i^j = s\right)$

For the Least Misery strategy we use:

$$\text{(6.2)} \quad Pr\left(c_j = s|v_1, \dots, v_m\right) = \sum_{x=s}^{d_{max}} \left(Pr\left(c_j = s|v_1, \dots, v_m\right) \cdot Pr\left(q_m^j = x\right)\right) + \sum_{x=s+1}^{d_{max}} \left(Pr\left(c_j = x|v_1, \dots, v_m\right) \cdot Pr\left(q_m^j = s\right)\right)$$

In both strategies we compute the probability that an item will receive a score of at most $s$ as follows:

$$\text{(6.3)} \quad Pr\left(c_j < s\right) = \sum_{x=\min(S)}^{s-1} Pr\left(c_j = x|v_1 .. v_m\right)$$

The following is a step by step running example, for the Majority strategy for $d = \{1,2,3\}$. The example is based on the voting distributions (VD's) presented in Table 11; note that $Pr(q_3^1 = 3) = 0.4$, $Pr(q_3^1 = 2) = 0.3$, $Pr(q_3^1 = 1) = 0.3$. We start by calculating $Pr(c_j = s)$. The calculation is done using a dynamic programming algorithm where each result is calculated using the previously calculated results. For instance, using Eq. (1), $Pr(c_1 = 6)$ based on the ratings of voters $v_1, v_2, v_3$:

$Pr(c_1 = 6|v_1 .. v_3) = Pr(c_1 = 5|v_1, v_2) \cdot Pr\left(q_3^j = 1\right) + Pr(c_1 = 4|v_1, v_2) \cdot Pr\left(q_3^j = 2\right) + Pr(c_1 = 3|v_1, v_2) \cdot Pr\left(q_3^j = 3\right)$. In the same manner: $Pr(c_1 = 5|v_1, v_2) = 0.14$, $Pr(c_1 = 4|v_1, v_2) = 0.36$, $Pr(c_1 = 3|v_1, v_2) = 0.24$ so that finally $Pr(c_1 = 6|v_1 .. v_3) = 0.236$. Next, we calculate $Pr(c_1 \leq s)$ using Eq (3): $Pr(c_1 < 6) = Pr(c_1 = 3) + Pr(c_1 = 4) + Pr(c_1 = 5)$.

To define top-$k$ with a confidence level we first define $PV$ as a vector of items, ordered according to their winning probability (Definition 19):

*Definition 19.(Ordered Vector of winning probabilities): PV[] is an array of decreasingly ordered items according to their winning probabilities.*

The probability that the necessary winner is within the top-$k$ is actually the aggregated winning probabilities of the first $k$ items in $PV$. The more preferences elicited from the users, the higher probability the necessary winner is within the top-*k*. The confidence level is a value which determines an upper bound for the probability of the necessary winner to be among the top-*k*. The preference elicitation process is terminated once the confidence level equals $1 - \alpha$. Formally, the termination condition is:

*Definition 20.(Termination with top-k approximate items): the preference elicitation process terminates for a given k and $\alpha$, when $\sum_{i=1}^{k} PV[i] \geq 1 - \alpha$ where $0 \leq \alpha \leq 1$.*

## 6.3 Evaluation

We present an empirical evaluation of the following statements: (a) Selection –outputting top-*k* items reduces the required number of queries (b) Approximation – there is a tradeoff between outputting an approximate winner, or approximate top-*k* items and outputting a definite winner or definite top-*k* items. The approximation accuracy improves as more data is collected. (c) Aggregation – the aggregation strategy affects the preference elicitation process. We examine two aggregation strategies: with emphasis towards the group and with emphasis towards the user (i.e., the Majority and Least Misery strategies).

We examine the performance of the DIG and ES algorithms presented in section 4. As mentioned in the related works section, to the best of our knowledge, there are no other algorithms that operate (or can be expanded to operate) under the same settings. Therefore, the baseline for measuring the effectiveness of our method is a random procedure (RANDOM), which randomly selects the next query. To account for the randomness of the RANDOM algorithm each experiment is repeated 20 times. In addition we evaluate the communication cost reduction (i.e., the reduction in amount of queries needed in order to reach the termination condition under the given strategy). We follow the evaluation procedure presented in chapter 3.

We evaluate the methods in terms of: (1) communication cost – we measure the number of queries required for finding the necessary winner (2) approximation accuracy. Our focus is on the analysis of the contribution of returning a winner within top-$k$ items, thus narrowing down the top-$N$ suggestions received by a recommendation system ($k \leq N$). An additional focus is on approximating a winner and on the aggregation strategies. The analysis of the scaling of the matrix sizes and the runtime has been evaluated in(Naamani Dery et al. 2014) Chapter 4.

We first present varying top-$k$ termination conditions (section 6.3.1).We then present an examination of the different confidence levels (section 6.2.1) and finally we compare the two strategies (section 6.3.3).

### 6.3.1 Selection of top-$k$ Items

We examined different top-$k$ termination conditions, from $k = 1$ (i.e., requiring one definite winner), to $k = 9$ (i.e., requiring the winner to be one of the top-9 items). The results are for the Majority aggregation strategy with a 100% confidence level ($\propto = 0$). Different confidence levels and a comparison between the performance of the Majority strategy and the Least Misery strategy are presented in the next sections. We first report the results of three levels of skewness of simulated data, followed by the Datasets: Netflix, Sushi, Pubs, and Restaurants.

We examine three different skewness levels of simulated data. Figure 31, Figure 32 and Figure 33 present results for a skewness level of (6), (0) and (-6) respectively. Axis x presents the termination conditions $k = 1, .., 10$. Axis y presents the percentage of the dataset queried in order to terminate and find a winner within the top-$k$. A larger $k$ means that the termination condition is relaxed and less queries are needed. Indeed, in all cases, as $k$ increases, the amount of queries decreases. The performance of RANDOM is not significantly affected by skewness levels. For a skewness level of -6 (Figure 31), DIG outperforms ES and RANDOM and requires the least amount of queries. For a skewness level of (0) and of (6), ES outperforms DIG and RANDOM for the top-1 to top-3 items. Then, DIG resumes charge and provides better results (Figure 32 and Figure 33).

We now turn to examine the real world datasets. On the Netflix dataset (Figure 34), the trend is similar to that obtained on the skewness level of 0 and 6. That is, for top-1 to top-3 ES is superior, and then DIG maintains the lead. Again, DIG displays a sharp curve while ES requires almost the same number of queries regardless of the termination point (the top-$k$). The same phenomenon is found on the Pubs dataset (Figure 36) and on the Restaurants dataset (Figure 37). However, on the Sushi dataset (Figure 35) DIG outperforms ES and RANDOM for all $k$.

The results can be explained by considering the properties of the heuristics and of the datasets. In a setting of a simulated skewness of (-6) the votes are skewed towards the winner and it is more obvious who the winner is. It is less obvious who the winner is when the skewness level is 0 or 6 in simulated data. Also, when $k$ is smaller, ES performs better, since ES is designed to seek for potential winning items. Therefore, the amount of queries ES requires is more or less constant regardless of the $k$ items required for output. DIG is designed to focus on reducing entropy. When $k$ is larger the entropy reduces faster. In the Sushi dataset the initial user-item distribution is uniform so all items have the same chance to be the winning item. Thus, the initial state in the Sushi dataset is similar to a simulated skewness data with (0). However in the Netflix, Pubs, and Restaurants datasets the distributions are estimated and there is a skewness pattern (see section 4.6) which enables DIG to outperform. Furthermore, when it is less obvious who the winner is (as in Netflix), the differences in the heuristics performance are smaller.

For all datasets, the Friedman Aligned Ranks test with a confidence level of 95% rejects the null-hypothesis that all heuristics perform the same. The Bonferroni-Dunn test concluded that DIG and ES significantly outperform RANDOM at a 95% confidence level.

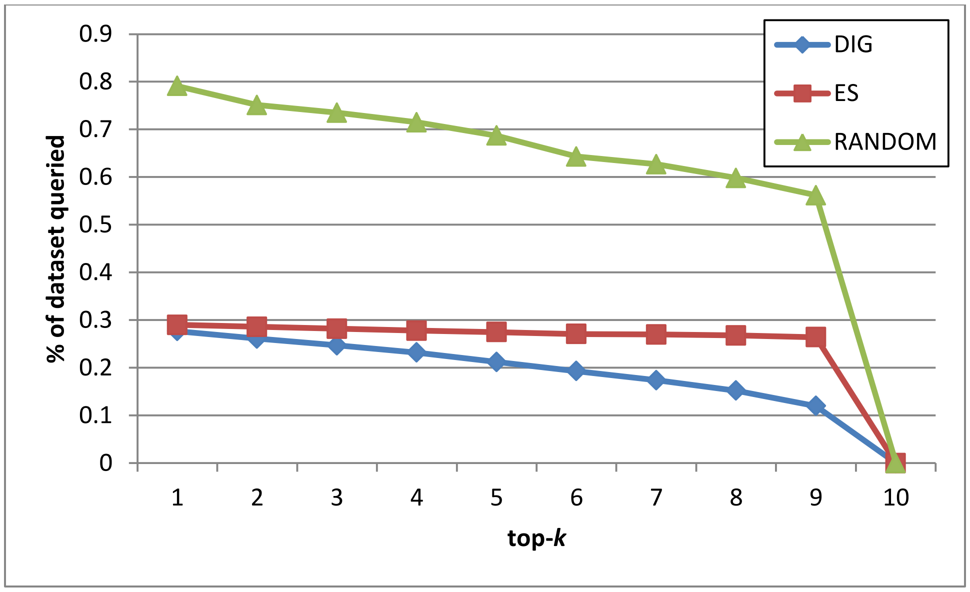


Figure 31: Heuristics comparison for top-*k* with skewness level (-6)

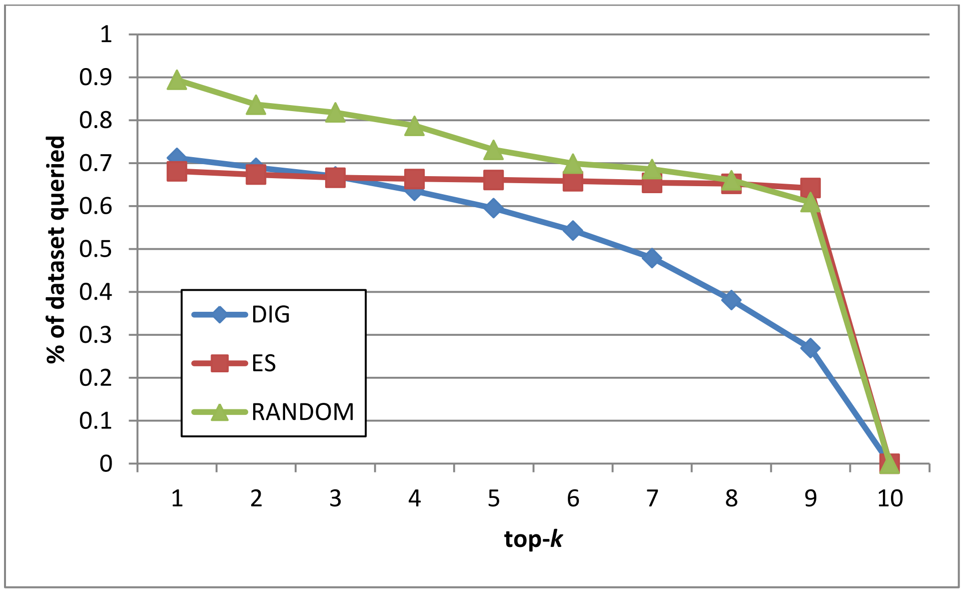


Figure 32: Heuristics comparison for top-*k* with skewness level (0)

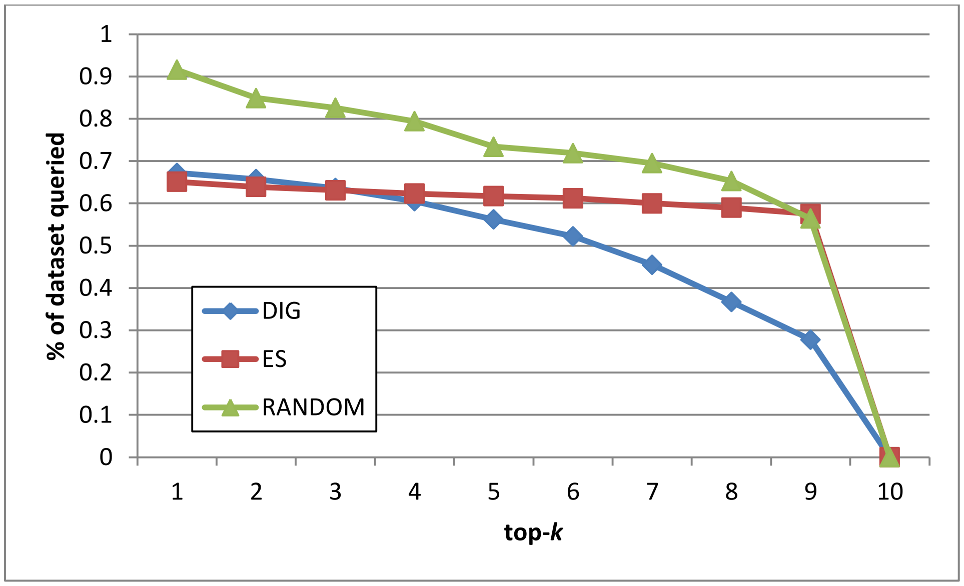


Figure 33: Heuristics comparison for top-k with skewness level (6)

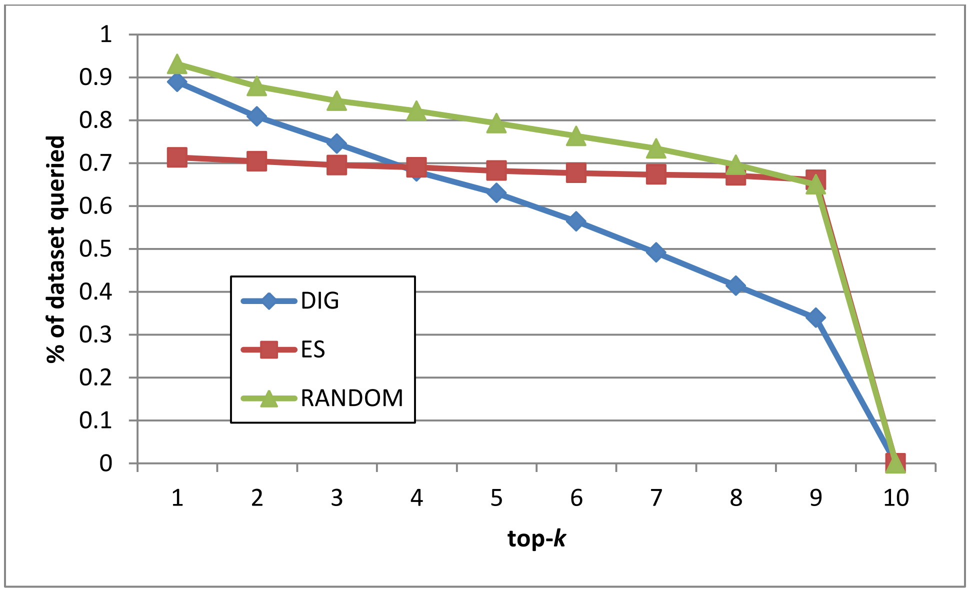


Figure 34: Heuristics comparison for top-*k* on the Netflix dataset

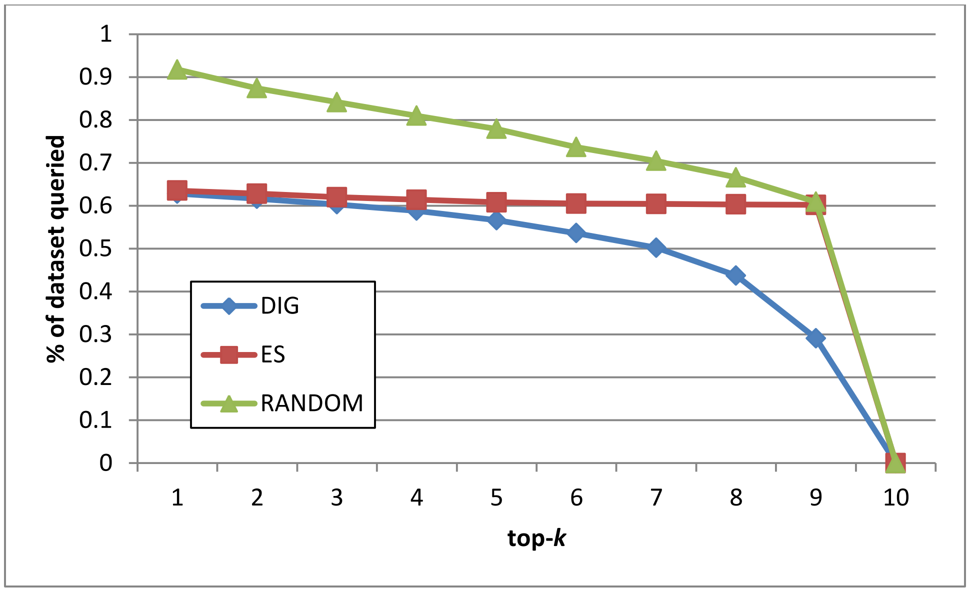


Figure 35: Heuristics comparison for top-*k* on the Sushi dataset

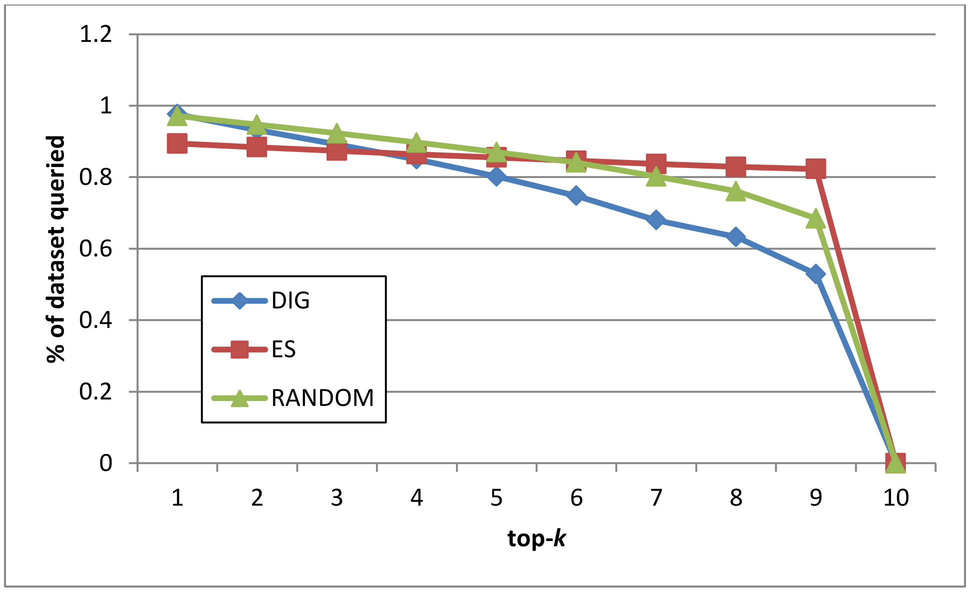


Figure 36: Heuristics comparison for top-*k* on the Pubs dataset

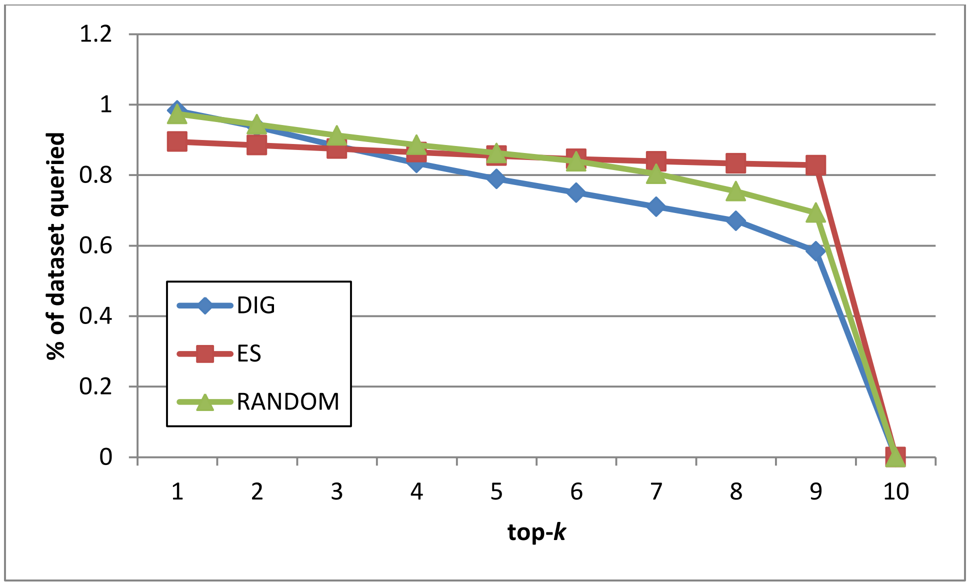


Figure 37: Heuristics comparison for top-$k$ on the Restaurants dataset

### 6.3.2 Approximation

We examined the amount of queries required under different confidence levels (Figure 38 and Figure 39), when a definite winner ($k = 1$) is required. For the simulated data, we set the skewness level to neutral (0). The results presented here are for the Majority strategy, while a comparison between the two aggregation strategies is presented in the next section. We also examine the accuracy of the approximations.

Axis x presents the required confidence level; from 50% to 100% (100% is a definite winner). Axis y presents the percentage of the dataset queried in order to terminate and find the top-$k$ items. For the simulated data, there is a steady increase in the required amount of queries (Figure 38) for all heuristics. DIG outperforms ES and RANDOM, while RANDOM is the least performer. The steady increase in the amount of queries for the simulated dataset and for the Sushi dataset (Figure 40), Pubs dataset (Figure 41) and Restaurants dataset (Figure 42) can be easily explained since more queries are needed in order to gain more information for a higher accuracy level. However, the results for the Netflix dataset behave differently and require a deeper explanation.

For the Netflix data (Figure 39), the increase in the required amount of queries is small for confidence levels 50%-95%. However, there is a big jump in the required number of queries when the desired confidence is 100% (a definite winner is required): from ~10 required queries to achieve a confidence level of 95%, to ~90 queries for a 100% confidence. The probability distributions for the Netflix dataset are estimated, whereas for the simulated data we have accurate (simulated) distributions. We show the probabilities accuracy for the datasets: simulated data with skewness level (0), Netflix and Sushi in Figure 43, Figure 44 and Figure 45 respectively. Axis x is the iteration number and axis y is the probability that the winner is indeed within the top-$k$ items. In this case, $k = 1$. For the simulated data (Figure 43) the probability accuracy increases steadily as more information, acquired in the iterations, becomes available. On the other hand, since the Netflix, Pubs and Restaurants probabilities are estimations, there is more noise until a 95% probability is reached (Figure 44). The Sushi dataset also contains probability estimations, but the estimation is more accurate (Figure 45). To conclude, when the probability estimation is accurate, there is linear relationship between the number of required queries and the approximation level. However, an inaccurate probability distribution results in a "jump" when the required confidence is a 100%.

For all datasets, the Friedman Aligned Ranks test with a confidence level of 95% rejected the null-hypothesis that all heuristics perform the same. The Bonferroni-Dunn test concluded that DIG and ES significantly outperform RANDOM at a 95% confidence level.

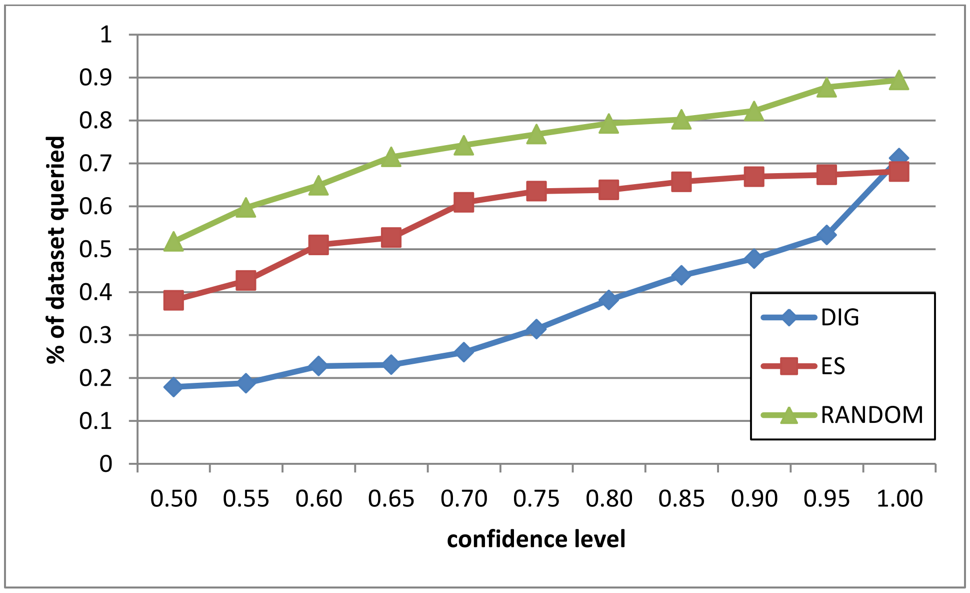


Figure 38: Approximations with simulated data with skewness (0)

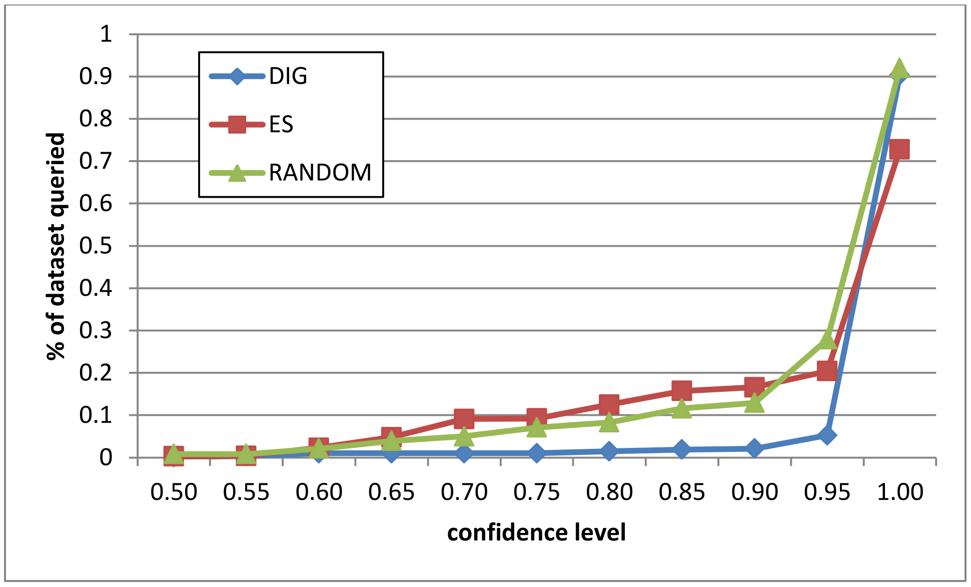


Figure 39: Approximations on the Netflix dataset

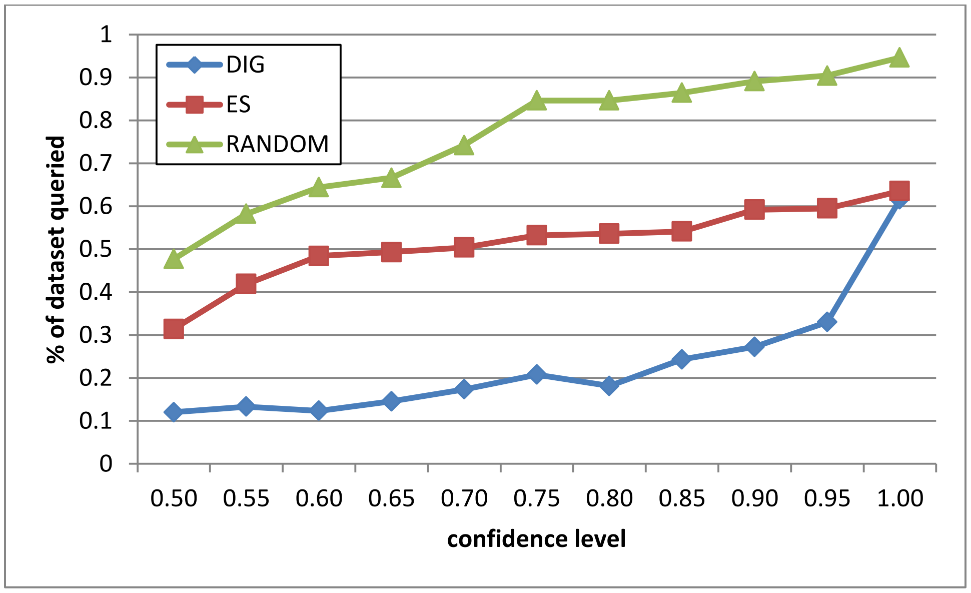


Figure 40: Approximations on the Sushi dataset

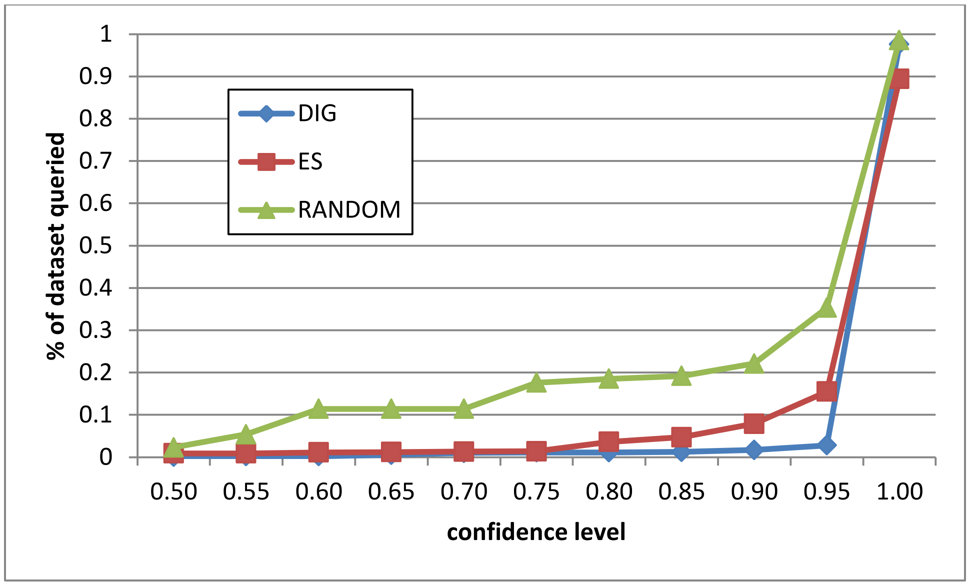


Figure 41: Approximations on the Pubs dataset

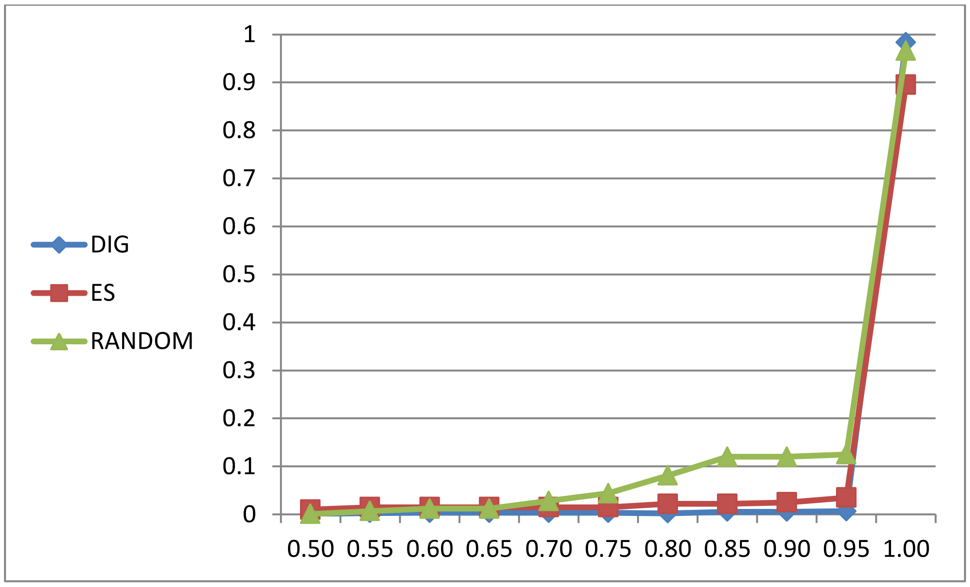


Figure 42: Approximations on the Restaurants dataset

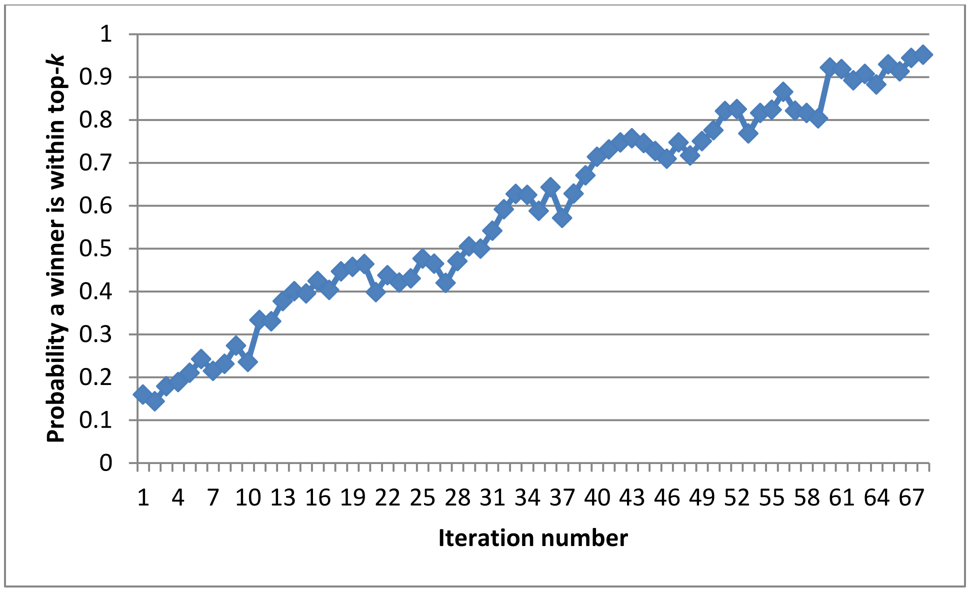


Figure 43: Simulated data: the probability the winner is within top-*k*

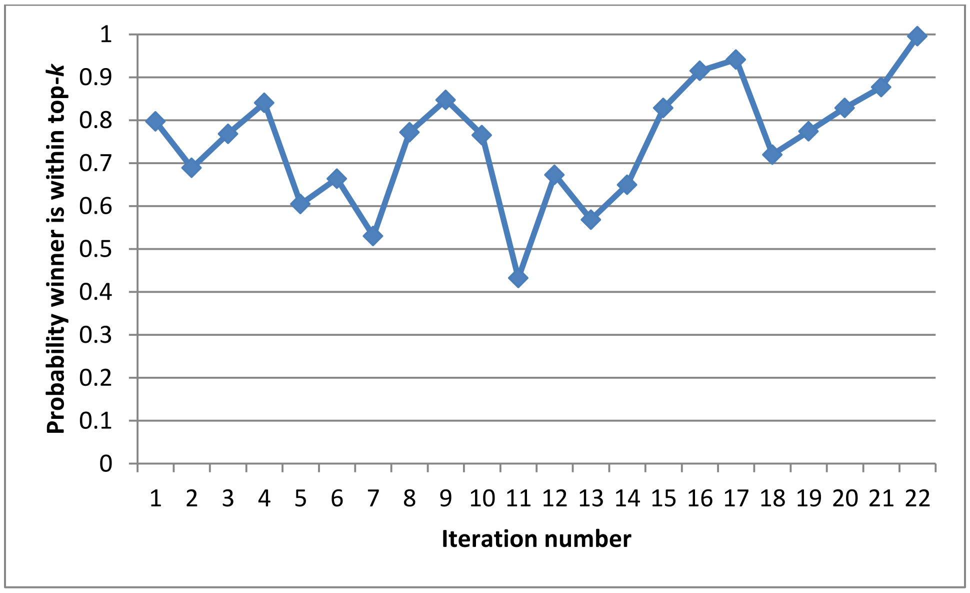


Figure 44: Netflix data: the probability the winner is within top-*k*

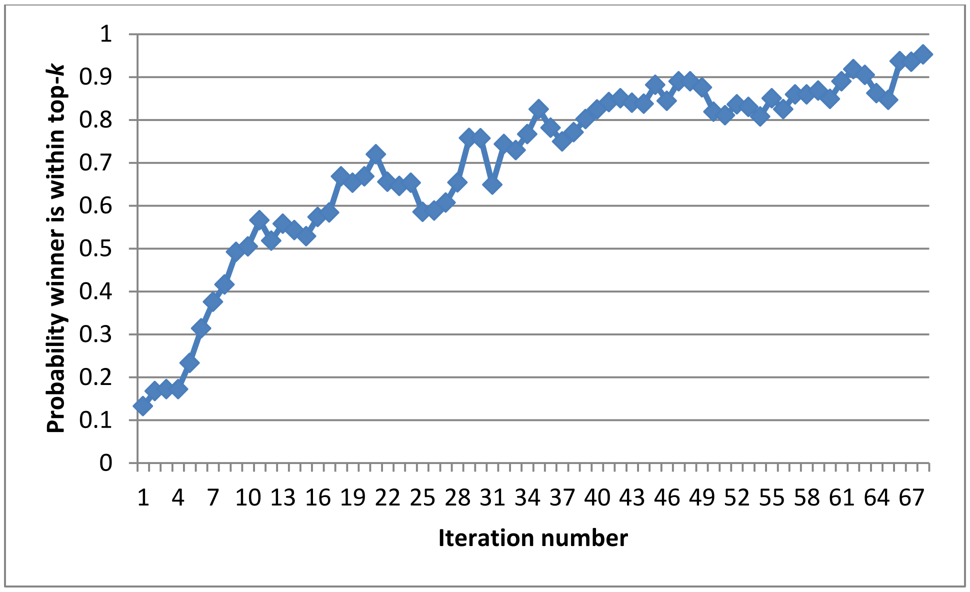


Figure 45: Sushi data - probability winner is within top-*k*

Another interesting question is whether the confidence level results are accurate. A confidence level refers to the percentage of all possible samples that can be expected to include the true population parameter. The confidence level $(1-\propto)$ is accurate, if the winner is indeed within the top-$k$ items in $(1-\propto)\%$ of the experiments. We analyzed the accuracy for the DIG algorithm (since it proved to be the best algorithm for approximation settings) for different confidence levels for $k = 1$. Table 13 presents the percentage of the experiments that held the winner, out of 10 experiments. As previously shown, since the estimation of the probability distribution of Netflix, Pubs and Restaurants datasets is less accurate, the results for Netflix are less accurate. The accuracy is effected by the bias in the user rating and is beyond the scope of this research. See (Koren and Sill2011) for further details on treating bias.

Table 13: Confidence Level Test

| Confidence Level | Simulated Data | Netflix Data | Sushi Data | Pubs Data | Restaurants Data |
|---|---|---|---|---|---|
| 0.5 | 60% | 50% | 80% | 30% | 70% |
| 0.55 | 60% | 50% | 80% | 30% | 90% |
| 0.6 | 60% | 50% | 70% | 30% | 80% |
| 0.65 | 60% | 50% | 80% | 20% | 80% |
| 0.7 | 80% | 50% | 70% | 40% | 80% |
| 0.75 | 90% | 50% | 90% | 30% | 80% |
| 0.8 | 100% | 60% | 90% | 40% | 80% |
| 0.85 | 100% | 60% | 90% | 40% | 80% |
| 0.9 | 100% | 60% | 90% | 40% | 80% |
| 0.95 | 100% | 70% | 90% | 60% | 80% |

### 6.3.3 Aggregation

We evaluated the two strategies: Majority (MAJ) and Least Misery (LM) on the DIG (Figure 46 and Figure 47) and ES heuristics (Figure 48 and Figure 49) for simulated data with different skewness levels: -6, 0, 6. Axis x presents the required top-$k$ items and axis y presents the percentage of the dataset queried. DIG and ES with MAJ perform the same for skewness levels 0 and 6, but it is better when the skewness is -6. However, for the DIG and ES with LM, skewness levels have no significant effect on the performance since skewness does not indicate the quantity of low scores in the dataset, and the low scores are exactly the issue that needs to be considered in LM.

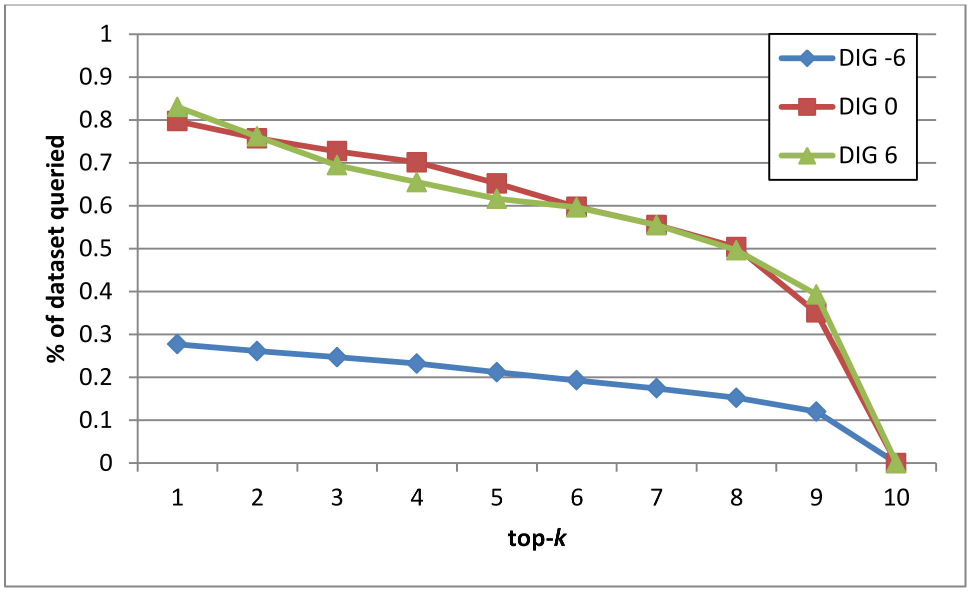


Figure 46: DIG with Majority (MAJ) strategy different skewness levels

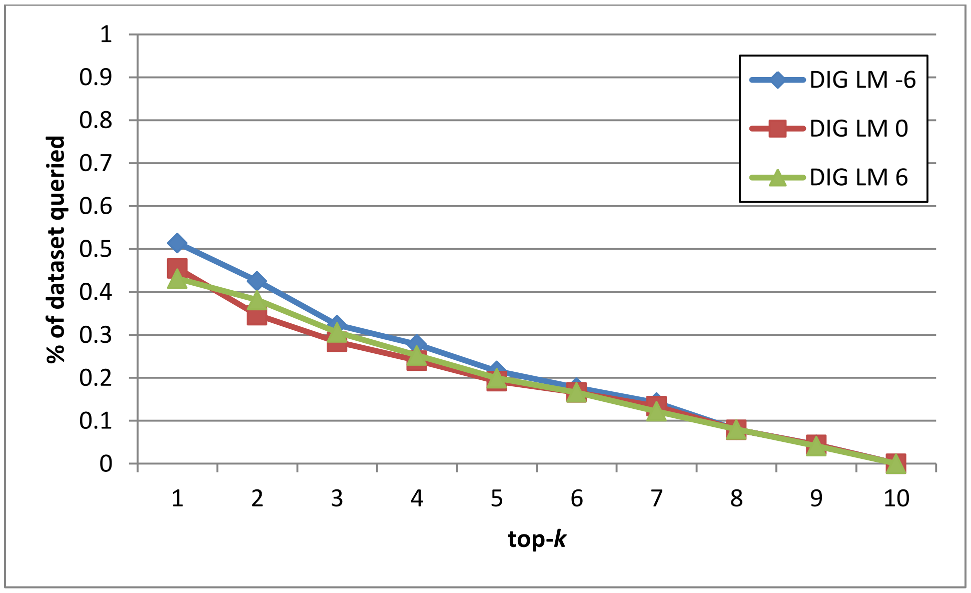


Figure 47: DIG with Least Misery (LM) strategy different skewness levels

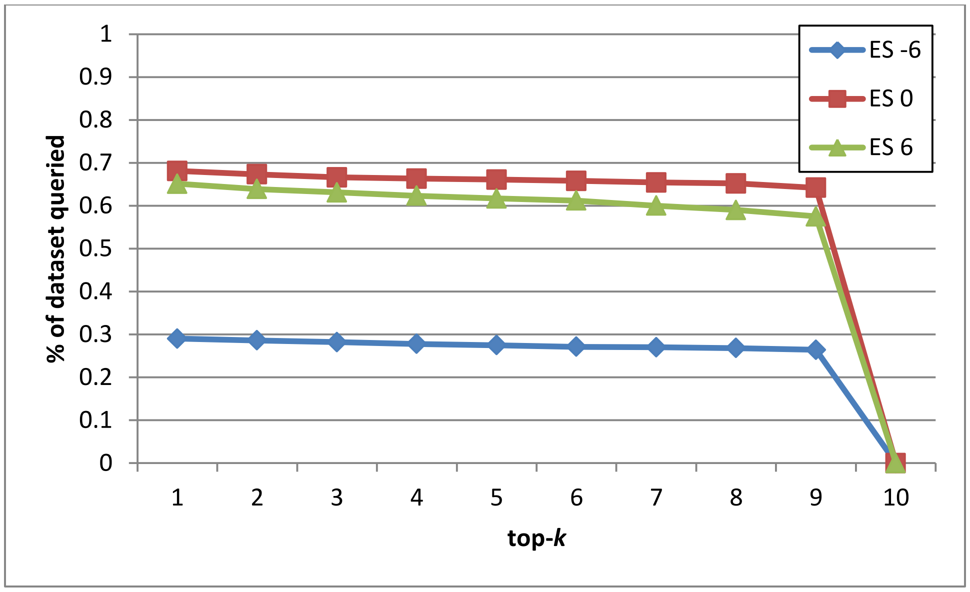


Figure 48:  ES with Majority (MAJ) strategy different skewness levels

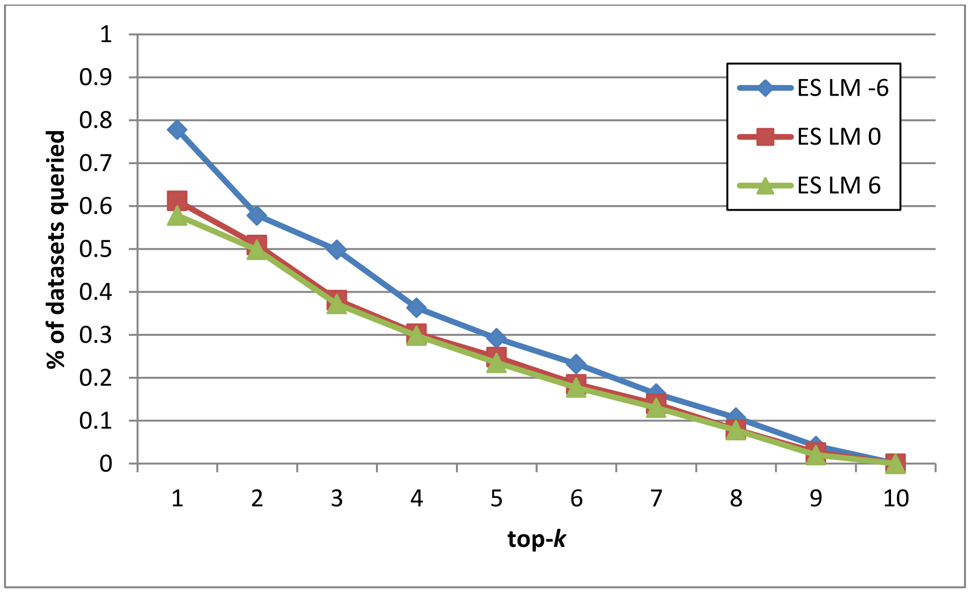


Figure 49: ES with Least Misery (LM) strategy different skewness levels

A comparison between DIG with MAJ and DIG with LM on simulated data on skewness level -6 (Figure 50) and on skewness level 0 (Figure 51) reveals that the LM strategy outperforms MAJ in situations such as these: in a uniform skewness (skewness level 0) and in $k > 4$ in skewness level -6. This can be explained by the fact that in a setting that is not skewed towards a certain candidate (i.e., any setting apart from -6), there might be more users that voted

"1" therefore, LM uses a tie-break to terminate. Thus, LM requires fewer queries in this situation. In the Netflix dataset (Figure 52) MAJ outperforms LM, further indicating the fact that LM has no additional value when there is no skewness towards a certain winner. Similarly, on the Sushi dataset (Figure 53), MAJ outperforms LM when $k < 5$ and then the trend changes and LM outperforms MAJ. On the pubs and restaurant datasets (Figure 54 and Figure 55) LM outperforms MAJ for both heuristics. These results might be explained by the data skewness.

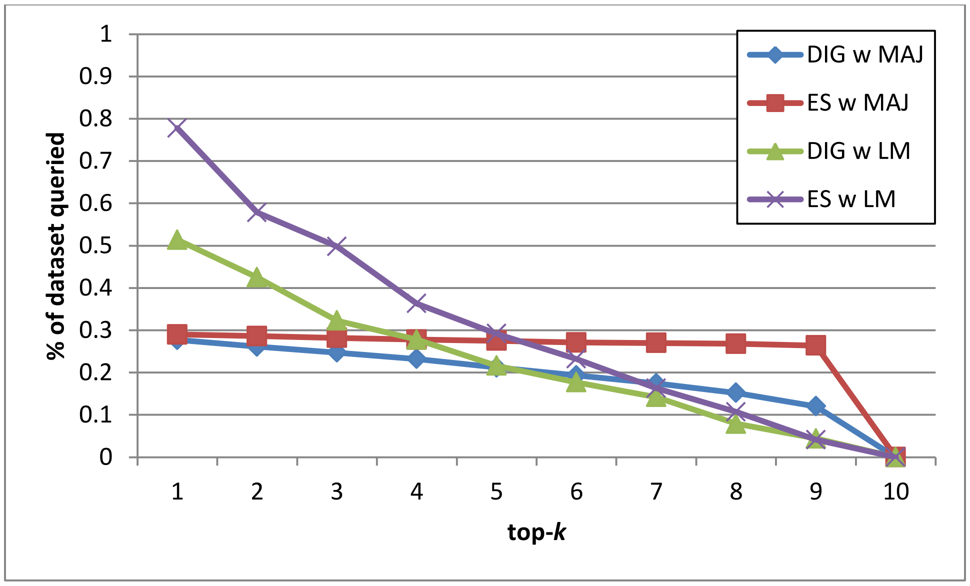


Figure 50: DIG with MAJ and DIG with LM on simulated data on skewness level -6

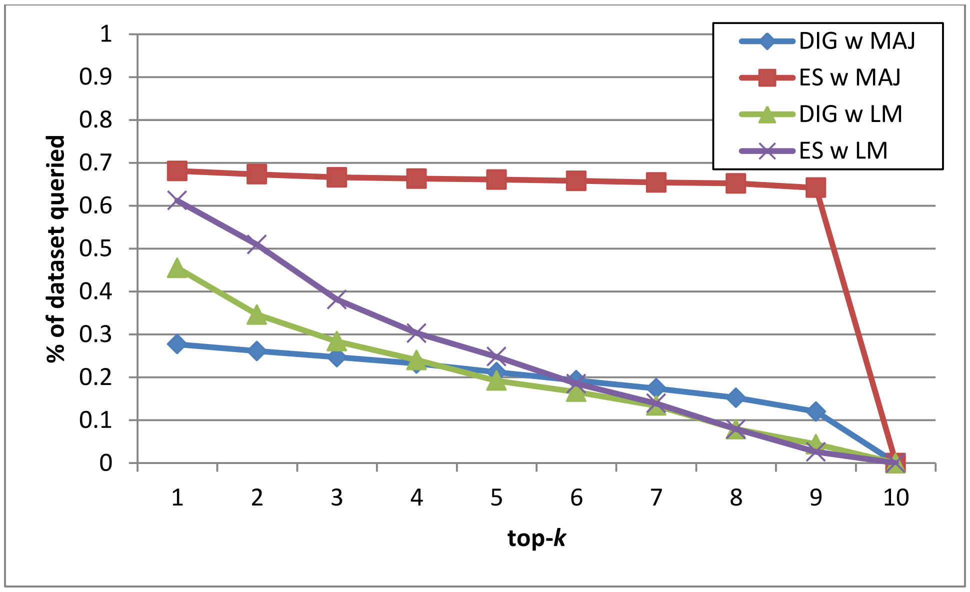


Figure 51: Skewness 0

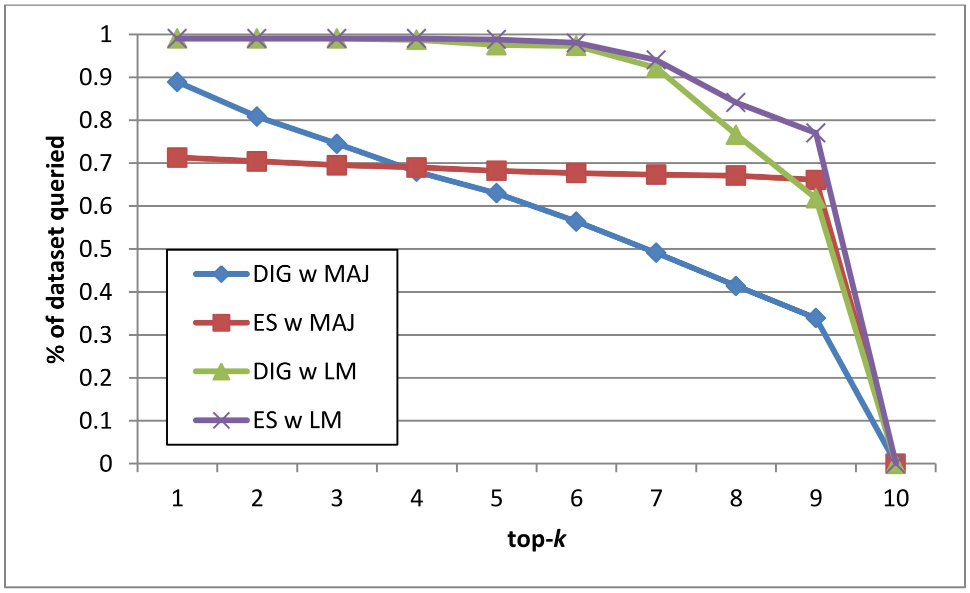


Figure 52: Netflix dataset: strategies comparison, top-*k*

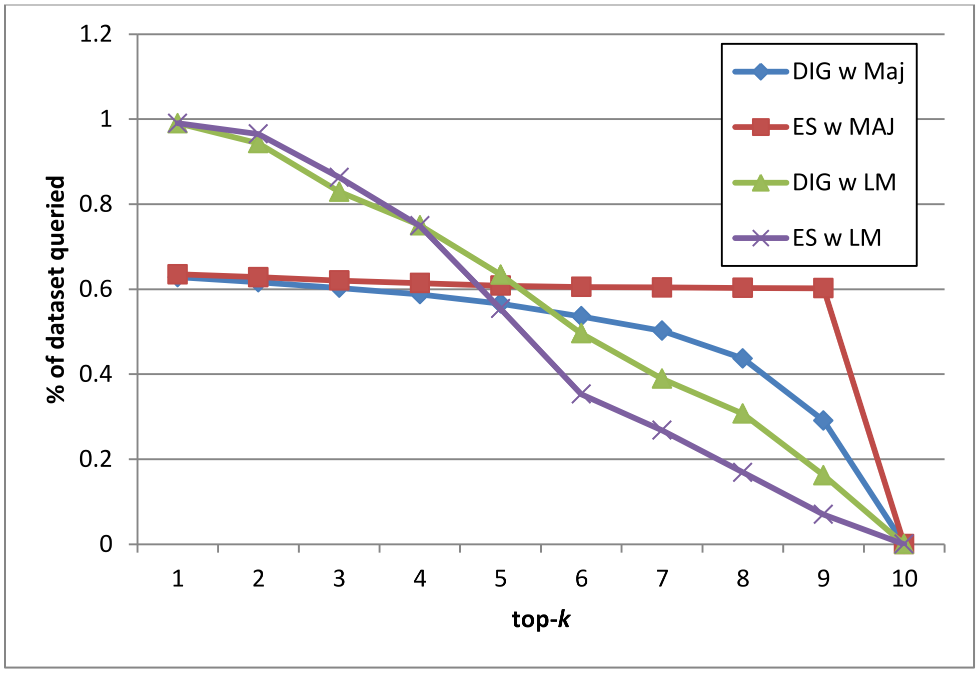


Figure 53: Sushi dataset: strategies comparison, top-*k*

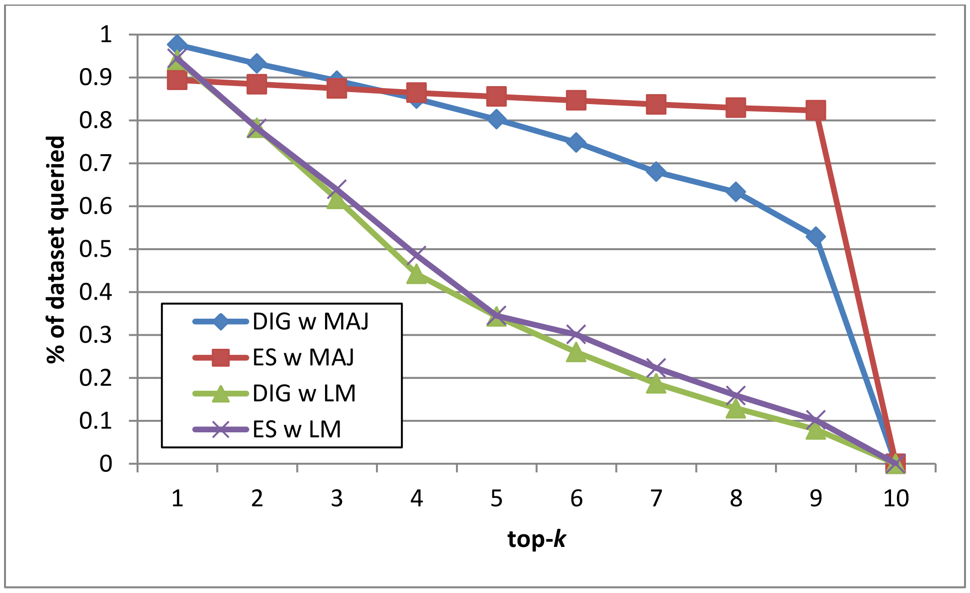


Figure 54: Pubs dataset: strategies comparison, top-*k*

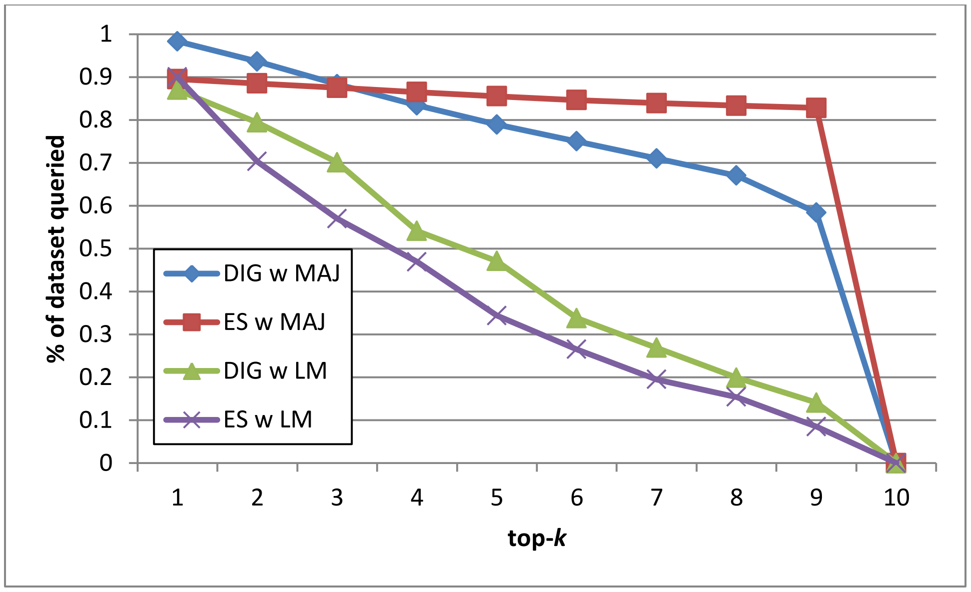


Figure 55: Restaurants dataset: strategies comparison, top-*k*

We evaluated MAJ and LM with respect to the approximation termination condition, with a constant value of $k = 1$ on the datasets: Netflix, Sushi, Pubs, and Restaurants (Figure 56 Figure 57, Figure 57 and Figure 58). Axis x presents the required confidence level and axis y presents the percentage of the dataset queried. There is no significant difference between MAJ and LM for DIG on the Netflix, Pubs, and Restaurants dataset. For ES, on the other hand, MAJ outperforms LM. This is since ES heuristic does not accommodate any consideration of Least Misery, as it always seeks for the item expected to win, and does not consider the least preferred items. The same results for ES are found on the Sushi dataset (Figure 57). However, for DIG on the Sushi dataset, LM outperforms MAJ for confidence levels 50%-95%. For confidence level 100%, MAJ outperforms LM. Namely, for one definite winner the system's entropy can be reduced faster for the Majority aggregation strategy than for the Least Misery strategy probably since Least Misery requires more queries in order to validate that none of the users are miserable.

For all datasets, the Friedman Aligned Ranks test with a confidence level of 90% rejected the null-hypothesis that all heuristics perform the same for different approximation levels. We did not execute the Bonferroni-Dunn test since there is not one algorithm that is preferred over the others.

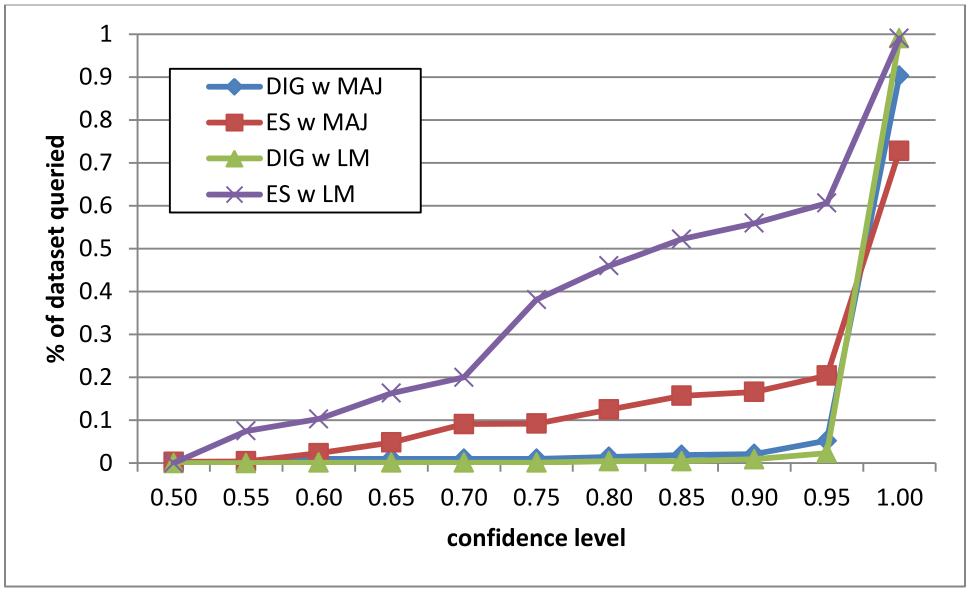


Figure 56: Netflix dataset: strategies comparison, approximation

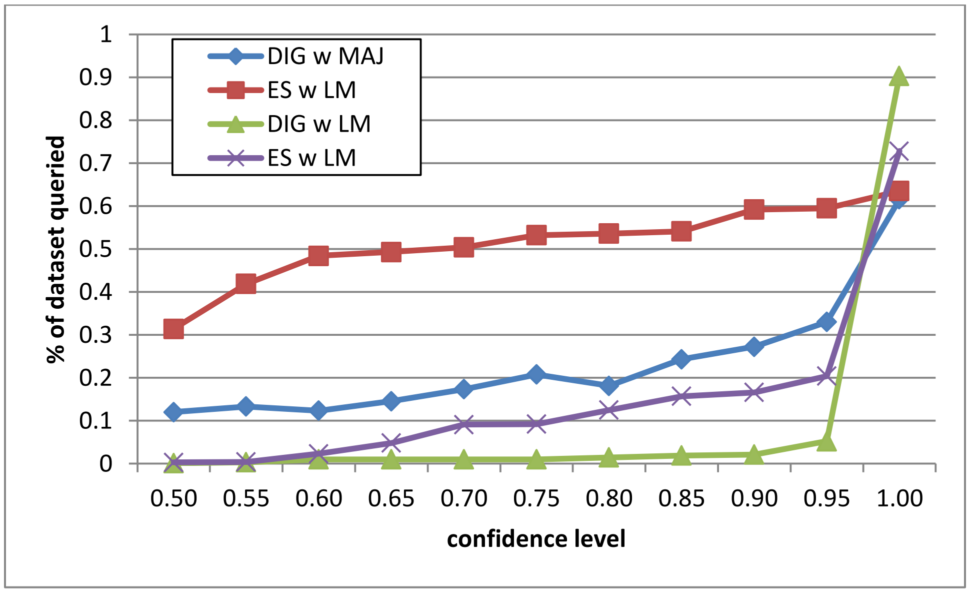


Figure 57: Sushi dataset: strategies comparison, approximation

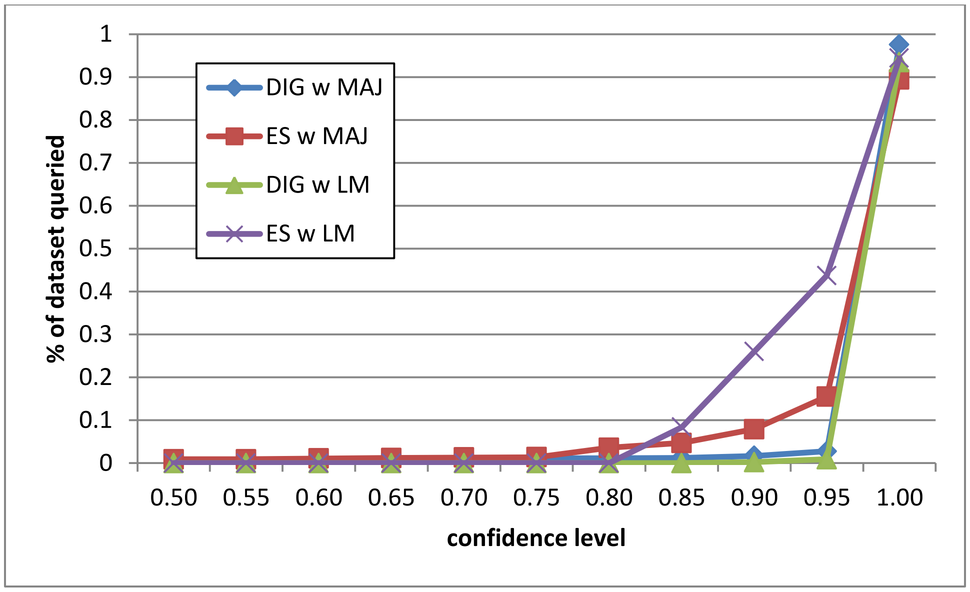


Figure 58: Pubs dataset: Strategies comparison, approximation

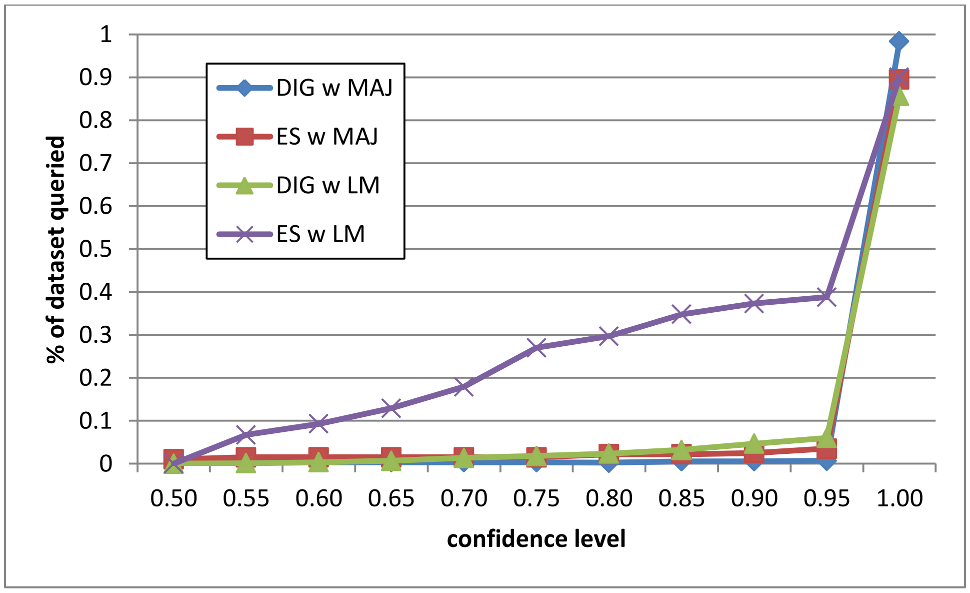


Figure 59: Restaurants dataset: strategies comparison, approximation

## 6.4 Discussion

In this chapter we suggested the consideration of the aggregation strategy and the termination conditions when attempting to reduce preference elicitation communication cost. We examined two termination conditions: *selection* and *approximation*. The first condition, *selection*, returns top-$k$ items where one of them is the winning item rather than just one ($k = 1$) definite winning item. The second termination condition, *approximation*, returns top-$k$ items with some confidence level $\alpha$ ($0 \leq \alpha \leq 1$), rather than top-$k$ items where one of them is the definite winner ($\alpha = 1$). Furthermore, we examined the Least Misery aggregation strategy and the Majority aggregation strategy.

The final goal of this chapter was to employ *selection*, *approximation* and *aggregation* in order to reduce the amount of queries needed during a preference elicitation process for a group of users that want to reach a joint decision. We focused on the Range voting protocol as it is very commonly applied for recommender systems. We implemented two heuristics whose primary aim is to minimize preference elicitation: DIG and ES. These are the only two publicly available heuristics that aim at reducing preference elicitation for the Range voting protocol. We performed an experimental analysis on two real-world datasets: the Sushi dataset (Kamishima et al. 2005) and the Netflix prize dataset (http://www.netflixprize.com). In order to analyze possible skewness levels in data, we simulated data with different skewness levels. We also estimated user-item probability distribution for all datasets. Lastly, we evaluated 2 datasets generated through a user-study.

In general, we showed that selecting the suitable aggregation strategy and relaxing the termination condition can reduce communication cost up to 90%. We also showed the benefits of the DIG heuristic for reducing the communication cost. In 0 we concluded that in most cases the ES heuristic outperforms the DIG heuristic. The ES heuristic focuses on identifying the current local maximum and queries the user that maximizes this item. The DIG heuristic focuses on reducing the system entropy. In this chapter we revealed that when the termination conditions are relaxed, DIG takes the lead.

We examined how the number of required queries is effected by the request to (1) return one definite winner, and (2) return top-$k$ items. In the latter case, the group members are left with $k$ items to select from (selection termination condition). With respect to the selection condition, there is an inverse linear connection: as $k$ is larger the amount of required queries is reduced. Only when the dataset is skewed towards a certain winner item, and also $k$ is set to $0 \leq k \leq 3$, does ES outperform DIG. This observation assists to determine the conditions in which each of these heuristics should be employed. Also, we can now state that, as expected intuitively, in cases where the group members are willing to accept a set of items rather than one winning item, the communication cost is reduced. For example, if a group's wish to select a movie can be satisfied with the system offering them a choice of top-3 movies rather than the system determining one movie for them, less queries to group members will be executed.

We studied (1) the tradeoff between finding the optimal winner and thus having an accurate result, and (2) the number of queries required for the process. For the approximation termination condition, we showed that the amount of required queries increases proportionally to the confidence level. We showed that DIG and ES can output accurate approximate recommendations. However, the accuracy is derived from the dataset's probability distribution accuracy. When the probability distribution is known or is estimated accurately, the recommendations are more accurate.

With respect to the aggregation strategy, we showed that the Majority strategy does not always outperform the Least Misery strategy. It is reasonable to assume that the strategy will be set according to the users' preferences and not according to the data. We demonstrated the feasibility of choosing either strategy on the datasets.

# Chapter 7

# Conclusions and Future Work

In this chapter we summarize and discuss our work, and finally offer directions for future research.

## 7.1 Summary

This study addresses the issue of preference elicitation for group decision making using voting rules. We presented a general, domain-free framework for preference management for groups, where the goal is to minimize the communication cost. We studied preference elicitation under the non-ranking (Range) and ranking (Borda) voting protocols. The goal of the preference elicitation process is to return a winning item while minimizing the communication costs.

We suggested an interactive incremental framework whose process consists of querying one member of the group at each step for either her rating for one item (user-item query) or for her preference between two items (user-item-item query). At each step the users' preference distributions are updated and a new query is found. We have suggested two approaches for heuristics that determine what query to select next (i.e., which group member to query regarding what item or items). One approach focuses on reducing the entropy of the winner in the system. The rationale behind this proposal is that reducing the entropy quickly will lead to the winner using a minimal amount of queries. The other approach focuses on maximizing the score of the item with the highest current score; under the same rational that expects to discover the winner item in minimal time. Both heuristics rely on probabilistic rating distributions. We have shown how these distributions can be estimated. The rating distributions are updated iteratively, allowing their accuracy to increase over time.

Although outputting a definite winner is the most accurate result, we also examined the effort-accuracy tradeoff and aggregation strategies for group preference elicitation. First, we suggested shortening the preference elicitation process by returning $k$ alternatives to the group members rather than returning just one item. Users might prefer to receive a few options rather than just one; so that if a chosen option is unavailable they can switch to another option without triggering more rounds of preference elicitation. Secondly, we suggested computing approximate winner or winners with some confidence level. On one hand, receiving an approximate winner item is less accurate than a definite winner, but on the other hand it further reduces the communication cost. Lastly, we suggested considering the aggregation strategy when combining the user preferences. We have shown that the aggregation strategy affects the communication cost required of the preference elicitation and compared two state-of-the-art aggregation strategies: the Majority based strategy and the Least Misery strategy.

We demonstrated the effectiveness of our framework by evaluating the heuristics on four real-world datasets. In addition, we examined possible effects of various data characteristics utilizing simulated datasets and manipulating their data parameters.

## 7.2 Discussion

We briefly present our main findings, and move on to describe their relations with the overlapping fields of social choice and of recommender systems. Lastly, we discuss the limitations of our work.

### 7.2.1 Main Findings

The main empirical findings were summarized within the respective chapters that describe: Preference elicitation using the Range voting protocol (Chapter 4), Preference elicitation using the Borda voting iterative voting (Chapter 5) and tradeoffs and aggregation strategies in preference elicitation (Chapter 6). The main empirical and theoretical findings are the following:

(a) **Preference elicitation using the Range voting protocol:** We have shown that heuristics can reduce the communication cost required for voting under the Range protocol by more than 50%. Different heuristics perform better under different settings: the DIG heuristic performs better when the voter-item distributions are not skewed towards a specific winner. The ES heuristic outperforms DIG when some pattern is found in the data. Furthermore, we have shown that updating the voter-item distributions increases the heuristics performance. The creation of the voter-item distribution is more accurate when created from a smaller (100x100) set of historical ratings than from a bigger set (1000x1000) of historical ratings.

(b) **Preference elicitation using the Borda voting protocol:** We have shown that the expected score for Borda heuristic (ESB) reduces the communication cost by more than 60%, since it focuses directly on finding a winner. The entropy based method for Borda (IGB) fails to reduce communication. The voter permutations probabilistic model cannot scale up, thus being a major disadvantage.

(c) **Tradeoffs and aggregation strategies in preference elicitation:** We have demonstrated that selecting the suitable aggregation strategy and relaxing the termination condition can reduce communication cost by up to 90%. When the termination conditions are relaxed, the entropy-based approximation method (DIG) takes the lead over the expected maximum heuristic (ES). We illustrated that DIG and ES can output accurate approximate recommendations. However, the accuracy is derived from the dataset's probability distribution accuracy. When the probability distribution is known or is estimated accurately, the winner approximation is more accurate.

To conclude, we have examined voting elicitation under the Range and Borda voting protocols, representing rating and ranking of items. Rating items is a task users are familiar with; it is used abundantly, websites being one example. However ranking items might be easier for users, specifically by the method we present where two items are presented to the user and the user states her preference between the two. Our framework allows the system administrator or the group members themselves to choose which task they prefer, rating or ranking, and the voting protocol is determined accordingly.

Our framework is domain free and can be used in any domain where users need to reach a joint decision. We have evaluated our framework on real-world datasets and on simulated datasets. The simulated datasets are important since they allow us to manipulate the dataset parameters and to examine data with different skewness of item preferences and thus simulate different circumstances. When running the framework on a new domain, if historical data exists (such as users' ratings to other items), we can find the skewness pattern and use the heuristic that best fits the data. For example, when given a new domain and a request to use ratings (and not rankings), if we see that the data is uniformly skewed, we will suggest to use the DIG heuristic.

### 7.2.2 Impact on Social choice and on Recommender Systems

Our findings append to a growing body of literature on preference elicitation using voting rules (Lu and Boutilier 2011; Kalech et al. 2011). Our research adds a unique contribution to preference elicitation in social choice in a number of perspectives that have previously been overlooked. First, we have studied preference elicitation using two different protocols: a ranking protocol (represented by the Borda protocol) and a non-ranking protocol (represented by the Range protocol). Previous research has focused only on the Borda protocol. However, ranking is worth considering since it is abundant and often used by different applications such as netflix.com and booking.com. Secondly, we have suggested various methods for reducing the amount of queries. In addition to heuristics which offer a necessary winner item, we have suggested (a) to return a list of top-$k$ items where one of them is the necessary winner; and (b) to approximate the necessary winners or top-$k$ items. These methods offer a decrease in the required amount of queries and have not been previously suggested. Finally, we examined the effect of aggregating the preferences in other strategies but the Majority based strategy. The Least Misery strategy is often needed in real-life scenarios yet has previously been overlooked (e.g., a group looking for a dining location may wish to avoid a fish restaurant if one of the group members dislikes fish).

From the recommender systems domain perspective, this study suggests a framework for preference elicitation that can be used as a second step procedure in group recommenders: to narrow down the predicted items list and present the group of users with definite or approximate necessary winners. Group recommender systems often focus on improving the systems accuracy and usually return a prediction to the group and not definite winning items. A group recommender system can process thousands of candidate items and return a list of top-$N$ items predicted as the most suitable to the group. We can enhance this by eliciting user preferences on these $N$ items and return a definite winner or top-$k$ items $(k \leq N)$ where one of the items is the winner or an approximate winner with some confidence level. This contribution may add to the usability of a group recommender system offering a platform that enables reaching a joint decision with minimal effort.

### 7.2.3 Limitations

As a direct consequence of this study, we encountered a number of limitations, which need to be considered:

a. Initial assumptions – this study assumes that the user always provides an answer to the query, independence of rating and equal communication cost. These limitations can be overcome by tweaking the model. For example, it is possible to model the probability that the user will answer the query. For a small number of voters and items it is possible to consider dependent probabilities. The communication cost be modeled as a weighted vector and added to the model.
b. Distribution accuracy - under both protocols, probabilistic data regarding the users' preferences were computed. For the Range protocol, voter-item distributions were computed. The accuracy of the distributions was found to influence the performance of the heuristics. The approximated distribution for the Netflix dataset was not accurate, as discussed in the evaluation section in Chapter 4. This limitation can be overcome by investing more time in researching ways to consider rating bias. However this subject was out of the scope of our research.
c. Model scaling - for the Borda voting, perhaps the main disadvantage of the presented framework is its lack of scalability, due to the need to hold a probabilistic model of all order permutations of items. Therefore while the number of voters can be increased, the number of items cannot be increased beyond 10 using standard computational power. This limitation can be partly overcome by relaxing the need to hold all permutations, thus trading off accuracy for less complexity. As discussed in section 5.2, we followed previous research and chose to hold all permutations in order to receive an accurate model.
d. Other aggregation strategies - the current study examined the two aggregation strategies most common in the literature. Extension to other available aggregation strategies does not require a fundamental change since the heuristics and the model do not change. The heuristics performance under

different strategies can be analyzed, but this also is out of the scope of our research.

e. Other voting protocols - the current study examined two voting protocols. Extending the study to other voting protocols is pretty straight forward since the only difference is the way the maximum and minimum score is computed, as discussed in Chapter 3.

## 7.3 Future Work

Further research might be conducted in order to address the limitations listed above, as well as other directions:

a. Distribution accuracy – the more accurate the initial distribution is, the fewer queries are needed in order to find a winner. We expect that further improving the accuracy of the Range protocol voter-item distribution, will lead to a decrease in the communication costs and therefore worthy of investigation. With regard to the Borda protocol, relaxing the accuracy of the distribution (i.e., a relaxation in the need to hold all permutations), will allow us to scale up the number of items, the tradeoff being an increase in the communication costs.
b. Bias in user feedback – in this work we followed (Koren and Sill2011) and corrected bias in user ratings when computing the rating distributions (section 4.2). However, this does not cover all possible bias. It has been shown that users have different rating patterns (Kuflik et al. 2012), and further research can hopefully plan and integrate a domain-free algorithm that considers user rating patterns.
c. Other aggregation strategies – different aggregation strategies are used for different purposes in the recommender systems domain. Investigating them could be of value.
d. Social networks – instead of deriving the probability distributions from historical data, the distributions can be computed from the social network to which the user belongs. Social networks typically demonstrate homophily, which is the tendency of individuals to bond with similar others. Hence it is probable that connected

individuals would share similar preferences. This phenomenon is often expressed in the phrase "birds of a feather flock together". Thus we assume that an individual's preferences may be derived from the social network topology.

e. User-study – the framework we suggested can be implemented as a real system. Then, a user study can be conducted, and the user satisfaction can be evaluated.
f. Approximations and the Borda voting protocol – this study presented winner approximations for the Range voting protocol. Approximations for the Borda protocol should be defined and evaluated in future research.

# Bibliography

Arrow, K. J. 1951. *Social Choice and Individual Values*. 2nd Edition 1963 ed. New Haven: Cowles Foundation.

Bachrach, Y., Betzler,N., and Faliszewski,P. .2010. Probabilistic Possible Winner Determination.*In Proceedings of The Twenty-fourth AAAI conference on Artificial Intelligence (AAAI), Atlanta, GA, USA,*.

Balakrishnan, S. and S. Chopra. .2012. Two of a Kind Or the Ratings Game? Adaptive Pairwise Preferences and Latent Factor Models. *Frontiers of Computer Science,* 6 (2): 197-208.

Baltrunas, L., Makcinskas,T., and Ricci,F. .2010. Group Recommendations with Rank Aggregation and Collaborative Filtering.*In Proceedings of the fourth ACM conference on Recommender systems*,119-126.ACM, .

Bellman, R. .1962. Dynamic Programming Treatment of the Travelling Salesman Problem. *Journal of the ACM (JACM),* 9 (1): 61-63.

Berkovsky, S. and Freyne,J. .2010. Group-Based Recipe Recommendations: Analysis of Data Aggregation Strategies.*In Proceedings of the fourth ACM conference on Recommender systems,*111-118. Barcelona:ACM, .

Betzler, N., S. Hemmann, and R. Niedermeier. .2009. A Multivariate Complexity Analysis of Determining Possible Winners Given Incomplete Votes. *Proc.of 21st IJCAI,* 2 (3): 7.

Betzler, N., Niedermeier,R., and Woeginger,G. J. .2011. Unweighted Coalitional Manipulation Under the Borda Rule is NP-Hard.*In Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume One,*55-60.AAAI Press, .

Betzler, N., A. Slinko, and J. Uhlmann. .2014. On the Computation of Fully Proportional Representation. *arXiv Preprint arXiv:1402.0580,*.

Boutilier, C., Lang,J., Oren,J., and Palacios,H. .2014. Robust Winners and Winner Determination Policies Under Candidate Uncertainty.*In Artificial Intelligence (AAAI),* Quebec, Canada:.

Brandt, F., Conitzer,V., and Endriss,U. 2013. "Computational Social Choice." Chap. 6, In *Multiagent Systems*, edited by Gehard Weiss. 2nd edition ed., 213-283: MIT Press. http://mitpress.mit.edu/books/multiagent-systems-1.

Braziunas, D. and C. Boutilier. .2009. Elicitation of Factored Utilities. *AI Magazine,* 29 (4): 79.

Breese, J. S., Heckerman,D., and Kadie,C. .1998. Empirical Analysis of Predictive Algorithms for Collaborative Filtering.*In Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence,*43-52.Morgan Kaufmann Publishers Inc., .

Carvalho, Lucas Augusto Montalvão Costa and Macedo,H. T. .2013. Users' Satisfaction in Recommendation Systems for Groups: An Approach Based on Noncooperative Games.*In Proceedings of the 22nd international conference on World Wide Web companion,*951-958.International World Wide Web Conferences Steering Committee, .

Chen, L. and P. Pu. .2012. Critiquing-Based Recommenders: Survey and Emerging Trends. *User Modeling and User-Adapted Interaction,* 22 (1-2): 125-150.

Chen, S., T. Lin, and L. Lee. .2014. Group Decision Making using Incomplete Fuzzy Preference Relations Based on the Additive Consistency and the Order Consistency. *Information Sciences,* 259: 1-15.

Conitzer, V. and Sandholm,T. .2005. Communication Complexity of Common Voting Rules.*In Proceedings of the 6th ACM conference on Electronic commerce,*78-87.ACM, .

Conitzer, V. .2009. Eliciting Single-Peaked Preferences using Comparison Queries. *Journal of Artificial Intelligence Research,* 35: 161-191.

Davies, J., Katsirelos,G., Narodytska,N., and Walsh,T. .2011. Complexity of and Algorithms for Borda Manipulation.*In AAAI,*657-662.

de Campos, L. M., J. M. Fernández-Luna, J. F. Huete, and M. A. Rueda-Morales. .2009. Managing Uncertainty in Group Recommending Processes. *User Modeling and User-Adapted Interaction,* 19 (3): 207-242.

Ding, N. and Lin,F. .2013. Voting with Partial Information: What Questions to Ask?*In Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems,*1237-1238.International Foundation for Autonomous Agents and Multiagent Systems, .

Domingos, P. and M. Pazzani. .1997. On the Optimality of the Simple Bayesian Classifier Under Zero-One Loss. *Machine Learning,* 29 (2): 103-130.

Elkind, E., Faliszewski,P., Skowron,P., and Slinko,A. .2014. Properties of Multiwinner Voting Rules.*In Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems,*53-60.International Foundation for Autonomous Agents and Multiagent Systems, .

Endriss, U. and Grandi,U. .2013. Binary Aggregation by Selection of the most Representative Voter.*In Proceedings of the 7th Multidisciplinary Workshop on Advances in Preference Handling,*.

Fishman, G. 1996. *Monte Carlo: Concepts, Algorithms, and Applications* Springer.

Freyne, J., S. Berkovsky, and G. Smith. .2013. Rating Bias and Preference Acquisition. *ACM Transactions on Interactive Intelligent Systems (TiiS),* 3 (3): 19.

Garcia, I., S. Pajares, L. Sebastia, and E. Onaindia. .2011. Preference Elicitation Techniques for Group Recommender Systems. *Information Sciences,* 189: 155-175.

García, S., A. Fernández, J. Luengo, and F. Herrera. .2010. Advanced Nonparametric Tests for Multiple Comparisons in the Design of Experiments in Computational Intelligence and Data Mining: Experimental Analysis of Power. *Information Sciences,* 180 (10): 2044-2064.

Gelain, M., Pini,M. S., Rossi,F., and Venable,K. B. .2007. Dealing with Incomplete Preferences in Soft Constraint Problems.*In Proceedings of the 13th international conference on Principles and practice of constraint programming,*286-300.Springer-Verlag, .

Gibbard, A. 1973. Manipulation of Voting Schemes: A General Result. *Econometrica: Journal of the Econometric Society,*: 587-601.

Goldberg, D., D. Nichols, B. M. Oki, and D. Terry. .1992. Using Collaborative Filtering to Weave an Information Tapestry. *Communications of the ACM,* 35 (12): 61-70.

Gorla, J., Lathia,N., Robertson,S., and Wang,J. .2013. Probabilistic Group Recommendation Via Information Matching.*In Proceedings of the 22nd international conference on World Wide Web,*495-504.International World Wide Web Conferences Steering Committee, .

Hazon, N., Aumann,Y., Kraus,S., and Wooldridge,M. .2008. Evaluation of Election Outcomes Under Uncertainty.*In Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2,*959-966.International Foundation for Autonomous Agents and Multiagent Systems, .

Herrera-Viedma, E., F. Chiclana, F. Herrera, and S. Alonso. .2007. Group Decision-Making Model with Incomplete Fuzzy Preference Relations Based on Additive Consistency. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions On,* 37 (1): 176-189.

Iyengar, S. S. and M. R. Lepper. .2000. When Choice is Demotivating: Can One Desire Too Much of a Good Thing? *Journal of Personality and Social Psychology,* 79 (6): 995.

Jameson, A. .2004. More than the Sum of its Members: Challenges for Group Recommender Systems.*In Proceedings of the working conference on Advanced visual interfaces,*48-54.ACM, .

Jameson, A. and B. Smyth. .2007. Recommendation to Groups. *The Adaptive Web,*: 596-627.

Kalech, M., S. Kraus, G. A. Kaminka, and C. V. Goldman. .2011. Practical Voting Rules with Partial Information. *Journal of Autonomous Agents and Multi-Agent Systems,* 22 (1): 151-182.

Kamishima, T., Kazawa,H., and Akaho,S. .2005. Supervised Ordering-an Empirical Survey.*In Data Mining, Fifth IEEE International Conference on,*4 pp.IEEE, .

Konczak, K. and Lang,J. .2005. Voting Procedures with Incomplete Preferences.*In Nineteenth International Joint Conference on Artificial Intelligence,* Edinburgh, Scotland:.

Koren, Y. and R. Bell. .2011. Advances in Collaborative Filtering. *Recommender Systems Handbook,*: 145-186.

Koren, Y. and Sill,J. .2011. OrdRec: An Ordinal Model for Predicting Personalized Item Rating Distributions.*In Proceedings of the fifth ACM conference on Recommender systems,*117-124.ACM, .

Kuflik, T., Wecker,A. J., Cena,F., and Gena,C. 2012. "Evaluating Rating Scales Personality." In *User Modeling, Adaptation, and Personalization,* 310-315: Springer.

Lang, J., Pini,M. S., Rossi,F., Venable,K. B., and Walsh,T. .2007. Winner Determination in Sequential Majority Voting.*In Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI),*1372-1377.

Lu, T. and Boutilier,C. .2013. Multi-Winner Social Choice with Incomplete Preferences.*In Proceedings of the Twenty-third International Joint Conference on Artificial Intelligence (IJCAI-13),*263-270. Beijing:.

Masthoff, J. .2011. Group Recommender Systems: Combining Individual Models. *Recommender Systems Handbook,*: 677-702.

Masthoff, J. .2004. Group Modeling: Selecting a Sequence of Television Items to Suit a Group of Viewers. *User Modeling and User-Adapted Interaction,* 14 (1): 37-85.

Mattei, Nicholas and Walsh, Toby. 2013. PrefLib: A Library of Preference Data. *Proceedings of Third International Conference on Algorithmic Decision Theory (ADT 2013).* Springer, *Lecture Notes in Artificial Intelligence,* November 13-15, 2013.

McCarthy, J. F. and Anagnost,T. D. .1998. MusicFX: An Arbiter of Group Preferences for Computer Supported Collaborative Workouts.*In Proceedings of the 1998 ACM conference on Computer supported cooperative work,*363-372.ACM, .

McCarthy, K., McGinty,L., Smyth,B., and Salamó,M. 2006. "The Needs of the Many: A Case-Based Group Recommender System." In *Advances in Case-Based Reasoning*, 196-210: Springer.

Naamani Dery, L., M. Kalceh, L. Rokach, and B. Shapira. .2014. <br />Reaching a Joint Decision with Minimal Elicitation of Voter Preferences. *Information Sciences,* 278: 466-487.

Nisgav, A. and B. Patt-Shamir. .2011. Improved Collaborative Filtering. *Algorithms and Computation,*: 425-434.

O'connor, M., Cosley,D., Konstan,J. A., and Riedl,J. .2002. PolyLens: A Recommender System for Groups of Users.*In ECSCW 2001,*199-218.Springer, .

Pfeiffer, T., Gao,X. A., Mao,A., Chen,Y., and Rand,D. G. .2012. Adaptive Polling for Information Aggregation.*In Twenty-Sixth AAAI Conference on Artificial Intelligence,*.

Pini, M. S., F. Rossi, K. B. Venable, and T. Walsh. .2009. Aggregating Partially Ordered Preferences. *Journal of Logic and Computation,* 19 (3): 475-502.

Popescu, G. and Pu,P. 2013. "Group Recommender Systems as a Voting Problem." In *Online Communities and Social Computing*, 412-421: Springer.

Procaccia, A. D., J. S. Rosenschein, and A. Zohar. .2008. On the Complexity of Achieving Proportional Representation. *Social Choice and Welfare*, 30 (3): 353-362.

Pu, P. and L. Chen. .2009. User-Involved Preference Elicitation for Product Search and Recommender Systems. *AI Magazine*, 29 (4): 93.

Regenwetter, Michel, Jason Dana, and Clintin P. Davis-Stober. 2011. "Transitivity of preferences." *Psychological Review* 118.1: 42.

Resnick, P. and H. R. Varian. .1997. Recommender Systems. *Communications of the ACM*, 40 (3): 56-58.

Rodríguez, R. M., L. Martínez, and F. Herrera. .2013. A Group Decision Making Model Dealing with Comparative Linguistic Expressions Based on Hesitant Fuzzy Linguistic Term Sets. *Information Sciences*,.

Rossi, F., K. B. Venable, and T. Walsh. .2011. A Short Introduction to Preferences: Between Artificial Intelligence and Social Choice. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 5 (4): 1-102.

Rubens, N., Kaplan,D., and Sugiyama,M. 2011. "Active Learning in Recommender Systems." In *Recommender Systems Handbook*, 735-767: Springer.

Satterthwaite, Mark Allen. 1975. "Strategy-proofness and Arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions." *Journal of economic theory* 10, no. 2: 187-217.

Senot, C., Kostadinov,D., Bouzid,M., Picault,J., and Aghasaryan,A. .2011. Evaluation of Group Profiling Strategies.*In Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Three*,2728-2733.AAAI Press, .

Shannon, C. E. .2001. A Mathematical Theory of Communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5 (1): 3-55.

Skowron, P., Faliszewski,P., and Slinko,A. .2013. Fully Proportional Representation as Resource Allocation: Approximability Results.*In Proceedings of the Twenty-Third international joint conference on Artificial Intelligence,*353-359.AAAI Press, .

Smith, W. D. .2001. Range Voting.

Stillwell, W. G., D. A. Seaver, and J. P. Schwartz. .1982. *Expert Estimation of Human Error Probabilities in Nuclear Power Plant Operations: A Review of Probability Assessment and Scaling,*.

Suzumura, K., Arrow,K. J., and Sen,A. 2010. *Handbook of Social Choice & Welfare*. Vol. 2 Elsevier.

Tversky, Amos. 1969. "Intransitivity of preferences." *Psychological review* 76.1: 31.

Walsh, T. .2007. Uncertainty in Preference Elicitation and Aggregation.*In Proceeding of the National Conference on Artificial Intelligence,*3.Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, .

Walsh, T. .2008. Complexity of Terminating Preference Elicitation.*In Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2,*967-974.International Foundation for Autonomous Agents and Multiagent Systems, .

Xia, L. and Conitzer,V. .2011. A Maximum Likelihood Approach Towards Aggregating Partial Orders.*In Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI),*446-451.

Yu, K., A. Schwaighofer, V. Tresp, X. Xu, and H. P. Kriegel. .2004. Probabilistic Memory-Based Collaborative Filtering. *Knowledge and Data Engineering, IEEE Transactions On,* 16 (1): 56-69.

Yu, Z., X. Zhou, Y. Hao, and J. Gu. .2006. TV Program Recommendation for Multiple Viewers Based on User Profile Merging. *User Modeling and User-Adapted Interaction,* 16 (1): 63-82.

# Appendix

## An Example for the Computation of the Probabilistic Voter Rating

### Distribution Model

The following is a step by step illustration of the probabilistic voter distribution model presented in section 4.2, using a running example. Consider the ratings given in Table 7 in section 4.2. The delta ratings (obtained using eq.4.4) are shown in Table 14. First, we compute voter to voter similarity using Cosine similarity (eq.4.5). The results are shown in Table 15. Next, we compute predicted ratings according to eq.4.6 (Table 10). Next, the similarities are aggregated into buckets according to their ratings (Table 17). For example, let us examine the aggregation for $v_1$ and $c_1$. In the first line of Table 10 we can see that the pair $v_1.c_1$ has no predicted rating of $d_g = 1$. There are three rounded predictions of $d_g = 2$; when the neighbor voters are: $v_1$, $v_2$, $u_{2,}$ $u_6$. The corresponding similarities (i.e., v1 with each of these neighbors) in Table 15 are aggregated: $1 + 0.266 + 0.552 + 0.369$. The result is updated in line one in column "2" (Table 17). Finally, we convert the results into a probability distribution such that each row sums up to 1 thus completing the computation of the initial voter-item rating distributions (Table 18). Computation of bias-free behavior can be achieved by applying alternative methods; however, we leave this to future research.

The voter-item distributions are dynamic: they change as new information is revealed. When a voter submits a rating on an item, the voter-item probability distribution is updated by calculating only the changes of the relevant places. Furthermore, a new rating contains valuable information which affects not only the user submitting the rating, but also the probability distribution of other voters for the same item.

Table 14: Bias free behavior according to eq.4.4

| | $v_1$ | $v_2$ | $v_3$ | $v_4$ | $u_1$ | $u_2$ | $u_3$ | $u_4$ | $u_5$ | $u_6$ |
|---|---|---|---|---|---|---|---|---|---|---|
| $v_1$ | 0 | 0 | 0 | -0.15 | 2.45 | 2.45 | 2.45 | 1.35 | -0.95 | 1.15 |
| $v_2$ | 0 | 0 | 0 | 0.85 | 2.45 | 2.45 | 2.45 | -0.65 | 2.05 | -0.85 |
| $v_3$ | 0 | 0 | 0 | 3.15 | -1.25 | 0.75 | 1.75 | 1.65 | 0.35 | 1.45 |
| $v_4$ | 0 | 0 | 0 | 3.05 | 1.65 | -0.35 | 0.65 | 0.55 | 1.25 | 1.35 |
| $u_1$ | 0.95 | -0.38333 | 0.283333 | 1.05 | -0.35 | -1.35 | -1.35 | 0.55 | 0.25 | 0.35 |
| $u_2$ | -1.05 | 0.616667 | -1.71667 | -1.95 | 0.65 | 1.65 | 1.65 | -0.45 | 0.25 | 0.35 |
| $u_3$ | -2.15 | -0.48333 | 1.183333 | 0.95 | -0.45 | -0.45 | -0.45 | 0.45 | 1.15 | 0.25 |
| $u_4$ | 0.15 | -0.18333 | -0.51667 | -1.75 | -0.15 | -0.15 | -0.15 | 0.75 | 1.45 | 0.55 |
| $u_5$ | 1.15 | -0.18333 | -0.51667 | 0.25 | -0.15 | -0.15 | -1.15 | -0.25 | 0.45 | 0.55 |
| $u_6$ | 0.45 | 0.116667 | 0.783333 | -0.45 | 0.15 | 0.15 | -0.85 | 1.05 | -1.25 | -0.15 |

Table 15: Voter-to-voter similarity according to eq.4.5

| | $v_1$ | $v_2$ | $v_3$ | $v_4$ | $u_1$ | $u_2$ | $u_3$ | $u_4$ | $u_5$ | $u_6$ |
|---|---|---|---|---|---|---|---|---|---|---|
| $v_1$ | 1 | 0.266 | -0.315 | -0.53 | -0.679 | 0.552 | -0.519 | -0.146 | -0.509 | 0.369 |
| $v_2$ | 0.266 | 1 | -0.453 | -0.174 | -0.625 | 0.437 | -0.209 | -0.134 | -0.325 | -0.436 |
| $v_3$ | -0.315 | -0.453 | 1 | 0.304 | 0.311 | -0.403 | 0.236 | -0.439 | -0.032 | -0.084 |
| $v_4$ | -0.53 | -0.174 | 0.304 | 1 | 0.619 | -0.614 | 0.309 | -0.493 | 0.281 | -0.28 |

Table 16: Predicted rating according to eq.4.6

| | $v_1$ | $v_2$ | $v_3$ | $v_4$ | $u_1$ | $u_2$ | $u_3$ | $u_4$ | $u_5$ | $u_6$ |
|---|---|---|---|---|---|---|---|---|---|---|
| $v_1.c_1$ | 2.25 | 2.25 | 0 | 0 | 0 | 1.64 | 0 | 0 | 0 | 2.39 |
| $v_1.c_2$ | 3.58 | 3.58 | 0 | 0 | 0 | 3.89 | 0 | 0 | 0 | 3.61 |
| $v_1.c_3$ | 2.92 | 2.92 | 0 | 0 | 0 | 1.94 | 0 | 0 | 0 | 3.18 |
| $v_2.c_1$ | 2.25 | 2.25 | 0 | 0 | 0 | 1.77 | 0 | 0 | 0 | 0 |
| $v_2.c_2$ | 3.58 | 3.58 | 0 | 0 | 0 | 3.83 | 0 | 0 | 0 | 0 |
| $v_2.c_3$ | 2.92 | 2.92 | 0 | 0 | 0 | 2.14 | 0 | 0 | 0 | 0 |
| $v_3.c_1$ | 0 | 0 | 1.95 | 1.95 | 2.23 | 0 | 1.43 | 0 | 0 | 0 |
| $v_3.c_2$ | 0 | 0 | 3.28 | 3.28 | 3.15 | 0 | 3.16 | 0 | 0 | 0 |
| $v_3.c_3$ | 0 | 0 | 2.62 | 2.62 | 2.69 | 0 | 2.88 | 0 | 0 | 0 |
| $v_4.c_1$ | 0 | 0 | 2.05 | 2.05 | 2.6 | 0 | 1.37 | 0 | 2.36 | 0 |
| $v_4.c_2$ | 0 | 0 | 3.38 | 3.38 | 3.11 | 0 | 3.22 | 0 | 3.32 | 0 |
| $v_4.c_3$ | 0 | 0 | 2.72 | 2.72 | 2.86 | 0 | 3.06 | 0 | 2.56 | 0 |

Table 17: Aggregated voter similarities

| | **1** | **2** | **3** | **4** | **5** |
|---|---|---|---|---|---|
| $v_1.c_1$ | 0 | 2.186 | 0 | 0 | 0 |
| $v_1.c_2$ | 0 | 0 | 0 | 2.186 | 0 |
| $v_1.c_3$ | 0 | 0.552 | 1.634 | 0 | 0 |
| $v_2.c_1$ | 0 | 1.702 | 0 | 0 | 0 |
| $v_2.c_2$ | 0 | 0 | 0 | 1.702 | 0 |
| $v_2.c_3$ | 0 | 0.437 | 1.266 | 0 | 0 |
| $v_3.c_1$ | 0.236 | 1.615 | 0 | 0 | 0 |
| $v_3.c_2$ | 0 | 0 | 1.851 | 0 | 0 |
| $v_3.c_3$ | 0 | 0 | 1.851 | 0 | 0 |
| $v_4.c_1$ | 0.309 | 1.585 | 0.619 | 0 | 0 |
| $v_4.c_2$ | 0 | 0 | 2.514 | 0 | 0 |
| $v_4.c_3$ | 0 | 0 | 2.514 | 0 | 0 |

Table 18: The normalized distribution

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $v_1.c_1$ | 0.139 | 0.443 | 0.139 | 0.139 | 0.139 |
| $v_1.c_2$ | 0.139 | 0.139 | 0.139 | 0.443 | 0.139 |
| $v_1.c_3$ | 0.139 | 0.216 | 0.367 | 0.139 | 0.139 |
| $v_2.c_1$ | 0.149 | 0.403 | 0.149 | 0.149 | 0.149 |
| $v_2.c_2$ | 0.149 | 0.149 | 0.149 | 0.403 | 0.149 |
| $v_2.c_3$ | 0.149 | 0.214 | 0.338 | 0.149 | 0.149 |
| $v_3.c_1$ | 0.18 | 0.382 | 0.146 | 0.146 | 0.146 |
| $v_3.c_2$ | 0.146 | 0.146 | 0.416 | 0.146 | 0.146 |
| $v_3.c_3$ | 0.146 | 0.146 | 0.416 | 0.146 | 0.146 |
| $v_4.c_1$ | 0.174 | 0.344 | 0.216 | 0.133 | 0.133 |
| $v_4.c_2$ | 0.133 | 0.133 | 0.468 | 0.133 | 0.133 |
| $v_4.c_3$ | 0.133 | 0.133 | 0.468 | 0.133 | 0.133 |

## An Example of the Computation of the Item Winning Probability

The following is a step by step illustration of item winning probability presented in section 4.3, using a running example. The example is based on the voting distributions (VD's) presented in Table 7; note that: $Pr(q_3^1 = 3) = 0.4,\ Pr(q_3^1 = 2) = 0.3,\ Pr(q_3^1 = 1) = 0.3$.

We begin by calculating $Pr(c_j = s)$. The results are presented in Table 19. The calculation involves a dynamic programming algorithm where each row is calculated using the results of the row above it. For instance, to calculate $Pr(c_1 = 6)$ based on the ratings of voters $v_1,\ v_2, v_3$, we use the probabilities that were computed in columns 3-5, line 2: $Pr(c_1 = 6) = 0.14 \cdot Pr(q_3^1 = 3) + 0.36 \cdot Pr(q_3^1 = 2) + 0.24 \cdot Pr(q_3^1 = 1) = 0.236$. This result is bolded in Table 19. Next, we calculate $Pr(c_1 \leq s)$ by aggregating the results of the cells in row 3 in Table 19. For item $c_{2,}$ we aggregate the results of row 6. This is presented in Table 20.

The probability that item $c_1$ is a winner with a certain aggregated rating $s$ is presented in Table 21. In our example of only two items, the probability of $c_1$ to win is: $Pr(NW = c_1) = Pr(c_1 = s) \wedge Pr(c_2 \leq s)$. For instance, the probability that $c_1$ is the winner with an aggregated rating of 5 is equal to the probability that its aggregated rating is 5 and the aggregated rating of $c_2$ is at most 5: 0.182*0.492=0.089. Next, we aggregate the item's probability to win over all possible ratings $s$. This is demonstrated in the last column of Table 21 (i.e., $Pr(NW = c_1) = 0.729$). Ties are broken according to the item positions in an increasing order of all items.

Table 19: The probability that an item has a score of *s*

| **Item** | **Voters** | $s = 3$ | $s = 4$ | $s = 5$ | $s = 6$ | $s = 7$ | $s = 8$ | $s = 9$ |
|---|---|---|---|---|---|---|---|---|
| $Pr(c_1 = s)$ | $v_1$ | 0.6 | 0 | 0 | 0 | 0 | 0 | 0 |
| | $v_1, v_2$ | 0.14 | 0.36 | 0.24 | 0.18 | 0 | 0 | 0 |
| | $v_1, v_2, v_3$ | 0.024 | 0.066 | 0.182 | **0.236** | 0.27 | 0.15 | 0.072 |
| $Pr(c_2 = s)$ | $v_1$ | 0.6 | 0 | 0 | 0 | 0 | 0 | 0 |
| | $v_1, v_2$ | 0.14 | 0.4 | 0.18 | 0.18 | 0 | 0 | 0 |
| | $v_1, v_2, v_3$ | 0.07 | 0.108 | 0.314 | 0.194 | 0.224 | 0.054 | 0.036 |

Table 20: The probability that an item has a score of at most *s*

| **Item** | **Voters** | $s = 3$ | $s = 4$ | $s = 5$ | $s = 6$ | $s = 7$ | $s = 8$ | $s = 9$ |
|---|---|---|---|---|---|---|---|---|
| $\Pr(c_1 \leq s)$ | $v_1, v_2, v_3$ | 0.024 | 0.09 | 0.272 | 0.508 | 0.778 | 0.928 | 1 |
| $\Pr(c_2 \leq s)$ | $v_1, v_2, v_3$ | 0.07 | 0.178 | 0.492 | 0.686 | 0.91 | 0.964 | 1 |

Table 21: The winning probability of an item

| **Item** | **Voters** | S=3 | S=4 | S=5 | S=6 | S=7 | S=8 | S=9 | **Total** |
|---|---|---|---|---|---|---|---|---|---|
| $\Pr(NW = c_1)$ | V1, V2, V3 | 0.001 | 0.011 | 0.089 | 0.162 | 0.245 | 0.144 | 0.072 | 0.729 |
| $\Pr(NW = c_2)$ | V1, V2, V3 | 0 | 0.003 | 0.028 | 0.053 | 0.114 | 0.042 | 0.033 | 0.273 |

# תקציר

המחקר הזה מתמקד בנושא של חילוץ העדפות עבור קבלת החלטות לקבוצה בעזרת חוקי הצבעות. אנו מציעים מסגרת כללית, רב תחומית לחילוץ העדפות, כאשר המטרה היא למזער את התקשורת עם המשתמשים. אנו מציגים יוריסטיקות מקוריות ומראים כיצד הן פועלות תחת פרוטוקלי הצבה של דירוגים ושל מדרגים, בפרט תחת הפרוטוקולים "ריינג'" ו"בורדה". אנו מציעים מסגרת אינטרקטיבית ואינקרמנטלית; בכל שלב משתמש אחד מתושאל לבי הדירוג שלו לפריט אחד או לגבי המרדג שלו לשני פריטים. אנו מציעים שתי גישות ליוריסטיקות שקובעות איזו שאילתה היא הבאה בתור (כלומר את מי לתשאל לגבי איזה פריט או אילו פריטים). יוריסטיקה אחת מחשבת את רווח המידע של כל שאילתה המועמדת. היוריסטיקה השנייה משתמשת בהתפלגות ההסתברותית של העדפות המשתמשים על מנת לבחור את הפריט שהכי סביר שיזכה. שתי היוריסטיקות מסתמכות על התפלגות הדירוגים. אנו מראים כיצד ניתן להעריך התפלגות זו. ההתפלגות מתעדכנת בצורה איטרטיבית, כך שהדיוק של ההתפלגות עולה עם הזמן.

למרות שתוצאה של פריט אחד שהוא המנצח הבטוח היא התוצאה המדויקת ביותר, אנו מסתכלים גם על שקלול תמורות שבין מאמץ לדיוק ועל אסטרטגיות קיבוץ בחילוץ העדפות. ראשית, אנו מציעים לסיים את חילוץ ההעדפות מוקדם יותר על ידי החזרה של $k$ פריטים כאשר אחד מהם הוא הפריט מנצח, במקום להחזיר פריט מנצח אחד בלבד. שנית, אנו מציעים להעריך את המנצח או המנצחים המשוערים ברמת בטחון מסויימת. מצד אחד, מנצח משוער הוא פחות מדיוק, מצד שני, מנצח משוער מוריד את עלות התקשורת עם המשתמשים. לבסוף, אנו מציעים להתחשב באסטרטגיית הקיבוץ כאשר מחברים את העדפות המשתמשים. אנו מראים שאסטרטיגיית הקיבוץ משפיעה על עלות התקשורת עם המשתמשים ואנו משווים שתי אסטרטגיות נפוצות: "מג'וריטי" ו"ליסט מיסרי". אנו מדגימים את היעילות של המסגרת שלנו על ידי בחינה של היוריסטיקות על ארבעה בסיסי נתונים אמתיים שונים. כדי לבחון את ההשפעות השונות של המידע אנו משתמשים גם בבסיסי נתונים מסומלצים בהם ניתן לשנות את הפרמטרים של המידע.

מילות מפתח: מערכות המלצה, חילוץ העדפות, חוקי הצבעות

העבודה נעשתה בהדרכת

ד״ר מאיר קלך
פרופ׳ ליאור רוקח
פרופ׳ ברכה שפירא

במחלקה להנדסת מערכות מידע

בפקולטה למדעי ההנדסה

# חילוץ העדפות עבור החלטות קבוצתיות בעזרת תאוריית ההצבעות

מחקר לשם מילוי חלקי של הדרישות לקבלת תואר "דוקטור לפילוסופיה"

מאת

ליהי דראי

הוגש לסנאט אוניברסיטת בן גוריון בנגב

תמוז תשע"ה יולי 2014

באר שבע