Collaborative Scoring with Dishonest Participants

Seth Gilbert
EPFL
Lausanne, Switzerland
seth.gilbert@epfl.ch

Rachid Guerraoui
EPFL
Lausanne, Switzerland
rachid.guerraoui@epfl.ch

Faezeh Malakouti Rad
Boston University
Boston MA, USA
faezeh@bu.edu

Morteza Zadimoghaddam
MIT
Cambridge MA, USA
morteza@mit.edu

ABSTRACT

Consider a set of players that are interested in collectively evaluating a set of objects. We develop a *collaborative scoring* protocol in which each player evaluates a subset of the objects, after which we can accurately predict each players' individual opinion of the remaining objects. The accuracy of the predictions is near optimal, depending on the number of objects evaluated by each player and the correlation among the players' preferences.

A key novelty is the ability to tolerate malicious players. Surprisingly, the malicious players cause no (asymptotic) loss of accuracy in the predictions. In fact, our algorithm improves in both performance and accuracy over prior state-of-the-art collaborative scoring protocols that provided no robustness to malicious disruption.

Categories and Subject Descriptors

C.2.4 [**Computer Communication Networks**]: Distributed Systems; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems

General Terms

Algorithms, Theory

Keywords

fault tolerance, randomized algorithms, collaborative filtering, recommendation systems

1. INTRODUCTION

Imagine a group of researchers (say, a program committee) that is attempting to evaluate a set of papers (say, SPAA submissions). Each researcher wants to know whether or not she/he likes each paper. However, none of the busy researchers have enough time to read all of the papers. Therefore, each researcher is assigned some subset of the papers to evaluate, and their scores are then used to determine, for each researcher, his/her (supposed) opinion on the remaining (unread) papers. (Notice that this differs from standard program committees in that, in the end, every researcher develops an opinion on every paper.) In fact, it is possible to accurately guess each committee member's opinion on his/her unread paper as long as there is sufficient correlation among the opinions of the researchers.

This process of collecting and correlating information is an example of *collaborative scoring*, which has been extensively studied (e.g., [2–5, 9]). Our goal in this paper is to design a collaborative scoring algorithm that minimizes the number of items each player must evaluate (i.e., the number of papers that each researcher has to read), while at the same time maximizing the accuracy of the predictions. Intuitively, there is an inherent trade-off between the number of items evaluated and the accuracy of the predictions.

Dishonest players.

One of the problems with collaborative solutions is that some of the players may violate the protocol or act dishonestly. For example, some researchers may be too busy and, instead of reading their assigned papers, they may simply choose scores at random (or based on the reputation of the authors). Alternatively, researchers might attempt to bias the algorithm toward their colleagues' papers[1]. In either case, some researchers may be tricked into liking a paper that they would otherwise dislike, if they had read it (or *vice versa*).

An important goal of this paper is to develop a collaborative scoring protocol that is robust to malicious interference[2]. Even if a reasonable fraction of the players collude in an attempt to subvert the system, the honest players are still

[1]While such practices are of course rare, perhaps non-existent, in the honest community of computer science researchers, these problems remain worthy of consideration in a broader context.

[2]For historical reasons, such malicious parties are often referred to as "Byzantine."

guaranteed near-optimal predictions as to their scores[3]. To the best of our knowledge, this is the first collaborative scoring algorithm that can tolerate dishonest players[4]. Interestingly, our algorithm also improves even over state-of-the-art algorithms that do not allow for dishonest behavior.

Correlation of preferences, accuracy of predictions.

The basic idea behind collaborative scoring is to leverage the correlation in preferences among players. For example, imagine that there are two researchers Alice and Bob with very similar opinions regarding the papers to be evaluated; in that case if Alice evaluates some paper, then Bob can reliably predict that, for that paper, he will have the same opinion as Alice. Thus if we have significant correlation among the researchers, and if we can determine efficiently which researchers share the same opinions, then we can develop very accurate predictions, even if the number of papers evaluated by each researcher is relatively small.

In fact, the accuracy of the predictions depends on the correlation of the players' preferences. For example, if the preferences are entirely independent, then collaboration provides no benefit: each researcher has to read every paper themselves. On the other hand, if there are large subsets of the players that share the same preferences, then collaboration can be quite effective. The question we address in this paper is how to discover such correlations, and how to leverage these correlations to provide good predictions.

Our results.

Assume we have n players and n objects. (Generalizing to more objects is straightforward.) Each player has some unknown opinion of each object. Our goal is to determine for each player whether she likes or dislikes each object. We present a collaborative scoring protocol with the following attributes:

- *Optimality:* Our protocol predicts each player's preferences with near-optimal levels of precision in the following sense. Given a bound B on the number of objects that can be evaluated by each player, and given a particular (though unknown) level of correlation among preferences, there is some minimum rate of error that can be achieved. Our algorithm achieves a *constant-factor approximation* of this minimum rate of error, in the worst-case, while requiring each player to evaluate $O(B \log^{O(1)} n)$ objects. Thus, by slightly augmenting the budget of each player, we achieve an asymptotically optimal rate of error, despite knowing nothing about the level of correlation among the preferences.

- *Fault-tolerance:* We achieve this (almost) optimal rate of error, despite up to $n/(3B)$ of the users behaving in a dishonest fashion.

Prior to this paper, the best known algorithm [2,3]—which does not tolerate dishonest players—requires each player to examine $O(B^2 \log^{O(1)} n)$ objects, and yet achieves only a B-approximation (instead of a constant-factor approximation) of the optimal rate of error[5].

Our basic strategy is to discover clusters of players that can cooperatively evaluate the objects. If we can identify, for each player, a sufficiently large set of other players that have similar preferences, then these players can share the work of evaluating the objects. For instance, if there are n objects and each player is part of a cluster containing at least n/B other players with similar preferences, then no player needs to evaluate more than B objects. Thus one of the main goals of the protocol is to find such clusters of players with similar preferences. In general, standard sampling techniques can be used to find such clusters. A key problem, however, is that the dishonest players may attempt to disrupt the process by "hijacking" some of the clusters. In order to compensate for this the samples would have to be too large; thus we augment the sampling with collaborative scoring techniques to reduce the cost.

More specifically, in order to find clusters of players with similar preferences, we want to examine the players' preferences on a smaller subset of the objects, specifically, a randomly chosen sample of size $\Theta(n \log n/D)$, where D represents the maximum divergence of preferences. It is too expensive, however, for each player to evaluate each of the objects even in this smaller sample, as each player can only evaluate $B \log^{O(1)} n$ objects. Since the sample is chosen at random, players that have similar preferences over the larger set have very similar preferences on the smaller set. Thus we can use a collaborative scoring algorithm optimized for players with very similar preferences, first described in [2,3]. This provides a good estimate of the preferences on the small sample, which can then be used to produce a clustering of the players. In order to ensure that no cluster is "hijacked" by dishonest players, we show that there are enough honest players in each cluster to dominate. Finally, the work of probing objects is divided among players in each cluster, with sufficient redundancy to overcome the dishonest players.

Roadmap.

In Section 2, we describe the basic model and define the problem of collaborative scoring and what it means for an algorithm to be optimal. In Section 4, we discuss some of the related work. In Section 5, we review some existing algorithms that we will use as sub-components of our protocol. In Section 6, we present and analyze our new algorithm in the absence of dishonest players, and in Section 7, we show how to cope with dishonest players. Finally, we conclude in Section 8.

2. BASIC MODEL

We consider a world consisting of a set Π of n *players* and a set Λ of n *objects*. (Generalizing for more objects than players is straightforward, and omitted for clarity.) Let P be the set of players and O the set of objects. Associated with each player $p \in P$ is a binary (0/1) vector $v(p)$ of size n that indicates whether player p likes or dislikes each object. We refer to $v(p)$ as player p's *preference* vector, and it is initially unknown. (In Section 8 we discuss some generalizations of this model.)

[3]See Section 2 for more intuition on what is meant here by *optimal*.

[4]There do exist collaborative *recommendation* systems that tolerate malicious users, e.g., [5,16], but these are solving a somewhat different problem, as discussed in Section 4.

[5]In [2,3], B is assumed to be constant, and hence it also claims to achieve a constant-factor approximation.

Notice that there may be some hidden structure on the distribution of preferences, for example, certain sets of players may have correlated preferences on certain subsets of the objects. However, we do not make any *a priori* assumptions on such structure. If such structure does exist, it is unknown to the players and must be discovered. (It is, in fact, just such correlations between players that makes collaborative scoring effective.)

We assume that some of the players are honest and some of the players are dishonest. A dishonest player may ignore the protocol, lying about its preferences and attempting to improperly influence the output of the protocol.

The game proceeds in synchronous rounds. In each round, each player can choose one object to probe. Every time a player probes an object, it learns its preference for that object. For example, when player p probes object k, it learns whether $v(p)_k$ is equal to 0 or 1.

Players have access to a public "bulletin board" (e.g., a distributed shared memory). In each round, the players can update and read the bulletin board after each probe. Without loss of generality, we assume that each honest player writes the result of each probe to the bulletin board. A dishonest player cannot modify the data written by honest players on the bulletin board.

Throughout the paper, when we say that an event occurs with high probability, we mean with probability $1-1/n$. By increasing the probe complexity by a constant factor, we can achieve a probability of error of $1/n^c$ for any constant c.

3. COLLABORATIVE SCORING

The problem of *B-budget collaborative scoring* is defined as follows. Each player may make up to $O(B)$ probes. The goal is to generate for each player p a vector $w(p)$ that minimizes $|w(p)-v(p)|$, i.e., the Hamming distance between the real preference vector $v(p)$ and the output vector $w(p)$. The *rate of error* is the maximum such difference for any player, for any set of initial preference vectors.

Initially, in Section 6, we assume that all the players behave correctly, obeying the protocol. In Section 7, we show that our protocol can tolerate up to $n/(3B)$ of the players behaving dishonestly.

Our goal is to devise an algorithm that, while using only $O(B \log^{O(1)} n)$ probes, performs asymptotically as well as any algorithm using only B probes. (One might think of this as a form of *resource augmentation*: by using somewhat more probes, the algorithm presented here can perform almost as well as an optimal B-budget collaborative scoring algorithm.) We now state more precisely what this claim means.

We begin by providing some intuition as to why Definition 1 in fact describes the best performance that a B-budget collaborative scoring protocol can achieve, in the worst case. While this may seem a somewhat non-intuitive definition of optimality, it does define a lower bound on what is achievable and hence a benchmark to which we can later compare the algorithm developed in this paper.

Given a subset of the players P, we define $D(P)$ to be the diameter of P: $D(P) = \max_{p,q\in P}(|v(p)-v(q)|)$. Assume we have some B-budget algorithm $\mathcal{ALG}$.

Consider some particular player p. Since each player can only probe B objects, it is easy to see that, in a sense, p must "collaborate" with at least $(n/B)-1$ other players in order to ensure that p has information on every object. (If p collaborates with any fewer players, then it has access to data on less than n objects. In this case, there is some object for which p has no information, i.e., it can only guess at random its preference.) Notice that p may certainly examine information from more than $(n/B)-1$ other players, and we in no way restrict $\mathcal{ALG}$ from collecting information in any way it chooses. However, in order to do better than random guessing, p must use information from probes performed by at least $(n/B)-1$ other players.

Ideally, player p would collaborate with the $(n/B)-1$ players that have preferences *most similar* to p. That is, player p would collaborate with a set of players P with minimum diameter. (Again, player p may also use information from farther away players.)

In fact, in the worst case, player p can do no better than to collaborate with the $(n/B)-1$ closest players. That is, if P is the set containing p with minimum diameter $D(P)$, then no algorithm can have a rate of error less than $D(P)/4$. Thus we define *optimality* in terms of the diameter of the set of players closest to p.

Formally, we define the notion of optimality (much as in [2, 3]) as follows:

Definition 1. *A collaborative scoring algorithm is said to be* asymptotically optimal with respect to some budget B *if there exists some constant c such that for every input set of preferences vectors, for every player p, with high probability:*

$$|w(p)-v(p)| \le \min_{P\subseteq\Pi,\ p\in P,\ |P|\ge n/B} cD(P)\ .$$

In the worst-case, every B-budget collaborative scoring algorithm may perform only as well as specified in Definition 1. We give here a simple proof of this claim, demonstrating a particular distribution of preferences that yields the specified rate of error.

Claim 2. *For every B-budget collaborative scoring algorithm $\mathcal{ALG}$, there is some distribution of preferences and some player p such that, with constant probability:*

$$|w(p)-v(p)| \ge \min_{P\subseteq\Pi,\ p\in P,\ |P|\ge n/B} D(P)/4\ .$$

Proof. Given a constant D such that $n/4 > D > 2B$, we define an input distribution of preference vectors as follows. Let P be a set of players of size n/B and let p be some arbitrary player in P. For every player $q \notin P$, assign its preference vector $v(q)$ at random. In addition, for player p, assign its preference vector $v(p)$ at random. Choose an arbitrary special set S of D objects, and for every player $q \in P\setminus\{p\}$, define $v(q)=v(p)$ on every object *except* those in the set S; for the objects in S, choose $v(q)$ at random.

Since the preference vectors of every player not in P are chosen at random (with respect to p), obviously the probes performed by players outside of P provide no information to p. Similarly, the probes performed by players in P provide no information to p on objects in S. Since p probes at most B objects, and since S contains at least $D > 2B$ objects, there are at least $D/2$ objects on which p has no information. Thus, no algorithm can do better for p than guessing its preferences on objects in S, and hence $\mathcal{ALG}$ has a rate of error of at least $D/4$, in expectation. At the same time, the diameter of $P \le D$ (and every other set of size n/B is of distance at least $n/4 > D$, whp), concluding the proof. $\square$

While the above proof uses a very specific distribution to demonstrate the worst-case notion of optimality, in fact, for a broad set of preference distributions, this notion of optimality really does capture the best that any B-budget collaborate scoring protocol can achieve[6].

In this paper, we give a new algorithm for solving the $B \log^{O(1)} n$-budget collaborative scoring problem; and this new algorithm is asymptotically optimal with respect to B.

4. RELATED WORK

The problem of determining preferences via collaborative scoring (or "collaborative filtering") has been widely studied. We focus here on the on-line solutions; other research focuses on off-line solutions that examine historical data to reconstruct preferences.

Early work in this area [1, 6–9, 12] defined the problem as one of reconstructing a matrix of preferences, and brought a series of linear algebraic techniques to bear on the problem. In [9], for example, they rely on singular-value decomposition, which requires some strong assumptions: users are partitioned into types that have orthogonal preferences; and users of the "dominant" type outnumber users of "subdominant" types.

Awerbuch et al. [4] and Alon et al. [2, 3] relax these different assumptions, introducing a combinatorial approach to the problem of collaborative scoring. They define the model that we use in this paper, and formulate precisely the problem of collaborative scoring. In [4], they introduce the algorithm that we here refer to as ZeroRadius (and the associated Select protocol), optimally solving the problem of collaborative scoring under the assumption that there are large clusters of users with *exactly* identical preferences. In [2, 3], they address the more general problem, which attempts to leverage correlation among users where preferences are similar, even when not identical. They develop an algorithm in which each user makes $O(B^2 \log^{O(1)} n)$ probes, and the resulting output is a B-approximation of optimal with respect to a budget of B. (As they assume that B is a constant, they refer to this as a constant-factor approximation of optimal.)

Another area of research has focused on the problem of *recommendation systems*, a problem closely related to collaborative scoring. The goal of a *recommendation system* is to provide each user with a small number of recommendations (e.g., one per day), with the goal of maximizing the number of "good" objects recommended. By contrast, a *collaborative scoring* system attempts to determine a user's score for every item. As one example of this line of research, Kleinberg and Sandler [11] develop a near optimal recommendation algorithm (based on mixture models).

While there has been no prior work (to our knowledge) on collaborative scoring robust to malicious players, there has been research on robust recommendation systems. In [5], Awerbuch et al. develop such a recommendation system; their algorithm runs in $O(n \log n)$ time, and is within a $\log n$ factor of optimal in the size of the recommendation set. In [16], Yu et al. presents a protocol that provides a continuous stream of recommendations (e.g., one per day), while maximizing the percentage of good recommendations. It too can tolerate dishonest players (known as "Sybils"), and relies on careful reputation management to choose recommendations. There is other work on robust recommendation systems that relies on pre-existing social networks (e.g., [14, 15].

Finally, Nisgav and Patt-Shamir [13] have recently developed algorithms for partitioning users based on their preferences. Their goal is not to determine preferences, but instead to group users into sets with similar preferences. It is possible that such a partitioning could be used as the basis for a collaborative scoring protocol (though it is unclear how the performance would compare), and their sampling techniques bear some similarity to the techniques used in this paper. However they do not consider the possibility of malicious failures.

5. BACKGROUND

In developing our algorithm, we use as building blocks three existing algorithms from [2–4]. To be self-contained, we briefly repeat them here. (They need little modification to tolerate dishonest players; see Section 7.)

5.1 Choose Closest Candidate

The first building block is a protocol for selecting among a set of candidate preference vectors. Assume a set of candidate vectors $w_1, w_2, \ldots, w_k$. Player p wants to determine which vector is closest to $v(p)$. The goal of the RSelect protocol, found in Figure 1, is to identify, with high probability, the best candidate.

Theorem 3 (Theorem 6.1 [2]). *Let w^* be the vector in $w_1, \ldots, w_k$ that is closest to $v(p)$. Then, with high probability,* RSelect *outputs a vector w such that $|v(p) - w| \leq O(|v(p) - w^*|)$, using only $O(k^2 \log n)$ probes.*

5.2 Special Case: Zero Radius Sets

A second building block, originally from [4], implements collaborative scoring under the assumption that for each $p \in P$, there exists a subset $S(p) \subseteq \Pi$ of size n/B with the *exact same* preferences as p. That is, the subset $S(p)$ has diameter of size 0. See Figure 1 for the pseudocode.

Theorem 4 (Theorem 3.1 [2]). *Assume that at least n/B' players have identical preferences to player p. Then, with high probability,* ZeroRadius$(\cdot, \cdot, B')$ *outputs preference vector $v(p)$ with $O(B' \log n)$ probes.*

5.3 Special Case: Small Radius Sets

A third algorithm from [2, 3] solves the collaborative scoring problem under the assumption that for each $p \in P$, there exists some subset $S(p)$ of size n/B where the diameter $D(S(p))$ is no greater than $\log n$. In this case, the diameter D is a parameter to the algorithm. The pseudocode can be found in Figure 1. The Select protocol is a deterministic version of RSelect with slightly difference performance guarantees.

Theorem 5 (Theorem 4.4 [2]). *Assume that at least n/B players have preferences within distance D of player p. Then, with high probability,* SmallRadius *outputs a preference vector $w(p)$ such that $|w(p) - v(p)| \leq 5D$, making $O(B \log n D^{3/2}(D + \log n))$ probes.*

[6]Notice this same notion of optimality can be found in [2, 3], where the *stretch* is divided by the diameter.

```
algorithm RSelect($(w_1, \ldots, w_k)_p$);
  For every pair of vectors $w, w' \in \{w_1, \ldots, w_k\}$ do:
    1. Let $X$ be the set of objects on which $w$ and $w'$ differ.
    2. Randomly probe $\Theta(\log n)$ objects from $X$.
    3. Eliminate $w'$ if at least 2/3 of the probed objects agree with $w$, and eliminate $w$ if at least 2/3 of the probed objects
       agree with $w'$. Otherwise, keep both $w$ and $w'$.
  Output any vector that remains.
```

```
algorithm ZeroRadius$(P, O, B')_p$        // players $P$, objects $O$, bound $B'$
  1. If $min(|P|, |O|) < O(B' \log(n))$, player $p$ probes all objects in $O$ and outputs their values.
  2. Otherwise, partition $P$ and $O$ randomly into two sets: place each player from $P$ in $P'$ or $P''$ with probability 1/2;
     place each object from $O$ in $O'$ or $O''$ with probability 1/2. The same partition is chosen by all players in $P$.
  3. Assume wlog that $p \in P'$. Player $p$ recursively executes ZeroRadius$(P', O', B')$ to determine the preferences for
     players in $P'$ on objects in $O'$.
  4. Let $V$ be a set of vectors for $O''$ such that each vector in $V$ is output by at least $|P''|/(2B')$ players in $P''$. Let $C$ be
     the set of objects for which there are different votes in $V$.
  5. While $C \neq \emptyset$ do: Choose an arbitrary object in $C$ and probe it. Remove from $V$ all objects that disagree with the
     probe, and update $C$.
```

```
algorithm SmallRadius$(P, O, D)_p$        // players $P$, objects $O$, diameter $D$
  Repeat $\Theta(\log n)$ times:
    1. Partition the objects $O$ randomly into $s = \Theta(D^{3/2})$ disjoint subsets: $O = O_1 \cup O_2 \cup \cdots \cup O_s$.
    2. For each $i \in \{1, \ldots, s\}$: all players execute ZeroRadius$(\cdot, \cdot, 5B)$ for the objects of $O_i$, allowing for $5B$ probes; let $U_i$
       be the set of vectors output by at least $n/(5B)$ players.
    3. Each player $p$ executes Select$(U_i, D)_p$, obtaining vector $u^i(p)$ for each $i \in \{1, \cdots, s\}$. Concatenate the vectors $u^i(p)$
       over all $i$, and add the result to the set $V$.
  Each player $p$ applies procedure Select$(V, D)_p$ and outputs the result.
```

Figure 1: Building Block Protocols (see [2, 3]). The pseudocode is given for player p. The procedure Select, not included here, is a deterministic version of RSelect that takes two parameters: a set of vectors V, and a diameter bound D such that for a least one vector $v' \in V$, $|v(p) - v'| \leq D$.

6. BASIC PROTOCOL

We now present and analyze the protocol for calculating each players' preferences, which we refer to as CalculatePreferences. The analysis in this section assumes that all the players are honest; in Section 7, we examine the problem of dishonest behavior.

6.1 Preliminaries

Initially, players have no knowledge about the correlation among their preferences. Thus, we begin by "guessing" a diameter D such that for every player p there is a set of a least n/B other players, including p, with diameter no greater than than D. Specifically, we execute our protocol $\lceil \log n \rceil + 1$ times, parameterized with $D = 1, 2, 4, \ldots, n$; the protocol works correctly when D is guessed correctly. For each player p, this process produces $O(\log n)$ candidate vectors $w_1, w_2, \ldots, w_{\log n}$, at least one of which near-exactly represents p's preferences. (This is depicted in Step 1 of Figure 2.) We then execute the RSelect$(w_1, \ldots, w_{\log n})_p$ protocol, for each p, to determine which of these candidate vectors is best.

We fix the target diameter D for the remainder of the paper. We assume throughout that for every player p, there exists a set of player $P' \subseteq P$ of size at least n/B, and containing p, such that $D(P') \leq D$. For at least one choice of D, this assumption will hold. (It should be noted that [2,3] follows this same strategy.)

We rapidly dispense with two easy cases. If the budget $B = \Omega(n/\log n)$, then every player probes every object, and the problem is trivially solved using $O(B \log^{O(1)} n)$ probes. On the other hand, if diameter $D < \log n$, then the SmallRadius algorithm from [2,3] solves the collaborative scoring problem using only $O(B \log^{O(1)} n)$ probes. For the remainder of this paper, we focus on the case where $B = O(n/\ln n)$ and $D \geq \log(n)$.

6.2 Overview

The key idea in our algorithm is to discover clusters of players with similar preferences that can cooperatively probe the objects. Specifically, if we can identify, for each player, a set of at least n/B other players that have similar preferences, then these players can each sample just B objects.

A natural approach is to rely on sampling to determine which players have similar preferences. Consider a randomly chosen set of objects of size $\Theta(n \log n/D)$. Such a set is large enough that it provides a good indication of whether two players have similar preferences. Unfortunately, it is too expensive for each player to probe each element in the set. Fortunately, any two players that have preference distance at most D over all the objects will differ in at most $O(\log n)$ objects on the smaller sample. Thus clusters of diameter D in the entire object space reduce to clusters of diameter $O(\log n)$ on this smaller sample. We can then use the SMALLRADIUS algorithm to efficiently determine each player's preference on this smaller set.

Finally, we use this information to construct a *neighbor graph* in which players with similar preferences share an edge, and we use this to group the players into clusters. The work of probing all n objects is then divided among the players in each cluster. The protocol is described in Figure 2.

6.3 Step 1: Selecting a Sample Set

Each object is added to set S with probability $10\ln(n)/D$. (For now: designate one player to make these random selection and publish them on the bulletin board; see Section 7 for the case of dishonest players.) We observe that the set S provides a good estimate of the similarity (or dissimilarity) of two players' preferences.

LEMMA 6. *For every pair of players p and q:*

1. *If $|v(p) - v(q)| < D$, then they differ in their preferences for at most $20\ln n$ objects, whp.*
2. *if $|v(p) - v(q)| \geq cD$, for $c \geq 3$, then they differ in their preferences for at least $5c\ln n$ objects, whp.*

PROOF. Fix players p and q. Let $A = \{o_1, o_2, \cdots, o_k\}$ be the set of objects on which p and q have different preferences. Note that $k \leq D$. For part (1), the expected number of elements in $A \cap S$ is at most $10\ln(n)$, and hence by a Chernoff bound, with high probability $|A \cap S| \leq 20\ln(n)$. For part (2), the expected number of elements in $A \cap S$ is at least $10c\ln(n)$, and hence by a Chernoff bound, with high probability $|A \cap S| \geq 5c\ln(n)$. Taking a union bound over all pairs $p, q \in P \times P$ concludes the proof. $\square$

6.4 Step 2: Probing the Sample Set

Next, we use the SMALLRADIUS algorithm to determine each player's preferences on the set S. By Lemma 6, we know that each set of players with diameter D differs in their preferences for at most $20\ln n$ objects in the set S. Thus, the SMALLRADIUS algorithm ensures that each player correctly discovers his preferences on the set S, with high probability.

LEMMA 7. *For every player p, let $z(p)$ be the output of the* SMALLRADIUS *algorithm on set S with distance $20\ln n$. Then for every pair of players p and q:*

1. *If $|v(p) - v(q)| \leq D$, then $|z(p) - z(q)| \leq 220\ln(n)$, with high probability.*
2. *If $|v(p) - v(q)| \geq 84D$, then $|z(p) - z(q)| \geq 220\ln(n)$, with high probability.*

PROOF. For part (1): According to Lemma 6, the number of differences between p and q for objects in S is at most $20\ln(n)$. By Theorem 5, we know that, restricted to objects in S, $|v(p) - z(p)| \leq 100\ln n$ and $|v(q) - z(q)| \leq 100\ln n$. Thus, $|z(p) - z(q)| \leq (2 \cdot 100)\ln(n) + 20\ln(n) = 220\ln n$.

For part (2): According to Lemma 6, the number of differences between p and q for objects in S is at least $(5 \cdot 84)\ln(n)$. By Theorem 5, we know that, restricted to objects in S, $|v(p) - z(p)| \leq 100\ln n$ and $|v(q) - z(q)| \leq 100\ln n$. Thus, $|z(p) - z(q)| \geq 420\ln(n) - (2 \cdot 100)\ln(n) = 220\ln n$. $\square$

6.5 Step 3: Calculating Clusters

We now use the output of the SMALLRADIUS algorithm on the sample set S to cluster the players in sets of size at least n/B, where each cluster has diameter $O(D)$. (Recall that we have assumed that for every player p, there exists a set of size n/B, containing p, with diameter $\leq D$.) Assume that for every player p, vector $z(p)$ is the output from SMALLRADIUS. We construct a graph $G = (P, E)$, adding an edge between players p and q if $|z(p) - z(q)| \leq 220\ln n$. We conclude as a corollary of Lemma 7:

LEMMA 8. *For all players p, q: (i) p has degree at least $n/B - 1$ in the neighbor graph; (ii) if (p, q) is an edge, then $|v(p) - v(w)| \leq 84D$.* $\square$

To construct the clusters, we begin with graph $G = (P, E)$, empty sets $V_1, V_2, \ldots$, and a counter $j = 1$. Repeat the following until there is no p in G with degree $\geq n/B - 1$:

1. Choose a player p that has degree $n/B - 1$ in G.
2. Add p and the ($\geq n/B - 1$) neighbors of p to V_j.
3. Remove p and the neighbors of p from G.
4. Increment j.

We have now constructed a sequence of sets $V_1, \ldots, V_\ell$ each of size at least n/B. Moreover, after this process, the graph G contains no player with $n/B - 1$ neighbors. Let q be one of the remaining vertices in the graph with degree less than $n/B - 1$. Since previously, by Lemma 8, q had $n/B - 1$ neighbors, there must be some $p \in V_j$ that was previously a neighbor of q; add q to V'_j, and remove it from G. Once every player has been removed from G, then combine V_j and V'_j to create the final sequence of clusters $V_1, \ldots, V_\ell$.

LEMMA 9. *The clustering has the following properties:*

1. *Every player is contained in exactly one of the sets $V_1, \ldots, V_\ell$.*
2. *For every $j \in [1, \ldots, \ell]$, each set V_j contains at least n/B players.*
3. *For every $j \in [1, \ldots, \ell]$, the diameter $D(V_j) = O(D)$.*

PROOF. The first two properties are immediate by construction. The third follows from the fact that every pair of nodes $p, q \in V_j$ are within distance 4 in the initial neighbor graph. According to Lemma 7, the total number of differences between p and q is then at most $4 \times 84D = 336D$. $\square$

6.6 Step 4: Sharing the Work

Finally, for each cluster and for each object, we choose $\Theta(\log n)$ of the players from the cluster uniformly random, and assign those players to probe the object. Each player p observes the $\Theta(\log n)$ values output for each object o by the assigned players in its cluster, and sets its output $w(p)_o$ to the value that is probed by a majority of the assigned players.

LEMMA 10. *In the final phase, no player probes more than $O(B\log n)$ objects, with high probability.*

```
algorithm CalculatePreferences_p;
  1. For $d = 0$ to $\lceil \log n \rceil$ do:
     (a) Let $D = 2^d$.
     (b) Add each object independently with probability $O(\log n/D)$ to the sample set $S$.
     (c) Execute SmallRadius$(\Pi, S, 20 \ln n)_p$ on the sample set $S$.
     (d) Construct a neighbor graph and cluster the players into sets $V_1, V_2, \ldots, V_\ell$ of size at least $n/B$ and diameter at
         most $D$.
     (e) For each cluster $V_i$ and for each object $o$, repeat $\Theta(\log n)$ times: choose at random one of the players in $V_i$ to
         probe object $o$. Each player $p \in V_i$ sets $w(p_i)_d$ to the value that is discovered by a majority of the players that
         probe $o$.
  2. Each player $p$ executes RSelect$(w(p))_p$, and outputs the result.
```

Figure 2: High-Level Description of the CalculatePreferences Algorithm

Proof. A player p probes each object with probability at most $O(B\log(n)/n)$, as every cluster is of size at least n/B. The expected number of probes per player is $O(B\log(n))$, and hence by a Chernoff bound, with high probability no player makes more than $O(B\log n)$ probes. □

6.7 Concluding Claims.

In the absence of malicious players, ComputePreferences is asymptotically optimal with respect to a budget of B, as it successfully identifies a cluster of size n/B with asymptotically minimum diameter (see Lemma 9), and uses this cluster to generate the output preference vector. We discuss this in more detail in Section 7, while considering the behavior of malicious players. We now examine the probe complexity of the entire protocol:

Lemma 11. *In ComputePreferences, no player makes more than $O(B\log^{O(1)} n)$ probes, whp.*

Proof. Each player repeats the protocol $O(\log n)$ times for varying choices of diameter D. In the SmallRadius algorithm, the diameter is $20\ln(n)$, and hence, by Theorem 5, in each iteration, the number of probes per player is $O(B\log^{3.5} n)$. In the final phase, by Lemma 10, each player makes $O(B\log n)$ probes. Finally, the RSelect protocol requires each player to make $O(\log^3 n)$ additional probes. □

Lastly, we argue that the protocol outputs preference vectors that are near to the real preference vectors. This follows from the observation that every cluster has diameter $O(D)$:

Lemma 12. *Let D be the minimum integer such that for every player p there exists a set of at least n/B players, including p, with diameter no greater than D. Then for every player p: $|w(p) - v(p)| = O(D)$.*

Proof. Let d be the smallest integer such that $2^d \geq D$. We consider the iteration where $D' = 2^d$ is the target diameter. Fix a player p, and let V_j be the cluster produced by the protocol containing p. Let s be the number of players in V_j. For object o_i, let x_i be the number of players q where $v(q) \neq v(p)$. By Lemma 9, we know that V_j has diameter $O(D')$, and hence $\Sigma_{i=1}^n x_i = O(sD')$.

We now argue that if $x_i < s/3$, then, with high probability, v's output is correct, i.e., $v(p)_i = w(p)_i$. In particular, there are $\Theta(\log n)$ objects assigned to probe object o_i. If $x_i < s/2$, then in expectation, at least $2/3$ of these probes return the correct value for p. Thus by a Chernoff bound, with high probability, at least a majority of the probes return the right value for p.

It remains to bound the number objects o_j for which $x_j \geq s/3$. Since we have $\Sigma_{i=1}^n x_i = O(sD')$, by a simple counting argument there cannot be more than $O(D')$ objects for which v outputs an incorrect preference, with high probability. Thus $|v(p) - w(p)_d| = O(D') = O(D)$. Finally, by Theorem 3, since one of the candidate vectors $w_1, \ldots, w_{\log n}$ is within distance $O(D)$ of $v(p)$, we know that the final output $w(p)$ is within distance $O(D)$ of $v(p)$. □

7. DISHONEST PLAYERS

We now address the problem of dishonest players. The algorithm presented in Section 6 has already been designed to mostly prevent dishonest disruption. There are two issues that must be resolved. First, the algorithm presented in Section 6 (along with the building block algorithms in Section 5) rely on random choices that are agreed upon by all honest players. In Section 7.1, we discuss how to generate such randomness in the presence of dishonest players. Second, the dishonest players may attempt to hijack the clusters formed by the CalculatePreferences protocol. In Section 7.2, we show that the CalculatePreferences protocol is still asymptotically (almost) optimal even if up to $n/(3B)$ of the players are dishonest.

7.1 Generating Shared Random Numbers

The CalculatePreferences protocol depends on random choices that are known to all players. For example, in step (1.b), the protocol selects a sample set S uniformly at random; and in step (1.e), players are randomly assigned objects to evaluate. If the dishonest players can bias these random choices, then the predicted preferences output by the protocol may be inaccurate.

Basic idea.

Consider the following simple solution. Prior to executing the CalculatePreferences protocol, the players collaboratively elect a leader at random via a protocol that guarantees an honest leader with constant probability (as discussed below). The leader then writes a set of randomly chosen bits to the public bulletin board, and these are used as a shared source of randomness throughout the protocol.

With constant probability, the leader is honest and these bits are truly random.

This entire process of choosing a leader and executing CalculatePreferences is iterated $\Theta(\log n)$ times, generating $\Theta(\log n)$ candidate output vectors. With high probability, at least one of these candidate vectors is generated from an execution in which an honest participant was elected leader. The players then execute RSelect to choose the best vector, ensuring that the final output is sufficiently accurate. (Note that RSelect is run locally at each player and does not depend on shared randomness.)

Electing an honest leader.

There exist in the literature several *Byzantine-tolerant* leader election algorithms (for shared-memory) that guarantee an honest leader is elected with constant probability. Here, we focus on the leader election protocol proposed by Feige [10] as a particularly nice solution to the problem.

The leader election protocol assumes that $(1+\delta)n/2$ players are honest, which is the case for all $B \geq 1$ as there are at most $n/(3B)$ dishonest players. The protocol also assumes a *full information* model in which each player can broadcast her information to all other participant; this is easily implemented using the shared bulletin board. The guarantee is that with probability $\Omega(\delta^{1.65})$, an honest leader is elected.

The idea underlying the protocol is to model the situation as a balls-and-bins game. Each player has a ball that it throw at random into one of the bins. The players that choose the lightest bin proceed to the next round, while the other players are eliminated. After repeating a sufficient number of times, one leader remains. The key principle ensuring correctness is that the lightest bin will have approximately the same fraction of honest players as the original set of players; the dishonest players cannot bias the fraction of honest player too much, as if they disproportionately join the lightest bin, it will cease to be the lightest.

7.2 Analysis with Dishonest Players

In this section, we show that the CalculatePreferences protocol achieves the same (asymptotic) rate of error, even when there up to $n/(3B)$ dishonest players, as long as the shared random bits are unbiased. We assume that the dishonest players may be colluding to bias the results.

Specifically we show that the dishonest players cannot bias the predicted preferences for too many objects, and thus they cannot increase the asymptotic rate of error. Recall that each cluster assigned $\Theta(\log n)$ players to probe each object (in step (2.e)), and these players vote on the predicted preference. If all the honest players in a cluster have the same preference for an object, it is easy for the honest players to "out-vote" the dishonest players since at most 1/3 of the players in a cluster are dishonest. The hard case, however, is when there are roughly the same number of players in a cluster that like an objects as dislike it. In that case, the votes of the dishonest players can bias the prediction. The crux of the argument, then, is that within each cluster, there are only $O(D)$ objects in which the honest players have significant disagreement as to the predicted preference.

Lemma 13. *For each cluster constructed in step (1.d), the dishonest players can influence the predicted preferences for at most $O(D)$ objects.*

Proof. Consider cluster V_i. Let X be the set of dishonest players that are part of this cluster. We examine their impact on step (1.e), the probing phase, focusing on a particular object o. Let A_1 be the set of players in V_i that are honest and like object o; let A_0 be the set of players in V_i that are not malicious and dislike object o. Without loss of generality, assume that $|A_1| \geq |A_0|$.

We now show that if $|A_1| > 5|A_0|$, then the result of the probing phase for object o in cluster V_i is 1, with high probability. Since $|V_i| \geq n/B$, as per Lemma 9, and $|X| \leq n/(3B)$, we know that $|A_0 \cup A_1| \geq 2|V_i|/3$. Assuming $|A_1| > 5|A_0|$, we can say that $|A_1| \geq 5|V_i|/9$. In step (1.e), object o is assigned at random (as the shared random bits are unbiased) to $c \log n$ players (for some constant c). Hence in expectation at least $5c \log n/9$ players in A_1 are assigned to probe object o. Thus, with high probability, the majority of players probing o are in A_1, and hence the dishonest players have no impact on the prediction. The symmetric claim also holds: if $|A_0| > 5|A_1|$, then the result of the probing phase for object o in cluster V_i is 0, with high probability, again preventing the dishonest players from having any impact.

We say object o is zero-strange if $5|A_0| \geq |A_1| \geq |A_0|$, and one-strange if $5|A_1| \geq |A_0| \geq |A_1$. We now show that there are at most $O(D)$ zero-strange objects, using the *double counting* method. (The symmetric claim about one-strange objects follows by the same argument.) We count the number of triples (p_1, p_2, o) where p_1 and p_2 are two honest players, and o is an object on which p_1 and p_2 disagree in their preferences. Since the number of differing preferences between any pair of players is $O(D)$ (by Lemma 9), the total number of such triples in a cluster is $O(|V_i|^2 D)$. On the other hand, for any strange object, we know that $|A_0| + |A_1| \geq 2|V_i|/3$, so we have that $|A_1| \geq |V_i|/3$, and $|A_0| \geq |A_1|/5 \geq |V_i|/15$. We conclude that the number of triples for a given object o is $|A_1||A_0| = \Omega(V_i^2)$, implying that there are at most $O(D)$ zero-strange objects. By the same logic, there are at most $O(D)$ one-strange objects. We conclude that the dishonest players can only impact a player's preferences with respect to $O(D)$ objects. □

As a final remark, it is relatively easy to see that the dishonest players cannot significantly impact the ZeroRadius and SmallRadius routines that are used as building blocks: in each case, all that matters is that there are sufficiently many honest players that participate to ensure a good outcome. We can now conclude with the main theorem of the paper:

Theorem 14. *The CalculatePreferences protocol is a $O(B \log^{O(1)} n)$-budget collaborative scoring algorithm that is asymptotically optimal with respect to a budget of B.*

8. CONCLUSION

In this paper, we have presented an algorithm for collaborative scoring that can tolerate some fraction of the players acting dishonestly. Moreover, the protocol is almost optimal in the sense that, given a budget of $O(B \log^{O(1)} n)$, it performs asymptotically as well as any algorithm with a budget of B, in the worst case.

There are several interesting open questions. One observation is that we have proposed a relatively restricted definition of collaborative scoring in order both to simplify the presentation and to make the analysis tractable. For example, players are restricted to binary preferences; in reality, players may rate items on a numerical scale. As a related example, we use the Hamming distance to measure the similarity of two players' preferences; in a real system (with non-binary preferences), other metrics may be more useful. We believe that many of the techniques developed in this paper generalize to these more realistic settings: the basic idea of using sampling to cluster players does not rely on these particular assumptions.

Another interesting scenario concerns the situation where different players have different budgets. For example, some players may be willing to probe a large number B_{big} of objects, while other players may be willing to probe only a small number B_{small} of objects. Again, the techniques in this paper should generalize to such a scenario: each cluster must be chosen to contain a sufficient total number of queries among all the members.

Another aspect to collaborative scoring is the communication complexity. Here, we assume a shared bulletin board which is, effectively, free to access. In reality, there are costs to sharing information, either via sending messages or accessing a shared memory. It remains an open question to minimize the underlying communication costs.

More technically, there are two obvious gaps. First, do we really need the augmented budgets to achieve asymptotic optimality? That is, can we develop an algorithm that uses only $O(B)$ queries, and yet is still asymptotically optimal with respect to a budget of B? Second, can we show, formally, a stronger lower bound on the worst-case performance? The notion of optimality in this paper is worst-case: there is a distribution of preferences for which it is impossible to do better. Yet we conjecture that there is a stronger notion of optimality: *for every* distribution of preferences, a player p can do no better than, say, the median distance to the closest n/B others.

Finally, there remains the question of tolerating a larger number of dishonest players. The key requirement, here, is that not too many malicious players are included in any one cluster. By preventing the malicious players from biasing the selection of clusters, it seems possible to adapt the techniques developed here to tolerate more dishonest players.

9. REFERENCES

[1] D. Achlioptas and F. McSherry. *Fast computation of low rank approximations.* In *Proceedings of the 33rd Annual Symposium on Theory of Computing*, pp, 611–618, 2001.

[2] N. Alon, B. Awerbuch, Y. Azar, and B. Patt-Shamir. *Tell me who I am: An interactive recommendation system.* In *Proceedings of the 18th Annual Symposium on Parallelism in Algorithms and Architectures*, pp. 1–10, 2006.

[3] N. Alon, B. Awerbuch, Y. Azar, and B. Patt-Shamir. *Tell me who I am: An interactive recommendation system.* Theory of Computing Systems, 45(2): 261–279, August, 2009.

[4] B. Awerbuch, Y. Azar, Z. Lotker, B. Patt-Shamir, and M. Tuttle. *Collaborate with strangers to find own preferences.* In *Proceedings of the 17th Annual Symposium on Parallelism in Algorithms and Architectures*, pp. 263–269, 2005.

[5] B. Awerbuch, B. Patt-Shamir, and D. Peleg. *Improved recommendation systems.* In *Proceedings of the 16th Annual Symposium on Discrete Algorithms*, pp. 1174–1183, 2005.

[6] Y. Azar, A. Fiat, A. Karlin, F. McSherry, and J. Saia. *Spectral analysis of data.* In *Proceedings of the 33rd Annual Symposium on Theory of Computing*, pp. 619–626, 2001.

[7] P. Drineas, A. Frieze, R. Kannan, S. Vempala, and V. Vinay. *Clustering in large graphs and matrices.* In *Proceedings of the 10th Annual Symposium on Discrete Algorithms*, pp. 291–299, 1999.

[8] P. Drineas and R. Kannan. *Fast monte-carlo algorithms for approximate matrix multiplication.* In *Proceedings of the 42nd Annual Symposium on Foundations of Computer Science*, p. 452, 2001.

[9] P. Drineas, I. Keredinis, and P. Raghavan. *Competitive recommendation systems.* In *Proceedings of the 34th Annual Symposium on Theory of Computing*, pp. 82–90, 2002.

[10] U. Feige. *Non-cryptographic selection protocols.* In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, p. 142, 1999.

[11] J. Kleinberg and M. Sandler. *Using mixture models for collaborative filtering.* In *Proceedings of the 36th Annual Symposium on Theory of Computing*, pp. 569–578, 2004.

[12] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. *Recommender systems: A probablistic analysis.* In *Proceedings of the 39rd Annual Symposium on Foundations of Computer Science*, p. 664, 1998.

[13] A. Nisgav and B. Patt-Shamir. *Finding similar users in social networks.* In *Proceedings of the 21st Annual Symposium on Parallelism in Algorithms and Architectures*, pp. 169–177, 2009.

[14] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao. *SybilLimit: A near-optimal social network defense against sybil attacks.* In *Proceedings of the Symposium on Security and Privacy*, pp. 3–17, 2008.

[15] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman. *SybilGuard: Defending against sybil attacks via social networks.* Transactions on Networking, 16(3):576–589, June 2008.

[16] H. Yu, C. Shi, M. Kaminsky, P. B. Gibbons, and F. Xiao. *DSybil: Optimal sybil-resistance for recommendation systems.* In *Proceedings of the 30th Annual Symposium on Security and Privacy*, pp. 283–298, 2009.