Collaboration of Untrusting Peers with Changing Interests

(Extended Abstract)

Baruch Awerbuch* Boaz Patt-Shamir† David Peleg‡ Mark Tuttle§

Abstract

Electronic commerce engines like eBay depend heavily on reputation systems to improve customer confidence that electronic transactions will be successful, and to limit the economic damage done by disreputable peers defrauding others. In a reputation system, participants post information about every transaction, and routinely check the posted information before taking any action to avoid other participants with a bad history. In this paper, we introduce a framework for optimizing reputation systems for objects. We study reputation systems in an asynchronous setting, and in the context of restricted access to the objects. Specifically, we study the cases where access may be restricted in time (objects arrive and depart from system) and in space (each peer has access to only a subset of the objects).

*Computer Science Department, Johns Hopkins University, 3400 N. Charles Street, Baltimore, MD 21218. Supported by NSF grants ANIR-0240551 and CCR-0311795. Research conducted while visiting CRL. Email: baruch@acm.org.

†Cambridge Research Laboratory, HP Labs, One Cambridge Center, Cambridge, MA 02142. On leave from Tel Aviv University. Email: boaz@hp.com.

‡Department of Computer Science and Applied Mathematics, The Weizmann Institute of Science, Rehovot 76100, Israel. Research conducted while visiting CRL. Email: david.peleg@weizmann.ac.il.

§Cambridge Research Laboratory, HP Labs, One Cambridge Center, Cambridge, MA 02142. Email: mark.tuttle@hp.com.

EC'04, May 17–20, 2004, New York, New York, USA.

Categories and subject descriptors: H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval — Information filtering; H.3.5 [Information Storage and Retrieval]: Online Information Services — Commercial services, Web-based services.
General terms: Algorithms, Reliability, Theory.
Keywords: Electronic commerce, reputation systems, peer-to-peer systems.

1 Introduction

The Internet clearly has a dark side. The junk mail filling our email folders is compelling evidence of the number of scam artists ready to take advantage of us, and the evening news shows discuss predators in chat rooms so frequently it is a wonder we communicate with any strangers electronically. On the other hand, most people feel quite comfortable buying and selling things over the web at eBay. This level of comfort with eBay is quite remarkable, and flows directly from eBay's elaborate reputation system [12]: After every transaction, the system invites each party to post its rating of the transaction on a public billboard the system maintains. Consulting the billboard is a key step before making a transaction.

Nonetheless, the possibility of fraud is everywhere with on-line commerce. A frequently cited scenario involves a group of sellers engaging in phony transactions, and rating these transactions highly to generate an appearance of reliability while ripping off other people. Another scenario involves a single seller behaving in responsible manner long enough to entice an unsuspecting buyer into a single large transaction, and then vanishing. Reputation systems are valuable, but not infallible.

In this paper, we introduce the following simple model for reputations systems. In our model, there are players and there are objects. Some of the objects are good and some are bad, and a player can

probe an object at some cost to determine whether the object is good. The goal of the players is to find a good object with as few probes as possible. Players collaborate by posting the results of their probes on a public billboard, and consulting the board when choosing an object to probe. The problem is that some of the players are dishonest, and can behave in an arbitrary fashion, including colluding and posting false reports on the billboard to entice honest players to probe bad objects.

We assume that players are asynchronous, and may probe objects at completely different rates. The objects may enter and leave the system over time, or they may be physically separated and inaccessible to some of the peers. In other words, the players' access to objects may be restricted in both time and space.

For example, imagine a manufacturing company buying parts from a set of suppliers day after day. Some suppliers sell reliable parts, and others sell defective parts. Some suppliers go out of business, and others enter the market. The manufacturer has to buy parts from suppliers without knowing the actual quality of the parts until buying a few and examining them. The supplier might offer refunds for defective parts, but there is still a cost to buying defective parts and dealing with unreliable suppliers. Manufacturers might even form an association — a sort of better business bureau — to keep track of suppliers and experience with supplier quality and supplier recommendations, but even such associations are vulnerable to manipulation via malicious slander by members. The manufacturer wants to buy its parts with as few headaches as possible.

As another example, consider manufacturing depending on "just-in-time" delivery of parts. In this case, physical proximity of the supplier could be important. Suppliers between Boston and New York might be able to sell into both markets, and similarly for suppliers between New York and Philadelphia, but suppliers near Boston are of no use to manufacturers near Philadelphia. New York might be willing to take recommendations from Boston and Philadelphia, but Boston might not find recommendations from Philadelphia to be helpful.

Contributions. In this paper, we make the following contributions.

We propose several algorithms based on a simple rule we call the BALANCED rule. There are several approaches a player might take to finding a good object. One strategy is to follow an *"exploration rule"* which says that a player should choose an object at random (uniformly) and probe it. This might be a good idea if there are a lot of good objects, or if there are a lot of dishonest players posting inaccurate reports to the billboard. Another strategy is to follow an *"exploitation rule"* which say that a player should choose another player at random and probe whichever object it recommends (if any), thereby exploiting or benefiting from the effort the other player has already put in to finding a good object. This might be a good idea most of the players posting recommendations to the billboard are honest. In many natural situations, however, the player will not know how many honest players or good objects are in the system. In this case, the natural course of action would be to balance between the two approaches as follows.

> **The BALANCED Rule**: Flip a coin. If the result is "heads," probe a random object. Otherwise, select a random player. If that player recommends an object, probe it, else probe a random object. (A player recommends any of the good objects it has probed, or says that it has found no good objects.)

We study variants of this rule in models where players have restricted access to the objects. This restricted access could be due to changes in the set of objects available or to different interests or physical access. We consider two such models:

- *The dynamic object model*: Objects can enter and leave the system over time.
- *The partial access model*: Each player has access to a different subset of the objects.

We show how variants of the BALANCED rule can be used to solve the dynamic and partial access problems described above. Our solutions work even in the presence of Byzantine players that collude in arbitrary ways to conspire against the honest players. We analyze our solutions in terms of the expected cost of the probes by the honest players in their search for good objects, which is just the number of times the honest players probe a bad object.

- For the dynamic object model in Section 3, where the set of object changes over time, we give a solution based on the BALANCED rule, and we prove a lower bound on the problem showing that our solution is very close to optimal.
- For the partial access model in Section 4, where a player has access only to a subset of the objects, we give a solution based on the BALANCED rule with nearly optimal cost. The most efficient solution one could hope for would be comparable

to a centralized solution. The cost of our solution matches a centralized solution for any group of players with a common interest (with overlapping subsets of objects). We also discuss some of the subtle issues involved in getting help from other players in the partial access model.

We have also subjected our algorithms to simulation on quite large instances, and our algorithms perform quite well in practice on the instances we have considered. In Section 5, we describe one large simulation demonstrating that our algorithms are quite resilient to the bias of the coin used by the BALANCED rule. Simulation also demonstrates that some balance is essential, in that neither the exploration nor exploitation rule functions well on its own.

Finally, in Section 6, we take some initial steps in the direction of a theory of trust in the context of reputation systems. We consider a model in which this game of finding good objects is repeated many times, and show that in some situations we can bound the amount of damage the dishonest players inflict on the honest players by a logarithmic factor.

Related work. In "collaborative filtering" (see, e.g., [5]), all players are honest but happen to have different tastes. The model proposed by [9] (and see also [1, 3, 8]) can be viewed as follows. Players are divided into types, and for each player type there is a taste modeled by a probability distribution over the objects. The value (utility) of an object i for a player of a given type is taken to be the probability that a player of that type chooses i. The objective is to design a centralized algorithm, that provides each player with a reputation for an object that maximizes the player's utility (based on choices of other players, that appear to have the same taste). Being centralized, algorithms for this model are not designed to withstand (and therefore can be easily fooled by) malicious players.

Other examples of reputation systems are web-searching algorithms (see, e.g., [2, 7]). The problem with this approach in our context is that their performance criteria do not match ours: These algorithms essentially compute the "popular vote" of the participants, which may have little to do with the concrete cost we are interested in. For example, consider using Google's algorithm [2] to rank the trustworthiness of eBay's users. Then a group of crooks create a huge number of fictitious crony buddies, writing excellent recommendations for each other and slandering all honest sellers and buyers. In this case, "popularity" style algorithms would solidify the monopoly of the crooks. Empirical evidence to that effect is provided in the work by Kamvar *et al.* [6], which studies trust in the context of authenticity of files downloaded in peer-to-peer systems. Using a variant of Kleinberg's algorithm [7], they assign a trust value to each peer. They comment that this approach is useful only if there are nodes that are known *a priori* to be trustworthy. Otherwise, "forming a malicious collective in fact heavily boosts the trust values of malicious nodes."

One can think of using Byzantine agreement protocols (see [11, 4] and the survey [10]). However, our problem is different. Our goal is to minimize the cost of acquiring input values. In contrast, the common settings of Byzantine agreement protocols are based on the assumption that each player is given all the inputs in advance, for free, rendering them meaningless for our purpose. Moreover, if dishonest players are majority, reaching Byzantine agreement is outright impossible.

2 Model

Our model begins with a set of n players and m objects. Some of the players are *honest* and some are *dishonest*. Some of the objects are *good* and some are *bad*. A basic step of a player is to *probe* an object to learn whether the object is good or bad. The *cost* of a probe is 1 if the object is bad and 0 if it is good. The goal of a player is, in general, to find a good object while incurring minimal cost.

To facilitate collaboration among the players in their search for a good object, the system maintains a *billboard* where players can post the results of their probes. By convention, each time a player p probes an object o, it posts the result of the probe in the (p, o) entry of the billboard. (Alternatively, we could have each player maintain a single entry of the billboard containing the name of some good object it has seen so far.) We assume that only the player p can write to an entry of the form (p, o). We also assume that entries are write-once, meaning that entries do not change once written, and that the billboard is reliable.

An execution of the system consists of a sequence of player steps. In each step, one player reads the billboard, optionally probes an object, and writes to the billboard. We assume that players are following a randomized protocol that chooses the object to probe based on the contents of the billboard. Honest players are required to follow the protocol, but dishonest players are allowed to behave in an arbitrary (or Byzantine) fashion, including posting incorrect information on the billboard. We think of the dishonest

players as being under the control of a malicious adversary that determines which objects to probe and what values to post.

An execution of an algorithm is uniquely determined by the algorithm, the coins flipped by the players while executing the protocol, and by three external entities:

1. the *player schedule* that determines the order in which players take steps,
2. the *dishonest players*, and
3. the *adversary* that determines the behavior of the dishonest players.

Our adversary is quite powerful, and may behave in an adaptive, Byzantine fashion. Formally, the adversary is a function from a sequence of coin flips to a sequence of objects for each dishonest player to probe and the results for the player to post on the billboard. The sequence of coin flips is the entire sequence of coins flipped during the execution, even including the coins flipped in the future. With this information, the adversary can reconstruct the entire state of the system at any point in time, and choose the next move for a dishonest player.

An *operating environment* is a triple consisting of a player schedule, a set of dishonest players, and an adversary. The operating environment encapsulates all of the nondeterministic choices made during an execution, leaving only the probabilistic choices to consider. When we deal with probabilities or compute expected values, we fix an operating environment and consider the distribution of executions with this operating environment induced by the coin flips. We note that in our model, a player must probe an object whenever it is scheduled. This assumption is made to disallow trivial protocols that never probe any objects, and could be replaced by the requirement that each player must eventually find a good object.

Each execution of the system yields a *probe sequence* consisting of the sequence of objects probed during the execution. When we compute the cost of a probe sequence, we count only the probes by the honest players, and ignore the probes by the dishonest players. The *cost* of a probe sequence is the number of probes of bad objects by honest players. Our goal is to minimize the cost for the honest players.

3 The dynamic object model

In this section we consider the dynamic model, where objects arrive and depart arbitrarily. We we present a simple on-line algorithm based on the BALANCED rule, and prove that the cost incurred to the honest players under this algorithm is nearly the best possible by any on-line algorithm.

We define the *dynamic object model* by extending the model of Section 2 in the following ways. Each step of an execution consists of two parts. First the set of objects changes, and then a player probes an object. The arriving and departing objects are globally announced, but not their values. We extend the operating environment in this model to include

4. the *object schedule* that determines when objects enter and leave the system, and their values.

We use m to denote an upper bound on the number of objects concurrently present in the system, and let β denote a lower bound on the fraction of good objects at any time, for some $0 \leq \beta \leq 1$.

3.1 Algorithm

The algorithm is an immediate application of the BALANCED rule from the introduction:

> **Algorithm** DYNALG: If the player has found a good object, then probe it again. If not, then apply the BALANCED rule.

In the analysis of DYNALG, it turns out that the following concept plays a central role.

Definition 3.1 *Given a probe sequence σ, $switches(\sigma)$ denotes the number of distinct objects in σ.*

Given an operating environment $\mathcal{E}$, let $\sigma_{\mathcal{E}}(\text{DYNALG})$ be the random variable whose value is the probe sequence of the honest players generated by DYNALG under $\mathcal{E}$. The next theorem says that the cost of DYNALG is very close to the cost of an optimal probe sequence σ^* for the honest players

Theorem 3.1 *For every operating environment $\mathcal{E}$ and every probe sequence σ^* for the honest players, the expected cost of $\sigma_{\mathcal{E}}(\text{DYNALG})$ is at most*

$$cost(\sigma^*) + switches(\sigma^*) \cdot (2 - \beta)(m + n \ln n)).$$

Proof: Fix an operating environment $\mathcal{E}$. Denote $\sigma = \sigma_{\mathcal{E}}(\text{DYNALG})$. Partition the sequence σ^* into subsequences $\sigma^* = \sigma_1^* \sigma_2^* \cdots \sigma_K^*$ such that for all $1 \leq i < K$, we have (1) all probes in σ_i^* are to the same object, and (2) the objects probed in σ_i^* and σ_{i+1}^* differ. Without loss of generality we may assume that for any $i \neq i'$, the object probed in σ_i^* is distinct from the object probed in $\sigma_{i'}^*$. It follows that $K = switches(\sigma^*)$. Next, partition σ into $\sigma = \sigma_1 \sigma_2 \cdots \sigma_K$, such that $|\sigma_i^*| = |\sigma_i|$ for all $1 \leq i \leq K$. Note that by definition, corresponding probes in σ_i^* and σ_i occur at the same step.

Fix an index $1 \le i \le K$, and consider the difference $cost(\sigma_i^*) - cost(\sigma_i)$. If the probes in σ_i^* are to a bad object, then trivially $cost(\sigma_i) \le cost(\sigma_i^*)$. To finish the proof, we show that if all probes in σ_i^* are to a good object, then $cost(\sigma_i) \le (2-\beta)(m + n \ln n)$.

So assume henceforth that all probes in σ_i^* are to a good object. Call an object *i-persistent* if it is good and if it is present in the system throughout the duration of σ_i^*. Since σ_i^* and σ_i occur at the same time, an i-persistent object is also available to DynAlg during σ_i. Call a probe *i-persistent* if it probes an i-persistent object. Partition the sequence σ_i into n subsequences $\sigma_i = D_0^i D_1^i D_2^i \cdots D_n^i$, where D_k^i consists of all probes in σ_i that are preceded by i-persistent probes of *exactly* k distinct honest players. In other words, the sequence D_{k-1}^i ends with the kth i-persistent probe of a new honest player, if it exists. If there are only k players doing i-persistent probes in σ_i, then $D_{k+1}^i, \ldots, D_n^i$ (and possibly D_k^i too) will be empty. Obviously, $cost(\sigma_i) = \sum_{k=0}^{n} cost(D_k^i)$.

Call a probe by a player *fresh* if the player's prior probe was to an object that is not currently a good object (meaning the player is now searching for a good object). The cost of a sequence is exactly the cost of the fresh probes.

Let us compute the expected cost of D_k^i by conditioning on the number of fresh probes in D_k^i. The expected cost of a single fresh probe in D_k^i is at most $1 - \frac{\beta}{2}$, since a player chooses to probe a random object with probability 1/2, and the object probed is good with probability at least β. The expected cost of ℓ fresh probes is therefore at most $(1-\frac{\beta}{2})\ell$. Each fresh probe in D_k^i finds a persistent object with some probability p_k (that we will compute in a moment). The probability that D_k^i contains exactly ℓ fresh probes is therefore $(1-p_k)^{\ell-1} p_k$. Putting this together, and conditioning on the number ℓ of fresh probes in D_k^i, it follows that the expected cost of D_k^i is at most

$$\sum_{\ell \ge 1} \left(1 - \frac{\beta}{2}\right) \ell \, (1-p_k)^{\ell-1} \, p_k = \left(1 - \frac{\beta}{2}\right) \frac{1}{p_k}.$$

Now let us compute the value of p_k. For $k = 0$, we have $p_0 \ge 1/2m$, since there is at least one i-persistent object (the object probed in σ_i^*), so each fresh probe in D_0^i finds an i-persistent object with probability at least $1/2m$. For $k > 0$, we have $p_k \ge k/2n$, since we have k probes to i-persistent objects preceding every probe in D_k^i, so every fresh probe goes to an i-persistent object with probability at least $k/2n$.

It follows that the expected cost of D_0^i is at most $(1-\frac{\beta}{2})2m$ and of D_k^i is at most $(1-\frac{\beta}{2})\frac{2n}{k}$, so the expected cost of σ_i is at most

$$\begin{aligned} & \mathbf{E}\left[cost(D_0^i)\right] + \sum_{k=1}^{n} \mathbf{E}\left[cost(D_k^i)\right] \\ \le \;& 2m(1-\frac{\beta}{2}) + \sum_{k=1}^{n} \frac{2n}{k} \cdot (1-\frac{\beta}{2}) \\ = \;& (2-\beta)(m + n \ln n) \; . \end{aligned}$$ ∎

Corollary 3.2 *The total cost incurred to all honest players using Algorithm* DynAlg *in the* static *object model is* $O(m + n \log n)$.

3.2 Lower Bound

We show that our algorithm is nearly the best possible, in the sense that there are scenarios in which any randomized algorithm has cost close to the upper bound guaranteed by Theorem 3.1. For simplicity of notation, we consider the case where at any given time there are at most $m+1$ objects, of which at least $\beta m + 1$ are good.

Theorem 3.3 *For any randomized algorithm R for the dynamic model and for any given m and $0 \le \beta < 1$, there exists an operating environment $\mathcal{E}$ and a probe sequence $\sigma_{\mathcal{E}}^*$ such that the expected cost of $\sigma_{\mathcal{E}}(R)$ is at least*

$$cost(\sigma_{\mathcal{E}}^*) + switches(\sigma_{\mathcal{E}}^*) \cdot \frac{1-\beta}{4} \cdot m \; .$$

We remark that the bound above holds even for centralized algorithms, where the set of honest players is common knowledge *a priori*.

Proof: By Yao's Lemma [13], it is sufficient to prove the lower bound hold for any deterministic algorithm, on average, for some probability distribution over operating environments. Define a probability distribution over a set of operating environments as follows. In all environments in the set, players are scheduled in round-robin fashion, and objects schedules consist of a sequence of K segments, each of length $\frac{m(1+\beta)}{1-\beta}$ probes. In each segment there are *persistent* and *transient* objects. Persistent objects arrive when the segment starts and depart when it ends. Transient objects arrive at some step and depart at the following step (so they are available for probing only once). In each segment, we have

- One persistent good object (with cost 0), denoted i_0.
- A set B_p of $m(1-\beta)/2$ persistent bad objects (with cost 1).
- In each time step t of the segment, a set C_t of $m(1+\beta)/2$ transient objects arrive, of which $m\beta$ are good and the other $m(1-\beta)/2$ are bad.

The probability distribution is defined by picking uniformly at random a permutation of the IDs of all persistent objects $\{m_0\} \cup B_p$, and a permutation of the IDs of each set of transient objects C_t. (Since any algorithm can discern persistent objects from transient objects, we might as well keep their name spaces disjoint.)

Now, fix any deterministic algorithm A and consider its behavior on an environment picked at random according to the distribution defined above. Since all objects are replaced when a new segment starts, we may consider each segment separately. We argue that the expected cost incurred by A in a segment is $\Omega((1-\beta)m)$. Let $\mathcal{I}_0$ denote the event that i_0, the single persistent good object, is eventually probed by A. First, consider the case where $\mathcal{I}_0$ holds, i.e., the object i_0 is eventually probed. Let t_0 be the number of probes of objects in B_p until A probes i_0 for the first time. Since each probe of an object in B_p has unit cost, the cost incurs to A in this case is at least t_0. Since i_0 has a random ID among all $|B_p|+1$ persistent objects, we have that

$$\mathbf{E}\left[t_0 \mid \mathcal{I}_0\right] \;=\; \frac{|B_p|+1}{2} \;=\; \frac{m(1-\beta)}{4}\;. \qquad (1)$$

Next, consider the case $\neg\mathcal{I}_0$, in which A never probes i_0. Let X_p denote the number of times A probes a persistent object from B_p. Then the number of probes of transient objects is the length of the segment, i.e., $\frac{m(1+\beta)}{(1-\beta)} - X_p$. Observe that the expected cost of probing a transient object is always $\frac{1-\beta}{1+\beta}$, because these objects are chosen at random, and the probes are independent trials. Since probing any persistent object in B_p costs one unit, we have

$$\begin{aligned}
&\mathbf{E}\left[\mathit{cost}(\sigma(A)) \mid \neg\mathcal{I}_0\right] \\
&\quad= \left(\frac{m(1+\beta)}{1-\beta} - \mathbf{E}\left[X_p \mid \neg\mathcal{I}_0\right]\right)\cdot\frac{1-\beta}{1+\beta} \\
&\qquad+ \mathbf{E}\left[X_p \mid \neg\mathcal{I}_0\right] \\
&\quad= \; m + \mathbf{E}\left[X_p \mid \neg\mathcal{I}_0\right]\cdot\frac{2\beta}{1+\beta} \\
&\quad\geq m\;. \qquad (2)
\end{aligned}$$

Combining Eq. (1) and Eq. (2), we get that the expected cost incurred by any deterministic algorithm A in a single segment is

$$\begin{aligned}
&\mathbf{E}\left[\mathit{cost}(\sigma(A))\right] \\
&\quad= \; \mathbf{P}\left[\mathcal{I}_0\right]\cdot\mathbf{E}\left[\mathit{cost}(\sigma(A)) \mid \mathcal{I}_0\right] \\
&\qquad+ \; \mathbf{P}\left[\neg\mathcal{I}_0\right]\cdot\mathbf{E}\left[\mathit{cost}(\sigma(A)) \mid \neg\mathcal{I}_0\right] \\
&\quad\geq \; \mathbf{P}\left[\mathcal{I}_0\right]\cdot\frac{m(1-\beta)}{4} \quad + \quad \mathbf{P}\left[\neg\mathcal{I}_0\right]\cdot m \\
&\quad\geq \; \frac{m(1-\beta)}{4}\;.
\end{aligned}$$

Thus, the expected cost of any deterministic algorithm in the entire execution is at least $Km\frac{1-\beta}{4}$ (recall that K is the number of segments in the environments in the set defined above). Applying Yao's Lemma, we can conclude that for any randomized online algorithm R, there exists an environment $\mathcal{E}_R$ from the set defined above such that the expected cost of R on $\mathcal{E}_R$ is at least $Km\frac{1-\beta}{4}$. To complete the proof, note that given $\mathcal{E}_R$, we can define a sequence $\sigma^*_{\mathcal{E}_R}$ that always probes the current persistent good object in $\mathcal{E}_R$. For this sequence, we have $\mathit{cost}(\sigma^*_{\mathcal{E}_R}) = 0$ and $\mathit{switches}(\sigma^*_{\mathcal{E}_R}) = K$. ∎

4 The partial access model

In this section we study the partial access model, where each player is able to access only a subset of the objects. The main problem with this model is that in contrast to the full access model (where each player can access any object), when we have partial access, it is difficult to measure the amount of collaboration a player can expect from other players in searching for a good object.

Our approach to overcome this difficulty is to concentrate on the amount of *collective work* done by subsets of players. First we prove upper bounds on the collective work of a group of players; and then we describe an example that demonstrates that, in some cases, even groups with common interests cannot help each other too much.

Notation. We model the partial access to the objects with a bipartite graph $G = (P, O, E)$, where P is the set of players and O is the set of objects, and a player j can access an object i only if $(j,i) \in E$. For each player j, let $\mathrm{obj}(j)$ denote the set of objects accessible to j, and let $\deg(j) \stackrel{\text{def}}{=} |\mathrm{obj}(j)|$. For each honest player j, let $\mathrm{best}(j)$ denote the set of good objects accessible to j. Let $\mathbf{N}(j)$ be the set of all players (honest and dishonest) that are at distance 2 from a given player j, i.e.,

$$\mathbf{N}(j) = \{k \in V \mid \mathrm{obj}(k) \cap \mathrm{obj}(j) \neq \emptyset\}\;.$$

4.1 Algorithm

Our algorithm is just DynAlg from the dyanmic model, except that we must adapt the Balanced rule to the restricted access model. In the new rule, a player j flips a coin. If the result is "heads," it probes an object selected uniformly at random from $\mathrm{obj}(j)$. If the result is "tails," it selects a player k uniformly

at random from $\mathbf{N}(j)$ and probes the object k recommends, if any; and otherwise it probes an object selected uniformly at random from $\mathrm{obj}(j)$.

For this algorithm, we have the following result.

Theorem 4.1 *Let Y be any set of honest players. Denote $\deg^*(Y) = \max\{\deg(j) \mid j \in Y\}$, and $\mathbf{N}^*(Y) = \max\{|\mathbf{N}(j)| \mid j \in Y\}$. Let $X(Y) = \bigcap_{j \in Y} \mathrm{best}(j)$. If $X(Y)$ in nonempty, then the total work of players in Y is at most*

$$2\left(\frac{\deg^*(Y)}{|X(Y)|} + |\mathbf{N}^*(Y)| \cdot \ln|Y|\right) .$$

Proof Sketch: The proof follows the argument used in the proof of Theorem 3.1. The notion of "persistent object" used in the proof of Theorem 3.1 is replaced by "an object in $X(Y)$," with the following subtle difference. In the partial access model, a player $j \in Y$ may be satisfied with an object not in $X(Y)$; in this case, the probes of j will not help in adding votes to objects in $X(Y)$, but on the other hand j will not incur additional cost to Y. Details are omitted. ∎

Let us interpret the result of Theorem 4.1. Consider any set Y of players with common interest $X(Y)$ (meaning any object in $X(Y)$ would satisfy any player in Y). Very roughly, Theorem 4.1 says that from the point of view of a player, its load is divided among the members of Y: the total work done by the group working together is roughly the same as the work of an individual working alone. This intuition follows from the fact that the first term in the bound is just an upper bound on expected amount of work until a player finds an object in $X(Y)$, and the second term is an upper bound on the total number of recommendations (times a logarithmic factor) a player has to go through. This is pleasing, because it indicates that the number of probes is nearly the best one can hope for. In particular, if $\deg^*(Y) \geq \mathbf{N}^*(Y)\ln|Y|$, the overall bound on the work is optimal up to a constant factor.

4.2 Collaboration across groups without common interest

It is interesting to consider sets of players who do *not* share a common interest. Of course, one can partition them into subsets (call them SIGs, for "special interest groups"), where for each SIG there is at least one object that will satisfy all its members. Theorem 4.1 guarantees that each SIG is nearly optimal, in the sense that the total work done by a SIG is not much more than the total work that must be done even if SIG members had perfect coordination (thus disregarding dishonest players). However, the collection of

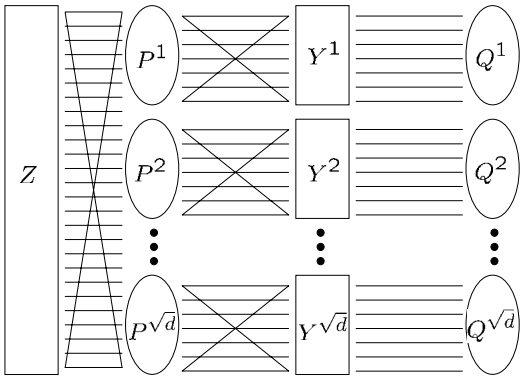

Figure 1: *Pathological example (see text). Bad objects and dishonest players are not shown.*

SIGs may be suboptimal, due to overlaps in the neighborhood sets (which contribute to the second term of the upper bound). It is natural to ask whether there always exists a "good" partition of players into SIGs, so that the overall work (summed over all SIGs) is close to optimal. The answer is negative in the general case, as the following example demonstrates.

Specifically, we show that it is possible that even if each good object would satisfy many honest players, the total amount of work, over all players, is close to the worst case (being the sum of work necessary if each player is working alone).

We define a parametric instance via a bipartite graph as follows (see Figure 1). Let $d > 0$ be a given parameter. The graph is regular with degree d. There are d^2 players, of which $2d^{3/2}$ are honest. Each honest player is connected to d objects, of which at least one is good. There are d^2 objects, of which $2d^{3/2}$ are good. Each object is connected to d players, such that each good object is connected to at least $\sqrt{d}$ honest players.

Nodes: Organize $2d^{3/2}$ honest players in $2d$ sets, $P^k = \{p_1^k, \ldots, p_{\sqrt{d}}^k\}$ for $k = 1, \ldots, d$ and $Q^k = \{q_1^k, \ldots, q_{\sqrt{d}}^k\}$ for $k = 1, \ldots, d$. Arrange $d^{3/2}$ good objects in d sets $Y^k = \{y_1^k, \ldots, y_{\sqrt{d}}^k\}$ for $k = 1, \ldots, d$. In addition, let Z be a set of $d^{3/2}$ good objects.

Edges: Connect each good object y_i^k to the honest players in $P^k \cup \{q_i^k\}$. In addition, connect every honest player in $P = \bigcup_k P^k$ to $d - \sqrt{d}$ additional good objects from Z. Use the dishonest players and bad objects to complete the graph into a d-regular graph.

Structure summary: By construction, every honest player in P is connected to d good objects and has no bad neighbors, and every honest player in $Q = \bigcup_k Q^k$ is connected to a single good object and $d-1$ bad objects. Moreover, the connections between the

honest players of Q and the good objects of Y are one-to-one, namely, no two players are connected to the same object.

Executions: Consider an execution. For concreteness, let us consider the round-robin schedule, where the execution proceeds in rounds. Clearly, all the honest players in P will probe a good object in the first round, and thus stop searching. Also, it is expected that a $1/d$ fraction of the honest players in Q will probe a good object in the first round, thus about $\sqrt{d}$ honest players from Q will quit. Also, for symmetry considerations, we may conclude that of the good objects in $Y = \bigcup_k Y^k$, a $1/\sqrt{d}$ fraction will be probed in the first round. We hereafter ignore the honest players connected to these d objects; we are willing to assume that those players also become satisfied soon after their first probe.

We now focus our attention on the remaining $\Omega(d^{3/2})$ honest players of Q, which after the first round continue to see d neighbors with no probes. Consider such an honest player q_j^k. It has $d-1$ bad neighbors, which will get no positive recommendations in the billboard. It has also a single good neighbor y_j^k. This good object has $\sqrt{d}$ additional good neighbors, p^k, which are now out of the game, and $d - \sqrt{d}$ bad neighbors, so it will also get no positive recommendations on the billboard. Hence the honest player q_j^k can only find its lone good neighbor by probing it itself, from among its d equally-looking neighbors. This will take, on average, $d/2$ steps. As this holds for $\Omega(d^{3/2})$ honest players of Q, the average work per honest player in this case is $\Omega(d)$.

Conclusion. In the example above, the average work per honest player is $\Omega(d)$. Note that since each player is incident to d objects, the mindless strategy of randomly sampling any accessible object results in $O(d)$ work per player in the worst case. Intuitively, this means that in some cases, honest players are getting much less help from their neighbors than the assistance one might hope for. While this example shows that we can't always expect help from collaboration, our simulations, described in the next section, show that collaboration is quite helpful in more natural examples.

5 Simulation

In our algorithm, the players flip a fair coin to determine whether to probe a random object or a random recommendation. One interesting question is how sensitive the algorithm is to the bias of this coin. Let p be the probability that a player chooses an object and $1-p$ be the probability it chooses a recommendation. Figure 2 shows the results of an interesting simulation of our algorithm for various values of p under the partial access model.

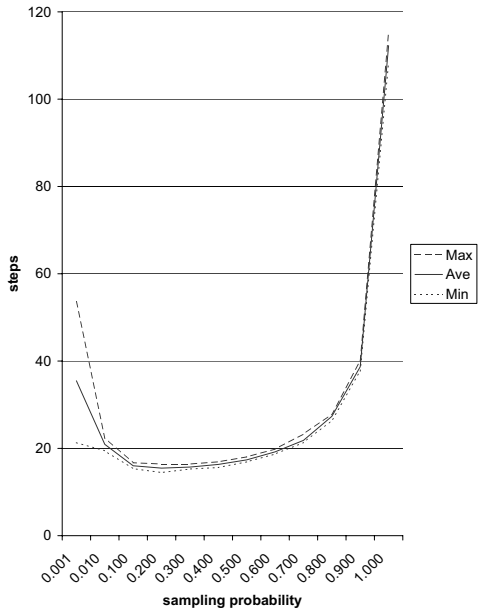

Figure 2: Simulation of the partial access algorithm

The access graph used in the simulation has 10,000 players and 10,000 objects. Each player has access to about 1,000 randomly chosen objects for about ten million edges in the graph. Roughly 5,000 honest players consider roughly 100 objects good, and the remaining 5,000 dishonest players consider the remaining 9,900 objects good.

The graph shows how the average number of steps by an honest player varies as p ranges from almost 0 to almost 1. The graph was generated as follows. Simulate a run of the partial access model on the graph. Compute the average number of steps taken by the honest players during that simulation. Do this ten times and compute this value ten times. Compute and plot the maximum, minimum, and average of these ten values. Do this once for each value of p. For small values of p the algorithm is using recommendations of other players and rarely sampling the objects themselves. For high values of p the algorithm is sampling the objects and rarely using recommendations of other players.

The graph suggests that the algorithm works fairly well for values of $p = 0.1$ through $p = 0.7$. It suggests that a little sampling is necessary, and a few recommendations can really help a lot, and the exact value of p is not very important (at least against this fairly benign adversary).

6 A repeated game scenario

We end this paper with a simple analysis of the role of trust, a first step toward our goal of developing a more general theory of trust. The main idea is that once a player is exposed as dishonest, no player will trust its recommendations in the future. We show that, in some situations, we can bound the damage done by dishonest players by exposing them. In our analysis (Lemma 6.4), we prove that the expected number of honest players that follow a recommendation made by a dishonest player is $O(\log n)$. One interpretation of this result is that "crime doesn't pay much." Consequently, if the system can ensure that the penalty for making a bogus recommendation is larger than $O(\log n)$, then no rational player would be dishonest.

We begin by defining what we call the *multi-epoch model*. In this model, computation proceeds in a sequence of synchronous *rounds*, where in each round, all players take steps concurrently. The execution is divided into *epochs* as follows. At the start of each epoch, all current objects leave the system, new objects arrive, and the players begin again the game of searching for a good object among these new objects. The start of each epoch is announced globally to all players.

We make two assumptions that have the effect of exposing dishonest players to the honest players at the end of each epoch. First, we assume that there are at most m objects in each epoch, of which exactly one is good (denote this object by i_0). Second, we assume that epochs are not too short; this allows all honest players to find the good object before the epoch ends, and thus, at the end of an epoch, the honest players know that any player recommending an object other than the good object must be dishonest.

The Skip Algorithm given in Figure 3 consists of exploration and exploitation steps, just like DynAlg. Now, however, each player keeps track of the set of potentially honest players — those players not proven to be dishonest — and exploitation steps now choose a random player from this set of potentially honest players. In addition, by fine-tuning the balance between exploration and exploitation steps, we obtain slightly better constants factors in our analysis than the perfect balance used in DynAlg would give us.

To bound the number of probes due to false recommendations, we note that in each round of Step 2 of the Skip Algorithm, the average number of honest players that follow any specific recommendation is at most α; since the number of rounds until all players find the good object is $O(\log n/\alpha)$, it follows that the total number of probes by honest players per vote is $O(\log n)$. Formally, we have the following result.

Code for an honest player:

```
0   repeat forever
1     for m/(αn) steps do                          exploration steps
1.1     Pick an object at random and probe it.
2     for (c log n)/α steps do                     exploitation steps
2.1     Pick a random player j among those
          not marked as "dishonest."
2.2     if j doesn't recommend any object, do nothing.
2.3     else let i be object recommended by j; probe i.
2.3.1     if i is bad, mark j as "dishonest."
2.3.2     else
            Recommend i;
            Mark all players recommending
              objects different than i as "dishonest;"
            Wait until next epoch is announced.
3   endrepeat
```

Figure 3: *The Skip Algorithm for the multi-epoch model. The parameter c is a constant to be determined later.*

Theorem 6.1 *If the Skip Algorithm is run for τ epochs and the length of each epoch is $\Omega\left(\frac{\log n}{\alpha} \cdot \left(\frac{m}{n} + \log n\right)\right)$ rounds, then the expected total number of probes done by all honest players is $O(\tau m) + O(K \log n)$, where K is the total number bogus recommendations throughout the execution.*

Proof Sketch: Fix an epoch. First, consider the exploration steps. Since the success probability of each exploration step by a single player (that is, the probability that the player exposes the good object i_0 in Step 1.1) is $1/m$, we have the following.

Lemma 6.2 *The expected total number of probes performed by honest players in Step 1.1 until i_0 is exposed is $\frac{m}{1-1/e}$.*

Similar considerations imply the following.

Lemma 6.3 *The probability that i_0 is not exposed after $c \log n$ iterations of the* **repeat** *loop is at most $1/n^c$.*

Now consider exploitation steps (the probes done in Step 2.3).

Lemma 6.4 *If B is the number of bogus recommendations made during the epoch, then the expected total number of probes done by honest players in Step 2.3 is at most $O(B \log n)$.*

Proof Sketch: First observe that by Lemma 6.2, the good object i_0 will be exposed after m expected exploration steps. Moreover, since the probability of

following a good recommendation in an exploitation step is proportional to the number of satisfied honest players at that step, it is not difficult to show that the once the good object has a vote, all honest players will expose it in $c \log n/\alpha$ exploitation steps for some constant $c > 0$. In each of these rounds, each honest player will probe a bad object with probability at most B/n. Summing over all honest players and all such rounds, the expected number of probes wasted by honest players on bad objects is at most $\frac{B}{n} \cdot \alpha n \cdot \frac{c \log n}{\alpha} = cB \log n$. ∎

As it is assumed that the epoch is sufficiently long to ensure that with high probability all honest players will find the good object, every dishonest player who lied in the current epoch is exposed, and will not be trusted in subsequent epochs. Hence each dishonest player can inflict wastage in at most one epoch. ∎

7 Conclusions

This paper shows that, in spite of asynchronous behavior, different interests, changes in time, and Byzantine behavior of unknown subset of peers, the honest peers miraculously succeed in collaborating, in the sense that the honest peers relatively rarely repeat mistakes of other honest peers. One interesting feature of our method is that we mostly avoid the issue of discovering who the faulty peers are. One obvious open question is how can be gained by trying to discover the faulty peers. Another open question is tightening the bounds for the partial access case.

References

[1] Y. Azar, A. Fiat, A. Karlin, F. McSherry, and J. Saia. Spectral analysis of data. In *Proc. 33rd Ann. ACM Symp. on Theory of Computing (STOC)*, pages 619–626, 2001.

[2] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[3] P. Drineas, I. Kerenidis, and P. Raghavan. Competitive recommendation systems. In *Proc. 34th Ann. ACM Symp. on Theory of Computing (STOC)*, pages 82–90, 2002.

[4] P. Feldman and S. Micali. An optimal probabilistic protocol for synchronous byzantine agreement. *SIAM J. Computing*, 26(4):873–933, 1997.

[5] D. Goldberg, D. Nichols, B. M. Oki, and D. B. Terry. Using collaborative filtering to weave an information tapestry. *Comm. of the ACM*, 35(12):61–70, 1992.

[6] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proc. 12th Int. World Wide Web Conference (WWW)*, 2003.

[7] J. Kleinberg. Finding authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, 1999.

[8] J. Kleinberg and M. Sandler. Convergent algorithms for collaborative filtering. In *Proc. 4th ACM conf. on Electronic Commerce (EC)*, pages 1–10, 2003.

[9] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Recommendation systems: A probabilistic analysis. In *Proc. IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 664–673, 1998.

[10] N. Lynch. *Distributed Algorithms*. Morgan Kaufmann, San Mateo, CA, 1995.

[11] M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 27(2):228–234, April 1980.

[12] P. Resnick, R. Zeckhauser, E. Friedman, and K. Kuwabara. Reputation sytems. *Comm. of the ACM*, 43(12):45–48, Dec. 2000.

[13] A. C. Yao. Probabilistic computations: toward a unified measure of complexity. In *Proc. 17th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 222–227, 1977.