Competitive Collaborative Learning

Baruch Awerbuch[a,1], Robert Kleinberg[b,2]

[a]*Department of Computer Science, Johns Hopkins University, 3400 N. Charles Street, Baltimore MD 21218, USA.*

[b]*Department of Computer Science, Cornell University, Ithaca, NY 14853, USA.*

Abstract

Intuitively, it is clear that trust or shared taste enables a community of users to make better decisions over time, by learning cooperatively and avoiding one another's mistakes. However, it is also clear that the presence of malicious, dishonest users in the community threatens the usefulness of such collaborative learning processes. We investigate this issue by developing algorithms for a multi-user online learning problem in which each user makes a sequence of decisions about selecting products or resources. Our model, which generalizes the adversarial multi-armed bandit problem, is characterized by two key features:

(1) The quality of the products or resources may vary over time.

(2) Some of the users in the system may be dishonest, Byzantine agents.

Decision problems with these features underlie applications such as reputation and recommendation systems in e-commerce, and resource location systems in peer-to-peer networks. Assuming the number of honest users is at least a constant fraction of the number of resources, and that the honest users can be partitioned into groups such that individuals in a group make identical assessments of resources, we present an algorithm whose expected regret per user is linear in the number of groups and only logarithmic in the number of resources. This bound compares favorably with the naïve approach in which each user ignores feedback from peers and chooses resources using a multi-armed bandit algorithm; in this case the expected regret per user would be polynomial in the number of resources.

Email addresses: `baruch@cs.jhu.edu` (Baruch Awerbuch), `rdk@cs.cornell.edu` (Robert Kleinberg).

[1] Partially supported by NSF grants CCF-0515080, ANIR-0240551, and CCR-0311795.

[2] Supported by a Fannie and John Hertz Foundation Fellowship and an NSF Mathematical Sciences Postdoctoral Fellowship. Portions of this work were completed while the author was a graduate student in the MIT Department of Mathematics and an NSF postdoctoral fellow in the UC Berkeley Computer Science Division.

1 Introduction

Only a fool learns from his own mistakes. The wise man learns from the mistakes of others.
— Otto von Bismarck

It is clear that leveraging trust or shared taste enables a community of users to be more productive, as it allows them to repeat each other's good decisions while avoiding unnecessary repetition of mistakes. Systems based on this paradigm are becoming increasingly prevalent in computer networks and the applications they support. Examples include reputation systems in e-commerce (e.g. eBay, where buyers and sellers rank each other), collaborative filtering (e.g. Amazon's recommendation system, where customers recommend books to other customers), and link analysis techniques in web search (e.g., Google's PageRank, based on combining links — i.e. recommendations — of different web sites). Not surprisingly, many algorithms and heuristics for such systems have been proposed and studied experimentally or phenomenologically [10,17,19,20,23–25]. Yet well-known algorithms (e.g. eBay's reputation system [11,15,22], the EigenTrust algorithm [16], the PageRank [10,20] and HITS [17] algorithms for web search) have thus far not been placed on a firm theoretical foundation.

Our goal in this paper is to provide a theoretical framework for understanding the capabilities and limitations of such systems in an idealized distributed environment. We propose a paradigm for addressing these issues using online learning theory, specifically a generalization of the adversarial multi-armed bandit problem [2,3]. Our approach aims to highlight the following challenges which confront the users of collaborative decision-making systems such as those cited above.

Malicious users. The above systems are vulnerable to fraudulent manipulation by dishonest ("Byzantine") participants.
Distinguishing tastes. Agents' tastes may differ, so that the advice of one honest agent may not be helpful to another.
Temporal fluctuation. The quality of resources varies of time, so past experience is not necessarily predictive of future performance.

While our learning theory paradigm is different from prior approaches, and the mathematical setting of our problem is quite distant from the applications listed above, the resulting algorithms exhibit a striking resemblance to algorithms previously proposed in the systems and information retrieval literature (e.g. EigenTrust [16]) indicating that our approach may provide a theoretical framework which sheds light on the efficacy of such algorithms in practice while suggesting potential enhancements to these algorithms.

1.1 Our approach

The problem we will consider is a generalization of the multi-armed bandit problem studied in [2,3]. In that problem there is a single learner and a set Y of m *resources*. In each of T consecutive trials, the learner chooses one of the resources while the adversary chooses a cost (taking values in $[0, 1]$) for each resource; both the learner and the adversary commit to their choices without knowledge of the other party's choice in the present trial. After the trial, the cost of the resource chosen by the learner is revealed, and that cost is charged to the learner. The goal is to choose a sequence of resources whose cost performs nearly as well, on average, as the best single resource in hindsight.

We generalize this by considering a set X of n *agents*, some of which (possibly a majority) may be dishonest. In each trial, each of the n agents chooses a resource, and the adversary chooses a cost for each resource. Each agent then learns the cost of the resource it selected, and this cost is charged to the agent. We assume that the honest agents belong to k *coalitions*, such that agents in the same coalition who choose the same resource at the same time will perceive the same expected cost. All agents may communicate with each other between trials, to exchange information (or possibly disinformation, in the case of dishonest agents) about the costs of resources they have sampled. However, agents are unaware of which coalitions exist and which ones they belong to. In particular, they are unaware of the identities of the dishonest agents.

The abstract notion of *cost* introduced in the preceding paragraph should be interpreted, in the context of most applications, to mean *disutility* rather than an actual monetary charge imposed on an agent. For example, if we are modeling the decision problem faced by consumers shopping on eBay, then the set of agents in this model corresponds to a set of consumers who are all in the market for one particular type of good. The resources correspond to sellers of this type of good. The cost of an agent (buyer) selecting a resource (seller) incorporates the price the buyer paid to the seller as well as the buyer's subjective assessment of the disutility incurred by purchasing the product from that seller, e.g. because of shipping delays or receiving the product in worse condition than advertised. However, many features of the eBay example don't precisely match the assumptions of our model — for example, the assumption that every agent participates in every trial is clearly unrealistic in the context of a large, relatively unstructured electronic marketplace — so one should interpret eBay-style reputation mechanisms as a distant application of the abstract methodology we develop here.

For an application which matches the details of our problem definition more closely, one may consider the following restaurant selection problem. A group of travelers (i.e., agents) arrive simultaneously in a city which none of them has visited before, and they spend several consecutive nights there. The city has many restaurants (i.e., resources), and the quality (i.e., negative cost) of each restaurant may vary from night to night and may depend on the subjective taste of the particular diner eating at the restaurant. Each night, each of the agents chooses to eat at one of the restaurants in the city, judges the quality of that restaurant, and posts his or her opinion (on a scale from 0 to 1) on a public bulletin board in the hotel where the travelers are staying. A subset of the agents are honest, and these agents can be partitioned into a small number of coalitions with identical tastes in food. Dishonest agents may post misleading feedback maliciously. None of the honest agents know who the other honest agents are, nor how they are partitioned into coalitions. The goal is to optimize the dining experience of the honest agents.

If an agent chooses to ignore the feedback from other agents, and simply runs the multi-armed bandit algorithm by itself, then the combined regret of all agents (the difference between the costs of their choices and those of the best choice in hindsight) grows linearly in the number of agents. On the other hand, if each honest agent knew in advance the identities of the other honest agents, as well as the partition of those agents into coalitions, then it seems reasonable that the combined regret of all agents should scale linearly in the number of *coalitions* and not the number of *agents*. Here, we show that this is possible (up to a factor of $\log(n)$) even if the agents *do not* know which other agents are honest nor how the honest agents are partitioned into coalitions. We present an algorithm whose combined regret is linear in $k \log(n)$, provided that the total number of agents and resources exceeds the number of honest agents by at most a constant factor.

Briefly, our algorithm works by having each agent select a resource in each trial using a random walk on a "reputation network" whose vertex set is the set of all agents and resources. Resources are absorbing states of this random walk, while the transition probabilities at an agent x may be interpreted as the probability that x will ask another agent x' for advice, or that x will choose a resource y without asking for advice. When an agent learns the cost of the resource chosen in a given trial, it uses this feedback to update its transition probabilities according to a multi-armed bandit algorithm. In this way, agents will tend to raise the probability of asking for advice from other agents who have given good advice in the past. In particular, though the initial transition probabilities do not reflect the partition of the honest agents into coalitions, over time the honest agents will tend to place greater weight on edges leading to other agents in the same coalition, since the advice they receive from such agents is generally better (on average) than the advice they receive from agents in other coalitions. Thus, the transition probabilities of the random walk can be interpreted as measuring the amount of trust that one agent places in another agent or resource.

An extended abstract of this work [5] appeared at COLT 2005. It is worth pointing out the two most substantial differences between that extended abstract and this paper. First, the algorithm in the earlier paper [5] depended on a complicated subroutine called the "biased bandit algorithm" (BBA), a version of the multi-armed bandit algorithm which was specifically designed for its use in the collaborative learning setting. Here, instead, we use a simpler algorithm called the "anytime bandit algorithm" (ABA), which was developed by Kleinberg in a later paper [18] that introduced multi-armed bandit algorithms which outperform the standard algorithm [2,3] when near-optimal strategies are highly prevalent among the set of all strategies. Our use of the ABA algorithm not only simplifies the description of the collaborative learning algorithm, but it also improves the bound on the expected regret per honest user: the dependence of the regret on $\log(n)$ is improved from $O(\log^4(n))$ to $O(\log(n))$. A second difference between this paper and the extended abstract [5] is that the extended abstract assumes that agents can communicate between rounds using a shared public channel. If communication instead takes place on private channels between two parties, this makes the problem more difficult because a Byzantine agent can give conflicting information to different honest agents on different private channels. In Section 5 we show how to adapt our algorithm to deal with private communication channels, under a mild synchronization assumption.

1.2 Comparison with existing work

The adversarial multi-armed bandit problem [2,3], discussed above, forms the basis for our work. We have indicated the ways in which our model generalizes the existing multi-armed bandit model to the setting of collaborative learning with dishonest users. Our work is also related to several other topics which we now discuss.

1.2.1 Collaborative filtering — spectral methods:

Our problem is similar, at least in terms of motivation, to the problem of designing collaborative filtering or recommendation systems. In such problems, one has a community of users selecting products and giving feedback on their evaluations of these products. The goal is to use this feedback to make recommendations to users, guiding them to subsequently select products which they are likely to evaluate positively. Theoretical work on collaborative filtering has mostly dealt with centralized algorithms for such problems. Typically, theoretical solutions have been considered for specific (e.g., stochastic) input models [8,12–14,21]. In such work, the goal is typically to reconstruct the full matrix of user preferences based on a small set of potentially noisy samples. This is often achieved using spectral methods. In constrast, we consider a general, i.e. adversarial, input model. Matrix reconstruction techniques do not suffice in our model. They are vulnerable to manipulation by dishonest users, as was observed in [6,7]. Dishonest users may disrupt the low rank assumption which is crucial in matrix reconstruction approaches. Alternatively, they may report phony data so as to perturb the singular vectors of the matrix, directing all the agents to a particularly bad resource.

In contrast, our algorithm makes recommendations which are provably good even in the face of arbitrary malicious attacks by dishonest users. To obtain this stronger guarantee, we must make a stronger assumption about the users: honest users are assumed to behave like automata who always follow the recommendations provided by the algorithm. (The work on collaborative filtering cited above generally assumes that users will choose whatever resources they like; the algorithm's role is limited to that of a passive observer, taking note of the ratings supplied by users and making recommendations based on this data.)

1.2.2 Collaborative filtering — random sampling methods:

The only previous collaborative filtering algorithm designed to tolerate Byzantine behavior is the "Random Advice Random Sample" algorithm in [6,7]; its average regret per user is $O(k \log n)$. The model in [6] deals with the static case, in which bad resources are consistently bad and good resources are consistently good; the only changes in the operating environment over time occur when resources arrive or depart. The algorithm in [6] uses the notion of "recommendation": once an agent finds a good resource, it sticks to it forever and recommends it to others. As time elapses, progressively more agents stick with the good advice. The bounds on regret and convergence time in [6] are analogous to ours, and are in fact superior to those in our Theorem 2.1 because the regret is independent of the number of trials. However, [6] does not handle costs which evolve dynamically as a function of time, and is limited to $\{0, 1\}$-valued rather than real-valued costs.

A related pair of papers [1,4] use random sampling methods to design algorithms for the problem of approximately reconstructing the *entire cost matrix*, again assuming costs are $\{0, 1\}$-valued and do not change over time. These algorithms have polylogarithmic convergence time provided that the set of honest agents can be partitioned into subsets, each containing a constant fraction of the agents who are weakly consistent, in the sense that their cost vectors have small Hamming distance from each other.

1.2.3 Reputation management in P2P networks:

Kamvar et al [16] proposed an algorithm, dubbed *EigenTrust*, for the problem of locating resources in peer-to-peer networks. In this problem, users of a peer-to-peer network wish to select other peers from whom to download files, with the aim of minimizing the number of downloads of inauthentic files by honest users; the problem is made difficult by the presence of malicious peers who may attempt to undermine the algorithm. Like our algorithm, EigenTrust defines reputation scores using a random walk on the set of agents, with time-varying transition probabilities which are updated according to the agents' observations. Unlike our algorithm, EigenTrust uses a different rule for updating the transition probabilities, and they demonstrate the algorithm's robustness against a limited set of malicious exploits, as opposed to the arbitrary adversarial behavior against which our algorithm is provably robust. The problem considered here is less general than the peer-to-peer resource location problem considered in [16]; for instance, we assume that in each trial, any agent may select any resource, whereas they assume that only a subset of the resources are available (namely, those peers who claim to have a copy of the requested file). Despite these differences, we believe that our work may shed light on the efficacy of EigenTrust while suggesting potential enhancements to make it more robust against Byzantine malicious users.

2 Statement of the Problem and the Results

Our algorithm operates in an environment consisting of a set X of n agents and a set Y of m resources. A subset $H \subseteq X$ is composed of *honest agents*, and the rest are dishonest. Honest agents are assumed to obey the distributed protocol to be specified, and to report their observations truthfully, while dishonest agents may behave in a Byzantine manner, disobeying the protocol or reporting fictitious observations as they wish. We will assume throughout that the number of honest agents is at least αn, where $\alpha > 0$ is a parameter which may be arbitrarily small. The agents do not initially know which ones are honest and which are dishonest, nor are they assumed to know the value of α.

In each of T consecutive rounds, a cost function $c_t : X \times Y \to [0, 1]$ is given. We think of the cost $c_t(x, y)$ as agent x's perception of the cost of resource y at time t. The costs may be set by an adaptive adversary who is allowed to choose c_t based on the agents' actions in rounds $1, \ldots, t-1$ but not on their random decisions in the present or future rounds; the adversary may also use randomization in determining c_t. Define two agents x_1, x_2 to be *consistent* if for all $y \in Y, t \in \{1, \ldots, T\}$, the costs $c_t(x_1, y), c_t(x_2, y)$ are random variables with the same expected value, conditional on the choices of the adversary and all agents in all rounds preceding t.[3] We will assume that the honest agents may be partitioned into k *coalitions*, such that two agents belonging to the same coalition are consistent; the honest agents do not initially know which coalitions the other honest agents belong to.

At the beginning of each round, each agent $x \in X$ must choose a resource $y = y_t(x) \in Y$. Any agent is allowed to choose any resource in any round. The cost of the choice is $c_t(x, y)$, and this cost (but not the cost of any other resource) is revealed to x. The agents may communicate with each other between rounds, and this communication may influence their decisions in future rounds. To simplify the exposition we will assume throughout most of the paper that all messages are exchanged using a shared public channel. In any round t all agents (including the Byzantine dishonest agents) must commit to their message on this channel before being able to read the messages posted by other agents in round t. This public-channel assumption is for expositional clarity only: in Section 5 we will indicate how to achieve the same results (with slightly worse bounds) in a message-passing model where agents may only exchange messages bilaterally on point-to-point communication links, subject to the assumption that all agents can synchronize clocks and that they have enough time to perform $\Omega(\log n)$ communication rounds in between consecutive decision rounds. (The Byzantine agents may eavesdrop on all such communications, whereas honest agents may not eavesdrop on any message if they are not the sender or receiver.) As might be expected, some subtleties arise in the message-passing model, due to ability of the Byzantine nodes to give differing advice to different parties, and to eavesdrop on others' messages.

The goal of the algorithm is to minimize the total cost incurred by honest agents. As is typical with online decision problems, we will evaluate the algorithm's performance by measuring the difference between the algorithm's expected total cost and the minimum expected cost that can be achieved by choosing a fixed resource for each agent, and instructing that agent to select this resource every time. This parameter is called *regret*[4] and will be denoted by R.

$$R = \mathbf{E}\left[\sum_{x\in H}\sum_{t=1}^{T} c_t(x, y_t(x))\right] - \min_{y:H\to Y} \mathbf{E}\left[\sum_{x\in H}\sum_{t=1}^{T} c_t(x, y(x))\right]. \tag{1}$$

The following two parameters, closely related to R, are also of interest:

[3] The randomness of the variables $c_t(x_1, y), c_t(x_2, y)$, conditional on the history preceding t, is due to the adversary's potential use of randomness in determining c_t.

[4] An even stronger notion of regret would compare the algorithm's cost with the *expectation of the minimum cost* of a fixed mapping from agents to resources, rather than using the *minimum of the expected cost* of such a mapping. In this paper we consider only the weaker notion of regret.

- The *normalized individual regret* $\hat{R} = R/\alpha nT$ is the regret per unit time of the average honest agent. For all of the algorithms we will consider, $\hat{R}$ converges to zero as $T \to \infty$.
- The *δ-convergence time* of such an algorithm, denoted by $\tau(\delta)$, is defined as the minimum value of T necessary to guarantee that $\hat{R} = O(\delta)$. Here, δ is a positive constant which may be arbitrarily close to zero.

2.1 Our results

We present a distributed algorithm, named TrustFilter, in Section 3. Let $\beta = 1 + m/n$.

Theorem 2.1. *Suppose the set of honest agents may be partitioned into k subsets $S_1, S_2, \ldots, S_k$, such that the agents in each subset are mutually consistent. Then the normalized regret $\hat{R}$ and δ-convergence time $\tau(\delta)$ of* TrustFilter *satisfy*

$$\hat{R} = O\left(\left(\frac{\beta}{\alpha}\right) k \log(n) \log(T) T^{-1/3}\right) \tag{2}$$

$$\tau(\delta) = O\left(\left(\frac{\beta}{\alpha\delta}\right)^3 k^3 \log^3(n) \log^3\left(\frac{\beta k \log n}{\alpha\delta}\right)\right). \tag{3}$$

The δ-convergence time bound follows from the regret bound. Typically we are interested in the case where α, β, δ, k are constants, hence we will summarize this result by saying that the algorithm has *polylogarithmic convergence time.*

3 The Algorithm TrustFilter

3.1 Intuition

As stated in the introduction, our algorithm is based on a Markov chain representing a random walk in a directed graph, whose vertices represent the set of resources and agents. We refer to this directed graph as the "reputation network." At each time, each agent picks an outgoing edge in the reputation network with appropriate probability, and then traverses this edge. If the edge leads to a resource, this resource is selected for sampling. Else, if the edge leads to another agent, advice is sought from that agent, who forwards the request for advice along a random edge of the reputation network (again chosen according to the current transition probabilities of the random walk) and so on until the random-walk path reaches a resource which the original agent selects for sampling. Depending on the observed cost of the sampled resource, the agent updates its transition probabilities in a manner to be specified later.

As an aid in developing intuition, consider the special case when the costs of resources are $\{0, 1\}$-valued and do not change over time. Hence, the objective of every agent is simply to find a resource with cost 0, if one exists, and then to use this resource for all subsequent trials. In this special case let us make the additional simplifying assumption that the values of α, β, δ are known to the algorithm designer. One may then use an algorithm in which the Markov chain is based on a random graph and the transition probabilities are updated according to a particularly simple rule. Specifically, at initialization time, each agent picks at random a small subset of the other agents and a small subset of the resources, takes their union, and sets equal transition probabilities on all outgoing edges leading to members of this set. (The word "small" in the foregoing sentence should be interpreted to mean, "independent of n, and depending only polynomially on $\alpha^{-1}, \beta, \delta^{-1}$.") All other outgoing edge probabilities are zero. Assume that agents adopt the following simple rule for updating their transition probabilities: if an agent chooses an outgoing edge and ends up selecting a resource with cost 0, it assigns probability 1 permanently to that resource and probability 0 to all other edges; otherwise it leaves the transition probabilities unchanged. This algorithm can be viewed as an alternative to the Random Advice Random Sample algorithm in [7], and it achieves logarithmic convergence time. The analysis of the algorithm is based on the observation that with high probability, almost all of the honest agents are contained in a giant strongly connected component of the reputation network with logarithmic diameter, and that good advice propagates through the reputation network at constant speed. Since we are interested in exploring the algorithm only as an aid in developing intuition for the more difficult case of time-varying costs, we will only sketch the proof. From the theory of random graphs, we know that when we restrict the reputation network to the honest agents, with high probability the induced subgraph has a "giant" strongly connected component S, containing all but an arbitrarily small constant fraction of the honest agents, in which every node can reach every other node using a path of length $O(\log n)$. With high probability there is an edge in the reputation network from some agent $x_0 \in S$ to a resource with cost 0 (assuming such a resource exists), and in a constant expected number of steps, x_0 will either directly sample this resource, or will stumble on another zero-cost resource by following the advice of others. Now we may argue that every other agent in S finds a resource with cost 0 in time $O(\log n)$. Indeed, if there is a path of length L, $x = x_L \to x_{L-1} \to \ldots \to x_0$, from x to x_0 in S, then we may prove by induction on L that x will find a zero-cost resource in $O(L)$ steps. Indeed, once x_{L-1} has found a zero-cost resource, only a constant number of steps are required (in expectation) before x either asks x_{L-1} for advice or follows some other advice to discover a zero-cost resource.

Our algorithm for the case of dynamic costs looks quite different from the algorithm for static costs presented in the preceding paragraph, but it is based on the same intuition: by structuring the reputation network as a random graph, good advice will propagate among honest agents at sufficient speed for the vast majority of them to identify the lowest-cost resource in logarithmic time. The main technical difference is that agents must update their transition probabilities using the multi-armed bandit algorithm, rather than shifting all of their probability mass to one outgoing edge as soon as they discover a resource with zero cost. This modification is necessary in order to deal with the fact that a resource which has zero cost at one time may not have zero cost at future times. More subtly, when agents are using the multi-armed bandit algorithm to update their transition probabilities, they must use an *anytime bandit algorithm* as defined in Section 3.2. [5] This is because the agents do not know how many other honest agents belong to their coalition, so they must consider all other vertices of the reputation network as potential neighbors to avoid being cut off from the lowest-cost resource by a ring of dishonest peers. Classical multi-armed bandit algorithms (e.g. [2,3]) will have a regret of $\Omega(\sqrt{\beta n T})$ in such a scenario, whereas we seek an algorithm whose regret depends polylogarithmically on n when the coalition size is $\Omega(\beta n)$. Accordingly, we use an anytime bandit algorithm ABA whose salient feature is that it satisfies a significantly better bound on regret when nearly-optimal strategies constitute a large fraction of all strategies. (The usage of the word *anytime* here parallels the use of the term *anytime algorithm* in the artificial intelligence literature, e.g. [9,26], to refer to optimization algorithms which generate imprecise answers quickly and proceed to construct progressively better approximate solutions over time, eventually converging to the optimal solution.)

3.2 The anytime bandit algorithm

The anytime bandit algorithm [18] is a multi-armed bandit algorithm with strategy set $\mathbb{N}$, the set of natural numbers. [6] By this, we mean that the algorithm outputs, in each time step t, a finitely-supported probability distribution π_t on $\mathbb{N}$ — based only on the inputs observed at times preceding t — and that its input at the end of step t is an ordered pair (i, c) where i is a random sample from π_t and $c \in [0, 1]$ is interpreted as the algorithm's cost for choosing i at time t. (We assume here that this cost is specified by an adaptive adversary who defines a cost function $c_t : \mathbb{N} \to [0, 1]$ in each step.) The algorithm's

[5] Anytime bandit algorithms were first defined in [18]. The algorithm is explained in Section 3.2 for the purpose of making our exposition self-contained.

[6] For the application in this paper, it would have been possible to use an anytime bandit algorithm with a finite strategy set, namely $X \cup Y$, but we will work with the infinite strategy set $\mathbb{N}$ — mapping its elements to the finite set $X \cup Y$ via a many-to-one correspondence — in order to keep our terminology consistent with [18].

objective is the same as that of the usual multi-armed bandit algorithm: to use the feedback from past trials to tune its probability distribution so that it becomes concentrated on the strategies with the lowest average cost. Since there are infinitely many strategies, it cannot accomplish this goal in a finite amount of time. Instead, the algorithm starts by trying to do nearly as well as the lowest-numbered strategies, and competes against the higher-numbered strategies only as time progresses. Thus, if the algorithm is stopped at any finite time T, one finds that the average cost of its choices in steps $1, 2, \ldots, T$ is nearly as good as the average cost of the best strategy in some initial segment $\{1, 2, \ldots, j\} \subset \mathbb{N}$, and the value of j tends to infinity with T.

Using the multi-armed bandit algorithm Exp3 of [2,3] as a subroutine, it is easy to describe the anytime algorithm ABA. For each $k \geq 0$, at time 8^k, it initializes an instance of Exp3 with strategy set $\{1, 2, \ldots, 2^k\}$. From time 8^k to $8^{k+1} - 1$, it uses this instance of Exp3 to define a probability distribution on $\mathbb{N}$ supported on the subset $\{1, 2, \ldots, 2^k\}$, and this is the distribution π_t which ABA outputs in step t. In each trial a strategy i_t is sampled from π_t, the cost $c_t(i_t)$ is revealed by the adversary, and this feedback is reported back to Exp3.

The performance of ABA is characterized by the following theorem. We refer the reader to [18] for a simple proof of the theorem, which is listed as Corollary 4.3 in that paper.

Theorem 3.1. *For every adaptive adversary, every natural number $j \in \mathbb{N}$, and every time limit $T > 0$, the regret of* ABA *satisfies the following bound, in which i_t denotes a random sample from the distribution π_t defined by* ABA*:*

$$\mathbf{E}\left[\frac{1}{T}\sum_{t=1}^{T}(c_t(i_t) - c_t(j))\right] = O\left(j \log(T) T^{-1/3}\right). \tag{4}$$

It is illuminating to compare this regret bound with the regret of the algorithm Exp3, which runs as a subroutine inside ABA. The normalized regret of Exp3 is $O\left(\sqrt{m \log m / T}\right)$ where m is the number of elements in the algorithm's strategy set. Thus, ignoring the dependence on j or m, the regret of Exp3 is $O(T^{-1/2})$ while the regret of ABA is $\widetilde{O}(T^{-1/3})$. This difference is explained by the fact that at time $T = 8^k$, the size of the strategy set used by ABA is $m = 2^k = T^{1/3}$. Hence, when evaluating the regret of the version of Exp3 running inside ABA at time T, we find that it is

$$O\left(\sqrt{\frac{m \log m}{T}}\right) = O\left(\sqrt{\frac{T^{1/3} \log\left(T^{1/3}\right)}{T}}\right) = \widetilde{O}\left(T^{-1/3}\right).$$

See Corollary 4.1 of [18] for alternate versions of the anytime bandit algorithm whose regret achieves a superior dependence on T at the cost of an inferior dependence on j. In fact, the exponent of T in the regret bound can be made arbitrarily close to $-1/2$ while retaining polynomial dependence on j, though the exponent of j tends to infinity as the exponent of T tends to $-1/2$. The version considered here, with regret depending linearly on j, is particularly convenient for our purposes because it dovetails cleanly with the random graph analysis conducted in Section 4.

For the analysis of TrustFilter in Section 3.5, it is necessary to extend the analysis of ABA to a more general feedback model which we call the "noisy feedback model." This generalization is described as follows. In each round t, instead of specifying one random cost function c_t, the adversary specifies *two* random cost functions c_t, c'_t satisfying

$$\forall i \in \mathbb{N}\ \mathbf{E}[c'_t(i) \,|\, \mathcal{F}_{<t}] = \mathbf{E}[c_t(i) \,|\, \mathcal{F}_{<t}],$$

where $\mathcal{F}_{<t}$ denotes the σ-field generated by all random variables revealed by the algorithm and adversary prior to time t. Rather than receiving $c_t(i_t)$ as feedback, the algorithm's feedback is $c'_t(i_t)$. However, the cost charged to the algorithm is still $c_t(i_t)$. The following easy proposition demonstrates that the regret of ABA is unaffected by the noisy feedback.

Proposition 3.2. *In the noisy feedback model, the regret experienced by algorithm* ABA *relative to strategy j still satisfies*

$$\mathbf{E}\left[\frac{1}{T}\sum_{t=1}^{T}(c_t(i_t) - c_t(j))\right] = O\left(j\log(T)T^{-1/3}\right).$$

Proof. Applying Theorem 3.1 to the sequence of cost functions $c'_1, c'_2, \ldots, c'_T$, gives that

$$\mathbf{E}\left[\frac{1}{T}\sum_{t=1}^{T}(c'_t(i_t) - c'_t(j))\right] = O(j\log(T)/T^{1/3}).$$

To finish proving the proposition, it suffices to prove that

$$\mathbf{E}\left[\sum_{t=1}^{T}(c'_t(j) - c_t(j))\right] = 0$$

and

$$\mathbf{E}\left[\sum_{t=1}^{T}(c'_t(i_t) - c_t(i_t))\right] = 0.$$

These follow from the fact that $\mathbf{E}(c'_t(i)) = \mathbf{E}(c_t(i))$ and $\mathbf{E}(c'_t(i_t)) = \mathbf{E}(c_t(i_t))$, both of which are consequences of the equation

$$\mathbf{E}[c_t(k) \,|\, i_t, \mathcal{F}_{<t}] = \mathbf{E}[c'_t(k) \,|\, i_t, \mathcal{F}_{<t}],$$

which holds for all $k \in \mathbb{N}$. $\square$

We conclude this section with a simple observation about ABA which will be needed later.

Lemma 3.3. *For every positive integer t, the distribution π_t which* ABA *outputs in step t satisfies $\pi_t(1) > 0$.*

Proof. The lemma follows immediately from the fact that Exp3 always assigns positive probability to each of its strategies, and the instance of Exp3 running inside ABA always has strategy set $\{1, 2, \ldots, 2^k\}$ for some $k \geq 0$. □

3.3 The algorithm TrustFilter

Here we present the algorithm TrustFilter, using ABA as a subroutine. Let y_0 be an arbitrary element of Y. When TrustFilter is initialized, each agent x initializes an instance ABA(x) of the anytime bandit algorithm, and it chooses a random function $g_x : \mathbb{N} \to X \cup Y$, such that $g_x(1) = y_0$ and the values $g_x(i)$ for $i \geq 2$ are independent uniformly-distributed elements of $X \cup Y$. We may consider the strategy set $\mathbb{N}$ of ABA(x) as comprising many "virtual copies" of each strategy $s \in X \cup Y$: the decision of ABA(x) to play strategy i corresponds, in TrustFilter, to a decision by agent x to play strategy $g_x(i)$. (The initialization step, as described above, requires the algorithm to make an infinite number of random choices at start-up time, which obviously is impossible in practice. In an actual implementation, the same result can be accomplished by lazy evaluation, i.e. the value of $g_x(i)$ is undefined until the algorithm reaches a step which requires it to examine the value of $g_x(i)$. At that moment, if $i > 1$, $g_x(i)$ is initialized to be a uniformly-distributed random sample from $X \cup Y$, drawn independently of previous random choices. If $i = 1$ then $g_x(i)$ is initialized to be y_0.)

At the beginning of each round t, each agent x queries its local bandit algorithm ABA(x) to obtain a probability distribution $\pi_t(x)$ on a finite subset of $\mathbb{N}$; this defines a probability distribution $\mu_t(x)$ on $X \cup Y$, according to the rule that

$$\mu_t(x, s) = \sum_{i \in g_x^{-1}(s)} \pi_t(x, i).$$

Here $\mu_t(x, s)$ denotes the probability assigned by $\mu_t(x)$ to any $s \in X \cup Y$, and similarly for $\pi_t(x, i)$ when $i \in \mathbb{N}$. If agent x is an honest agent, it posts this distribution $\mu_t(x)$ on the public channel. Of course, if x is a Byzantine agent, it may announce an arbitrary distribution rather than using the one which would be supplied by ABA(x). For all $x \in X$ let $\nu_t(x)$ denote the distribution which x chooses to post on the public channel. Lemma 3.3 ensures that $\nu_t(x, y_0) > 0$ for all honest agents x. Accordingly, if we define $p_t(x)$ to be equal to $\nu_t(x)$ if $\nu_t(x, y_0) > 0$ and otherwise to be a distribution which assigns probability 1 to

y_0, then we have the following properties:

(1) The distribution $p_t(x)$ is common knowledge to all agents at time t.
(2) For all $x \in X$, $p_t(x, y_0) > 0$.
(3) If x is an honest agent, then $p_t(x) = \nu_t(x) = \mu_t(x)$.

The distributions $p_t(x)$ determine a Markov chain with state space $X \cup Y$, in which the elements of Y are absorbing states and each element $x \in X$ has transition probabilities governed by the distribution $p_t(x)$. A sample path of this Markov chain, starting from a state $x \in X$, can be described as a realization of the following stochastic process: x generates a request for advice and sends it to a random neighbor sampled according to $p_t(x)$. If the random neighbor is an element $y \in Y$, the advice request stays at y; if it is an element $x' \in X$, then x' forwards the advice request to one of its neighbors sampled from the distribution $p_t(x')$, and the process continues recursively. In Section 3.4 below, we prove that with probability 1, the process is absorbed at an element $y \in Y$. Letting $q_t(x, y)$ denote the probability that the random walk starting from x is absorbed at y, we will also prove that there is an efficient algorithm to compute $q_t(x, y)$, given the transition probabilities of the Markov chain (which can easily be computed from the contents of the public channel at time t).

To select a resource $y = y_t(x) \in Y$, x samples a number $i_t \in \mathbb{N}$ from the distribution $\pi_t(x)$ and puts $s = g_x(i_t)$. It then samples a resource y randomly using the probability distribution $q_t(s)$, i.e. the distribution which assigns probability $q_t(s, y)$ to each element $y \in Y$. It learns the cost $c_t(y)$, and returns the feedback $(i_t, c_t(y))$ to $\mathsf{ABA}(x)$.

3.4 Analysis of the Markov chain

As explained above, the behavior of the algorithm $\mathsf{TrustFilter}$ at time t is governed by a Markov chain with state space $X \cup Y$, in which every state in Y is an absorbing state, and the transition probability from any state $x \in X$ to any state $s \in X \cup Y$ is equal to $p_t(x, s)$. If we define

$$p_t(y, s) = \begin{cases} 1 \text{ if } s = y \\ 0 \text{ otherwise} \end{cases}$$

for all $y \in Y$, $s \in X \cup Y$, then the transition matrix of this Markov chain is a matrix M_t whose rows and columns are indexed by the elements of $X \cup Y$, and whose entries are given by $(M_t)_{rs} = p_t(r, s)$. We say that a sequence of states $s_1, s_2, \ldots$ is absorbed at a state s if there exists some integer $r_0 > 0$ such that $s_r = s$ for all $r > r_0$.

Lemma 3.4. *If $s_1, s_2, \dots$ is a sample path of the Markov chain M_t, then with probability 1 the sample path is absorbed at an element of Y.*

Proof. If $s_{r_0} = y \in Y$ then $s_r = y$ for all $r > r_0$ because y is an absorbing state of M_t. Hence the lemma is equivalent to the assertion that the sample path contains an element of Y with probability 1. Recall that the transition probabilities p_t were defined so that $p_t(x, y_0) > 0$ for all $x \in X$. Letting $\varepsilon_t = \min_{x \in X} p_t(x, y_0)$, we find that $\Pr(\{s_1, s_2, \dots, s_r\} \subseteq X) \leq (1 - \varepsilon_t)^{r-1}$. As $\varepsilon_t > 0$, this establishes that the expected number of state transitions that take place before hitting Y is finite. By the First Borel-Cantelli Lemma, it follows that with probability 1 the sample path eventually hits Y. $\square$

Recall from Section 3.3 that for any state $s \in X \cup Y$, we define $q_t(s, y)$ to be the probability that a sample path of M_t is absorbed at y, conditional on the sample path starting at s. Lemma 3.4 ensures that for all s, $\sum_{y \in Y} q_t(s, y) = 1$. Hence $q_t(s)$ is a well-defined probability distribution on Y. The remainder of this section describes a formula for computing $q_t(s)$ algebraically.

Let Q_t denote the matrix whose rows are indexed by $X \cup Y$, whose columns are indexed by Y, and whose entries are given by the formula $(Q_t)_{sy} = q_t(s, y)$.

Lemma 3.5. $M_t Q_t = Q_t$.

Proof. Let $s_1, s_2, \dots$ denote a sample path of the Markov chain M_t. Let the notation $s_1, s_2, \dots \to y$ denote the event that $s_1, s_2, \dots$ is absorbed at y. For any $s \in X \cup Y, y \in Y$ we have

$$\begin{aligned}
(Q_t)_{sy} &= q_t(s, y) \\
&= \Pr(s_1, s_2, \dots \to y \mid s_1 = s) \\
&= \sum_{r \in X \cup Y} \Pr(s_2 = r \mid s_1 = s) \Pr(s_1, s_2, \dots \to y \mid s_1 = s, s_2 = r) \\
&= \sum_{r \in X \cup Y} p_t(s, r) q_t(r, y) \\
&= \sum_{r \in X \cup Y} (M_t)_{sr} (Q_t)_{ry} \\
&= (M_t Q_t)_{sy}
\end{aligned}$$

$\square$

We pause here to make an observation which will be useful later.

Corollary 3.6. *For any $x \in H$,*

$$\sum_{y\in Y}\sum_{i\in\mathbb{N}} \pi_t(x,i)q_t(g_x(i),y)c_t(x,y) = \sum_{y\in Y} q_t(x,y)c_t(x,y).$$

Proof. We can write the left side as

$$\sum_{y\in Y}\sum_{s\in X\cup Y}\sum_{i\in g_x^{-1}(s)} \pi_t(x,i)q_t(s,y)c_t(x,y).$$

Recalling that $\sum_{i\in g_x^{-1}(s)} \pi_t(x,i) = \mu_t(x,i)$ and that $\mu_t(x) = p_t(x)$ because $x \in H$, we see that the expression above is equal to

$$\begin{aligned}\sum_{y\in Y}\sum_{s\in X\cup Y} p_t(x,s)q_t(s,y)c_t(x,y) &= \sum_{y\in Y}\sum_{s\in X\cup Y} (M_t)_{xs}(Q_t)_{sy}c_t(x,y)\\ &= \sum_{y\in Y}(M_tQ_t)_{xy}c_t(x,y)\\ &= \sum_{y\in Y}(Q_t)_{xy}c_t(x,y)\\ &= \sum_{y\in Y} q_t(x,y)c_t(x,y),\end{aligned}$$

as claimed. $\square$

For any matrix L with rows indexed by a set I and columns indexed by a set J, if $U \subseteq I$ and $V \subseteq J$ are any subsets of the rows and columns, let us use the notation $L[U,V]$ to denote the submatrix of L consisting of all entries belonging to a row and column whose indices are in U and V, respectively. Define

$$\begin{aligned}A &= Q_t[X,Y]\\ B &= M_t[X,X]\\ C &= M_t[X,Y]\end{aligned}$$

so that we have

$$M_t = \begin{pmatrix} B & C \\ 0 & I \end{pmatrix}, \qquad Q_t = \begin{pmatrix} A \\ I \end{pmatrix}, \tag{5}$$

where I denotes the m-by-m identity matrix and 0 denotes the zero matrix. The problem of computing Q is equivalent to the problem of computing A. From (5) we have

$$M_tQ_t = \begin{pmatrix} BA + C \\ I \end{pmatrix},$$

which together with Lemma 3.5 implies

$$\begin{aligned} BA + C &= A \\ (I - B)A &= C \end{aligned}$$

The following lemma allows us to deduce that $A = (I - B)^{-1}C$, which completes our description of the formula for computing the distributions $q_t(s)$.

Lemma 3.7. *The matrix $I - B$ is invertible.*

Proof. For any $x \in X$, let B_x denote the x-th row of B, i.e. the row vector whose z-th component is $p_t(x, z)$. Observe that

$$\|B_x\|_1 = \sum_{z \in X} p_t(x, z) \leq \left(\sum_{s \in S} p_t(x, s) \right) - p_t(x, y_0) < 1.$$

For any non-zero vector v we have

$$\|Bv\|_\infty = \max_x |B_x v| \leq \max_x \|B_x\|_1 \|v\|_\infty < \|v\|_\infty,$$

hence $Bv \neq v$. This implies that $(I - B)v \neq 0$ whenever $v \neq 0$, hence $I - B$ is invertible. □

3.5 Analysis of Algorithm TrustFilter

In this section we analyze the algorithm TrustFilter, thereby proving Theorem 2.1.

Proof of Theorem 2.1. Let $\mathcal{F}_{<t}$ be the σ-field generated by the random variables

$$\{g_x(i) \,:\, x \in X, i \in \mathbb{N}\} \cup \{y_u(x) \,:\, x \in X, u < t\} \cup \{c_u(y) \,:\, y \in Y, u < t\},$$

i.e. the entire history preceding round t. In particular, $\mathcal{F}_{<1}$ denotes the σ-field generated by the random variables $\{g_x(i) \,:\, x \in X, i \in \mathbb{N}\}$. For $x \in X, s \in X \cup Y$, let

$$\tilde{c}_t(x, s) = \sum_{y \in Y} q_t(s, y) \mathbf{E}[c_t(x, y) \,|\, \mathcal{F}_{<t}].$$

From the standpoint of agent x, the bandit algorithm ABA(x) is running in the noisy feedback model with cost functions $C_t(i) = \tilde{c}_t(x, g_x(i))$ and random feedback variables $C'_t(i)$ distributed according to the cost $c_t(x, y)$ of a random resource $y \in Y$ sampled according to $q_t(g_x(i))$. For $u \in H, v \in X \cup Y$, let

$$\ell(u, v) = \min\{i \in \mathbb{N} \,:\, g_u(i) = v\}. \tag{6}$$

It follows from Proposition 3.2 that for each $u \in H$ and $v \in X \cup Y$,

$$\frac{1}{T}\mathbf{E}\left[\sum_{t=1}^{T}\sum_{i\in\mathbb{N}}\pi_t(u,i)\tilde{c}_t(u,g_i(u)) - \sum_{t=1}^{T}\tilde{c}_t(u,v) \,\middle|\, \mathcal{F}_{<1}\right]$$
$$= O\left(\ell(u,v)\log(T)\,T^{-1/3}\right). \tag{7}$$

Furthermore, applying Corollary 3.6 we have

$$\begin{aligned}\sum_{i\in\mathbb{N}}\pi_t(u,i)\tilde{c}_t(u,g_i(u)) &= \mathbf{E}\left[\sum_{i\in\mathbb{N}}\sum_{y\in Y}\pi_t(u,i)q_t(g_u(i),y)c_t(u,y) \,\middle|\, \mathcal{F}_{<t}\right]\\ &= \mathbf{E}\left[\sum_{y\in Y}q_t(u,y)c_t(u,y) \,\middle|\, \mathcal{F}_{<t}\right]\\ &= \mathbf{E}\left[\tilde{c}_t(u,u) \mid \mathcal{F}_{<t}\right].\end{aligned}$$

Hence we may rewrite (7) as

$$\frac{1}{T}\mathbf{E}\left[\sum_{t=1}^{T}(\tilde{c}_t(u,u) - \tilde{c}_t(u,v)) \,\middle|\, \mathcal{F}_{<1}\right] = O\left(\ell(u,v)\log(T)\,T^{-1/3}\right). \tag{8}$$

Recall that two agents $u, x \in H$ are called *consistent* if $\mathbf{E}[c_t(u,y) \,|\, \mathcal{F}_{<t}] = \mathbf{E}[c_t(x,y) \,|\, \mathcal{F}_{<t}]$ for all t, y. Thus, when $u, x \in H$ are consistent, $\tilde{c}(u,s) = \tilde{c}(x,s)$ for all $s \in X \cup Y$. It follows that we may rewrite equation (8) as

$$\mathbf{E}\left[\frac{1}{T}\sum_{t=1}^{T}(\tilde{c}_t(x,u) - \tilde{c}_t(x,v)) \,\middle|\, \mathcal{F}_{<1}\right] = O\left(\ell(u,v)\log(T)\,T^{-1/3}\right), \tag{9}$$

provided that x and u are consistent elements of H. Let $S \subseteq H$ be a consistent coalition containing x. For $u \in S$, let

$$\bar{c}(u) = \mathbf{E}\left(\frac{1}{T}\sum_{t=1}^{T}\tilde{c}_t(x,u) \,\middle|\, \mathcal{F}_{<1}\right).$$

Then (9) may be rewritten as

$$\bar{c}(u) - \bar{c}(v) = \ell(u,v)\cdot O\left(\log(T)\,T^{-1/3}\right). \tag{10}$$

Note that for a resource $y \in Y$,

$$\bar{c}(y) = \mathbf{E}\left(\frac{1}{T}\sum_{t=1}^{T}c_t(x,y) \,\middle|\, \mathcal{F}_{<1}\right),$$

i.e. $\bar{c}(y)$ is the conditional expectation of the average cost of y according to x, conditioned on the random functions sampled during the initialization of the algorithm. Also,

$$\bar{c}(x) = \mathbf{E}\left(\sum_{t=1}^{T} c_t(x, y_t(x)) \,\middle|\, \mathcal{F}_{<1} \right),$$

i.e. $\bar{c}(x)$ is the conditional expectation of the average cost of the resources sampled by x. Let y^* denote a resource with minimum expected cost for members of S. If x and y^* are joined by a directed path $x = u_0, u_1, u_2, \ldots, u_j = y^*$ in $S \cup Y$, we may sum up the bounds (10) over the edges of this path to obtain

$$\bar{c}(x) - \bar{c}(y^*) = \sum_{i=1}^{j} (\bar{c}(u_{i-1}) - \bar{c}(u_i)) = \sum_{i=1}^{j} \ell(u_{i-1}, u_i) \cdot O\left(\log(T)\, T^{-1/3}\right). \quad (11)$$

Letting $d(x, y^*)$ denote the length of the shortest path from x to y^* in the directed graph with vertex set $S \cup Y$ and edge lengths given by $\ell(\cdot, \cdot)$, we may rewrite (11) as

$$\bar{c}(x) - \bar{c}(y^*) = d(x, y^*) \cdot O\left(\log(T) T^{-1/3}\right). \quad (12)$$

Observe that the left side is the expected normalized regret of agent x, conditioned on $\mathcal{F}_{<1}$. The random edge lengths $\ell(u, v)$ on the $m+n$ outgoing edges of u are derived from the random function $g_u : \mathbb{N} \to X \cup Y$ using the formula (6). For graphs with random edge lengths specified according to this distribution, we analyze the expected distance between two given vertices in Section 4. Applying Proposition 4.1 from that section, we may conclude that the expectation of the right side of (12) is $O\left((\beta/\alpha(S)) \log(\alpha(S)n) \log(T)\, T^{-1/3}\right)$, where $\alpha(S)$ denotes the ratio $|S|/n$. It follows that the normalized regret and δ-convergence time for agents in the coalition S satisfy

$$\hat{R} = O\left(\left(\frac{\beta}{\alpha(S)} \right) \frac{\log(\alpha(S)n) \log(T)}{T^{1/3}} \right) \quad (13)$$

$$\tau(\delta) = \widetilde{O}\left(\left(\frac{\beta}{\alpha(S)\delta} \right)^3 \log^3(\alpha(S)n) \log^3\left(\frac{\beta \log n}{\alpha(S)\delta} \right) \right). \quad (14)$$

Note that (14) can be interpreted as saying that the large consistent coalitions learn to approximate the cost of the best resource much more rapidly than do the small coalitions, which accords with one's intuition about collaborative learning. To obtain Theorem 2.1, we must average over the k consistent coalitions $S_1, \ldots, S_k$. We may multiply the regret bound for a coalition S in (13) by the size of S, to obtain an upper bound of $O\left(\beta n \log n \log(T)\, T^{-1/3}\right)$ on the sum of the normalized regret of all users in S. Summing over k such coalitions, the cumulative normalized regret of all honest users is $O\left(k\beta n \log n \log(T)\, T^{-1/3}\right)$, so the average individual normalized regret and the convergence time satisfy:

$$\hat{R} = O\left(k \cdot \left(\frac{\beta}{\alpha}\right) \frac{\log(n)\log(T)}{T^{1/3}}\right) \tag{15}$$

$$\tau(\delta) = \tilde{O}\left(k^3 \cdot \left(\frac{\beta}{\alpha\delta}\right)^3 \log^3(n) \log^3\left(\frac{\beta k \log n}{\alpha\delta}\right)\right). \tag{16}$$

□

4 A random graph lemma

Let $G = (V, E)$ denote the directed graph with vertex set $V = X \cup Y$, in which each $x \in X$ has outgoing edges to every element of V (including a self-loop to itself), and each $y \in Y$ has no outgoing edges. Suppose $|X| = n$ and $|V| = \beta n$. Let $X_0 \subseteq X$ be any subset consisting of at least αn elements, and let y_0 be an arbitrary element of Y. For each $x \in X$, sample a random function $g_x : \mathbb{N} \to V$ by putting $g_x(1) = y_0$ and, for all $i > 1$, letting $g_x(i)$ be independent uniformly-distributed elements of V. Define an edge length $\ell(u, v)$ for each directed edge (u, v) by specifying that

$$\ell(u, v) = \min\{i \in \mathbb{N} : g_u(i) = v\}.$$

Note that $\ell(u, v)$ is well-defined with probability 1, since $\Pr(\nexists i \; g_u(i) = v) = 0$.

Proposition 4.1. *For all $x \in X_0, y \in Y$, the expected length of the shortest path from x to y in $X_0 \cup Y$ is $O((\beta/\alpha)\log(\alpha n))$.*

Proof. The fact that $g_v(1) = y_0$ for every v is a nuisance in the proof, so we begin by adopting a trick to circumvent this nuisance. For each edge $e = (u, v)$ define the *modified length* of e to be the number $\lambda(u, v) = \ell(u, v) - 1$. Let G_0 be the induced subgraph on the vertex set $X_0 \cup Y$. For $u, v \in X_0 \cup Y$ let $d(u, v)$ (resp. $\delta(u, v)$) denote the minimum total length (resp. modified length) of a path in G_0 from u to v. Note that $\ell(u, v) \leq 2\lambda(u, v)$ except when $\ell(u, v) = 1$, and the case $\ell(u, v) = 1$ occurs only when $v = y_0$, in which case v has no outgoing edges. It follows that for every path P, $\ell(P) \leq 2\lambda(P) + 1$; here $\ell(P)$ and $\lambda(P)$ denote the length and modified length of P, respectively. Hence, to prove the proposition, it suffices to put an upper bound on $\mathbf{E}(\delta(x, y))$.

For an integer $r \geq 0$, let $B_0(x, r)$ denote the set of all $v \in X_0$ such that $\delta(x, v) \leq r$. Let $b(r)$ denote the number of vertices in $B_0(x, r)$ and let

$$r^* = \min\{r : b(r) \geq \alpha n/3\}.$$

We will prove separate bounds on the expected values of r^* and $\delta(x, y) - r^*$. To bound the expected value of r^*, we make use of the following lemma.

Lemma 4.2. *For all $r \geq 0$,*

$$\mathbf{E}\left[\ln(b(r)) - \ln(b(r-1)) \mid b(r-1)\right] \geq \frac{\alpha}{6\beta}\mathbf{1}_{[b(r-1)\,\leq\,\alpha n/3]},$$

where $\mathbf{1}_{[\mathcal{E}]}$ denotes the indicator random variable of an event $\mathcal{E}$.

Proof. Let us assume that $b(r-1) \leq \alpha n/3$, as otherwise the lemma is trivial. In that case, for each $v \in B_0(x, r-1)$ let $i_v = 1+r-\delta(x,v)$, and let $w(v) = g_v(i_v)$. We have $\lambda(v, w(v)) \leq r - \delta(x,v)$, which implies

$$\delta(x, w(v)) \leq \delta(x,v) + \lambda(v, w(v)) \leq r.$$

Hence $w(v) \in B_0(x,r)$. We will prove that in expectation, at least $\frac{\alpha}{3\beta}b(r-1)$ of the vertices $w(v)$ belong to $X_0 \setminus B_0(x,r-1)$. Label the elements of $B_0(x,r-1)$ as $v_1, v_2, \ldots, v_{b(r-1)}$. For $1 \leq j \leq b(r-1)$, let

$$S_j = B_0(x, r-1) \cup \left(X_0 \cap \{w(v_1), w(v_2), \ldots, w(v_j)\}\right).$$

Note that $|S_j| \leq 2b(r-1) \leq 2\alpha n/3$ for all j, so X_0 always contains at least $\alpha n/3$ elements which are not in S_j. The following bound holds for all j:

$$\Pr(w(v_j) \in X_0 \setminus S_{j-1} \mid S_{j-1}) \geq \frac{\alpha}{3\beta}. \tag{17}$$

This is because $w(v_j)$ is uniformly distributed among the βn elements of the set $X \cup Y$, at least $\alpha n/3$ of which belong to $X_0 \setminus S_{j-1}$, and the random variables $w(v_j), S_{j-1}$ are independent. Summing (17) for $j = 1, 2, \ldots, b(r-1)$, we find that the expected number of elements of $S_{b(r-1)} \setminus B_0(x, r-1)$, conditional on the event $b_0(r-1) \leq \alpha n/3$, is at least $\frac{\alpha}{3\beta}b(r-1)$. Since $S_{b(r-1)} \subseteq B_0(x,r)$ we have $|S_{b(r-1)}| \leq b(r)$. Note also that $|S_{b(r-1)}| \leq 2b(r-1)$. Combining all these bounds, we obtain

$$\begin{aligned}\mathbf{E}\left(\min\left\{\frac{b(r)}{b(r-1)}, 2\right\} \,\middle|\, b(r-1)\right) &\geq \mathbf{E}\left(\frac{|S_{b(r-1)}|}{b(r-1)} \,\middle|\, b(r-1)\right) \\ &\geq 1 + \frac{\alpha}{3\beta}\mathbf{1}_{b(r-1)\leq\alpha n/3}.\end{aligned}$$

Let $y = \min\{\frac{b(r)}{b(r-1)}, 2\}$. Using the identity $\ln(x) \geq \frac{x-1}{x} \geq \frac{x-1}{2}$ which is valid for all $1 \leq x \leq 2$, we find that

$$
\begin{aligned}
\mathbf{E}\left(\ln\left(\frac{b(r)}{b(r-1)}\right) \,\middle|\, b(r-1)\right) &\geq \mathbf{E}\left(\min\left\{\ln\left(\frac{b(r)}{b(r-1)}\right),\, \ln(2)\right\} \,\middle|\, b(r-1)\right) \\
&= \mathbf{E}(\ln(y) \,|\, b(r-1)) \\
&\geq \mathbf{E}\left(\frac{y-1}{2} \,\middle|\, b(r-1)\right) \\
&\geq \frac{\alpha}{6\beta}\mathbf{1}_{b(r-1)\leq \alpha n/3}.
\end{aligned}
$$

□

Lemma 4.2 implies that

$$\frac{6\beta}{\alpha}\mathbf{E}(\ln(b(r)) - \ln(b(r-1))) \geq \Pr(b(r-1) \leq \alpha n/3) = \Pr(r^* \geq r). \tag{18}$$

Summing (18) for $r = 1, 2, \ldots$ we find that

$$
\begin{aligned}
\frac{6\beta}{\alpha} \lim_{r\to\infty} \mathbf{E}(\ln(b(r))) &\geq \sum_{r=1}^{\infty} \Pr(r^* \geq r) \\
\frac{6\beta}{\alpha} \ln(\alpha n) &\geq \mathbf{E}(r^*),
\end{aligned} \tag{19}
$$

where the last line was derived using the fact that $B_0(x, r) \subseteq X_0$ — and hence $b(r) \leq \alpha n$ — for all r.

Having established an upper bound on $\mathbf{E}(r^*)$, we turn now to the task of bounding $\mathbf{E}(\delta(x, y) - r^*)$. Let $B = B_0(x, r^*)$. For any $v \in B$ and any positive integer a, let $w_a(v) = g_v(1 + r^* + a - \delta(x, y))$. As before, observe that $\lambda(v, w_a(v)) \leq r^* + a - \delta(x, v)$ and consequently,

$$\delta(x, w_a(v)) \leq \delta(x, v) + \lambda(v, w_a(v)) \leq r^* + a.$$

For any positive integer d, the event $\delta(x, y) - r^* > d$ implies that $y \neq w_a(v)$ for all $v \in B$ and $a \leq d$. Conditional on the values of r^* and B, the random variables $w_a(v)$ $(1 \leq a \leq d, v \in B)$ are all mutually independent and uniformly distributed in $X \cup Y$. Consequently,

$$\Pr(\delta(x, y) - r^* > d) \leq \prod_{a=1}^{d} \prod_{v\in B} \Pr(y \neq w_a(v)) = \left(1 - \frac{1}{\beta n}\right)^{d|B|}. \tag{20}$$

Summing over $d = 0, 1, 2, \ldots$, we obtain

$$\begin{aligned}\mathbf{E}(\delta(x,y)-r^*) &\le \sum_{d=0}^{\infty}\left(\left(1-(\beta n)^{-1}\right)^{|B|}\right)^d \\ &= \frac{1}{1-(1-(\beta n)^{-1})^{|B|}}.\end{aligned}$$

Recalling that $|B| = b(r^*) \ge \alpha n/3$, we have

$$\left(1-(\beta n)^{-1}\right)^{|B|} \le e^{-|B|/(\beta n)} \le e^{-\alpha/(3\beta)}.$$

Using the inequality $e^{-x} \le 1 - \frac{x}{2}$, which is valid for $0 \le x \le 1$, we obtain

$$1-\left(1-(\beta n)^{-1}\right)^{|B|} \ge \frac{\alpha}{6\beta},$$

which implies

$$\mathbf{E}(\delta(x,y)-r^*) \le \frac{6\beta}{\alpha}. \tag{21}$$

Combining (19) with (21), and recalling that $d(x,y) \le 2\delta(x,y)+1$, we obtain

$$\mathbf{E}(d(x,y)) = O\left(\frac{\beta}{\alpha}\log(\alpha n)\right)$$

as desired. $\square$

5 Message-passing implementation of TrustFilter

In formulating the model in Section 2 we assumed the existence of a shared public channel, to which all agents could post their probability distribution in each decision epoch. It is natural to wonder whether the same outcome can be achieved in a more distributed communication model. In this section we will assume that each pair of agents can exchange messages over a private channel. We will assume that every Byzantine agent can eavesdrop on every channel (even if both parties participating in the channel are honest), i.e. it can read but not modify the messages sent on such a channel. An honest agent can only read messages on the channels in which it is a participant. We will also assume that the agents' clocks are sufficiently synchronized that they can determine a sequence of $L = \lceil \log(n) \rceil$ non-overlapping time windows in each decision epoch. More precisely, we assume that each honest agent x can define time windows $W_i(x)$ for $1 \le i \le L$, such that if $a_i(x), b_i(x)$ are the start and end of $W_i(x)$ as measured according to some (non-observable) global clock, then $\max_x b_i(x) < \min_x a_{i+1}(x)$ for $1 \le i < L$, where the maximum and minimum are taken over the set of all honest agents.

There is an obvious protocol in this message-passing model which attempts to emulate TrustFilter. Each agent initiates a request for advice at the beginning of the decision epoch, and this advice is forwarded randomly through the reputation network using the transition probabilities determined by the state of the ABA at each node, until reaching a resource, at which point a response is returned to the agent who originated the request for advice. However, the message-passing model allows the Byzantine nodes to perpetrate sophisticated exploits which are not available in the public-channel model. For example, they may forward different advice requests according to different transition probabilities, or even adjust their transition probabilities *during* the decision epoch after observing which neighbors forward advice requests to them. Because of adversarial exploits such as these, we are not able to prove that this naive emulation of TrustFilter is effective.

Instead, we analyze an algorithm based on simulating a directed acyclic graph G with levels $0, 1, \ldots, L$ and a complete directed bipartite graph joining level i to level $i-1$, for $1 \leq i \leq L$. The vertex set in level 0 is identified with Y, while the vertex set in each positively-numbered level is identified with X. In this way, each agent x is identified with L nodes $x_1, x_2, \ldots, x_L$ of G. The agent instantiates L separate instances of the ABA algorithm, one for each corresponding node of G. For each such ABA instance, the strategy set $\mathbb{N}$ is mapped to the set $\delta_{out}(x_i)$ of outgoing edges of the corresponding node of G using a random one-to-one correspondence from $\{1, 2, \ldots, d\}$ to $\delta_{out}(x_i)$ combined with an arbitrary constant function from $\{d+1, d+2, \ldots\}$ to $\delta_{out}(x_i)$. (Here, $d = n$ for nodes above level 1, and $d = m$ in level 1.) The random one-to-one correspondence is sampled independently for each of the nodes $x_1, x_2, \ldots, x_L$. Using a message-passing protocol to be described below, in each decision epoch the nodes of G will each choose a resource by emulating the algorithm TrustFilter. (This emulation, unlike the one described above, will not be susceptible to adversarial exploits because the nodes in different levels will be forced to commit to their decisions during disjoint time windows.) Note that it is not possible for every node of G to receive feedback on its decision, because each agent x corresponds to L nodes of G which may select different resources in a decision epoch, whereas x itself is allowed to select only one resource. To address this, x chooses a random level — assigning probability $1-\delta$ to level L and $\delta/(L-1)$ to all other levels — and selects the resource y chosen by the instance of x at that level. The resulting cost is supplied as feedback only to the instances of x which selected y in this decision epoch; all other instances of x in G receive no feedback.

The message-passing protocol operates as follows. As noted above, each agent x defines L consecutive non-overlapping time windows $W_i(x)$ during the decision epoch. At the start of $W_i(x)$, the node $x_i \in V(G)$ chooses an outgoing edge using its ABA instance. If $i = 1$, the other endpoint of this edge designates the resource selected by x_i. If $i > 1$, the other endpoint of this edge corresponds to an agent x', and x sends a request for advice to this agent. If x' responds before the end of time window $W_i(x)$, then x chooses the resource specified in x''s response. Otherwise x chooses an arbitrary resource. An honest agent receiving a request for advice during its time window $W_i(x)$ responds with the resource it chose during window $W_{i-1}(x)$. A Byzantine agent, of course, responds however it wants.

The analysis of this algorithm closely parallels the analysis of TrustFilter. The δ-convergence time is slower by a factor of $O(\log(n)/\delta)$, corresponding to the fact that most of the nodes of G receive feedback once in every $\delta/\log(n)$ decision epochs (on average) rather than in every decision epoch. Most of the other steps of the analysis are identical with the analysis of TrustFilter. However, there are two salient differences which illustrate the subtlety of the message-passing model. The definition of the simulated cost function $\tilde{c}_t(x,s)$ is clear-cut only if s is a resource or an honest agent. (In these two cases, $\tilde{c}_t(x,s) = \sum_{y \in Y} q_t(s,y)\mathbf{E}[c_t(x,y)\,|\,\mathcal{F}_{<t}]$ as before.) If s is a Byzantine agent and y denotes the advice which s would give to x assuming that x sends a request to s, then $\tilde{c}_t(x,s)$ must be defined as the expectation of $c_t(x,y)$ conditioned on all messages transmitted in all prior decision epochs as well as those transmitted in the current decision epoch prior to x receiving a response from s. Recall that the analysis of TrustFilter requires the fact that $\tilde{c}_t(v,w) = \tilde{c}_t(w,w)$ when v, w are consistent. Here we must prove something similar: that if v, w are honest and consistent and (v,w) in an edge of G, then the random variables $\tilde{c}_t(v,w)$ and $\tilde{c}_t(w,w)$ have the same conditional expectation. In the public-channel setting, this statement follows immediately from the definition of "consistent", whereas here there is a small subtlety. In order to prove $\mathbf{E}[\tilde{c}_t(v,w)\,|\,\mathcal{F}_{<t}] = \mathbf{E}[\tilde{c}_t(w,w)\,|\,\mathcal{F}_{<t}]$ in the private-channel setting, it is necessary that the advice given by Byzantine nodes downstream from w is independent of the event that v requests advice from w. This explains why it is necessary for G to be a directed acyclic graph, and why it is necessary for the advice to propagate "backwards" (from the set of resources back to the agents at the top level) rather than the conceptually more natural protocol in which the agents at the top level *initiate* requests for advice and these progress downwards until reaching a resource node at the bottom level of G.

6 Open Problems

In this paper we have introduced and analyzed an algorithm for a simple model of collaborative learning. A key feature of our model is the presence of a large number of dishonest agents who are assumed to behave in an arbitrary Byzantine manner. However, other aspects of our model are quite idealized, and there are some very natural extensions of the model which more closely reflect the reality of collaborative learning systems such as eBay's reputation system and peer-to-peer resource discovery systems. It would be desirable to identify algorithms for some of the following extensions.

(1) This paper was concerned with a *synchronous* decision-making problem in which each agent must choose one resource in each decision round. One may instead study the *asynchronous* case, in which only a subset of the agents act as decision-makers in each round and the rest are inactive.
(2) We assumed that any agent could choose any resource at any time. One may instead study cases in which an agent x is restricted to a choose from a subset $S(x, t) \subseteq Y$ at time t. Useful special cases include the case in which $S(x, t)$ does not depend on t and the case in which it does not depend on x. (In the latter case, it is not even clear how to formulate the proper notion of "regret.")
(3) We assumed a very strict consistency condition for two agents x_1, x_2 in the same coalition: at *every* time t, for *every* resource y, the random variables $c_t(x_1, y), c_t(x_2, y)$ should have the same expected value, conditioned on past history. Consider relaxations of this criterion, for instance:
   - $|\mathbf{E}[c_t(x_1, y) \,|\, \mathcal{F}_{<t}] - \mathbf{E}[c_t(x_2, y) \,|\, \mathcal{F}_{<t}]| < \varepsilon$.
   - $\mathbf{E}[c_t(x_1, y^*) \,|\, \mathcal{F}_{<t}] = \mathbf{E}[c_t(x_2, y^*) \,|\, \mathcal{F}_{<t}]$, where y^* is the best resource for both x_1 and x_2. No such equation is required to hold for other resources y.
   - The mixture model: for each t, the functions $f_x(y) = \mathbf{E}[c_t(x, y) \,|\, \mathcal{F}_{<t}]$ belong to a k-dimensional linear subspace of the vector of functions $Y \to \mathbb{R}$, as x ranges over X.
(4) Study more structured collaborative decision-making problems, e.g. selecting routing paths in a network, some of whose nodes are identified with the agents.

Finally, it would be desirable to discover non-trivial lower bounds for the convergence time of collaborative learning algorithms. At present the trivial lower bound of $\Omega(m/\alpha n)$ — the minimum number of rounds needed to ensure that the best resource is sampled by at least one honest agent with constant probability — is essentially the only known lower bound.

References

[1] Noga Alon, Baruch Awerbuch, Yossi Azar, and Boaz Patt-Shamir. Tell me who I am: An interactive recommendation system. In *Proceedings of the 18th Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 1–10, 2006.

[2] Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. Gambling in a rigged casino: The adversarial multi-armed bandit problem.

In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, pages 322–331. IEEE Computer Society Press, Los Alamitos, CA, 1995.

[3] Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The non-stochastic multi-armed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002.

[4] Baruch Awerbuch, Yossi Azar, Zvi Lotker, Boaz Patt-Shamir, and Mark R. Tuttle. Collaborate with strangers to find own preferences. In *Proceedings of the 17th Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 263–269, 2005.

[5] Baruch Awerbuch and Robert Kleinberg. Competitive collaborative learning. In *Proceedings of the 18th Annual Conference on Learning Theory (COLT)*, pages 233–248, 2005.

[6] Baruch Awerbuch, Boaz Patt-Shamir, David Peleg, and Mark Tuttle. Collaboration of untrusting peers with changing interests. In *Proceedings of the 5th ACM Conference on Electronic Commerce (EC)*, pages 112–119, 2004.

[7] Baruch Awerbuch, Boaz Patt-Shamir, David Peleg, and Mark Tuttle. Improved recommendation systems. In *Proceedings of the 16th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1174–1183, 2005.

[8] Yossi Azar, Amos Fiat, Anna Karlin, Frank McSherry, and Jared Saia. Spectral analysis of data. In *Proceedings of the 33rd ACM Symposium on Theory of Computing (STOC)*, pages 619–626, 2001.

[9] M. Boddy. Anytime problem solving using dynamic programming. *Proceedings of National Conference on Artificial Intelligence (AAAI)*, 1991.

[10] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. In *Proceedings of the 7th International World Wide Web Conference (WWW)*, pages 107–117, 1998.

[11] Chrysanthos Dellarocas. Analyzing the economic efficiency of ebay-like reputation reporting mechanisms. In *Proc. 3rd ACM Conference on Electronic Commerce (EC)*, pages 171–179, 2001.

[12] Petros Drineas, Iordanis Kerenidis, and Prabhakar Raghavan. Competitive recommendation systems. In *Proceedings of the 34th ACM Symposium on Theory of Computing (STOC)*, pages 82–90, 2002.

[13] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, December 1992.

[14] Thomas Hofmann and Jan Puzicha. Latent class models for collaborative filtering. In *Proceedings of the International Joint Conference in Artificial Intelligence (IJCAI)*, pages 688–693, 1999.

[15] D. Houser and J. Wooders. Reputation in auctions: Theory, and evidence from ebay. *Journal of Economics and Management Strategy*, 15(2):353–369, 2006.

[16] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The Eigentrust algorithm for reputation management in P2P networks. In *Proceedings of the 12th International World Wide Web Conference (WWW)*, pages 640–651, 2003.

[17] Jon Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, 1999.

[18] Robert Kleinberg. Anytime algorithms for multi-armed bandit problems. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 928–936, 2006.

[19] Pattie Maes, Robert H. Guttman, and Alexandros G. Moukas. Agents that buy and sell. *Communications of the ACM*, 42(3):81–91, 1999.

[20] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the Web. Technical report, Stanford Digital Library Technologies Project, 1998.

[21] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW)*, pages 175 – 186, 1994.

[22] Paul Resnick, Ko Kuwabara, Richard Zeckhauser, and Eric Friedman. Reputation systems. *Comm. ACM*, 43(12):45–48, 2000.

[23] Bin Yu and Munindar P. Singh. A social mechanism of reputation management in electronic communities. In *Cooperative Information Agents*, pages 154–165, 2000.

[24] Giorgos Zacharia, Alexandros Moukas, and Pattie Maes. Collaborative reputation mechanisms in electronic marketplaces. In *HICSS '99: Proceedings of the Thirty-second Annual Hawaii International Conference on System Sciences-Volume 8*, page 8026, Washington, DC, USA, 1999. IEEE Computer Society.

[25] Oren Zamir and Oren Etzioni. Web document clustering: A feasibility demonstration. In *Research and Development in Information Retrieval*, pages 46–54, 1998.

[26] Shlomo Zilberstein. Operational rationality through compilation of anytime algorithms, 1993.