Monitoring Algorithms for Negative Feedback Systems

Mark Sandler
Google Inc
76 9th Avenue
New York, NY,
U.S.A.
sandler@google.com

S. Muthukrishnan
Google, Inc
76 9th Avenue
New York, NY,
U.S.A.
muthu@google.com

ABSTRACT

There are many online systems where millions of users post original content such as videos, reviews of items such as products, services and businesses, etc. While there are general rules for good behavior or even formal Terms of Service, there are still users who post content that is not suitable. Increasingly, online systems rely on *other* users who view the posted content to provide feedback.

We study online systems where users report negative feedback, i.e., report abuse; these systems are quite distinct from much studied, traditional reputation systems that focus on eliciting popularity of content by various voting methods. The central problem that we study here is how to *monitor* the quality of negative feedback, that is, detect negative feedback which is incorrect, or perhaps even malicious. Systems address this problem by testing flags manually, which is an expensive operation. As a result, there is a tradeoff between the number of manual tests and the number of errors defined as the number of incorrect flags the monitoring system misses.

Our contributions are as follows:

- We initiate a systematic study of negative feedbacks systems. Our framework is general enough to be applicable for a variety of systems. In this framework, the number of errors the system admits is bounded over the worst case of *adversarial users* while *simultaneously* the system performs only small amount of manual testing for multitude of *standard users* who might still err while reporting.
- Our main contribution is a randomized monitoring algorithm that we call Adaptive Probabilistic Testing (APT), that is simple to implement and has guarantees on expected number of errors. Even for adversarial users, the total expected error is bounded by εN over N flags for a given $\varepsilon > 0$. Simultaneously, the number of tests performed by the algorithm is within a constant factor of the best possible algorithm for standard users.
- Finally, we present empirical study of our algorithm that shows its performance on both synthetic data and real data accumulated from a variety of negative feedback systems at Google. Our study indicates that the algorithm performs better than the analysis above shows.

WWW 2010, April 26–30, 2010, Raleigh, North Carolina, USA.
ACM 978-1-60558-799-8/10/04.

Categories and Subject Descriptors

H.1.2 [**Information Systems**]: Models and Principles—*User/Machine Systems*; I.2.0 [**Computing Methodologies**]: Artificial Intelligence—*General*

General Terms

Human Factors,Algorithms,Experimentation

Keywords

Negative Feedback Systems, User Reputation, Probabilistic Analysis

1. INTRODUCTION

The World Wide Web made information dissemination much easier and more efficient, from communication and commerce to content. In particular, not only did more offline content come online and access to it made easier, but generation and publication of content has become easier. Millions of users publish blogs, self-made videos, and post comments or reviews of products and businesses such as hotels, restaurants, and others. Users also post and answer questions, tag pictures and maps, form and nurture social networks, etc. This giant publication system works by implicit understanding that content should be appropriate (e.g., avoid porn), legal (e.g., no illegally copied content), believed to be correct (e.g., when tagging maps and answering questions factually), or respectful of others privacy (e.g., with blogging and social networking), etc.; explicitly, these are enforced by Terms of Service that gives the platform provider the right to remove content, cancel access, report to police, etc. However, for a variety of reasons — non-professional users, monetary incentives, lack of cost to action, etc. — such user-generated content is always suspect. The challenge is how to identify inappropriate content at the Internet scale where millions of pieces of content being generated every day. While sophisticated algorithms that explicitly identify inappropriate content are used in cases (e.g., with finding copyright violations), a common solution has been to rely on the community, i.e., *other* users, to identify and "flag" inappropriate content.

We refer to such systems where users flag or report inappropriate content as *negative feedback systems.* Exam-

ples of negative feedback systems include Amazon reviews[1], or YouTube comments[2]. There are many other examples. These systems should be distinguished from others that rely on user participation, in particular those that identify popular items based on user feedback. These use various voting schemes to tap into the *wisdom of crowds* and aggregate positive feedback, for say ranking content.[3] Such ratings systems rely on the fact that single popular item can score thousands of ratings. In contrast, negative feedback systems are expected to take action quickly, in some cases even after a *single* flag on certain content.

The community-based policing in negative feedback systems faces a different challenge. How do the systems know when a user's flag is genuine and something to be acted upon? Flags may be simply incorrect due to an error on the user's part (mistaken about an answer to a query), or maliciousness (flag and remove a good review for a competitor), or even cluelessness; in some cases, there may be revenge flags. Known systems solve this problem by manually evaluating some or all the flags. There are two underlying assumptions here. First, humans can identify a correct flag. Second, human testing of flags is more scalable than human testing of the original contents. These are reasonable assumptions in many applications. As an example, consider YouTube comments: humans can readily spot incorrect flags of spam, and the number of flags is several orders of magnitude smaller than the number of either user-generated videos or comments.[4] However, as the systems grow in size, even testing of all flags becomes prohibitively expensive and becomes prone to denial of service style attacks. The challenge is then reduced to a tradeoff: number of flags tested by humans versus the number of incorrect flags the system misses. Testing all the flags by humans will be prohibitive but detect all incorrect flags, and testing none will allow far too many incorrect flags. In this paper, we study this tradeoff.

1.1 Our Formulation

Consider a set of *items* $\mathcal{I}$, and a user u who generates a sequence of *flags* $i_1, i_2, \ldots, i_N$. These flags correspond to the items i_j that the user u deems abusive. The flag could either be *true* or *false*. A flag i_j is True means that the item i_j violates Terms of Service; a false flag indicates that the item is not abusive and user committed an error (whether honestly or maliciously) when reporting it. For each flag, the monitoring algorithm performs one of the three actions, $\mathcal{A} = \{accept, reject, test\}$, and the outcome is $\mathcal{S} = \{accepted, rejected, positive, negative\}$. The first two states correspond to the case when algorithm accepted the report and took appropriate action against the item (say, removed or demoted it) or rejected the flag (did not perform any action against content) *without* any further testing. The last two correspond to the case where the algorithm chooses to perform external, human-based testing and discovered the true state of the flag. If the algorithm chooses to test, depending on the outcome of the test the system will either accept or reject the item. We further assume that an item is not flagged multiple times; in fact, multiple flags are seldom seen in our applications and can be handled as a sequence of independent, single flags. In fact as we discuss later, very few flags per item, is a crucial difference between our system and traditional reputation systems. Formally,

- *User strategy* is a function $\mathcal{U} : \mathcal{A}^* \times \mathbf{r} \rightarrow \{\mathbf{true}, \mathbf{false}\}$ that takes as an input vector $\mathcal{A}^*$ of past actions of the monitoring algorithm, a random vector $\mathbf{r}$, and generates the next flag.
- The *monitoring algorithm* $\mathcal{R} : \mathcal{S}^* \times \mathbf{r} \rightarrow \mathcal{A}$ is a function that takes as an input the vector $\mathcal{S}^*$ of past outcomes, a random vector $\mathbf{r}$ and returns the action for the current flag.

Interestingly, we do not assume any structure between items (such as, two videos were posted by the same user) or model the correlation between items with flags (such as, reviews in poor language tend to get flags, or videos with DJ mixes get fewer flags). Our approach here is to focus on users alone, and ignore the signals from the content. This may be seen as analogous to web search where link analysis, without content analysis, gives lot of information [5, 9]. The additional motivation for us is that focusing on user analysis makes our model applicable across content types (be they video, review text or others) and general. Finally, notice that the monitoring algorithm has no access to user strategy and it only learns something about user when it decides to test the flag, and not in other actions. This is in contrast with standard learning with experts framework where algorithm learns something on each action or step.

Say user u places N flags. Given a randomized monitoring algorithm A we measure:

- $t_u^A(N)$, the expected number of tests performed by algorithm A on the sequence of N flags by user u, and
- $e_u^A(N)$, the expected number of errors — an incorrect flag that is accepted or a correct flag that is rejected — by the algorithm for user u.

1.2 Our Contributions

To the best of our knowledge this is the first effort to formalize the problem of monitoring negative feedback. More formally, we study the tradeoff between $t_u^A(N)$ and $e_u^A(N)$.

- Our main contribution is the design of a randomized algorithm called *Adaptive Probabilistic Testing* (APT) to process flags in real time as they are being submitted by users, and a detailed theoretical analysis of it. We prove that APT satisfies the following:
  - *[Adversarial Users]* $e_u^{APT}(N) \leq \varepsilon N$ in expectation against *any* user u and any fixed ε, $0 \leq \varepsilon \leq 1$. The user strategy could be adversarial and user can observe actions of the monitoring algorithms once they are performed and adjust his strategy arbitrarily.
  - *[Standard Users]* We denote by STD(p) a standard user who errs with some probability p independently on each flag. Let OPT be the optimal algorithm for monitoring such a user with

[1] See "Report This" link in product reviews. e.g., `http://www.amazon.com/Introduction-Algorithms-Third-Thomas-Cormen/dp/0262033844/`

[2] See `http://www.youtube.com/watch?v=4TpRAp0WWLs` where reported spam comments are hidden by default.

[3] For example, see `http://digg.com/about/`.

[4] Of course, since humans do not test content that are not flagged by any user, there could potentially be false negatives. This, however, is a reasonable outcome, since it indicates that the content is not seen by too many users.

$e^{\mathrm{OPT}}_{\mathrm{STD}(p)}(N) \le \varepsilon N$. We show that for APT algorithm we have:

$$t^{APT}_{\mathrm{STD}(p)}(N) \le 4\, t^{\mathrm{OPT}}_{\mathrm{STD}(p)}(N) + o(N).$$

in other words, the adaptive testing algorithm performs within constant factor of the best possible algorithm for that particular type of user.

- We present an experimental study of our algorithm with synthetic and real data collected from various Google systems. Our algorithm satisfies $e_u(N) \le \varepsilon N$ almost always (not only in expectation), and at the same time, behaves significantly better than our theoretical analysis, more like $t^{APT}_{\mathrm{STD}(p)}(N) \le t^{\mathrm{OPT}}_{\mathrm{STD}(p)}(N) + o(N)$.

Thus the framework we use for evaluating any monitoring algorithm involves two properties: (1) the number of errors should be bounded in all cases including adversarial users, and simultaneously, (2) in a system where overwhelming majority of the users are nonmalicious, the monitoring algorithm should perform almost as well as the best possible algorithm that satisfies (1).[5] It is our experience that practitioners do need *both* of the properties above. Systems need to be robust when it comes to spammers, but also graceful for majority of users who tend to be honest. Notice that it is not trivial for a monitoring algorithm to satisfy these two properties *simultaneously*. For example, a naive approach would be to test a user at the beginning to see if the user is standard, determine p and then thereafter run OPT for user STD(p). This however will not satisfy the first property because a strategy available to a user is to pretend to be STD(0) at the beginning (e.g. never lie) and then switch to STD(1). In fact, our algorithm is far more graceful: if an adversarial user becomes standard only for certain consecutive number of flags and is arbitrary elsewhere, our algorithm will automatically behave like OPT for most of that portion when user is standard. We do not formally abstract this property for the various portions, and only focus on the two properties above over the entire sequence of flags.

1.3 Related Work and Technical Overview

Our model is very general and can be applied to many real world negative feedback systems. Because of its generality however it is reminiscent of many other problems. In particular it is reminiscent of "online label prediction problems", where the task is to predict the label on each item. But typically in such problems, there is an underlying well-structured class of hypotheses and errors are measured with respect to best in this class. In contrast, users may have arbitrary strategy in our problems, and no structured class of hypotheses may fit their behavior. In multiarmed bandit [2], expert learning setting with limited feedback [6], apple-tasting [7], and other learning problems, for each online item, one is given the correct label for all experts or arms after the action is performed, but in our problem, we obtain the correct label *only* when we test an item. Hence our monitoring algorithms have to work more agnostically.

Our approach is also reminiscent of large class of "reputation problems" where a user's reputation is measured based on his agreement with other users or item quality and is used as weight to measure how much the system values user feedback. Reputation problems arise in user driven systems such as Wikipedia [1, 4] and others [3, 8]. Such reputation schemes do not apply directly to negative feedback systems, since our systems do not have opportunity to adjust a users' errors based on other users.[6]

Despite the plethora of work in related areas, and practical motivations, to the best of our knowledge there is very little work done in the area of negative feedback systems. In fact, the only work we are aware of is empirical paper by Zheleva et al [10] that considers a problem of computing trust score of e-mail users reporting spam/non-spam. Their results considers a community-based scoring and a fixed scoring function and present empirical results, however, their algorithm provides no guarantees against malicious users.

Approaching our problem from the first principles, it is immediately clear that the monitoring algorithm has to use randomization to determine what flags to test. The algorithm we design is simple and natural, keeping a testing probability p_i that is adjusted based on feedback from tests, up or down based on whether tests reveal correct or incorrect flags. The main difficulty is its analysis because of the dependence of the state of the algorithm to the entire trajectory of testing probabilities and user strategies. In particular, the analysis of expected behavior relies on careful upper bounding the hitting times of sequences of Bernoulli trials with stopping probabilities $p_{i_1} \ge p_{i_2} \ge p_{i_l} \ldots$. The stopping probability itself changes non-deterministically, in fact influenced by user strategy, over time in such a way that there is no explicit upper bound on expectation. Instead, we rely on expectations of hitting times conditioned on the fact that stopping probability stays above some fixed value, and then generalize the results.

2. ALGORITHM AND INTUITION

In this section we describe our algorithm and provide intuition why it works; we defer full analysis to Section 3.

We begin by presenting a monitoring algorithm which only performs two actions: *test* and *accept*. This algorithm has applications of its own such as when leaving abusive content after it was reported is unacceptable for legal reasons. Similarly, a monitoring algorithm which either *rejects* or *tests* has applications of its own in systems where accidentally removing content bears high cost. Later we generalize these two algorithms into an universal monitoring algorithm with at most $\varepsilon_1 N$ false positive (accepts erroneous flags) and at most $\varepsilon_2 N$ false negative errors (ignores correct flags).

The high level idea behind the algorithm is as follows. When flag i arrives, the algorithm flips a p_i-biased coin and tests the flag with probability p_i and accepts it otherwise. After each flag, the new probability p_{i+1} is computed. The crux is to determine the update rule for the consequent of p_i's. A naive approach would be to set $p_1 = \cdots = p_k = 1$ for some k, and use the fraction of incorrect flags out of k as p_j for $j > k$. However, if user changes strategy after k flags, this method will fail. A different approach is to fix some window size w and test a fraction in each window i to estimate the number n_i of incorrect flags, and use n_i/w

[5] For calibration, $t^{\mathrm{OPT}}_{\mathrm{STD}(p)}(N) = c(\varepsilon, p)N$ tests if $p > \varepsilon$ and $O(1)$ otherwise. This is far fewer than what an optimal algorithm needs for an adversarial user.

[6] If some item was demoted or deleted on request of a user, it will no longer be equally visible to other users, and thus likely to become uncorrectable.

as testing probability for subsequent window. Again, if user changes strategy between windows, this method will fail and make far too many errors. Thinking about this further, one will realize that we need a more flexible way to combine testing probability, and knowledge from each testing. Our approach is based on the following simple observation – if we test flags with probability p, then upon discovery of a false flag, the expected number of missed false flags is $\frac{1-p}{p}$, independently of user strategy. Indeed, since *each* false flag is tested with probability p, the expected stopping time on a sequence of false flags is $\frac{1-p}{p}$. Thus, intuitively if between two consecutive discoveries of false flags the testing probability was in the range $[p', p'']$ then we have both lower and upper bound on the expected number of missed flags in between as $\frac{1-p'}{p'}$ and $\frac{1-p''}{p}$. The formal proof of this statement is not obvious since the actual probability p' is changing non-deterministically and is in fact not bounded away from zero in advance. A technical achievement in our paper is to indeed develop this idea formally as in Theorem 3.1, but this intuition suffices. Using this, we can keep track of approximately how many false items we have accepted (e.g. number of false positive errors), and thus can chose new p_i in such a way so that we satisfy the constraint on the number of false positives in expectation. See Algorithm 1 for the full details.

Algorithm 1 Test-Accept Algorithm

Input: A stream of flags.
Output: For each flag we either accept or test it. If the flag is tested the algorithm learns its true state
Description:

1. Set testing probability $p_i = 1$, estimated number of *false* skipped flags at $L = 0$;
2. For each flag i,
   (a) Test flag with probability p_i and accept otherwise.
   (b) If flag is tested and the flag is false, set $L \leftarrow L + \frac{1-p_i}{p_i}$
   (c) Set the new testing probability $p_{i+1} \leftarrow \frac{1}{\varepsilon i + 1 - L}$.

Test-accept and test-reject cases are symmetric (with default action *accept* action being replaced by *reject*, and in step 2b, L increases if the flag is true). For completeness Algorithm 2, provides full details on test-reject algorithm.

Combining Two Cases.

The idea behind test-accept-reject algorithm is that we just run test-accept and test-reject algorithms in parallel, with only one of them being *active* and producing the next action. After every step, both algorithms advance one step, and we re-set active algorithm to the one which has lower probability of testing. The complete algorithm is given on Figure 3.

3. ANALYSIS

We first analyze errors by the monitoring algorithm against an adversarial user; later, we analyze the number of tests it performs against a standard user.

Algorithm 2 Test-Reject Algorithm

Input: A stream of flags.
Output: For each flag we either reject or test it. If the flag is tested the algorithm learns its true state
Description:

1. Set testing probability $p_i = 1$, estimated number of *true* skipped flags at $L = 0$;
2. For each flag i,
   (a) Test flag with probability p_i and reject otherwise.
   (b) If flag is tested and it is *true*, set $L \leftarrow L + \frac{1-p_i}{p_i}$
   (c) Set the new testing probability $p_{i+1} \leftarrow \frac{1}{\varepsilon i + 1 - L}$.

Algorithm 3 Adaptive Probabilistic Testing algorithm

Input: Stream of flags, constants ε_1, ε_2
Output: For each flag, output *test*, *accept* or *reject*
Description:

- Let $\mathcal{A}$ and $\mathcal{B}$ be the test-accept and test-reject algorithms respectively.
- For each flag, the algorithms $\mathcal{A}$ and $\mathcal{B}$ are run in parallel to produce probabilities $p_i^{\mathcal{A}}$ and $p_i^{\mathcal{B}}$.
- If $p_i^{\mathcal{A}} < p_i^{\mathcal{B}}$, set algorithm $\mathcal{A}$ as active, $\mathcal{B}$ as passive. Else, set $\mathcal{B}$ as active and $\mathcal{A}$ as passive.
- Active algorithm flips a coin performs its action and updates its state. Passive algorithm is executed to update its p_i, but the suggested action is not performed.

3.1 Adversarial Users

We begin by analyzing the test-accept algorithm. For each flag this algorithm tests each flag with certain probability and accepts it otherwise. Thus the only type of error admitted is false positives, where algorithm accepts a false flag. Intuitively, how many undetected false flags there are between two detected ones? We begin by estimating the run length until the first detected flag, if the testing probabilities is some non-increasing sequence $\{p_i\}$.

LEMMA 3.1. *Let $\{r_i\}$ be a sequence of Bernoulli trials with parameter p_i, where $\{p_i\}$ is monotonically non-increasing sequence, and p_i itself can depend on r_j, for $j < i$. Let $Q \in [0, \infty]$ be the hitting time for the sequence $\{r_0, r_1, \dots\}$. In other words random variable Q is equal to the first index i, such that $r_i = 1$. Then for any γ, we have the expectation bound:*

$$\mathrm{E}[Q | p_Q \geq \gamma] \leq (1-\gamma)/\gamma \text{ and } \mathrm{E}[Q | p_Q] \leq (1-p_Q)/p_Q \quad (1)$$

and further the realizations are concentrated around the expectation:

$$\Pr[Q > c/\gamma \,|\, p_Q \geq \gamma] \leq e^{-c} \text{ and } \Pr[Q > c/p_Q] \leq e^{-c} \quad (2)$$

Proof. Consider sequence $r_i^{(\gamma)}$, such that $r_i^{(\gamma)} = r_i$ if $p_i \geq \gamma$ and is Bernoulli trial with probability γ otherwise. And suppose $Q^{(\gamma)}$ is a hitting time for $r_i^{(\gamma)}$. Then

$$\mathrm{E}\left[Q^{(\gamma)}\right] = \mathrm{E}\left[Q^{(\gamma)} | p_Q \geq \gamma\right] \Pr[p_Q \geq \gamma] + \mathrm{E}\left[Q^{(\gamma)} | p_Q < \gamma\right] \Pr[p_Q < \gamma] \geq \mathrm{E}\left[Q^{(\gamma)} | p_Q \geq \gamma\right]$$

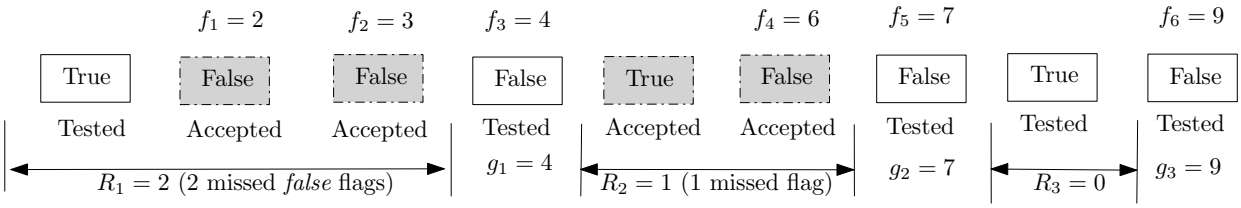


Figure 1: A sequence of flags. Gray flag indicates that the flag was not tested and its true state is unavailable to the algorithm. f_i indicates indices of all false flags (both discovered and not), g_i indicates realization of indices of discovered false flags. R_i is a realization of random variable "the number of undiscovered flags between two sequential g_i's".

where in the first transition we used the linearity of expectation and in the second we used the fact that for any fixed sequence $P = \{p_i\}$, $\mathrm{E}\left[Q^{(\gamma)}|p_Q \geq \gamma, P\right] < \mathrm{E}\left[Q^{(\gamma)}|p_Q \leq \gamma, P\right]$. On the other hand $Q^{(\gamma)}$ and Q are equal to each other if $p_Q \geq \gamma$. Thus, we have

$$\mathrm{E}\left[Q|p_Q \geq \gamma\right] = \mathrm{E}\left[Q^{(\gamma)}|p_Q \geq \gamma\right] \leq \mathrm{E}\left[Q^{(\gamma)}\right]$$

similarly for probabilities we have:

$$\begin{aligned}\Pr\left[Q \geq \frac{c}{\gamma}|p_Q \geq \gamma\right] &= \Pr\left[Q^{(\gamma)} \geq \frac{c}{\gamma}|p_Q \geq \gamma\right] \\ &\leq \Pr\left[Q^{(\gamma)} \geq \frac{c}{\gamma}\right]\end{aligned}$$

Now we just need to show $\mathrm{E}\left[Q^{(\gamma)}\right] \leq (1-\gamma)/\gamma$ and $\Pr\left[Q^{(\gamma)} \geq c\gamma\right] \leq e^{-c}$. Observe that $Q^{(\gamma)}$ can be upper bounded by geometric random variable $G \geq 0$ with parameter γ. Indeed, let us suppose g_i is 1 with probability $\min\{1, \gamma/p_i\}$ if $r_i^{(\gamma)} = 1$, and is 0 otherwise. Unconditionally each g_i is 1 with probability γ. Thus, hitting time G for $\{g_i\}$ is a geometric random variable, and by definition $G \geq Q$. Since expectation of G is $(1-\gamma)/\gamma$ we have the first part of our lemma. The second part of equation (1) follows from the definition of conditional expectation. To prove the equation (2), we note that

$$\Pr\left[G > c/\gamma\right] = (1-\gamma)^{\frac{c}{\gamma}+1} \leq e^{-c}$$

since it is exactly the probability that a sequence of Bernoulli trials with identical probability γ does not hit 1 after $\frac{c}{\gamma}$ steps. Since $Q^{(\gamma)} \geq G$ in the entire space, we have the desired bound. ■

THEOREM 3.2. *For the test-accept algorithm, the expected number of errors $e_u(N) \leq \varepsilon N$ for an adversarial user u.*

Proof. We count the expected number of undetected false positives so far after we test the ith flag. The crux is to consider the underlying sequence of false flags and corresponding testing probability, and hide all the true flags inside the probability changes p_i and apply lemma 3.1.

Suppose the false flags have occurred at positions f_1, f_2 $\ldots f_l$. We do not know what those f_i are, but our goal is to show that for any sequence the desired lower bounds holds. Denote r_i a random variable that indicates whether i-th false flag has been accepted without testing. In other words r_i is a sequence of bernoulli trials each occurring with probability $1 - p_{f_i}$.

Consider $g_0, g_1, \ldots g_{l'}$ where $g_0 = 0$, and g_i is an index of the ith *detected* false flag. In other words $\{g_i\}$ is a random subsequence of $\{f_i\}$ where algorithm *detected* false flag. Note that while f_i are unknown, g_i are the steps of the algorithm where we test and discover false flags and thus are known. Let R_i denote a random variable that is equal to the number of false flags between flags g_{i-1} and g_i. We illustrate all the notation we used with an example on Figure 3.1. It is easy to see that $R_i = \sum_{j:g_{i-1} \leq f_j < g_i} r_{f_j}$. And thus $\sum_{i=1}^{l'} R_i = \sum_{i=1}^{l} r_{f_i}$, therefore it is sufficient for us to estimate $\mathrm{E}\left[\sum_{i=1}^{l'} R_i\right]$. Note that R_i is a hitting time for the sequence of $p_{g_{i-1}}, \ldots p_{g_i}$, where the sequence if over hidden false flags p_{g_i} is not bounded a-priori. Since our algorithm does not increase testing probability if it does not detect false flags by Lemma 3.1

$$\mathrm{E}\left[R_i|p_{g_i}\right] \leq \frac{1-p_{g_i}}{p_{g_i}}.$$

Further, note that for fixed p_{g_i} the expectation is bounded independently of all earlier probabilities and therefore:

$$\begin{aligned}\sum_{i=1}^{l'} \mathrm{E}\left[R_i|p_{g_0} \ldots p_{g_i}\right] &= \sum_{i=1}^{l'} \mathrm{E}\left[\mathrm{E}\left[R_i|p_{g_i}\right]|p_{g_1}, \ldots p_{g_{i-1}}\right] \\ &\leq \sum_{i=1}^{l'} \frac{1-p_{g_i}}{p_{g_i}} \leq \varepsilon N.\end{aligned}$$

where the last transition follows from the step 2c of Algorithm 1, where we have $p_{g_i} = \frac{1}{\varepsilon(g_i-1)+1-L}$ and thus $\frac{1-p_{g_i}}{p_{g_i}} \leq \varepsilon g_i - L_{g_i}$, where $L_{g_i} = \sum_{j=1}^{i} \frac{1-p_{g_j}}{p_{g_j}}$. Hence $\sum_{i=1}^{l'} \frac{1-p_{g_i}}{p_{g_i}} \leq \varepsilon N$.

To finish the proof:

$$\sum_{i=1}^{l'} \mathrm{E}\left[R_i\right] = \mathrm{E}\left[\sum_{i=1}^{l'} \mathrm{E}\left[R_i|p_{g_1}\right]\right]$$

Expanding the right hand part, we have:

$$\mathrm{E}\left[\mathrm{E}\left[R_1|p_{g_1}\right]+\sum_{i=2}^{l'}\mathrm{E}\left[R_i|p_{g_1}\right]\right]$$
$$=\mathrm{E}\left[\mathrm{E}\left[R_1|p_{g_1}\right]+\mathrm{E}\left[R_2|p_{g_1},p_{g_2}\right]+\sum_{i=3}^{l'}\mathrm{E}\left[R_i|p_{g_1},p_{g_2}\right]\right]$$
$$=\cdots=\mathrm{E}\left[\sum_{i=1}^{l'}\mathrm{E}\left[R_i|g_1,\ldots g_i\right]\right]$$
$$\leq \mathrm{E}\left[\sum_{i}^{l'}\frac{1-p_{g_i}}{p_{g_i}}\right]\leq \varepsilon N$$

■

Similarly to above, the same results apply to the test-reject algorithm. Combining these two results together we have:

THEOREM 3.3. *For APT monitoring algorithm the expected number of false positives is at most $\varepsilon_2 N$ and the expected number of false negatives is at most $\varepsilon_2 N$.*

Proof. Indeed, lets $\mathcal{A}$ denote the set of items where *test-accept* algorithm was active, and let $\mathcal{B}$ denote the set of items where *test-reject* algorithm was active. During the $\mathcal{B}$ phase, the test-accept algorithm did not produce any mistake (since no item was accepted), thus the expected number of errors still test-accept is $\varepsilon_1 N$. More formally, we define R_i as aa hitting times, of detecting false flags, with an extra constraint that the algorithm must have been active, and the analysis carries through. ■

3.2 Standard Users

In this section, we consider standard users. Recall that for a standard user, each flag is incorrect with some unknown probability p_u. This models two dimensions about users in negative feedback systems. First, even genuine users err sometimes, but it is not correlated across items, and hence we assume it is with some fixed, independent, but unknown probability p_u. Second, some of non-malicious users might be clueless and err; in such cases, the error probability p_u is again considered independently random, not correlated with the items. We abstract these as STD(p) users. Note that p may be small as in the first case or large as in the second case. What is the minimum number of tests we need to perform to guarantee at most $\varepsilon_1 N$ of false positive? Since the user is random the only parameters we can tune are the number of tests T, the number of accepts A and the number of rejects R with the goal of minimizing T, since it does not matter which flags got tested:

$$T+A+R=N,\ Ap\leq\varepsilon_1 N,\ R(1-p)\leq\varepsilon_2 N,\ \min T$$

Thus if $p\leq\varepsilon_1$ then we can accept all the flags and not do any testing. On the other hand if $p\geq 1-\varepsilon_2$, then we can reject all flags and again not perform any testing. In the general case it can be shown that the total fraction of tested flags will be at least $1-\frac{\varepsilon_1}{p}-\frac{\varepsilon_2}{1-p}$. In the case when $\varepsilon_2=0$ we get the total fraction of flags that needs to be tested is at least $\frac{p-\varepsilon_1}{p}$ and if $\varepsilon_1=0$ it becomes $\frac{1-p-\varepsilon_2}{1-p}$.

We now analyze the behavior of our algorithm and show that for a standard user the algorithm is competitive with respect to the optimal algorithm described above. Equivalently, we prove that if $p\leq\varepsilon$ then the expected number of tests is $o(N)$ and if $p\geq\varepsilon$ then it is bounded by 4 OPT. As empirical evaluation in Section 4 shows, the analysis below is very likely to be not tight, and providing a tighter constant is an interesting open problem.

THEOREM 3.4. *For a* STD(p) *user with N flags, each false with probability p, the test-accept algorithm performs in expectation $\beta N^{\frac{p}{\varepsilon}}$ tests if $p\leq\varepsilon$, and $\gamma N+c$ tests otherwise, where $\gamma=4\frac{p-\varepsilon}{p}$ and c and β are $O(1)$. Similarly test-reject algorithm performs at most $\beta N^{\frac{1-p}{\varepsilon}}$ if $p\geq 1-\varepsilon$ and $4\frac{1-p-\varepsilon}{1-p}N+c$ otherwise.*

Proof. Suppose our target is ε fraction of errors. It is easy to see that the algorithm can be reformulated as follows. At step i test with probability $\frac{1}{1+\varepsilon i}$, and every time the item is tested, the probability of testing resets back to 1 with probability p. The question then becomes what is the expected number of tests we will perform? The full proof is given in the appendix. ■

Finally, we analyze the performance of the *APT*algorithm.

THEOREM 3.5. *Consider* STD(p) *a user u with N flags. The number of tests performed by the APT algorithm is at most $4\ OPT+2\max(\varepsilon_1,\varepsilon_2)N+o(N)$.*

Proof.

The total number of tests is the lesser of the number of tests performed by either of the *test-accept* and *test-reject* algorithms in isolation. Thus it is sufficient to only consider $\varepsilon_1\leq p\leq 1-\varepsilon_2$, otherwise, by Theorem 3.4 the expected number of tests is o(N). For the latter case we have, the expected number of tests is:

$$t^{APT}_{STD(p)}(N)\leq 4n\min(1-\frac{\varepsilon_1}{p},1-\frac{\varepsilon_2}{1-p}). \qquad (3)$$

If $p\geq 1/2$, the number of tests performed by the optimal algorithm is $t^{OPT}_{STD(p)}(N)\geq N(1-\frac{\varepsilon_1}{p}-\frac{\varepsilon_2}{1-p})\geq N(1-\frac{\varepsilon_1}{p}-2\varepsilon_2)$. Similarly, for $p\leq 1/2$ the number of tests is bounded by: $t^{OPT}_{STD(p}(N)\geq N(1-\frac{\varepsilon_1}{p}-\frac{\varepsilon_2}{1-p})\geq n(1-\frac{\varepsilon_1}{1-p}-2\varepsilon_1)$, combining these two inequalities with equation (3) we have the desired result. ■

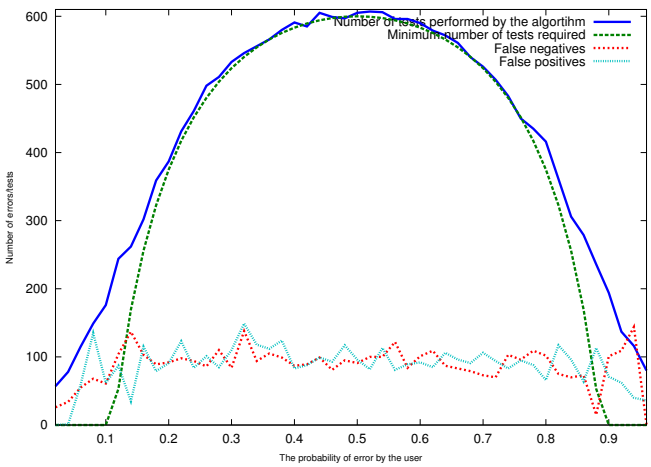
4. EXPERIMENTS

In this section we perform two kinds of experiments. First, on synthetic data, we compare our algorithm with the optimal algorithm which knows user strategy in advance. The primary goal here is to show that not only the algorithm performs only *constant* times as optimal, but to further demonstrate that the *constant* is very close to 1.

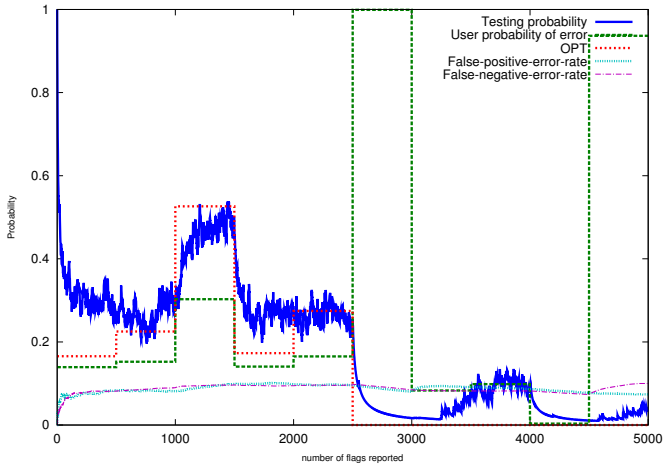
Second, we present results from running our algorithm on real data consisting of abuse reports submitted to several Google properties. An important observation here is that since our algorithms are not using the actual content in any way, the only important quantity is the number of reports per user. In all our experiments we assume that acceptable error level for either false positive or false negative type of errors is 0.1.

Synthetic data. We first demonstrate the performance (number of tests and number of errors) of the algorithm against standard users. To achieve this we plot the optimal

Figure 2: Experiments on synthetic data



(a) The number of tests and the number of errors admitted by APT algorithm when user happens to be STD(p) vs that by the optimal algorithm that knows p.



(b) Performance of the APT algorithm for STD(p) user that changes p over time.

number of tests for fixed acceptable error $\varepsilon = 0.1$ and p changing from 0.01 to 1 for STD(p) against average number of tests performed by APT. For each p we assume 1000 flags and run the experiments 30 times, to get accurate estimate of the expected number of tests performed. The results are presented in Figure 3(a). It is clear that our algorithm is in fact much closer to the optimal than the theoretical analysis above suggests, and $t^{APT}_{\mathrm{STD}(p)}(N)$ is more like $t^{\mathrm{OPT}}_{\mathrm{STD}(p)}(N) + o(N)$.

On the Figure 3(b) we consider a user who changes his error probability over time. The step-function with extreme values between 0 and 1 is user real error rate. The best optimal test rate (if the algorithm knew the underlying constant), is step function bounded by 0.5. The line, closely following the latter, shows the testing probability for the APT algorithm when STD(p) user keeps changing p. It is clear that APT algorithm automatically adjusts its testing rate nicely to be nearly close to the best testing for STD(p) for whatever p the user uses for a period of time.

Experiments with real data. In this section we present experimental results that we perform using real data, which contains a subset of abuse reports accumulated by various Google services over the period of time of about 2 years. The dataset contained roughly about 650 randomly selected anonymized users who submitted at least 50 reports to the system (some submitted considerably more). Their total contribution was about 230,000 flags. The algorithm computed testing probability independently for every user, and thus our guarantees apply for every user independently. Our goal was to measure to average testing rate as a function of the total number of flags, since it translates to immediate reduction of amount of manual labor required. On Figure 3(c) we plot the number of total flags arrived into system (blue curve), vs the total number of tested flags (green curve). The bottom three curves show the actual fraction of admitted errors vs the acceptable error levels.

To illustrate the actual fraction of tested flags we refer to Figure 3(d). As one can see the testing ratio in general hovers around 0.35, which means that only roughly 1 in every 3 user flags needs to get tested, and the remainder can be acted on automatically.

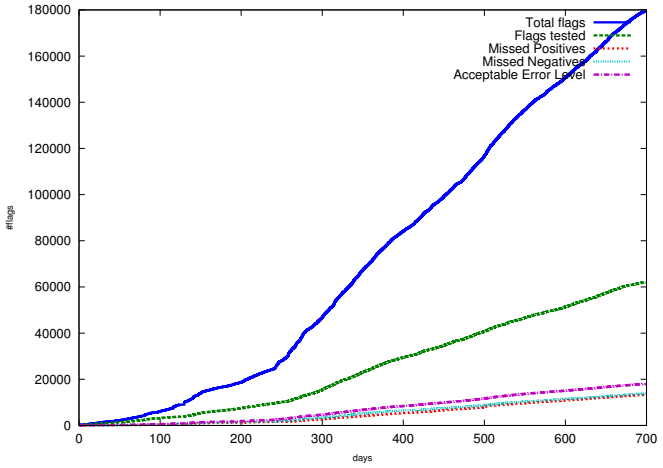
5. FUTURE WORK AND CONCLUSIONS

We described a simple model for monitoring negative feedback systems, and presented the APT algorithm with expected number of errors $\leq \varepsilon N$ for even adversarial users; for a standard user STD(p), the expected number of tests is close to that of the optimal algorithm for STD(p). Practitioners look for such algorithms that are resistant to adversary users, while still being gracefully efficient to standard users. We have found these algorithms useful for some of the Google systems.

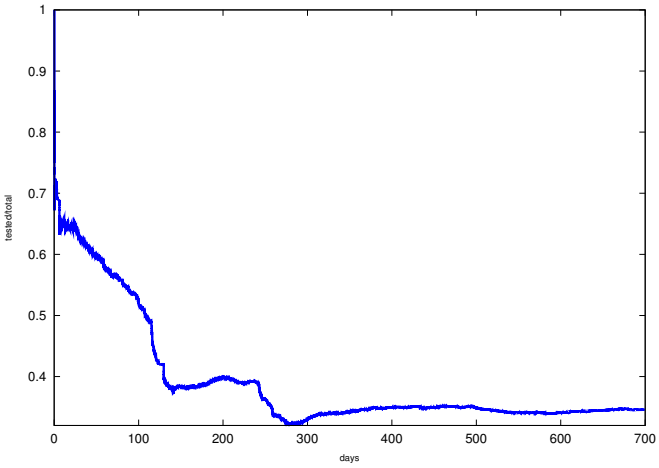
From a technical point of view, the immediate open question is if our analysis of APT algorithm can be improved since our experiments indicate $t^{APT}_{\mathrm{STD}(p)}(N)$ behaves more like $t^{\mathrm{OPT}}_{\mathrm{STD}(p)}(N) + o(N)$. Further, could we extend the expected case analysis in this paper to high concentration bounds? We are able to analyze the APT algorithm and show that the $e^{APT}_u(N)$ is within twice the expectation with overwhelming probability for any user, under certain mild conditions. This result appears in Appendix B. We need to use martingale inequalities to prove these bounds, but they require bounded difference, whereas hitting times of our algorithm are not bounded. We overcome this problem by suitably modifying Azuma's inequality. This analysis may be of independent interest. Similar high concentration bounds for $t^{APT}_{\mathrm{STD}(p)}(N)$ would be of interest.

From a conceptual point of view, negative feedback systems are abundant on the Internet, and we need more research on useful monitoring algorithms. For example, an extended model is to

consider items having attributes and typical user error as some function of those attributes that we need to learn. Similarly considering standard (e.g. non malicious) user whose behavior evolves over time is also important. A potential first step in this direction is to compare the performance of a monitoring algorithm with the optimal (but not necessarily spam resistant) algorithm that is required to have



(c) The number of flags that the system tested vs. the total number of flags vs. the number of errors



(d) The ratio between number of tested flags and the total number of flags as a function of time

Figure 3: Performance of the algorithm on flags received from real users.

bounded number of error on every *prefix* sequence of user flags for users that change their failure probability gracefully over time. Another potentially interesting direction which has practical justification is to allow monitoring algorithms to take retroactive actions such as deciding to test an item which was accepted or rejected earlier. Finally, a rich direction is to not consider each user individually, but group them according to their past history of reports. This allows a reduction in the amount of testing for users who provide only a few flags since such users can contribute a significant fraction of flags in many real-world systems. Such a grouping can be dynamic and depend on users' strategies as well as other properties, and guarantees will be relative to the group. We hope our work here spurs principled research on monitoring algorithms for negative feedback systems that are much needed.

5.1 Acknowledgements

Authors would like to thank Nir Ailon, Raoul-Sam Daruwala, Yishay Mansour and Ruoming Pang and anonymous reviewers for interesting discussion and good feedback on the paper.

6. REFERENCES

[1] B. Adler and L. de Alfaro. A content-driven reputation system for the Wikipedia. In *Proc. of the 16th Intl. World Wide Web Conf.(WWW 2007)*.

[2] D. Berry and B. Fristedt. *Bandit Problems*. Chapman and Hall, 1985.

[3] R. Bhattacharjee and A. Goel. Avoiding ballot stuffing in eBay-like reputation systems. In *Proceedings of the 2005 ACM SIGCOMM workshop on Economics of peer-to-peer systems*, pages 133–137. ACM New York, NY, USA, 2005.

[4] R. Bhattacharjee and A. Goel. Algorithms and incentives for robust ranking. In *SODA*, pages 425–433. Society for Industrial and Applied Mathematics Philadelphia, PA, USA, 2007.

[5] 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.

[6] N. Cesa-Bianchi and G. Lugosi. *Prediction, learning, and games*. Cambridge University Press, 2006.

[7] D. Helmbold, N. Littlestone, and P. Long. Apple tasting. *Information and Computation*, 161(2):85–139, 2000.

[8] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 640–651, New York, NY, USA, 2003. ACM.

[9] J. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[10] E. Zheleva, A. Kolcz, and L. Getoor. Trusting spam reporters: A reporter-based reputation system for email filtering. *ACM Trans. Inf. Syst.*, 27(1):1–27, 2008.

APPENDIX

A. PROOF OF 3.4

Proof of Theorem 3.4. We reformulate the algorithm equivalently: at step i test with probability $\frac{1}{1+\varepsilon i}$, and every time we reset the probability p. The question then becomes what is the expected number of tests we will perform? Let X_i be the random variable that is 1 if there were no reset until step i, and ith flag was tested (whether or not the probability was reset on flag i.) Further let P_i be the probability that there was no reset until step i (whether or not reset happened on step i). Obviously $P_i = \prod_{j=1}^{i-1}(1 - \frac{p}{1+\varepsilon j})$ and $\mathrm{E}[X_i] = \frac{P_i}{1+\varepsilon i}$. The number of tests that happened before and including the first reset is the random variable: $\mathrm{E}\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} \frac{P_i}{1+\varepsilon i}$. Let F_n denotes the random variable indicating how many tests were performed during the first n steps. Let $F_{n,i}$ denote random variables indicating how many tests were performed by the

algorithm after i-th step respectively, provided that the reset occurred at position i, and is 0 if the reset hasn't occurred. Also let G_n denote the number of tests performed by the algorithm before and including the first reset. Note that the algorithm has no memory and thus $F_{n,i} = P_i \frac{p}{1+\varepsilon i} F_{n-i}$. On the other hand since each test causes a reset with probability p the expected number of tests before the first reset can be expressed as

$$\begin{aligned}\mathrm{E}[G_n] &= \sum_{i=1}^{n} \frac{P_i}{1+\varepsilon i} = \frac{1}{p}(1-P_{n+1}) \\ &\le \min(\frac{1}{p}, \frac{\ln(1+\varepsilon n)}{\varepsilon}) \le \frac{\ln(1+\varepsilon n)}{\varepsilon}.\end{aligned}$$

Where in the first upper bound, we used

$$\sum i = 1^n \frac{1}{1+\varepsilon i} \le \int_1^n \frac{1}{\varepsilon x + 1} dx \le \frac{\ln(1+\varepsilon n)}{\varepsilon}.$$

Thus:

$$\mathrm{E}[F_n] = \mathrm{E}\left[\sum_{i=1}^{n} F_{n,i} + G_n\right] \quad (4)$$

$$= \sum_{i=1}^{n} \frac{P_i}{1+\varepsilon i} p \mathrm{E}[F_{n-i}] + \frac{\ln(1+\varepsilon n)}{\varepsilon} \quad (5)$$

Now we estimate $\sum_{i=1}^{n} \frac{P_i}{1+\varepsilon i}[1 + p\mathrm{E}[F_{n-i}]]$. First of all we have:

$$\ln[P_i] = \ln(1 - \frac{p}{1+\varepsilon}) + \sum_{j=2}^{i-1} \ln(1 - \frac{p}{1+\varepsilon j})$$

After some algebra, where we approximate the sum and integrating, we have

$$P_i \ge \frac{(1-p+\varepsilon)^{\frac{p}{\varepsilon}+1}}{(1+\varepsilon)(1-p+(i-1)\varepsilon)^{\frac{p}{\varepsilon}}}$$

Case $p \le \varepsilon$**.** We need to show that $\mathrm{E}[F_n] \le \beta n^\alpha$ where $\alpha = \frac{p}{\varepsilon}$ and $\beta = 2\ln(1+\varepsilon n)\frac{1+\varepsilon}{\varepsilon}$. The proof is by induction. The base is obvious, to prove the induction hypothesis we have:

$$\begin{aligned}\mathrm{E}[F_n] &\le \frac{\ln(1+\varepsilon n)}{\varepsilon} + \sum_{i=1}^{n} \frac{P_i \times p \times \mathrm{E}[F_{n-i}]}{1+\varepsilon i} \\ &\le \frac{\ln(1+\varepsilon n)}{\varepsilon} + (1-P_{n+1})F_{n-1} \\ &\le \frac{\ln(1+\varepsilon n)}{\varepsilon} + \beta n^\alpha - \frac{\beta(1-p+\varepsilon)^{\alpha+1} n^\alpha}{(1+\varepsilon)(1-p+\varepsilon n)^\alpha} \\ &\le \frac{\ln(1+\varepsilon n)}{\varepsilon} + \beta n^\alpha - \beta \frac{n^\alpha}{(1+\varepsilon)(1+\varepsilon^\alpha n^\alpha)} \\ &\le \frac{\ln(1+\varepsilon n)}{\varepsilon} + \beta n^\alpha - \frac{\beta}{2(1+\varepsilon)} \le \beta n^\alpha\end{aligned}$$

where first we used $\sum_{i=1}^{n-1} \frac{pP_i}{1+\varepsilon i} = 1 - P_n$, then that $(1-p+\varepsilon) \ge 1$ and $(1-p+\varepsilon n)^\alpha \le (1+\varepsilon n)^\alpha \le 1 + (\varepsilon n)^\alpha$ $\alpha < 1$, and finally, $\frac{n^\alpha}{1+\varepsilon^\alpha n} \ge 1/2$.

Case $p \ge \varepsilon$**.** We have to prove $\mathrm{E}[F_n] \le \gamma n + c$. If $p \ge 4/3\varepsilon$ the result is obvious since $4\frac{p-\varepsilon}{p} \ge 1$. We consider $p \le 4/3\varepsilon$ only. After some algebra, we get the following bound:

$$\frac{P_i}{1+\varepsilon i} \ge \frac{(1-p+\varepsilon)^{\frac{p}{\varepsilon}+1}}{(1+\varepsilon)(1-p+i\varepsilon)^{\frac{p}{\varepsilon}+1}}$$

Substituting that into (4) we have:

$$\begin{aligned}\mathrm{E}[F_n] &\le \frac{1}{p} + \sum_{i=1}^{n-1} \frac{pP_i\gamma(n+c-i)}{1+\varepsilon i} \\ &= \frac{1}{p} + (1-P_n)(\gamma n + c) - \sum_{i=1}^{n-1} \frac{pP_i\gamma i}{1+\varepsilon i} \\ &\le \gamma n + c + \frac{1}{p} - P_n c - \gamma p \sum_{i=1}^{n-1} \frac{i(1-p+\varepsilon)^{\alpha+1}}{(1+\varepsilon)(1-p+\varepsilon i)^{\alpha+1}}\end{aligned}$$

Thus we need to prove:

$$\frac{1}{p} - P_n c\gamma - \gamma p \sum_{i=1}^{n-1} \frac{i(1-p+\varepsilon)^{\alpha+1}}{(1+\varepsilon)(1-p+\varepsilon i)^{\alpha+1}} \le 0 \quad (6)$$

Let $q = 1-p$:

$$\begin{aligned}&\sum_{i=1}^{n-1} \frac{i}{(q+\varepsilon i)^{\alpha+1}} \ge \frac{1}{(q+\varepsilon)^{\alpha+1}} + \int_{x=2}^{n} \frac{x}{(q+\varepsilon x)^{\alpha+1}} = \\ &= \frac{1}{(q+\varepsilon)^{\alpha+1}} + \frac{-(q+\alpha\varepsilon x)}{(-1+\alpha)\alpha\varepsilon^2(q+\varepsilon x)^\alpha}\Big|_{x=2}^{n} \\ &= \frac{1}{(q+\varepsilon)^{\alpha+1}} + \frac{-(q+px)}{(-\varepsilon+p)p(q+\varepsilon x)^\alpha}\Big|_{x=2}^{n} \\ &= \frac{1}{(q+\varepsilon)^{\alpha+1}} + \frac{1+p}{(p-\varepsilon)p(q+2\varepsilon)^\alpha} - \frac{q+pn}{(p-\varepsilon)p(q+\varepsilon n)^\alpha}\end{aligned}$$

substituting we have:

$$\begin{aligned}&\frac{p}{1+\varepsilon} \sum_{i=1}^{n} \frac{i(q+\varepsilon)^{\alpha+1}}{(q+\varepsilon i)^{\alpha+1}} \ge \\ &\ge \frac{p(q+\varepsilon)^{\alpha+1}}{1+\varepsilon}\Big(\frac{1}{(q+\varepsilon)^{\alpha+1}} + \frac{1+p}{(p-\varepsilon)p(q+2\varepsilon)^\alpha} - \\ &- \frac{q+pn}{(p-\varepsilon)p(q+\varepsilon n)^\alpha}\Big) \\ &\ge \frac{p}{1+\varepsilon} + \frac{(1+p)(q+\varepsilon)^{\alpha+1}}{(1+\varepsilon)(p-\varepsilon)(q+2\varepsilon)^\alpha} - \frac{(q+pN)(q+\varepsilon)^{\alpha+1}}{(1+\varepsilon)(p-\varepsilon)(q+n\varepsilon)^\alpha}\end{aligned}$$

Note that the last term asymptotically behaves as $O(n^{1-\alpha}) = o(1)$, and on the other hand we have $P_n \ge \frac{(q+\varepsilon)^{\alpha+1}}{(1+\varepsilon)(q+(n-1)\varepsilon)^\alpha}$, thus by adjusting c independently of n we can guarantee:

$$cP_n - \frac{(q+pN)(q+\varepsilon)^{\alpha+1}}{(1+\varepsilon)(p-\varepsilon)(q+N\varepsilon)^\alpha} \ge 1$$

thus we have:

$$\begin{aligned}&cP_n + \frac{p}{1+\varepsilon} \sum_{i=1}^{n} \frac{i(q+\varepsilon)^{\alpha+1}}{(q+\varepsilon i)^{\alpha+1}} \ge \\ &\ge \frac{p}{1+\varepsilon} + \frac{(1+p)(q+\varepsilon)^{\alpha+1}}{(1+\varepsilon)(p-\varepsilon)(q+2\varepsilon)^\alpha} - 1 \\ &\ge \frac{p}{1+\varepsilon} - 1 + \frac{(1+p)(q+\varepsilon)}{(1+\varepsilon)(p-\varepsilon)}(1 - \frac{p}{1-p+2\varepsilon}) \\ &\ge \frac{(p-1-\varepsilon)(p-\varepsilon) + (1-p^2)(1-p+\varepsilon)}{(1+\varepsilon)(p-\varepsilon)} \\ &\ge \frac{(q+\varepsilon)^2(q-p^2+\varepsilon)}{(1+\varepsilon)(p-\varepsilon)} \ge \frac{1}{4(p-\varepsilon)}\end{aligned}$$

Using this bound and substituting $\gamma = \frac{4(p-\varepsilon)}{p}$ we immediately have (6) satisfied, as needed. ■

B. CONCENTRATION BOUNDS ON NUMBER OF ERRORS

In this section we show that under certain conditions the number of missed false flags is also tightly concentrated around its expectation. In particular we introduce an extra constraint to algorithm 1, that requires that each $p_i \geq \sqrt{\frac{1}{i}}$. Curbing at this probability adds $O(\sqrt{N})$ extra tests, the adversarial case is not affected, and the standard user expectation only adds $o(N)$, thus the analysis carries through. The proof is based on application of Azuma inequality to the sequence of R_i with a twist that since R_i are unbounded we need to bound each R_i with high probability.

LEMMA B.1. *Suppose $X_1 \dots X_n$ be a submartingale such that $\Pr[X_{i+1} - X_i \geq c_i] \leq \frac{\delta}{n}$, then*

$$\Pr[X_n \geq \lambda] \leq \exp[\frac{\lambda}{2\sum c_i^2}] + \delta$$

Proof. In order to prove the inequality we introduce bounded random variables that are almost always equal to X_i, and then use Azuma inequality combined with union bound. Let $Y_i = X_i - X_{i-1}$, and let $Y_i' = \min(Y_i, c_i)$, and $X_i' = \sum_{i=1}^n Y_i'$, then using Azuma inequality we have $\Pr[X_m' \geq \lambda] \leq \exp[-\frac{\lambda}{2\sum c_i^2}]$, Consider the underlying probability space and let $\Omega(.)$ denotes the subspace where condition (.) is satisfied.

We have $\Omega(X_n' \neq X_n) \subseteq \Omega(Y \geq c_i, \text{ for some } i)$, thus $\Pr[X_n' \neq X_n] \leq \frac{n\delta}{n} = \delta$. On the other hand we have $\Omega(X_n \geq \gamma) \subseteq \Omega(X_n' \neq X_n) \cup (\Omega(X_n' \geq \gamma) \cap \Omega(X_n' = X_n))$. Thus:

$$\begin{aligned}\Pr[X_n \geq \lambda] &\leq \Pr\left[X_n' \neq X_n\right] + \Pr\left[X_n' \neq X_n\right] \\ &\leq \delta + \exp[-\frac{\lambda}{2\sum_{i=1}^n c_i^2}]\end{aligned}$$

as needed. ■

THEOREM B.2 (CONCENTRATION). *If all $p_i \geq \sqrt{\frac{1}{i}}$ we have*

$$\Pr[(e(N) - \varepsilon N) \geq \varepsilon N] \leq 2\exp[-\varepsilon^{2/3}N^{1/6} + \log N]$$

Note that the constraint on p_i is on the algorithm execution (e.g. we control all p_i), and not on the stopping times outcomes, thus we are not introducing any hidden conditioning and do not change distributions. *Proof.* The idea of the proof is to apply Azuma inequality and use lemma B.1 to get the desired bound.

Fix $\delta = \exp[-\varepsilon^{2/3}N^{1/6} + \log N]$ and denote $c = \log N/\delta$, and consider $T_i = \frac{1-p_{g_i}}{p_{g_i}}$. By using lemma 3.1 we have $\mathrm{E}[R_i - T_i | T_i, R_0 \dots R_{i-1}] \leq 0$ for any T_i, and thus $\mathrm{E}[R_i - T_i | R_0 \dots R_{i-1}] < 0$ therefore $\sum_{i=0}^i (R_i - T_i)$ is a submartingale. On the other hand, we have:

$$\Pr\left[R_i - T_i \geq c\sqrt{N}\right] \leq \Pr\left[R_i - T_i \geq \frac{c}{p_{g_i}}\right] \leq \exp[-c] \leq \frac{\delta}{N}.$$

Where we have used that $g_i \leq N$, constraint on $p_{g_i} \geq \frac{i}{\sqrt{g_i}}$, and the lemma 3.1 to bound $\Pr\left[R_i - T_i \geq \frac{c}{p_{g_i}}\right]$.

Thus we can use lemma B.1 with bound $c_i \leq \log(N/\delta)\sqrt{N}$. For any l' and $\lambda = \varepsilon N$ we have:

$$\begin{aligned}\Pr\left[\sum_i^{l'} R_i - T_i \geq \varepsilon N\right] &\leq \delta + \exp[-\frac{\varepsilon^2 N^2}{2l'\sqrt{N}\log^2(N/\delta)}] \\ &\leq \exp[-\frac{\varepsilon^2\sqrt{N}}{(\log N - \log\delta)^2}] + \delta\end{aligned}$$

Now we substitute for $\log\delta$ into the exponent and we get as needed:

$$\begin{aligned}\Pr\left[\sum_i^{l'} R_i - T_i \geq \varepsilon N\right] &\leq \delta + \exp[-\frac{\varepsilon^2\sqrt{N}}{(\log N - \log\delta)^2}] \\ &\leq \exp[-\frac{\varepsilon^2\sqrt{N}}{(\varepsilon^{2/3}N^{1/6})^2}] + \delta \leq 2\delta.\end{aligned}$$

■