On Auditing Elections When Precincts Have Different Sizes

Ronald L. Rivest
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139
rivest@mit.edu

Draft of April 29, 2007[*]

Abstract

We address the problem of auditing an election when precincts may have different sizes, and suggest methods for picking a sample of precincts to audit that precinct size into account. One method yields *optimal* auditing strategies together with an exact measure of its effectiveness (probability of detecting corruption of a given size).

We restrict attention to *basic* auditing strategies, in which each precinct P_i is audited *independently* with some probability p_i determined by the auditor. The auditing probability for a precinct will depend on the size of the precinct, with larger precincts audited more frequently; when all precincts have the same size they will have the same probability of being audited.

We first show how, given the auditor's simpel auditing strategy, how to efficiently compute an *optimal* strategy for the adversary, using dynamic programming. This also yields the *exact* probability that the auditor will detect the adversary's corruption in one or more precincts.

We then show how to embed the above procedure in an efficient iterative optimization computation that appears to always converge upon the optimal auditing strategy.

Finally, we present the "logistic auditing strategy," which is very easy to compute and appears to yield optimal or nearly optimal auditing strategies when the amount of corruption being sought is noticeably larger than the maximum precinct size.

In the logistic approach, the auditor picks each precinct to be audited independently with a probability p that depends on the size v of the precinct (in votes), as follows:

$$p = 1 - \exp(-v/w) \qquad (1)$$

where w is an adjustable globabl parameter that can be chosen to achieve a given confidence level (aka statistical power), to achieve a given expected number of precincts audited, or to achieve a given expected auditing workload. We call this approach a "logistic audit" since equation (1) is an instance of the familiar "logistic function" [11].

As a precinct gets larger, the probability that it is audited increases; as the precinct size passes the threshold w the probability of being audited passes $1 - 1/e \approx 63\%$.

The logistic approach also enables estimation of the probability of detecting fraud: if the adversary corrupts precincts containing a total of at least C votes, then the auditor will detect fraud with probability at least

$$1 - \exp(-C/w) \ .$$

[*]The latest version of this paper can always be found at http://theory.csail.mit.edu/~rivest/Rivest-OnAuditingElectionsWhenPrecinctsHaveDifferentSizes.pdf

For example, by choosing $w \approx C/3$, the auditor will detect fraud of magnitude C with probability at least 95%; choosing $w \approx C/4.605$ gives a 99% confidence level (statistical power).

1 Introduction

Suppose we have an election with n precincts, $P_1, \ldots, P_n$.

Suppose the number of voters who voted in precinct P_i is v_i; we call v_i the "size" of precinct P_i. Let the total number of such voters be $V = \sum_i v_i$. Assume without loss of generality that $v_1 \geq v_2 \geq \cdots \geq v_n$.

Suppose further that in precinct P_i we have both electronic records and paper records for each voter. The electronic records are easy to tally.

For the purposes of this paper, the paper records are used only as a source of authoritative information when the electronic records are audited. They may be considered more authoritative since the voters may have verified them directly.

Auditing is desirable since a malicious party (the "adversary") may have manipulated some of the electronic tallies so that a favored candidate wins the election. (It is also possible that a simple software bug caused the electronic tallies to be inaccurate. However, in this note we are concerned primarily with detecting malicious adversarial behavior, as that is the more challenging task.)

A precinct can be "audited" by re-counting by hand the paper records of that precinct, to confirm that they match the electronic totals for that precinct. (We ignore here the fact that hand-counting may be inaccurate, and assume that any discrepancies are due to fraud on the part of the adversary. In practice, the discrepancy might have to be larger than some prespecified threshold to trigger a conclusion of fraud in that precinct.)

See the overview [6] for information about current election auditing procedures. In this paper we ignore many of the complexities of real elections, in order to focus on our central issue here: how to select a sample of precincts to audit when the precincts have different sizes.

This situation is closely related to the classic notion of an "inspection game", with an "inspector" (the auditor) and an "inspectee" (the adversary). Inspection games fit within the standard framework of game theory. With optimal play, both auditor and adversary use randomized strategies. See Avenhaus et al. [2] for discussion. (In our case, the adversary may choose to use a deterministic strategy, so inspection games have a bit more generality than we need.)

1.1 Auditing objectives and costs

After the election is over, the auditor selects a sample of precincts in which to perform a post-election audit. In each selected precinct the paper ballots are counted by hand, and the totals compared with the electronic tallies.

The auditor wishes to assure himself (and everyone else) that the level of error and/or fraud in the election is likely to be low or nonexistant.

If the audit finds no (significant) discrepancies between the electronic and paper tallies, the auditor announces that no fraud was discovered, and the election results may be certified by the appropriate election official.

If, on the other hand, significant discrepancies are discovered between the electronic and paper tallies, then additional investigations may be needed to determine the nature and extent of the problem. For example, state or federal law may require a full recount of the paper ballots.

When beginning the audit, the auditor knows the size v_i of each precinct P_i.

The auditor also knows the *margin of victory* m of the winning candidate over the runner-up—this is the extra fraction of voters who voted (according to the electronic tallies) for the apparently victorious candidate over the runner-up. As we shall see, the smaller the margin, the more auditing may be appropriate.

We consider three distinct scenarios for the auditing objective depending on the level of effort to be undertaken by the auditor:

- The auditor may have a *legal requirement on the number u of precincts to be audited.* For example, state law may require that 1% of the precincts be audited. (This number may also be mandated as a certain function of the margin of victory.) (See Section 8.)
- The auditor may have a *budgetary constraint on the number A of votes to be recounted.* The workload and cost of auditing is more-or-less proportional to the total number A of votes recounted, rather than to the number u of precincts audited.) (See Section 5.3.)
- The auditor may wish to achieve a certain confidence level that the declared election result is correct—that is, that the true level of error or fraud is unlikely to have affected the election outcome. (Section 5.1). (Our term "confidence level" is equivalent to the notion of "statistical power" in the statistics literature; it is the probability of rejecting a false null hypothesis.)

The logistic approach suggested here can handle all three of the above scenarios.

As a function of the precinct sizes, the margin of victory, and the auditing objective, the auditor will determine how to randomly select an appropriately-sized sample of the precincts to be audited.

1.2 Adversarial Objectives

We assume the adversary wishes to corrupt (or has corrupted) a set of precincts whose total size is at least a given value C, where $0 < C \leq V$.

We now outline the assumptions first suggested by Dopp and Stenger [5] for determining a lower bound on the value of C that would have sufficed to have changed the election outcome.

Let us assume the reported margin of victory is mV (votes), for some margin value m, $0 \leq m \leq 1$. Thus, the reported vote total for the adversary's candidate (the apparent winner) minus the reported vote total for the next possible real winner (the runner-up) needs to have been manipulated by an amount mV, for the adversary's candidate to "win" the election.

We assume that the adversary is willing to "flip" up to $s = 20\%$ of the votes in each precinct chosen for corruption. (If the adversary changes more votes than that within a corrupted precinct, too much suspicion would be generated.) If precinct P_i is corrupted, the advantage gained for the adversary's candidate is then $2sv_i = 0.4v_i$ (the margin of victory changes by two votes for each vote switched).

In this way, the total net improvement (from the adversary's point of view) in the margin of victory for the adversary's candidate is $2sC$. Thus, the adversary must have chosen C so that $2sC \geq mV$ if the election result was actually affected. In general, a corruption level

$$C = \frac{mV}{2s} \tag{2}$$

votes (sum of the sizes of the corrupted precincts) would give the adversary enough leverage to have changed the election outcome. (Equation (2) is due to Dopp and Stenger [5].) Of course, this reasoning applies whether the precincts all have the same size or whether they have different sizes.

We may also assume that the adversary knows the general form of the auditing strategy. Indeed, the auditing strategy may be mandated by law, or described in public documents. While the adversary may not know which specific precincts will be chosen for auditing, he is assumed to know the method by which those precincts will be chosen, and to know the probability that any particular precinct will be chosen for auditing.

In this situation, the adversary can be assumed to use a *deterministic* strategy, and to pick the precincts to be corrupted in a way that is a deterministic function of the margin m of victory, the vote-shift fraction s, and the public details of the auditing strategy. For example, if it is known that the auditing strategy will pick precincts uniformly at random, then the adversary may do best by corrupting a few of the largest precincts only, in order to be able to achieve his goal of corrupting precincts totalling C votes while corrupting as few precincts as possible. (The dynamic programming algorithm of Section 2 gives the general solution to this problem of picking the precincts to be corrupted, both for the case that precincts are picked uniformly at random by the auditor, and for the case that the auditing probabilities are non-uniform.)

We assume here that the adversary wishes to corrupt precincts totalling C votes while minimizing the *probability of detection*—that is, while minimizing the chance that one or more of the precincts chosen to be audited will be one that has been corrupted.

Once C is determined, we then assume that each precinct size is "trimmed" to be at most C: that is, v_i is adjusted downwards as necessary so that it is at most C:

$$v_i \leftarrow \min(v_i, C) \text{ for } 1 \leq i \leq n \ . \qquad (3)$$

A large precinct having $C > v_i$ is sufficiently large so that all of the corruption can be hidden within it; the excess of v_i over C is not needed by the adversary, and should be ignored in the computations—two precincts of size greater than C should have the same probability of being audited. The above trimming procedure ensures this.

1.3 Auditing Strategy

How should the auditor select precincts to audit?

The auditor wishes to maximize the *probability of detection*: the probability that the auditor audits at least one precinct that has been corrupted.

The auditor's strategy should of course be *randomized*, as is usual in game theory. (If the adversary knew exactly which precincts would be audited, then the game is not interesting.)

We first review the case that all precincts have the same size, and then proceed to handle the case of interest in this paper, when precincts have a variety of sizes.

1.4 Same-size precincts

In this section we quickly review the situation when all n of the precincts have the same size v (so $V = nv$).

When all precincts have the same size, then the auditor should pick an appropriate number u of precincts uniformly at random to audit. See Neff [7], Saltman [9], or Aslam et al. [1] for discussion and procedures for calculating appropriate audit sample sizes.

Let us assume that b precincts have been corrupted (they are "bad"), so $C = bv$. Then the probability of detecting at least one corrupted precinct is just

$$1 - \frac{\binom{n-b}{u}}{\binom{n}{u}} \ .$$

By choosing u so that

$$u \geq (n - (b-1)/2)(1 - (1-c)^{1/b})$$

one achieves a confidence level (statistical power) of c that at least one corrupted precinct will be detected, if there are at least b corrupted precincts (See Aslam et al. [1].)

1.5 Basic strategies for varying precinct sizes

The question addressed in this paper is: how should one pick a sample of precincts to audit when the precincts have *different* sizes? What then do the optimal strategies for auditor and adversary look like?

We assume in this paper that the auditor adopts what we will call a "basic" strategy, wherein each precinct is audited *independently* with a probability determined a priori by the auditor. While this represents some restriction on the flexibility of the auditor, we feel that this restriction is not significant in practice. (Furthermore, some lifting of this restriction is explored in Section 5.) Restricting attention "basic" strategies will make the math easier.

We thus assume that the auditor will audit each precinct P_i independently with some probability p_i, where each p_i satisfies $0 \leq p_i \leq 1$. Thus, the auditor's auditing strategy is completely determined by the vector $\mathbf{p} = (p_1, p_2, \ldots, p_n)$.

We assume that vectors $\mathbf{p} = (p_1, p_2, \ldots, p_n)$ and $\mathbf{v} = (v_1, v_2, \ldots, v_n)$ are public knowledge and known to everyone, including the adversary.

We note that the sum $u(\mathbf{p})$ of the p_i's is the expected number of precincts to be audited, which is typically greater than 1.

The expected workload for the auditor (in terms of the expected number of votes to be counted) is

$$v(\mathbf{p}) = \sum_i p_i v_i \ . \qquad (4)$$

2 Optimal Adversarial strategy against a basic auditing strategy

How should the adversary best counter a basic auditing strategy?

The adversary wishes to corrupt precincts with a total of C votes, while minimizing the probability of detection.

When the adversary corrupts precinct P_i, he gains v_i towards his goal of corrupting C votes, but takes a chance p_i of being detected. The adversary succeeds with P_i (i.e., is undetected) with probability $1 - p_i$. For convenience, we let

$$q_i = 1 - p_i$$

denote that probability that P_i will *not* be audited.

If the adversary corrupts a set Q of precincts, then the total gain, in terms of the number of votes in the corrupted precincts, is

$$v(Q) = \sum_{i \in Q} v_i$$

while the overall chance of escaping detection, assuming that each precinct is audited independently, is

$$e(Q) = \prod_{i \in Q} q_i \ .$$

We let $d(Q)$ denote the corresponding detection probability:

$$d(Q) = 1 - e(Q) \ .$$

Each precinct P_i for i in Q contributes v_i to the vote total corrupted, and contributes $\ln(q_i)$ to $\ln(e(Q))$, the logarithm of the escape probability $e(Q)$.

We now describe a simple dynamic programming [4, Chapter 15] algorithm for computing an *optimal* strategy for the adversary; that is, for computing a set of precincts Q to be corrupted with total size at least C that maximizes the chance of escaping detection.

While the adversary's problem is NP-hard (it is a variant of the NP-hard 0-1 knapsack problem), the adversary can take advantage of the fact that the total number V of voters isn't so large in practice. (Perhaps several million at most.) Thus, it is OK to use an algorithm whose running time is polynomial in both n and V.

Here is a sketch of a dynamic programming algorithm for computing exactly an optimal set Q of precincts for the adversary to corrupt, and the corresponding optimal detection probability $d(Q)$.

We let Z_{ij} for $0 \leq i \leq n$ and $0 \leq j \leq V$ denote the maximum of $\ln(e(Q))$ where $Q \subseteq \{1, 2, \ldots, i\}$ and $v(Q) = j$. Then

$$Z_{0,0} = 0 \ ,$$

$$Z_{0,j} = -\infty \text{ for } j > 0,$$

for $i > 0$ and $j < v_i$ we have:

$$Z_{i,j} = Z_{i-1,j}$$

and for $i > 0$ and $j \geq v_i$ we have:

$$Z_{i,j} = \max\{Z_{i-1,j}, Z_{i-1,j-v_i} + \ln(q_i)\} \ . \qquad (5)$$

Then the optimal Q is determined by finding the $j \geq C$ maximizing Z_{nj} (which is $\ln(e(Q))$) by iteratively backtracking through the Z computation to figure out how this maximum escape probability was derived. (This is just like computing the optimal longest-common substring in [4, Step 4, page 354]; it depends only on the information as to which argument was the "max" in the relevant steps of equation (5) as you work backwards.) Code for this computation is available from the author.

This dynamic programming computation is in fact very efficient, taking a mere fraction of a second, even for large jurisdictions.

So, it is easy for the adversary to compute an optimal set Q of precincts to corrupt (and thus also its corresponding minimized detection probability $d(Q)$), given the vectors $\mathbf{v}$ and $\mathbf{p}$.

Let $Q_{opt}(\mathbf{p}, C)$ be the Q determined by the optimal adversary, and let $d_{opt}(\mathbf{p}, C)$ denote the corresponding detection probability.

When all precincts have different sizes but are equally likely to be audited, we have the case studied by Dopp [5] and by Stanislevic [10]; they gave heuristic ("greedy") approximations for computing the optimal adversarial strategy.

(The cost of computing an (nearly) optimal adversarial strategy can be reduced by choosing some factor $k > 1$, replacing each v_i by $\lfloor v_i/k \rfloor$, and replacing C by $\lceil C/k \rceil$. This will speed up the computation by a factor of k, and give an optimal solution to this "rounded" problem, which can either by used as is, or used as a starting point for a search for the optimal strategy to the original problem.)

3 Computing an optimal basic auditing strategy

Now we turn to the question of computing an optimal auditing strategy $\mathbf{p}$, given v.

We first note, for the record, that the optimal auditing strategy has a simple monotonicity property, given that expected number of precincts to be audited is fixed.

Lemma 1 *Let $\mathbf{v} = (v_1, v_2, \ldots, v_n)$ be a non-increasing vector of precinct sizes, and let $\mathbf{p} = (p_1, p_2, \ldots, p_n)$ be the corresponding optimal audit probabilities, for some fixed target $u = \sum_i p_i$ for the expected number of precincts to be audited. Then, without loss of generality, $v_i > v_j$ implies $p_i \geq p_j$ for all i, j.*

Proof: Otherwise switch p_i and p_j. We now argue that this (obviously) preserves $\sum_i p_i$ and doesn't affect the detection probability for any adversarial attack (where the attack also switches i and j when possible). Consider a subset Q of precincts that might be attacked. If Q doesn't involve either i or j, then $d(Q)$. If Q involved only j, then a modified attack $Q' = Q - \{j\} \cup \{i\}$ has $d(Q') = d(Q)$, and still has $v(Q') \geq C$ since $v_i \geq v_j$. If Q involved only i, then $d(Q)$ has increased. Replacing i by j in the attack may not preserve $v(Q') \geq C$, so the adversary may not be able to compensate. If Q involves both i and j, then $d(Q)$ is unchanged. In all cases, the detection probability for the optimum attack has not decreased. ■

We note for the record, however, that $v_i = v_j$ does *not* imply that $p_i = p_j$ in the optimal strategy.

Example: Let $\mathbf{v} = (10, 10, 1, 1)$, $C = 31$, and $u = 1$. Then there is no optimal basic audit strategy with $p_1 = p_2$; an optimal audit strategy will have one of p_1, p_2 equal to 1 and all other probabilities equal to 0.

The preceding example can be used to show that there may be multiple "local maxima" when searching for a globally optimal audit strategy. It also shows that an iterative optimization procedure may need to include some "symmetry breaking" method.

We also note that the above lemma only applies when the auditor's constraint is the number u of precincts audited (rather than, say, the expected number of votes in the audited precincts).

Since we have determined an optimal adversarial strategy, we can view this as a "search" problem for the auditor.

We assume here that the auditor is searching for the best way to audit u precincts, on the average. That is, the auditor wishes to find a probability vector p with $\sum_i p_i = u$ and which is as effective as possible against the optimal adversary—that is, which maximizes $d_{opt}(\mathbf{p}, C)$.

Since the auditor can also compute $d_{opt}(\mathbf{p}, C)$, he can use any one of a variety of search or optimization techniques to compute (or approximate) an optimal $\mathbf{p}$.

The following heuristic technique seems to be quite effective as a search strategy; it determines p_i to be proportional to the product of v_i and the number of times t_i that the adversary has corrupted P_i in the search so far.

1. Initialize counter $t_i = 1$ for $1 \leq i \leq n$.

2. Choose p_i's so that they are proportional to product of t_i and v_i. Let $T = \sum_i t_i v_i$, and Let $p_i = t_i v_i u/T$.

3. Ensure that all p_i's satisfy $0 \leq p_i \leq 1$, by shifting "excess" from all positions i where $p_i > 1.0$ to places j where $p_j < 1.0$, favoring those places j with minimum positive values of $1.0 - p_j$ to "top off" first.

4. Determine $d_{opt}(\mathbf{p}, C)$ and the corresponding Q_{opt}, using the dynamic programming algorithm given above.

5. For each $i \in Q_{opt}$, increment t_i by 1.

6. Return to step 2.

This loop can be iterated until the algorithm appears to have converged.

In testing, this loop often converges to an optimal auditing strategy reasonably rapidly. There are of course many other constrained optimization methods that could be employed instead of the heuristic method outlined above, once you are given the ability to compute the "quality" $d_{opt}(\mathbf{p}, C)$ of a candidate solution $\mathbf{p}$. (I note that d_{opt} is a continuous function of p, although the derivatives may be discontinuous.)

The search can be considerably sped up by finding a good approximate solution by using a "greedy" adversary rather than the optimal adversary. The greedy adversary takes precincts in order of decreasing value q_i^{1/v_i}, until precincts with total size C or more hav been taken. The greedy adversary is easier to compute, so the optimization run considerably quicker for computing the optimal auditing strategy against the greedy adversary. Then one can use that strategy as a starting point for computing the optimal strategy against the optimal adversary.

The logistic strategy (see Section 4 next), is an excellent way to initialize the search for an optimal auditing strategy; it appears to frequently actually be optimum, and may often be reasonably used in practice *instead* of the optimal auditing strategy.

Various other improvements can be made to make this iterative search technique more robust and efficient; the above is just an initial attempt at getting good results in a reasonable amount of time.

One very nice thing about such an iterative improvement technique is that for each basic strategy $\mathbf{p}$ considered, one obtains an *exact* lower bound on the chance of detecting fraud of size C or greater. Thus, one can stop the search at any time, and have in hand a reasonably good basic strategy $\mathbf{p}$, as well as an exact measure of how good it is.

4 Logistic auditing strategy

Although the preceding sections show how to compute an optimal auditing strategy, we now present a heuristic auditing strategy that is much simpler to compute, and which often gives extremely good results. We call it the "logistic" strategy.

Intuitively, the auditor wants to make the "value" for each precinct to the adversary roughly the same; the value v_i gained for the adversary of each precinct should be balanced by the risk of detection.

The auditor should set things up so that, for example, the adversary is indifferent between corrupting a single precinct of size $v_\ell = (v_i + v_j)$ or corrupting two precincts of sizes v_i and v_j respectively. The chance of escaping detection on P_ℓ or escaping detection on *both* of P_i and P_j should be the same.

This implies that the auditor should *not* audit each P_i with probability:

$$q_i = \exp(-v_i/w) \tag{6}$$

where w is some fixed constant. Thus

$$q_\ell = q_i q_j$$

as desired if $v_\ell = v_i + v_j$.

This is the same as saying that q_i^{1/v_i} is constant. so that the (e.g. greedy) adversary doesn't really prefer one precinct over another. The auditor won't be wasting effort auditing precincts the adversary doesn't care about.

Our logistic auditing strategy thus yields

$$p_i = 1 - \exp(-v_i/w) \tag{7}$$

Thus, we propose here that the auditor should use the "logistic" auditing strategy of equation (7), where w may be chosen to satisfy constraints on auditing cost or confidence level.

As v_i increases, the probability of auditing P_i increases, starting off at 0 for $v_i = 0$ and increasing in an s-shaped curve as v_i increases, and levelling off approaching 1 asymptotically for large v_i. The chance of auditing P_i passes $(1 - 1/e) \approx 63\%$ as v_i exceeds w.

The value w can be thought of as approximating a "threshold" value: precincts larger than w have a fairly high probability of being audited, while those smaller than w have a smaller chance of being audited.

As w decreases, the auditing gets more stringent: more precincts are likely to be audited. As w increases, auditing becomes less stringent: fewer precincts are likely to be audited.

One can use any of several standard packages for root-finding to find a value of w that meets given constraints, such as that $\sum_i p_i$ is a given value u. (We used the routine `brentq` from the Python library `scipy.optimize`.)

An important and very convenient property of the logistic audit is that for any set Q of precincts that the adversary may choose to corrupt satisfying

$$s(Q) = \sum_{i \in Q} v_i \geq C \ ,$$

the chance of detection is at least

$$1 - \prod_{i \in Q} \exp(-v_i/w) \geq 1 - \exp(-C/w) \ . \tag{8}$$

This holds no matter what strategy the adversary uses.

In particular, we note that if the adversary can not find a set of precincts whose size totals exactly C votes, then he will choose a set of precincts with size C' where C' is somewhat larger than C, but then the detection probability also becomes larger, since

$$1 - \exp(-C'/w) > 1 - \exp(-C/w) \ .$$

4.1 Relation to picking precincts uniformly

As the amount C of corruption being sought becomes smaller and smaller, the logistic strategy devolves into a strategy of picking precincts uniformly. The reason is that the initial "trimming" of the v_i's given by equation (3) makes all of the v_i's equal (to each other and to C) in the limit. When all of the v_i's are equal, then their auditing probabilities are equal, giving the uniform auditing strategy.

4.2 Relation to picking precincts with probability proportional to size (pps)

An earlier draft of this paper explored a related strategy, whereby precincts were selected for auditing with probability proportional to their size (pps). This had some problems to deal with, as the probabilities computed sometimes exceeded 1 and had to be trimmed. The logistic approach, by contrast, never has audit probabilities exceeding 1. The pps strategy was motivated not by probability of detection, but by the magnitude of the corruption detected; the logistic audit seems better motivated. But the logistic and pps strategies are closely related, since

$$p_i = 1 - \exp(-v_i/w) \approx v_i/w$$

when v_i is small relative to w, so that the pps scheme can be viewed as an approximation to a logistic audit.

4.3 Relation to the optimal auditing strategy

Given the strong motivation behind the logistic strategy, it is perhaps not surprising that it performs exceptionally well in practice.

Empirically, when C is at least two or three times larger than the size of the largest precinct, the logistic strategy often gives essentially optimal results.

It is only in cases where C is not so large relative to the precinct sizes that that the logistic strategy appears to fail.

As one example, with $n = 3$, $\mathbf{v} = (3, 4, 5)$, $C = 8$, $u = 2$, the logistic strategy gives $\mathbf{p} = (0.5698, 0.6753, 0.7549)$ with detection probability $d = 0.8946$, while an optimal auditing strategy is $\mathbf{p} = (0.5, 0.5, 1.0)$ with detection probability 1.0. (Note that the adversary *must* attack P_3 in order to achieve $C = 8$ corrupted votes.

Thus, in practice, one can either just use the logistic strategy as an effective heuristic by itself for computing an auditing strategy $\mathbf{p}$, or one can use it as an extremely good way of getting a starting value for the iterative improvement strategy for computing the optimal $\mathbf{p}$; the latter only seems to be necessary when some precincts are large compared to C.

Either way, don't forget that given $\mathbf{p}$, we have from Section 2 a method for computing the *exact* confidence level (statistical power) for $\mathbf{p}$.

5 Practice

We discuss here how to use the logistic auditing strategy in practice.

5.1 Sampling to achieve a given level of confidence

How can an auditor audit enough to achieve a given confidence level?

The relationship of equation (8) gives a very nice way for the auditor to choose w: by choosing

$$w = \frac{C}{-\ln(1-c)} \tag{9}$$

the auditor achieves confidence of at least c in catching fraud of size at least C, no matter what the adversary's strategy is. For example, by choosing $w \approx C/3$, the auditor achieves confidence level 95% of detecting fraud of size C or greater.

If the auditor starts with the margin of victory, then by using equation (2), the auditor can choose

$$w = \frac{mV}{-2s\ln(1-c)} \tag{10}$$

to determine w in terms of the margin m of victory, the vote-shift amount s per precinct, and the desired confidence level c. For example: for a $m = 1\%$ margin of victory with $s = 0.20$ and confidence level $c = 0.95$, the auditor may choose

$$w = \frac{0.01V}{-2 * 0.20 * (-3)} = 0.0083V$$

(w just under one percent of V) to obtain the desired level of confidence.

5.2 Sampling for a given expected number of precincts to be sampled

Once u is determined in this manner, the auditor can either sample the precincts independently with the logistically determined probabilities, or he can use the procedure of Section 8 to actually select *exactly* u precincts to be audited.

5.3 Sampling to achieve a given workload

In this section, we'll assume that the auditor also starts with an auditing work target A.

The simplest way to proceed is then to just pick each precinct independently with probability p_i. Then the auditing budget will be correct, but only on the average.

It is not clear how to proceed if you want the *actual* workload (number of votes audited) to be (approximately) A, rather than having the *expected* workload be A. (In the same way that the Appendix shows how to pick *exactly* u precincts instead of having an *expected value* of u precincts sampled. Getting the actual workload exactly right is a difficult problem; it is effectively an instance of the well-known NP-complete "knapsack" problem.

Perhaps it would be OK in practice to repeat the drawing of sampling according to the above procedure until auditing budget and the actual auditing work from the strategy chosen are close enough. But the effect of such a procedure on the probability of detecting fraud seems hard to calculate.

5.3.1 Experiment

Mark Lindeman supplied a dataset of 640 precinct sizes (vote counts) for the Ohio congressional district 5 race (OH-05) in 2004, ranging from 1637 (largest) to 132 (smallest), a difference in size by a factor of more than 12.

Let us assume an audit of this district, when the margin of victory is $m = 1\%$. Assume that the adversary will change at most $s = 20\%$ of the votes in a precinct.

Let us assume that an audit of 3% is made: 19 precincts.

With a logistic strategy, the chance of detecting fraud sufficient to have changed the election result is at least 38.31%.

With a uniform strategy for picking 19 precincts, the chance of detecting fraud sufficient to have changed the election result is only 18.08%, *less than half as large as for the logistic strategy.*

Even with a uniform strategy with the same expected workload (number of votes counted), auditing 21.42 precincts on the average, the chance of detecting fraud sufficient to have changed the election result is only 20.13%.

Thus, we see that for real elections, using a logistic audit can sometimes *double* the effectiveness of the audit in terms of the chance of detecting outcome-changing fraud.

It is interesting that when we tried our iterative improvement method, we did not find any improvement over the logistic audit for this problem; it may be that for this example the logistic approach is optimal or extremely close to it.

(The dataset and Python program for this experiment are available at `http://theory.csail.mit.edu/~rivest/pps/oh5votesonly.txt` and `http://theory.csail.mit.edu/~rivest/pps/diff.py`.)

6 Related and Prior Work

See Stanislevic [10] for some discussion of this problem, including a conservative way of handling precincts of different sizes when the auditor is constrained to sampling precincts according to a uniform distribution. This approach was developed independently by Dopp et al. [5]. The idea is to assume that the adversary corrupts the larger precincts first, in order to determine a good lower bound on the number of precincts that must be corrupted, and then to use an approach for auditing that samples precincts uniformly.

See Avenhaus et al. [2] for an excellent survey of "Inspection Games," of which the present problem is an example.

See Neff [7], Cordero et al. [3], Saltman [9], Dopp et al. [5], Rivest [8] for additional discussion of the mathematics of auditing, and additional references to the literature.

7 Discussion

It would be preferable in general, rather than having to deal with precincts of widely differing sizes, if one could some divide the records for the larger precincts into "bins" for "pseudo-precincts" of some smaller standard size. (You can do this for say paper absentee ballots, by dividing the paper ballots into nominal standard precinct-sized batches before scanning them.) It is harder to do this if you have DRE's with wide disparities between the number of voters voting on each such machine.

Perhaps legislation should express mandatory lower bounds on the amount of auditing that needs to be performed in terms of the fraction of votes in the audited precincts, rather than merely in terms of the fraction of the number of precincts.

But the best approach for legislation is probably merely to mandate a lower bound on the confidence level that must be achieved in terms of the probability that fraud of sufficient magnitude to have changed the election outcome will be detected.

8 Conclusions

We have presented algorithms for computing an optimal auditing strategy, as well as a powerful "logistic" approach to computing an auditing strategy.

Acknowledgments

Thanks to Mark Lindeman for helpful discussions and the Ohio dataset of precinct sizes, and to Christos Papadimitriou for referring me to the literature on "inspection games." Thanks also to Kathy Dopp, Silvio Micali, and Howard Stanislevic for constructive suggestions.

References

[1] Javed Aslam, Raluca Popa, and Ronald L. Rivest. On estimating the size and confidence of a statistical audit, 2007. Available at: `http://theory.csail.mit.edu/~rivest/AslamPopaRivest-OnEstimatingTheSizeAndConfidenceOfAStatisticalAudit.pdf`.

[2] Rudolf Avenhaus, Bernhard Von Stengel, and Shmuel Zamir. Inspection games. In R. J. Aumann and S. Hart, editors, *Handbook of Game Theory*, volume III. January 30 1998. Available at: `http://citeseer.ist.psu.edu/212144.html`.

[3] Arel Cordero, David Wagner, and David Dill. The role of dice in election audits — extended abstract, June 16 2006. To appear at IAVoSS Workshop on Trustworthy Elections (WOTE 2006). Preliminary version available at: `http://www.cs.berkeley.edu/~daw/papers/dice-wote06.pdf`.

[4] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms (Second Edition)*. MIT Press / McGraw-Hill, 2003.

[5] Kathy Dopp and Frank Stenger. The election integrity audit, 2006. `http://electionarchive.org/ucvAnalysis/US/paper-audits/ElectionIntegrityAudit.pdf`.

[6] ElectionLine.org. Case study: Auditing the vote, March 2007. Available

at: http://electionline.org/Portals/1/Publications/EB17.pdf.

[7] C. Andrew Neff. Election confidence—a comparison of methodologies and their relative effectiveness at achieving it (revision 6), December 17 2003. Available at: http://www.votehere.net/papers/ElectionConfidence.pdf.

[8] Ronald L. Rivest. On estimating the size of a statistical audit, 2006. Unpublished. Available at: http://theory.csail.mit.edu/~rivest/Rivest-OnEstimatingTheSizeOfAStatisticalAudit.pdf.

[9] Roy G. Saltman. Effective use of computing technology in vote-tallying. Technical Report NBSIR 75–687, National Bureau of Standards (Information Technology Division), March 1975. Available at: http://csrc.nist.gov/publications/nistpubs/NBS_SP_500-30.pdf.

[10] Howard Stanislevic. Random auditing of e-voting systems: How much is enough?, revision August 16, 2006. Available at: http://www.votetrustusa.org/pdfs/VTTF/EVEPAuditing.pdf.

[11] Wikipedia. Logistic function. Available at: http://en.wikipedia.org/wiki/Logistic_function.

Appendix. Sampling a fixed number of precincts

In this section we consider the question of how to sample a given number u of precincts, but doing so with given probabilities for each precinct. (The number u of precincts to be audited may result, for example, from a legal requirement.)

Thus, we consider the case that

$$\sum_i p_i = u \ .$$

We would like a sampling procedure that given a vector $\mathbf{p}$ of probabilities that sum to u, returns each time with a subset of the precincts, such that the subset always has size *exactly* u, and such that precinct P_i is sampled with probability p_i.

We assume that $p_1 \geq p_2 \geq \ldots \geq p_n$, without loss of generality.

There are several approaches one might take to solving this problem; the solution we describe here attempts to maximize the independence of the choices to select each precinct.

We assume that we have already available two procedures: one for generating random real numbers in the interval $[0, 1]$, and one for selecting *uniformly* at random a set of u elements from a given set of n elements. (See Cordero et al. [3] for a discussion of practical procedures for doing so, given the available of ten-sided dice.)

Here is the procedure. Let M be initialized to the empty set of precincts.

1. If there are no precincts in the list, return M as the subset generated, and stop.

2. If $p_n = 0$, remove precinct P_n from further consideration; it will not be included in the u-subset generated on this run. Restart this procedure at step 1 with the reduced list of $n - 1$ precincts.

3. If $p_1 = 1$, remove precinct P_1 from the list of precincts being considered, reduce u by 1, and add precinct P_1 to M. (Precinct P_1 will be "mandatory" to include in the u-subset generated on this run.) Restart this procedure at step 1 with the reduced list of $n - 1$ precincts and the reduced value of u.

4. (Now each remaining precinct has a probability strictly greater than 0 and strictly less than 1.) The precincts remain sorted in order, so that $1 > p_1 \geq p_2 \geq \cdots \geq p_n > 0$.) Let
$$x = \frac{(1 - p_1)n}{n - u}$$
$$y = \frac{p_n n}{u}$$

and let

$$z = \min(x, y) \ .$$

Note that z is positive and at most 1. With probability z select a u-subset S uniformly at random from the set of all u-subsets of $\{P_1, P_2, \ldots, P_n\}$, return $S \cup M$ (the union of S and M), and stop.

5. Otherwise (i.e., with probability $1 - z$, which must be nonzero for us to get to this step), for each precinct P_i, update its probability from p_i to

$$\frac{p_i - zu/n}{1 - z}$$

and restart the procedure from step 1 with this new set of probabilities. (Note that they remain sorted in order.)

This procedure halts in at most $n - u$ iterations, and always gives the correct marginal probabilities. (The way p is chosen ensures that the new probabilities for the following round are non-negative and at most one. To show termination, it suffices to show that when $p < 1$ the total number of probabilities equal to 1 or equal to 0 in a round increases by at least one going into the next round: when $p = x$ the number of ones increases, and when $p = y$ the number of zeros increases.)

Example

Suppose we have four precincts with respective vote counts:

$$v_1 = 40, v_2 = 30, v_3 = 20, \text{ and } v_4 = 10 \ .$$

Suppose further we wish to pick exactly two of them with probabilities given by the logistic approach, so we determine that $w \approx 33.4$ (via numerical optimization) and that

$$\begin{aligned} p_1 &= 1.0 - \exp(-40/33.4) \approx 0.698069, \\ p_2 &= 1.0 - \exp(-30/33.4) \approx 0.592685, \\ p_3 &= 1.0 - \exp(-20/33.4) \approx 0.450517 \text{ and} \\ p_4 &= 1.0 - \exp(-10/33.4) \approx 0.258728 \ . \end{aligned}$$

Running the algorithm, we develop the following procedure for doing the selection.

1. Pick a number r uniformly at random from the interval $0 \le r \le 1$.

2. If $0 \le r < 0.517458$, then select two distinct precincts uniformly at random from $\{P_1, P_2, P_3, P_4\}$, and return them as the sample.

3. If $0.517458 < r \le 0.647064$, then take two of precincts $\{P_1, P_2, P_3\}$ (chosen uniformly) and return this pair of distinct precincts as the sample.

4. If $0.647064 < r \le 0.857833$, then take precinct P_1 and one of precincts $\{P_2, P_3\}$ (chosen uniformly) and return this pair of distinct precincts as the sample.

5. If $0.857833 < r \le 1.00$, then return $\{P_1, P_2\}$ as the sample.

While the above strategy may seem a bit magical, it can be confirmed that each precinct is selected with the correct probability. For example, precinct P_2 is selected with probability

$$\begin{aligned} 0.592685 \ = \ & 0.517458 * (1/2) + \\ & (0.647064 - 0.517458) * (2/3) + \\ & (0.857833 - 0.647064) * (1/2) + \\ & (1 - 0.857833) \end{aligned}$$

A simple program can print out the strategy for a given set of probabilities; see the program `diff.py` for such a program.