Election Manipulation with Partial Information

Michelle Blom[1], Peter J. Stuckey[2], and Vanessa J. Teague[1]

[1][michelle.blom,vjteague]@unimelb.edu.au
School of Computing and Information Systems
The University of Melbourne
[2]peter.stuckey@monash.edu
Faculty of Information Technology
Monash University

Abstract. We consider the case of manipulating the results of Instant Runoff Voting (IRV) elections. Previous work in this area looked at *posthoc* manipulation with complete information, where the manipulator may alter ballots after reading the whole election profile. In this paper we examine the much more realistic, but challenging, problem of manipulating ballots during the election process, having observed only some ballots. The aim of the manipulator is to modify as few ballots as possible to ensure their candidate's victory with high probability. We show that this it quite feasible in practice to generate efficient manipulations with a high probability of success. We also add some extra conditions on the manipulations to it less likely they will be detected by naive methods.

1 Introduction

Instant Runoff Voting (IRV), also known as Alternative Vote (AV), is a system of preferential voting in which voters rank candidates in order of preference. Tallying proceeds by eliminating the least popular candidate and redistributing their votes to the next-preferred candidate—see Section 2 for a precise algorithm. It is used in presidential, parliamentary and local government elections in countries such as Australia, Fiji, Papua New Guinea, Ireland, Bosnia/Herzogovinia, the UK and United States [6].

IRV elections can be complicated—an early alteration in the elimination order can cascade into an entirely different election result. It is NP-hard even to compute the true margin of victory [7]. But the hard examples are somewhat contrived, and real elections follow patterns which make it relatively easy in practice to compute both margins [5, 3] and successful manipulations [2]. However, prior work has required complete information about all the votes cast in the election. In contrast, the related problem of manipulating ones own vote to increase the likelihood of a desired outcome has been studied in the context of both full and partial knowledge of the preferences of other voters [4] .

In this work, we examine for the first time whether an attacker with only partial information can devise a successful manipulation with a high probability of success. This models a realistic attacker (such as a corrupt scanner or voting machine) who must generate manipulated electronic records on the fly after reading only a few of them. The problem we consider is known as 'election control by replacing voters' in the computational social choice literature. We find that it remains easy to generate successful manipulations, though of course success is not guaranteed. Our emphasis is on practical attacks in a realistic setting. We do not prove optimality nor a bound on the probability of success, but we demonstrate successful practical attacks by simulating the algorithm on state election data from Australia. We also examine a number of ways the attacker could refine the manipulation to be less likely to raise suspicion.

We assume the adversary has sufficient access to the election system to read ballots as they are received, and to modify each ballot before it is recorded. Unlike in prior work [2], our adversary cannot modify ballot recordings once made, but must perform manipulations based on the partial sample seen so far. This models a man-in-the-middle attack where ballots are intercepted between the voter and the final counting, and altered by the adversary. It is precisely the attacker model of a corrupted scanning process in which scanned ballots must be immediately output, but may be misrecorded on the fly. It is also appropriate for an e-voting process with immutable logs, in which the manipulated votes must be written immediately onto the log and cannot later be altered.

We look at an adversary that computes some manipulation rules to apply at periodic intervals, after they have seen some ballots scanned through. The attacker succeeds if their desired candidate wins. We compare a number of ways of generating the rules, and how effective they are at actually achieving a desired winner change. We find that if the adversary sees the first quarter or half of the votes then, with a few seconds of computation, it can compute a manipulation close to the optimal, and then apply it to the next $3/4$ or half of the votes with a success probability usually higher than 90%.

Avoiding suspicion The last round margin (LRM) of an IRV election – the difference in tallies of the final two remaining candidates, divided by two and rounded up – is commonly used as an indicator of how close the election was. Blom *et al.* [3] have shown that the true margin of victory (MOV) of the election – the smallest number of votes one would have to alter to change who won the election – is generally equal to the last round margin, but not always. In some cases, the MOV can be much smaller than the last round margin. The Australian Electoral Commission (AEC) use the "margin between the two leading candidates" after all remaining candidates have been eliminated, and their preferences distributed, to determine whether an automatic recount of cast votes should be automatically performed.[1] In practice, the LRM plays a major role in determining whether further scrutiny of the outcome is performed.

Our first suspicion-avoiding extension is for the adversary to generate manipulations that result in a large last-round margin. The adversary wants to alter the smallest number of electronic records so that their desire of changing the outcome is realised, while at the same time ensuring that the last round margin of the manipulated election is larger than a given threshold. Another important way of avoiding suspicion is to minimize changes in first preferences. This is significant because, in preferential elections, first-preference tallies are often manually counted in a polling place independently of the scanning process. Modifying first preferences is therefore much more likely to be detected than other alterations. We find, surprisingly, that this constraint usually makes an insignificant difference to the number of ballots that need to be changed. A clever

[1] https://www.aec.gov.au/Elections/candidates/files/hor-recount-policy.pdf. The AEC definition of a "margin" is the difference in tallies of two candidates (not divided by two).

Initially, all candidates remain standing (are not eliminated)
While there is *more than one* candidate standing
  For every candidate c standing
    Tally (count) the votes in which c is the highest-ranked candidate of those standing
  Eliminate the candidate with the smallest tally
The winner is the one candidate not eliminated

Fig. 1: The IRV vote tallying process: the candidate with the smallest tally is repeatedly eliminated, with the ballots in their tally redistributed according to their next preference.

surreptitious manipulation is almost as easy as an obvious one. These attacks would be defeated if a rigorous risk-limiting audit was applied. The trick with LRMs would be defeated by a careful computation of the true MOV [3]. The purpose of this paper is to demonstrate that these rigorous methods are necessary, because an active adversary can defeat heuristic methods of assessing when to re-examine an election result.

Our contributions Using the Australian New South Wales (NSW) 2015 Legislative Assembly election as a case study, we simulate the attack by randomly determining the order in which ballots arrive at the scanner, computing a best-guess manipulation at periodic intervals, and then applying that manipulation to later ballots as they are scanned. We then compare the outcome of the modified election to that of the unmodified election to see whether the manipulation succeeded.

We report the number of ballots that this adversary needs to modify, and the number of first preferences on ballots they need to modify, in order to have a high chance of altering the election outcome, assuming that the proportion of the ballots they saw before computing the manipulation were a random sample of all ballots.

2 Preliminaries and prior work

Votes are tallied in an IRV election in a series of rounds (see Figure 1). In each round, the candidate with the smallest number of votes (their tally) is eliminated, with the last remaining candidate declared the winner. All votes in an eliminated candidate's tally are distributed to the next most-preferred (remaining) candidate in their ranking.

We denote an IRV election, with a set of candidates $\mathcal{C}$, as $\mathcal{B}$. A sequence of candidates π is represented in list notation (e.g., $\pi = [c_1, c_2, c_3, c_4]$, which means that c_1 is the highest preference, c_2 the next-preferred, and so on). Such sequences represent both votes and the order in which candidates are eliminated. An election $\mathcal{B}$ is defined as a multiset[2] of votes, each vote $b \in \mathcal{B}$ a sequence of candidates in $\mathcal{C}$, with no duplicates, listed in order of preference (most preferred to least). The first candidate appearing in a sequence π is denoted $\mathit{first}(\pi)$ (e.g., $\mathit{first}([c_2, c_3]) = c_2$). In each round of vote counting, there are a current set of eliminated candidates $\mathcal{E}$ and a current set of candidates still standing $\mathcal{S} = \mathcal{C} \setminus \mathcal{E}$. The winner c_w of the election is the last standing candidate.

[2] A multiset allows for the inclusion of duplicate items.

Our definitions for a candidate's tally, the margin of victory (MOV), and last round margin (LRM) of an IRV election are replicated from [2].

Definition 1. ***Tally*** $\mathbf{t}_{\mathcal{S}}(\mathbf{c})$ *Given candidates $\mathcal{S} \subseteq \mathcal{C}$ are still standing in an election $\mathcal{B}$, the tally for candidate $c \in \mathcal{C}$, denoted $t_{\mathcal{S}}(c)$, is defined as the number of votes $b \in \mathcal{B}$ for which c is the most-preferred candidate of those remaining.*[3] *Let $p_{\mathcal{S}}(b)$ denote the sequence of candidates mentioned in b that are also in $\mathcal{S}$.*

$$t_{\mathcal{S}}(c) = \ | \, [b \in \mathcal{B} \mid c = \mathit{first}(p_{\mathcal{S}}(b))] \, | \tag{1}$$

Definition 2. ***Margin of Victory (MOV)*** *The MOV in an election $\mathcal{B}$ with candidates $\mathcal{C}$ and winner $c_w \in \mathcal{C}$, is the smallest number of votes in $\mathcal{B}$ whose ranking must be modified (by an adversary) so that a candidate $c' \in \mathcal{C} \setminus \{c_w\}$ is elected.*

Definition 3. ***Last Round Margin (LRM)*** *The LRM of an election $\mathcal{B}$, in which two candidates $\mathcal{S} = \{c, c'\}$ remain with $t_{\mathcal{S}}(c)$ and $t_{\mathcal{S}}(c')$ votes in their tallies, is equal to half the difference between the tallies of c and c' rounded up.*

$$LRM = \left\lceil \frac{|t_{\mathcal{S}}(c) - t_{\mathcal{S}}(c')|}{2} \right\rceil \tag{2}$$

In the design of our adversary, we consider the concept of an elimination margin (EM) – the margin by which a candidate is eliminated in a given round. We find that a manipulator can be much more effective when controlling the elimination margin of the two runner-ups (the candidate eliminated just prior to the runner-up, and the runner-up themselves) rather than just the LRM of the runner-up and winner. In settings where there is a genuine three candidate race for winner, successful manipulations typically require the elimination of a specific candidate in third place whose preferences will mostly flow to the desired winner. Consider a manipulation designed to elect a candidate a with a certain margin, but where tallies of the candidates up for elimination in that crucial third place position are similar. Given uncertainty in what types of ballots a manipulator will see as it is changing votes on the fly, it is likely to be more successful it aims to eliminate the required candidate in third place with a larger elimination margin.

Definition 4. ***Elimination Margin (EM)*** *The EM in a round of counting i, in which candidate c_i is eliminated and candidates $\mathcal{S} \setminus \{c_i\}$ are still standing, is equal to half the smallest difference in tallies between $t_{\mathcal{S}}(c_i)$ and $t_{\mathcal{S}}(c')$ for $c' \in \mathcal{S} \setminus \{c_i\}$ rounded up.*

$$EM_i = \min_{c' \in \mathcal{S} \setminus \{c_i\}} \left\lceil \frac{t_{\mathcal{S}}(c') - t_{\mathcal{S}}(c_i)}{2} \right\rceil \tag{3}$$

Example 1. Consider the election with ballots shown in Table 1(a). The true election result is shown in Table 1(b), with c the winner with a last round margin of 13. The margin of victory is actually only 5. Changing five ballots from $[c, a]$ to $[b, c]$ results the totals shown in Table 1(e). In this election a wins as shown in Table 1(f). In the true election, b is eliminated with an EM of $\lceil 9/2 \rceil = 5$ votes, and a with an EM of 13. $\square$

[3] Square brackets have been used to denote a multiset.

Table 1: IRV example, with (a) the number of votes cast with each listed ranking over candidates a, b, c, and (b) tallies after each round of vote counting (c) the ballot count 25% into the election and the estimated complete election (d) the tallies of each round of vote counting in the estimated election (e) the number of votes recorded after manipulation, and (f) the tallies after each round of vote counting in the manipulated election

Ranking	Count $\mathcal{B}$
$[a]$	55
$[c, a]$	30
$[b, c]$	36
$[c]$	15

(a)

Candidate	Round 1	Round 2
a	55	55
b	36	—
c	45	81

(b)

Ranking	Partial Count $\mathcal{B}_\mathcal{T}$	Estimated Completion $\hat{\mathcal{B}}$
$[a]$	11	44
$[c, a]$	9	36
$[b, c]$	10	40
$[c]$	4	16

(c)

Candidate	Round 1	Round 2
a	44	44
b	36	—
c	52	88

(d)

Ranking	Manipulated Count
$[a]$	55
$[c, a]$	25
$[b, c]$	41
$[c]$	15

(e)

Candidate	Round 1	Round 2
a	55	80
b	41	41
c	40	—

(f)

2.1 Modifying an Election Outcome

In this paper we make use of algorithms for determining minimal manipulations of IRV elections in order to change their outcome, developed by Blom *et al* [2]. They consider the case where all recorded ballots can be manipulated *after* they have all been scanned. This is clearly a necessary starting point for determining a more restricted manipulation.

These algorithms are based on the *margin-irv* algorithm for computing the true MOV of an IRV election. A description of *margin-irv* can be found in Blom *et al.* [3]. We summarise the algorithm in this section, and explain how it can be modified to compute the smallest number of vote changes required to: (i) bring about a change in the outcome of the election; (ii) produce a manipulated election with certain properties, modelled as side constraints; and (iii) produce a manipulated election that minimises discrepancies between a manual hand count of first preference tallies (based on non-manipulated paper ballots) and the first preference tallies of the manipulated election.

The *margin-irv* algorithm Consider an IRV election $\mathcal{B}$ with candidates $\mathcal{C}$ and winner $w \in \mathcal{C}$. The *margin-irv* algorithm starts by adding $|\mathcal{C}| - 1$ partial elimination sequences to a search tree, one for each of alternate or desired winner $c \in \mathcal{C} \setminus \{w\}$. These partial sequences form a frontier F, with each sequence containing a single candidate – an alternate winner. Note that a partial sequence [a, b, c] represents an election outcome in which a and b are the last two candidates eliminated, and c the winner. All other candidates are assumed to have been eliminated in some prior round.

An adversary is likely to to have a desired winner $a \neq w$. We can modify *margin-irv* by initializing the frontier F with the single sequence $[a]$. Then it only considers election sequences where a wins. The same idea is used by Blom *et al.* [1].

For each partial sequence $\pi \in F$, we compute a lower bound on the number of vote changes required to realise an elimination sequence that *ends* in π. These lower bounds are used to guide construction of the search tree, and are computed by both solving an Integer Linear Program (ILP), and applying several rules for lower bound computation. These rules are described in Blom *et al.* [3]. The ILP, denoted DISTANCETO, computes a lower bound on the smallest number of vote changes required to transform the election $\mathcal{B}$, with an elimination sequence π', to one with an elimination sequence that ends in π. When applied to a complete order π, containing all candidates, DISTANCETO exactly computes the smallest number of votes changes required to realise the outcome π. The largest of the lower bounds computed by the rules of Blom *et al.* [3] and the DISTANCETO ILP is assigned to each partial sequence π as it is added to F. The DISTANCETO ILP is defined in Section 2.2. To enforce additional constraints on the nature of any manipulated election, we add these constraints to each ILP solved.

The partial sequence $\pi \in F$ with the smallest assigned lower bound is selected and *expanded*. For each $c \in \mathcal{C}$ that is not already present in π, we create a new sequence with c appended to the front. For example, given a set of candidates e, f, and g, with winning candidate g, the partial sequence $\pi = [f]$ will be expanded to create two new sequences $[e, f]$ and $[g, f]$. We evaluate each new sequence π' by assigning it a lower bound on the number of votes required to realise any elimination order ending in π'.

While exploring and building elimination sequences, *margin-irv* maintains a running *upper bound* on the value of the true margin. Without any side constraints designed to inject desirable properties into a manipulated election, this upper bound is initialised to the last round margin of the original election. To enforce additional constraints on the properties of any manipulated election, we need to manipulate at least as many, and often more, votes than required to simply change the original outcome. Consequently, we must set the upper bound maintained by *margin-irv* to a higher value. In this context, we set the initial upper bound to the total number of votes cast in the election. This is clearly always a correct upper bound on any manipulation.

When a sequence π containing all candidates is constructed, the DISTANCETO ILP computes the exact number of vote manipulations required to realise it, while satisfying all desired side constraints. If this number is lower than our current upper bound, the upper bound is revised, and all orders in F with a lower bound greater than or equal to it are pruned from consideration (removed from F). This process continues until F is empty (we have considered or pruned all possible alternate elimination sequences). The final value of the running upper bound is the true electoral MOV (with side constraints).

2.2 DISTANCETO with Side Constraints

We now present the DISTANCETO Integer Linear Program (ILP) used to compute lower bounds on the degree of manipulation required to realise an election outcome ending in a given candidate sequence, and the (exact) smallest number of vote changes required to realise a given (complete) alternate elimination sequence. This ILP, without added side constraints, was originally presented by Magrino *et al.* [5].

We consider additional side constraints to inject desirable properties into any manipulated election, and describe how we can minimise both the total number of vote changes required to elect a desired winner, and the total changes made to first preference tallies between the true and manipulated election profiles.

Let $\mathbf{R}$ denote the set of possible (partial and total) rankings R of candidates $\mathcal{C}$ that could appear on a vote, N_R the number of votes cast with ranking $R \in \mathbf{R}$, and N the total number of votes cast. Let $\mathcal{R}_{j,i}$ denote the subset of rankings ($\mathcal{R}_{j,i} \subset \mathbf{R}$) in which c_j is the most preferred candidate still standing (i.e., that will count toward c_j's tally) at the start of round i (in which c_i is eliminated). For each $R \in \mathbf{R}$, we define variables:

q_R integer number of votes to be changed into R;
m_R integer number of votes with ranking R in the unmodified election to be changed into something other than R; and
y_R number of votes in the modified election with ranking R.

Given a partial or complete order π, the DISTANCETO ILP is:

$$\min \sum_{R \in \mathbf{R}} q_R \tag{4}$$

$$N_R + q_R - m_R = y_R \qquad \forall R \in \mathbf{R} \tag{5}$$

$$\sum_{R \in \mathbf{R}} q_R = \sum_{R \in \mathbf{R}} m_R \tag{6}$$

$$\sum_{R \in \mathcal{R}_{i,i}} y_R \leq \sum_{R \in \mathcal{R}_{j,i}} y_R \qquad \forall c_i, c_j \in \pi \;.\; i < j \tag{7}$$

$$n \geq y_R \geq 0, \;\; N_R \geq m_R \geq 0, \;\; q_R \geq 0 \qquad \forall R \in \mathbf{R} \tag{8}$$

Constraint (5) states that the number of votes with ranking $R \in \mathbf{R}$ in the new election is equal to the sum of those with this ranking in the unmodified election and those whose ranking has *changed to* R, minus the number of votes whose ranking has been *changed from* R. Constraint (7) defines a set of *special elimination constraints* which force the candidates in π to be eliminated in the stated order. Constraint (6) ensures that the total number of votes cast in the election does not change as a result of the manipulation. The objective minimises the total number of ballot changes required to manipulate the election and enforce a desired elimination sequence.

The above ILP does not include any additional side constraints – properties that we want the manipulated election to satisfy besides resulting in a different winner to that of the original election. Manipulated elections found by *margin-irv* in this setting are almost always evidently close, with a last round margin of 0 or 1 vote. This makes sense as the algorithm is trying to manipulate as few votes as possible, breaking any ties in favour of an alternate outcome. An adversary with the ability to modify electronic records of cast votes, however, will want to create a manipulated election that is not evidently close. An election with a tie in the final round of counting, or a difference of several votes in the tallies of the final two remaining candidates, is likely to be closely scrutinised. Australian IRV elections in which the final tallies of the last two candidates differ by less than 100 votes, for example, trigger an automatic recount.

Given the widespread use of the last round margin as the indicator of how close an IRV election is, rather than the true MOV of the election, our adversary can use this to their advantage. Consider a candidate elimination sequence π, containing at least n candidates from a set $\mathcal{C}$. Let the last n candidates in the sequence π be denoted by c_k, c_{k+1}, ..., c_{k+n}, with c_{k+n} denoting the winning candidate according to π. Adding the following side constraint to DISTANCETO ensures that margin by which each candidate c_i for $k \leq i \leq k+n-1$ is eliminated (the EM, Definition 4) is at least Δ votes. This allows us to ensure that both the last round margin of the manipulated election, and the elimination margin of any number of prior rounds, is at least a certain size.

$$\sum_{R \in \mathcal{R}_{i,i}} y_R \leq \sum_{R \in \mathcal{R}_{j,i}} y_R + 2\,\Delta \qquad \forall i \in \{k, \ldots, k+n-1\} \quad (9)$$

An important side constraint we will make use in the design of our manipulators is *limiting the manipulation*. Since in our scenario the attacker will modify ballots in the middle of the election its important that we calculate manipulations that do not remove ballots already recorded. Suppose C_R is the number of ballots already recorded for ranking R. We must ensure that the modified election has at least C_R ballots with ranking R, since they cannot be changed. Adding the following constraint ensures this.

$$y_R \geq C_R \qquad \forall R \in \mathbf{R} \quad (10)$$

2.3 Minimising Change to First Preference Counts

To both minimise discrepancies between any manual count of first preference tallies, and that computed by counting software applied to a manipulated election, we solve the DISTANCETO ILP of Section 2.2 twice for each complete elimination sequence π.

For every such π that *margin-irv* encounters (these are the leaves of the generated search tree), we first solve the ILP with an objective to minimise first preference tally discrepancies. Let $t_{\mathcal{C}}(c)$ denote the first preference tallies of each candidate $c \in \mathcal{C}$. The first preference tally for candidate $c \in \mathcal{C}$ in any solution of our ILP, $t'_{\mathcal{C}}(c)$, is given by:

$$t'_{\mathcal{C}}(c) = \sum_{R \in \mathbf{R}. c = first(R)} y_R \quad (11)$$

Let FPD denote the total number of first preference tally discrepancies between any manipulated profile found by DISTANCETO ILP, and the true election profile of the original election (Equation 12). For each complete candidate elimination sequence π, containing all candidates, we first solve the DISTANCETO ILP with the objective shown in Equation 13. Let fpd denote the value of FPD in the optimal solution found when solving DISTANCETO with this objective. We then constrain the ILP to ensure that any subsequently found solutions must satisfy the constraint in Equation 14.

$$FPD = \sum_{c \in \mathcal{C}} |t_{\mathcal{C}}(c) - t'_{\mathcal{C}}(c)| \tag{12}$$

$$\min FPD \tag{13}$$

$$FPD \leq fpd \tag{14}$$

We re-solve our DISTANCETO ILP with the objective shown in Equation 4 – to minimise total ballot changes given the constraint limiting permitted change to first preference counts. The result is a manipulated election profile that first aims to minimise the total number of discrepancies between the true and manipulated elections, as a first priority, and the total number of ballot changes as a second.

3 Manipulation Algorithms

We consider a setting in which all ballots are assumed to be scanned at a central location, and the resulting electronic records passed into counting software that computes the result of the election. We assume the scanner has been compromised, and the attacker is able to manipulate (change) the ranking on the electronic record of ballots at the moment they are scanned. Alternatively, we can also consider a setting where votes are scanned at disparate locations, but the attacker is able to intercept and modify the scanned record before it is passed to the counting software. In both cases the attacker manipulates the record of ballots before they are used for counting. In either case, first preference counts may have been completed manually at polling booths and/or at the central scanning location. We assume the attacker has some reasonable estimate of the total number of ballots E expected in the election. Note that in places like Australia with compulsory voting the expected number of ballots E is known with high confidence, in other jurisdictions there will be more variance. Across all experiments in which we simulate our manipulators, we use $E = |\mathcal{B}|$ where $\mathcal{B}$ is the historical set of ballots.

The algorithm applied by our manipulators is as follows. The parameter α, determining how often the attacker computes a set of manipulation rules, and k, the number of rounds on which to apply an elimination margin constraint, are given as input.

1. Let E be the expected number of ballots for the election. Let $n = \lceil \alpha E \rceil$ be a proportion α of this expected number of votes.
2. Collect the first n ballots $\mathcal{B}_{\mathcal{T}}$, passing them on to the counting software unmanipulated. Let $\mathcal{B}_{\mathcal{M}} = \mathcal{B}_{\mathcal{T}}$, the current profile of the manipulated election.
3. Compute an approximate complete election profile $\hat{\mathcal{B}}$ by extending the (manipulated) ballots $\mathcal{B}_{\mathcal{M}}$ processed so far with ballots uniformly drawn from the set of true (unmanipulated) ballots seen so far $\mathcal{B}_{\mathcal{T}}$.
4. Use the methods of [2] to determine a minimal manipulation $\mathcal{M}$ of $\hat{\mathcal{B}}$ in order to achieve the desired winner with an EM of Δ applied to the last k rounds. Note that this manipulation may be null if the desired winner already win $\hat{\mathcal{B}}$ by Δ.
5. Examining the minimal manipulation made in $\mathcal{M}$ and the assumed unseen ballots $\mathcal{U} = \hat{\mathcal{B}} - \mathcal{B}_{\mathcal{M}}$, determine a set of manipulation rules $\mathcal{R}$ which will ensure that applying $\mathcal{R}$ to $\mathcal{U}$ will result in manipulation $\mathcal{M}$.

6. Intercept the next n ballots. If an incoming ballot b matches one of the manipulation rules, $r \in \mathcal{R}$, replace b by $r(b)$ before passing it on to the counting software.
7. Let $\mathcal{B}_{\mathcal{T}}$ be the true ballots seen so far. Let $\mathcal{B}_{\mathcal{M}}$ be the manipulated ballots processed so far – the current state of the manipulated election profile. If all ballots have been processed, the algorithm is complete, otherwise we return to step 3.

We now examine the individual steps in the approach in detail.

Completing an Election At any given point during the scanning of ballots, our adversary has seen a proportion δ of the total number of expected ballots, E. The manipulated election profile at this point contains δE ballots. Some of these ballots have been manipulated (altered from their true state), and others have been left unchanged. To compute a set of manipulation rules to apply to future ballots, the adversary needs to estimate what a complete election profile could look like (i.e., a profile containing the current set of manipulated ballots, and an estimate of future ballots). Let $\hat{\mathcal{B}}$ denote this estimated profile. To compute $\hat{\mathcal{B}}$, we start with the current set of ballots in $\mathcal{B}_{\mathcal{M}}$ and add $E - |\mathcal{B}_{\mathcal{M}}|$ further ballots. These additional ballots are drawn uniformly at random (with replacement) from $\mathcal{B}_{\mathcal{T}}$ – the set of ballots, unmodified, that have been seen so far. Each sampled ballot is added to $\hat{\mathcal{B}}$.

Example 2. Suppose the first quarter of ballots of the election in Table 1(a), $\mathcal{B}_{\mathcal{T}}$, is as shown in Table 1(c). Then an estimation of the complete election $\hat{\mathcal{B}}$ might be determined as shown in the same table. The result of the estimated election is shown in Table 1(d). Candidate c remains the winner with a LRM of 22. □

Computing a Manipulation We consider two methods of computing a set of manipulation rules to apply to future scanned ballots, given an estimate of what the eventual election profile could look like, $\hat{\mathcal{B}}$, assuming the adversary makes no further changes. Each method first simulates the outcome of $\hat{\mathcal{B}}$ to determine whether any further manipulation is needed. If the desired candidate wins, no manipulation rules are generated. Otherwise, a set of rules indicating what kind of ballots to look for during the scanning process, and what to replace them with when they are seen, are formed. When a rule is followed by either of our manipulators, that rule is removed from their rule set.

Our first method for generating such rules uses *margin-irv*, as described in Section 2.1, to determine a minimal manipulation $\mathcal{M}$ of the election $\hat{\mathcal{B}}$ so that the candidate selected by the attacker wins. Note that we add the side constraints of Eq (10) where $C_R = |\{r \mid r \in \mathcal{B}_{\mathcal{M}}, r = R\}|$ is the current count of ballots of the form R.

We then translate this minimal manipulation into a set of rules for the attacker to follow. Our second method does not compute a minimal manipulation of $\hat{\mathcal{B}}$, but simply computes the difference between final tallies of the eventual winner w, and the desired winner w', $\Delta_{w,w'}$. The adversary will seek to remove $\lceil \Delta_{w,w'}/2 \rceil$ votes in which w is preferenced first, and replace them with a vote in which w' is preferenced first.

Example 3. Imagine the attacker wants candidate a to win with a last round margin of 20. One such manipulation (certainly not the minimal one) is to change the ballots to sum to the counts in Table 1(e) which results in election shown in Table 1(f).

The manipulation needs to remove one $[c]$ vote and 11 $[c,a]$ votes and add one $[b,c]$ vote and 11 $[a]$ votes, to end with these tallies from the estimated completion $\hat{\mathcal{B}}$. □

MOV-based Manipulator A minimal manipulation $\mathcal{M}$ found by *margin-irv* specifies, for each type of ballot $R \in \mathbf{R}$, the number of ballots of that type that should be *added to* or *removed from* the election profile to achieve a desired elimination sequence π. The manipulation $\mathcal{M}$ simply records for each ranking R of candidates: how many ballots are modified q_R to take on the new ranking R, and how many ballots with ranking R are modified m_R to show a different ranking. Its not possible that both q_R and m_R are non-zero for the same R, otherwise there is a smaller manipulation with the same effect.

In order for such a manipulation to be found in a reasonable time frame, the ILP of Section 2.2 operates over *equivalence classes* of ballot rankings, $\tilde{\mathbf{R}}$, rather than all possible rankings over a set of candidates $\mathcal{C}$, $\mathbf{R}$. Given an elimination sequence to achieve, π, each ranking in $\mathbf{R}$ is reduced to a ballot class in $\tilde{\mathbf{R}}$. The original ranking is reduced by removing all candidates that would be eliminated by the time that ballot could possibly be placed in their tally. All ballots in the same class will move between the tally piles of the same set of candidates, at the same times. For example, consider an election with candidates a, b, c, and d, and a desired elimination sequence $\pi = [a,c,d,b]$. Ballots with rankings $[c,a,b,d]$, $[c,b,a]$, and $[c,a,b]$, are reduced to the equivalent class $[c,b]$.

The minimal manipulation $\mathcal{M}$ found by *margin-irv* defines: a candidate elimination sequence to be achieved, π, in which the desired winner is victorious; a set of ballots D, in equivalence class form, to remove from $\hat{\mathcal{B}}$; and a set of ballots A, in equivalence class form, to add to $\hat{\mathcal{B}}$. For each ballot to add to the profile, there is a ballot to remove – leaving the total number of cast ballots unchanged (i.e., $|A| = |D|$).

Our MOV-based manipulator creates a manipulation rule for every ballot in D. For the i^{th} ballot in D, d_i, a rule of the form:

$$reduce_{\pi}(b) = d_i \rightarrow a_i$$

is formed, stating that if the manipulator sees a ballot b with a ranking that could be reduced to the equivalence class d_i (assuming the eventual elimination sequence will be π), this ballot should be replaced with the i^{th} ballot in A, a_i.

Example 4. The elimination order desired by the attacker is $\pi = [c,b,a]$. The equivalence classes of the seen ballot types are $[a]$, $[c,a]$, $[b]$ and $[c]$ respectively. The manipulation in terms of these equivalence classes is $+11[a]$, $-11[c,a]$, $+1[b]$ and $-1[c]$. We end up with 12 rules: 11 copies of $[c,a] \rightarrow [a]$ and 1 copy of $[c] \rightarrow [b]$.

If we perform the manipulation on the *actual* remaining ballots we will find enough ballots to change resulting in a final manipulated count $B_{\mathcal{M}}$ of $[a] : 66$, $[c,a] : 19$, $[b,c] : 37$ and $[c] : 14$. The election will first eliminate c and then b with a winning with an LRM of 24. Because the initial ballots were less favorable to the attackers candidate than in the full election the manipulation is larger than required. □

First Preference Manipulator Recall that our first preference manipulator seeks to take $\Delta_{w,w'}$ ballots (divided by two and rounded up) in which a candidate w is preferenced first, and replace them with a ballot in which candidate w' is preferenced first. This manipulator creates $\lceil \Delta_{w,w'}/2 \rceil$ rules of the form:

$$[w, \ldots] \rightarrow [w']$$

The pattern on the left hand side of this rule matches all ballots in which candidate w is preferenced first. Such ballots are replaced by the ballot $[w']$ containing a single preference for w'. If the manipulator is seeking to achieve a last round margin of a given size, Δ, it creates $\lceil \Delta_{w,w'}/2 \rceil + \Delta$ rules of the above form. Note that this naive manipulator is able to influence the last round margin of an election, but not the elimination margin of losing candidates in prior rounds. The manipulation is also heuristic – there is no guarantee that it will result in a desired winner, as it does not consider how preferences might flow between candidates. For example, robbing the original winner of some of their primary vote may result in their early elimination, distributing enough votes to cause an alternate candidate $c \neq w'$ to win. The MOV-based manipulator has more control over potential outcomes as it can alter the later preferences on each ballot.

Example 5. The difference in tallies between the actual winner c of the estimated election $\hat{\mathcal{B}}$ and the desired winner a is 44. The first preference manipulator then requires moving 22 + 20 votes from c to a in order to attain a LRM of 20 for a. The manipulation is then 42 copies of $[c, \ldots] \rightarrow [a]$.

When we apply this manipulation to the actual ballots we find there are only 21 $[c, a]$ and 11 $[c]$ votes arriving in the remainder of the election which are all converted to $[a]$ votes. The final manipulated count is $[a] : 87$, $[c, a] : 9$, $[b, c] : 36$, $[c] : 4$. The winner is a with a LRM of 30 over b. These rather gross manipulations may bring the election result into question, since the final tallies are far from the actual tallies. □

4 Results and Conclusions

We take the cast ballot data available for 5 seats of the Australian New South Wales (NSW) 2015 Legislative Assembly election, and simulate the use of our first preference and more intelligent MOV-based manipulators. The goal of each manipulator is to bring about the election of a specific candidate. For each type of manipulator, we simulate its application in each seat over 100 trials. In each trial, the order which the cast ballots arrive at the scanner is randomised. We compute, over the 100 trials: the average number of ballot changes made by the manipulator; the average total change in first preference tallies resulting from the manipulations; the number of simulations in which the manipulator achieved its desired winner, and achieved its desired winner with a last round margin sufficient to avoid an automatic recount; and the average number of manipulation rules generated by the manipulator after the processing of each αE batch of ballots. The latter statistic corresponds to the average number of *intended* manipulations the adversary still expects to need at each stage of processing.

All experiments have been conducted on a machine with an Intel Xeon Platinum 8176 chip (2.1GHz), and 1TB of RAM. We have used $\alpha = 25\%$ across each batch of 100 simulations. Each batch of 100 simulations have been initialised with the same random seed controlling the order in which ballots arrive at the scanner.

We first consider the relative performance of our first preference manipulator, and a MOV-based manipulator that does not attempt to minimise change across first preference tallies in the true and altered election. We then consider the effectiveness of a

Table 2: Performance of the first preference manipulator (aiming to enforce a LRM of at least Δ) and MOV-based manipulator (enforcing an elimination margin of Δ over the last 2 rounds of counting). The MOV-based manipulator is not using first preference discrepancy minimisation. We report the number of simulations (/100) in which the manipulators are successful, the average number of ballot manipulations performed, the average resulting change to first preference counts, and the number of manipulation rules (int. ballot changes) generated after each 25% proportion of ballots has been seen.

	First Preference Manipulator				**MOV-based Manipulator**			
2Δ	**100**	**300**	**500**	**1000**	**100**	**300**	**500**	**1000**
Ballina: MOV of 1,130; LRM of 1,267; 47,865 ballots cast								
Desired winner achieved	**100**	**100**	**100**	**100**	76	98	91	95
Avg ballot changes	3198	3198	3198	3198	**1171**	**1265**	**1328**	**1407**
Avg FP count changes	6397	6397	6397	6397	**2059**	**2283**	**2408**	**2591**
Avg int. ballot changes (1)	4699	4799	4899	5149	1218	1310	1396	1629
(2)	0	0	0	0	670	774	754	824
(3)	0	0	0	0	272	310	291	145
Balmain: MOV of 1,731; LRM of 1,731; 46,952 ballots cast								
Desired winner achieved	**84**	**94**	**98**	**100**	71	**94**	93	97
Avg ballot changes	1860	1935	2004	2203	**1779**	**1865**	**1907**	**2023**
Avg FP count changes	3720	3871	4008	4405	**3215**	**3443**	**3556**	**3875**
Avg int. ballot changes (1)	1747	1847	1947	2197	1776	1891	1968	2223
(2)	66	54	34	0	313	304	287	146
(3)	47	34	22	5	111	87	50	29
Campbelltown: MOV of 3,096; LRM of 3,096; 45,124 ballots cast								
Desired winner achieved	**90**	**98**	**99**	**100**	72	92	93	**100**
Avg ballot changes	3232	3313	3392	3590	**3161**	**3237**	**3294**	**3512**
Avg FP count changes	6464	6627	6784	7181	**6166**	**6319**	**6452**	**6903**
Avg int. ballot changes (1)	3130	3230	3330	3580	3151	3272	3357	3586
(2)	75	63	41	5	615	673	770	1023
(3)	27	20	21	5	130	91	61	12
Heffron: MOV of 5,824; LRM of 5,835; 46,367 ballots cast								
Desired winner achieved	**85**	**96**	**97**	**100**	71	95	99	**100**
Avg ballot changes	5928	6006	6087	6327	**5878**	**5973**	**6068**	**6255**
Avg FP count changes	11856	12013	12175	12656	**11357**	**11450**	**11744**	**12346**
Avg int. ballot changes (1)	5862	5962	6062	6312	5883	5979	6047	6320
(2)	744	844	944	1194	2680	2820	2746	3028
(3)	50	29	10	0	589	566	601	747
Lismore: MOV of 209; LRM of 1,173; 47,208 ballots cast								
Desired winner achieved	**100**	**99**	**98**	**100**	81	91	90	95
Avg ballot changes	4651	4696	4727	4742	**339**	**396**	**434**	**510**
Avg FP count changes	9304	9392	9455	9486	**601**	**716**	**796**	**974**
Avg int. ballot changes (1)	4371	4475	4572	4814	294	242	514	730
(2)	192	195	151	110	150	150	150	117
(3)	89	44	44	0	52	52	51	23

MOV-based manipulation that attempts to minimise such discrepancies. The computational requirements of each of our manipulators varies. The first preference manipulator is able to compute a set of manipulation rules in less than a second, the MOV-based manipulators (without first preference tally change minimisation) several seconds, while minimising first preference tally changes extends rule generation time by up to a minute.

Table 2 reports the performance of our first preference and MOV-based manipulators on the IRV elections held across our 5 case study seats: Ballina; Balmain; Campbelltown; Heffron; and Lismore. We report the true MOV, LRM, and number of ballots cast in each election alongside the effectiveness of each manipulator across 100 simulated trials. We have found that the MOV-based manipulator is more effective in achieving a change in winner when it seeks to enforce a reasonably sized elimination margin (EM) on the last two rounds of counting, rather than a margin on just the last round. In all our reported results, the MOV-based manipulators enforce an EM of Δ between the runner-up and winner, and the runner-up and their runner-up. The two settings – enforcing an LRM vs an EM on the last two rounds – are similarly effective in certain seats (Campbelltown, Heffron, and Lismore), with the latter more effective in Ballina. Note that the detailed results of this comparison have been omitted for brevity.

Ballina is a seat where the identity of the candidate coming in third significantly influenced which of the remaining two candidates won. The manipulator needed the Greens candidate to place third so that their preferences flowed to the manipulator's desired candidate from the Country Labor Party. Simply enforcing a certain LRM in this instance resulted in manipulated elections in which the tallies of the two runner-ups were similar when determining third place. When forming a manipulation, there is no guarantee that all generated manipulation rules will be applied, and no guarantee that these rules will bring about the desired change in winner. The rules are computed based on hypothetical completions of partially known election profiles. Forming manipulation rules designed to ensure the Greens candidate was eliminated in this third-last position, with a reasonably sized elimination margin, more reliably achieved the desired result.

Table 2 shows that in general, our MOV-based manipulator requires less ballot changes on average to reliably (more than 90% of the time) bring about a desired change in winner. The first preference manipulator is often more successful, as more of its manipulation rules are likely to be applied in practice. A rule that is looking for a ballot with a certain candidate ranked first is more likely to be applied than one that looking for a ballot with a specific ranking of candidates. While a manipulation based on first preferences may, in some circumstances, underestimate the number of ballot changes required to alter an election outcome, it generally overestimates the degree of manipulation required. By focusing on preference flow in Lismore, the MOV-based manipulator reliably achieves a desired outcome by changing orders of magnitude less ballots, on average. In Balmain and Campbelltown, the first preference manipulator generates a slightly smaller 'intended manipulation', forming less manipulation rules, on average, after seeing the first 25% batch of ballots. The MOV-based manipulator applies fewer of its generated rules, in these two contests, leading to fewer ballot changes on average.

It may be the case that ballots are manually examined to compute first preference tallies for each candidate, while the full count is performed by a computer. In this setting, a large discrepancy between the first preference tallies reported by the software, and that of the manual count, is likely to arouse suspicion. Table 3 compares the average discrepancy in first preference counts (between the true, unmanipulated elections, and those altered by our manipulators) when using a MOV-based manipulator that *does not* focus on minimising these discrepancies, and one that *does*. Note that if a number of ballots N is shifted from the first preference tally of one candidate to another, this

Table 3: Number of trials (/100) in which two MOV-based manipulators achieve a desired winner change. The first does not minimise first preference count discrepancies, while the second does. We report the number of successful manipulations in which the resulting election avoided an automatic recount (LRM $\geq$ 100/2 = 50 votes).

	MOV-based Manipulator No FP change minimisation				**MOV-based Manipulator Minimise FP count changes**			
2Δ	**100**	**300**	**500**	**1000**	**100**	**300**	**500**	**1000**
Ballina: MOV of 1,130; LRM of 1,267; 47,865 ballots cast								
Desired winner achieved	**76**	98	91	**95**	72	**99**	**100**	88
Avoid auto recount	**76**	98	91	**95**	72	**99**	**100**	88
Avg ballot changes	**1171**	**1265**	**1328**	**1407**	1185	1296	1463	1735
Avg FP count changes	2059	2283	2408	2591	**230**	**324**	**512**	**714**
Avg int. ballot changes (1)	1218	1310	1396	1629	1216	1368	1457	2042
(2)	670	774	754	824	789	870	998	1235
(3)	272	310	291	145	398	434	502	403
Balmain: MOV of 1,731; LRM of 1,731; 46,952 ballots cast								
Desired winner achieved	**71**	**94**	93	**97**	50	93	**100**	92
Avoid auto recount	**48**	**82**	83	**94**	26	75	**100**	87
Avg ballot changes	**1779**	**1865**	**1907**	**2023**	1784	1958	2147	2502
Avg FP count changes	3216	3443	3556	3875	**849**	**970**	**1079**	**1136**
Avg int. ballot changes (1)	1776	1891	1968	2223	1783	1954	2130	2585
(2)	313	304	287	146	1173	1241	1316	1429
(3)	112	87	50	29	577	612	637	545
Campbelltown: MOV of 3,096; LRM of 3,096; 45,124 ballots cast								
Desired winner achieved	**72**	92	93	**100**	59	**93**	**100**	95
Avoid auto recount	**57**	**85**	89	**100**	31	79	**100**	92
Avg ballot changes	**3161**	**3237**	**3294**	**3512**	4034	4115	4245	4366
Avg FP count changes	6166	6319	6452	6903	**3051**	**3122**	**3329**	**3654**
Avg int. ballot changes (1)	3151	3272	3357	3586	4213	4325	4380	4661
(2)	615	673	770	1023	2134	2108	2173	2116
(3)	130	91	61	12	875	901	906	758
Heffron: MOV of 5,824; LRM of 5,835; 46,367 ballots cast								
Desired winner achieved	**71**	**95**	99	**100**	45	93	**100**	**100**
Avoid auto recount	**49**	**89**	96	**100**	31	83	**100**	**100**
Avg ballot changes	**5878**	**5973**	**6068**	**6255**	6595	6719	6897	7651
Avg FP count changes	11357	11450	11744	12346	**10607**	**10910**	**11414**	**12029**
Avg int. ballot changes (1)	5883	5979	6047	6320	6635	6748	6904	7611
(2)	2680	2820	2746	3028	2976	3164	3291	3392
(3)	589	566	601	747	1402	1466	1568	1697
Lismore: MOV of 209; LRM of 1,173; 47,208 ballots cast								
Desired winner achieved	**81**	**91**	90	95	72	**91**	**96**	**95**
Avoid auto recount	**77**	83	86	87	67	**91**	**96**	**95**
Avg ballot changes	**339**	**396**	**434**	**510**	**339**	438	521	621
Avg FP count changes	601	716	796	974	**215**	**283**	**368**	**542**
Avg int. ballot changes (1)	294	242	514	730	384	535	719	1140
(2)	150	150	150	117	206	293	354	240
(3)	52	52	51	23	97	86	64	77

is viewed as a discrepancy of $2N$ votes. The discrepancy minimising manipulator was able to reduce change in first preference counts by 2.5 times, on average, between a factor of 1.1 and 9, while requiring only a small increase in total ballot changes. In Ballina, we are able to reliably realise a desired winner change while producing a first preference count discrepancy that is significantly lower than the MOV or LRM of the election. Heffron and Campbelltown, with their large margins of victory, are more challenging to manipulate in a non-obvious manner.

Irrespective of whether first preference count changes are being minimised or not, our MOV-based manipulator can successfully alter the outcomes of elections, while avoiding an automatically triggered recount.

Minimising first preference count changes requires a more subtle manipulation, with the later preferences on ballots being altered more often. The result is that the manipulator must aim to achieve larger elimination margins in the last two eliminations to reliably achieve a desired winner change. The manipulator can be too ambitious however, and try to achieve elimination margins that are not realistically achievable. A limitation of the MOV-based manipulators is that if they cannot find a manipulation of a given hypothetical complete election profile that satisfies all desired side constraints, they give up and fail to generate a set of manipulation rules. A more effective strategy would relax these constraints – with smaller requirements on elimination margins – until the algorithm of Section 2.1 is able to find a manipulation.

Conclusions The experiments show that it is quite feasible for an attacker to manipulate an election to change the winner with high confidence in the scenario we examine. Using MOV-based manipulation and minimising first preference changes the attacker can avoid an automatic recount, and often significantly reduce the number of first preference changes. Hence we can conclude that rigorous risk limiting audits of elections is warranted, since simple counting based approaches to auditing can be defeated.

References

1. Michelle Blom, Peter J. Stuckey, and Vanessa Teague. Computing the margin of victory in preferential parliamentary elections. In *Proceedings of the E-Vote-ID 2018: Third International Joint Conference on Electronic Voting*, 2018.
2. Michelle Blom, Peter J. Stuckey, and Vanessa Teague. Election manipulation 100. In *Proceedings of the Fourth Workshop on Advances in Secure Electronic Voting (Voting'19)*, 2019.
3. Michelle Blom, Vanessa Teague, Peter J. Stuckey, and Ron Tidhar. Efficient computation of exact IRV margins. In *Proceedings of the 22nd European Conference on Artificial Intelligence*, 2016.
4. Vincent Conitzer, Toby Walsh, and Lirong Xia. Dominating manipulations in voting with partial information. In *AAAI Conference on Artificial Intelligence*, 2011.
5. T.R. Magrino, R.L. Rivest, E. Shen, and D.A. Wagner. Computing the margin of victory in IRV elections. In *USENIX Accurate Electronic Voting Technology Workshop: Workshop on Trustworthy Elections*, USENIX Association Berkeley, CA, USA, 2011.
6. R. Richie. Instant Runoff Voting: What Mexico (and Others) Could Learn. *Election Law Journal*, 3:501–512, 2004.
7. Lirong Xia. Computing the margin of victory for various voting rules. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, EC '12, pages 982–999, New York, NY, USA, 2012. ACM.