ECAI 2016
G.A. Kaminka et al. (Eds.)

doi:10.3233/978-1-61499-672-9-480

Efficient Computation of Exact IRV Margins

Michelle Blom and **Vanessa Teague** and **Peter J. Stuckey** and **Ron Tidhar** [1]

Abstract. Computing the margin of victory (MOV) in an Instant Runoff Voting (IRV) election is NP-hard. In an IRV election with winning candidate w, the MOV defines the smallest number of cast votes that, if modified, result in the election of a candidate *other than* w. The ability to compute such margins has significant value. Arguments over the correctness of an election outcome usually rely on the size of the electoral margin. Risk-limiting audits use the size of this margin to determine how much post-election auditing is required. We present an efficient branch-and-bound algorithm for computing exact margins that substantially improves on the current best-known approach. Although exponential in the worst case, our algorithm runs efficiently in practice, computing margins in instances that could not be solved by the current state-of-the-art in a reasonable time frame.

1 Introduction

Instant Runoff Voting (IRV) is a system of preferential voting in which voters rank candidates in order of preference. IRV is used for all parliamentary lower house elections in Australia, parliamentary elections in Fiji and Papua New Guinea, presidential elections in Ireland and Bosnia/Herzogovinia, and local elections in numerous locations world-wide, including the UK and United States [19]. Given candidates c_1, c_2, c_3, and c_4, each vote in an IRV election is a (*possibly partial*) ranking of these candidates. A vote with the ranking $[c_1, c_2, c_3]$ expresses a first preference for candidate c_1, a second preference for c_2, and a third for c_3. The tallying of votes proceeds by distributing each vote to its first ranked candidate. The candidate with the smallest number of votes is eliminated, with their votes redistributed to subsequent, less preferred candidates. Elimination proceeds in this fashion, until a single candidate w remains, who is declared the winner. The *margin of victory* (MOV) of the election is the smallest number of cast votes that must be modified (their ranking replaced with an alternate ranking) to ensure that a candidate *other than* w is the last candidate standing and is elected.

Exact computation of IRV electoral margins is NP-hard [22]. It is difficult to compute either the true runner-up of an IRV election, or the margin by which they lost. Disputing an election outcome, or proving that it is correct, generally requires some argument comparing the electoral margin to the precision of the process. For example, risk-limiting audits require knowledge of the MOV to determine how much auditing is required [15]. A close election, in which the MOV is small, requires more auditing than one with a large margin. As more jurisdictions move toward electronic voting or post-election digitisation of votes, software for election analysis, including MOV computation, will become increasingly important.

Automatic recounting of ballots, for example, is triggered in many jurisdictions if the last round margin (the difference between the tallies of the last two remaining candidates, divided by two and rounded up) of an IRV election falls below a threshold. The 2013 federal election for the Australian seat of Fairfax, in Queensland, for example, had a last round margin of just 4 votes, triggering a recount. The actual margin of victory for an IRV election, however, may be much lower than its last round margin. The 2011 federal election for the Australian seat of Balmain, New South Wales, had a last round margin of 1239 votes, with 2477 votes separating the last two remaining candidates – a Liberal and a Green. The actual margin of victory, however, was at most 388 votes, with 775 votes separating the Greens and Labor in a prior round of elimination. Last round margins will therefore trigger recounts in only a portion of eligible IRV elections.

This paper contributes an efficient algorithm, denoted margin, for exact IRV margin computation that substantially improves on the current best-known approach by Magrino et al. [16], denoted MRSW throughout this paper. On a data set of 29 IRV elections held in the United States between 2007 and 2014, MRSW computes margins in several hundred seconds in 26 of the 29 instances, but fails to compute a margin within 72 hours in the remainder. Although exponential in the worst case, our algorithm runs efficiently in practice on all real IRV election instances for which we could obtain data. On all IRV instances in our data set, our algorithm computes exact margins in less than 6 seconds. The *significance* of our improved algorithm is that MOV computation *is now practical* in elections with large numbers of candidates. In the 2007 San Francisco Mayoral election (with 18 candidates) our algorithm computes the MOV is less than 2 seconds while MRSW times out after 72 hours. We compute the MOV in the Minneapolis 2013 Mayoral election (36 candidates) and the Oakland 2014 Mayoral election (17 candidates) in less than 5 seconds, while MRSW times out after 72 hours. In the 2015 Australian New South Wales (NSW) state election (lower house), 93 IRV elections were held to elect a member of parliament in 93 distinct electorates. Our algorithm computes the MOV in each of these elections in less than 0.04 seconds. MRSW requires up to several hundred seconds.

An obvious, but inefficient, algorithm for computing exact IRV margins is to consider every possible order in which candidates could be eliminated, and use a linear program (LP) solver such as CPLEX to compute the exact number of manipulations (vote modifications) necessary to achieve it. A manipulation replaces the ranking of a vote (e.g., $[c_1, c_2, c_3]$) with a different ranking (e.g., $[c_2, c_1]$). The MOV is the smallest number of such modifications required to realise the election of a different candidate. An insight of Magrino et al. [16] is that this LP can be used to compute a *lower bound* on the number of manipulations required to realise an elimination order *ending* in a particular sequence of candidates. The branch-and-bound algorithm of Magrino et al. [16] guides a search of the space of partial orders, using these lower bounds, for a complete elimination order (involving all candidates) requiring fewest vote manipulations to realise.

Our algorithm has the same basic structure as MRSW but introduces a new, easily computed, and often *tighter*, lower bound on the

[1] The University of Melbourne, Australia, email: michelleb@unimelb.edu.au

number of vote manipulations required to realise an elimination order ending in a specific candidate sequence. Computing this lower bound does not require the solving of an LP, and is often higher in value than that computed by MRSW. Combining our new lower bounds with those generated by the LP of MRSW allows us to prune larger portions of the space of possible elimination orders, earlier in search, and disregard many partial orders without the need to solve an LP. This significantly reduces the time required to compute the MOV. Like Magrino et al. [16], we compute margins under the assumption that any manipulation applied to cast votes must leave the number of votes *unchanged*. We further extend the work of Magrino et al. [16] by presenting two variations of our algorithm in which this assumption is not required. This allows us to answer important practical questions. If there were lost votes, could their inclusion have altered the election outcome? If some people voted twice, could it have made a difference to the outcome? The answer to these questions can be obtained by calculating the MOV under the assumption that votes can only be *added* to the election (*addition only*) or removed (*deletion only*). This type of manipulation is known as *voter control*. We consider both settings in this paper.

The problem of computing margins in elections is related to that of *bribery* in the literature (see [11, 10, 12, 4, 5, 13, 6]). In the bribery problem, voters can be bribed in order to change their votes. Much work has analysed the susceptibility of various voting rules (e.g., Condocert-based or plurality voting) to bribery, and the complexity of manipulating an election with bribery. In the shift bribery problem, for example, each voter has prices (bribes) for which they are willing to shift the position of a candidate in their vote forward by i positions [5]. The bribery problem seeks to find a lowest cost set of bribes such that an alternate candidate (to the original winner) is elected. If each voter will change their vote at a price of 1, this lowest cost is equivalent to the electoral MOV. While the complexity of the bribery problem, and election manipulation in general, has been well-studied, the work presented in this paper differs in that it presents a practical and implementable algorithm for *computing* electoral margins in IRV elections, outperforming the current state-of-the-art.

The key contributions of this paper are as follows. We present: a new, efficient method of computing a lower bound on the number of vote manipulations required to realise the elimination of candidates in an IRV election in a specific order; a modification of the MOV-calculation algorithm of Magrino et al. [16] in which this bounding procedure is used; a comparison of our approach with that of Magrino et al. [16] on 29 IRV elections held in the United States between 2007 and 2014 (25 of which appear in the work of Magrino et al. [16]), and 93 IRV elections from the 2015 NSW state election; and two adaptations of our algorithm for settings in which votes can only be *added* or *removed*, but not both.

This paper is structured as follows. Section 2 examines related work in the complexity and computation of IRV margins. Definitions and concepts underlying our approach are presented in Section 3. Section 4 describes our improved algorithm for the computation of margins, highlighting where it deviates from that of Magrino et al. [16]. Section 5 evaluates our algorithm on a suite of IRV instances. Section 6 examines two variations of our algorithm in which votes may be *deleted* from or *added* to an election profile, but not both.

2 Related Work

Computing the exact MOV in an IRV election is NP-hard [22]. Determining whether a single voter can manipulate an IRV election to achieve a desired outcome is also NP-hard [2]. This result has been extended by Conitzer et al. [8, 9] to show that for a weighted variant of IRV in which there are more than 3 candidates (and votes are weighted), finding a manipulation for which a specific candidate is elected (constructive manipulation) is NP-complete. For elections of more than 4 candidates, finding a destructive manipulation (ensuring that a specific candidate is not elected) is also NP-complete. The complexity of manipulating plurality and Condorcet-based elections by adding or deleting voters (equivalent to the addition or deletion of votes considered in this paper) is examined by Bartholdi III et al. [3]. The complexity of strategic voting in schemes for which votes can be *partial* rankings over candidates is investigated by Narodytska and Walsh [17]. See Rothe and Schend [20] for a review of such complexity results. In some of these works, IRV is referred to as a form of Single Transferable Vote (STV) with a single winner.

In this paper we seek to determine the smallest number of votes cast in an IRV election that, if modified, will result in a different outcome (a different winner). Similar questions have been considered for alternate voting rules. The complexity of manipulating an election with bribery is considered by Faliszewski et al. [12], under a number of voting schemes: Condocert-based; approval voting; scoring rules; veto rules; and plurality. Their aim is to find a manipulation to achieve a desired election result, while minimising the cost of bribes given to voters for changing their vote.

An algorithm for computing upper and lower bounds on the IRV MOV has been developed by Cary [7]. The "Winner Elimination" upper bound, for example, finds the most efficient way to eliminate the winner in each round, returning the least-cost (involving fewest vote changes) of these. Bounds on the MOV for IRV and other voting schemes have also been provided by Sarwate et al. [21]. A lower bound is computed by picking sets of candidates to eliminate in order to maximise the difference between the number of votes allocated to the candidates in these sets, and to the remaining candidate with the fewest votes. The bounds defined by Cary [7] and Sarwate et al. [21] can be computed in polynomial time, but are not necessarily tight (i.e., they may differ significantly from the true margin).

In 31 IRV elections conducted in the United States and Ireland, Sarwate et al. [21] compare their computed bounds to known exact margins. Lower bounds equaled exact margins in 18 elections, and in the others fell below exact margins by 0.6% to 19% of the total votes cast. Computed upper bounds were typically within a few votes of exact margins, with a number of exceptions. For the 2009 Aspen City Council election, the lower and upper bound of Sarwate et al. [21] differ from the exact margin by 2.5% (62 votes) and 9.9% (254 votes) of the total number of votes cast. Our algorithm finds the exact MOV in this election within 1.5 seconds. In the 2008 race for Pierce County assessor, their lower and upper bound differ from the exact margin by 0.6% (1945 votes) and 1.6% (5079 votes) of the total number of votes. Our algorithm computes the MOV within 0.02 seconds.

Magrino et al. [16] present a branch-and-bound algorithm for computing exact IRV margins. Applied to 25 IRV elections in the United States, this approach successfully computes exact margins in all but one instance. This algorithm, referred to as MRSW in this paper, considers the space of possible alternate elimination orders of a set of candidates $\mathcal{C}$, in which the actual winner $c_w \in \mathcal{C}$ is *not* the last remaining candidate. Given one such order, a linear program (LP) computes the smallest number of votes (of those cast) that must be modified in order to realise this elimination order. When applied to a partial sequence of candidates, π', this LP computes the smallest number of vote changes required to achieve this order of elimination in a *reduced election profile*, in which all candidates not in π' have been eliminated (and their votes redistributed). It is clear that this

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

Figure 1. An informal definition of the IRV counting algorithm.

Ranking	Count
$[c_2, c_3]$	4
$[c_1]$	20
$[c_3, c_4]$	9
$[c_2, c_3, c_4]$	6
$[c_4, c_1, c_2]$	15
$[c_1, c_3]$	6

(a)

Candidate	Rnd1	Rnd2	Rnd3
c_1	26	26	26
c_2	10	10	—
c_3	9	—	—
c_4	15	24	30

(b)

Table 1. An example IRV election profile, stating (a) the number of votes cast with each listed ranking over candidates c_1, c_2, c_3, and c_4, and (b) the tallies after each round of vote counting and elimination.

number is a lower bound on the number of vote changes required to achieve any *complete* elimination order (involving all candidates) ending in π'. The MRSW algorithm builds a tree of partial elimination orders, with the smallest LP evaluation obtained for each visited leaf (a complete order) providing an upper bound on the electoral margin. Partial orders whose associated lower bound is *larger* than the best known upper bound are pruned from the search.

The main restricting cost of MRSW is the number of nodes that are explored and evaluated via the LP. Our algorithm dramatically reduces the number of partial elimination orders (nodes) explored, relative to MRSW, through the use of a bounding rule assigning tighter (higher) lower bounds to nodes close to the root of the tree. We are thus able to prune larger portions of the search space, earlier.

3 Preliminaries

The tallying of votes in an IRV election proceeds by a series of rounds in which the candidate with the lowest number of votes is eliminated (see Figure 1) 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.

Let $\mathcal{C}$ be the set of candidates in an IRV election $\mathcal{B}$. We refer to sequences of candidates π in list notation (e.g., $\pi = [c_1, c_2, c_3, c_4]$), and use such sequences to represent both votes and elimination orders. We will often treat a sequence as the set of elements it contains. 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 preferred). Let $\mathit{first}(\pi)$ denote the first candidate appearing in sequence π (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.

Definition 1 Projection $\mathbf{p}_\mathcal{S}(\pi)$ We define the projection of a sequence π onto a set $\mathcal{S}$ as the largest subsequence of π that contains only elements of $\mathcal{S}$. (The elements keep their relative order in π).

For example: $p_{\{c_2,c_3\}}([c_1, c_2, c_4, c_3]) = [c_2, c_3]$ and $p_{\{c_2,c_3,c_4,c_5\}}([c_6, c_4, c_7, c_2, c_1]) = [c_4, c_2]$.

Each candidate $c \in \mathcal{C}$ has a *tally* of votes. Votes are added to this tally upon the elimination of a candidate $c' \in \mathcal{C} \setminus c$, and are redistributed from this tally upon the elimination of c.

Definition 2 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 a 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. Recall that $p_\mathcal{S}(b)$ denotes the sequence of candidates mentioned in b that are also in $\mathcal{S}$.

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

Definition 3 Margin of Victory (MOV) The MOV in an election with candidates $\mathcal{C}$ and winner $c_w \in \mathcal{C}$, is the *smallest* number of votes whose ranking must be modified (by an adversary) so that a candidate $c' \in \mathcal{C} \setminus c_w$ is elected.

If several candidates receive the same number of votes, at any stage of the IRV count, we assume that the adversary can decide which of the candidates is eliminated. This assumption is made by Magrino et al. [16]. If this is not the case, the MOV of Definition 3 slightly underestimates (but never overestimates) the true margin.

Definition 4 Last Round Margin ($LRM_\mathcal{B}$) The last round margin of 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.

$$LRM_\mathcal{B} = \lceil \frac{|t_\mathcal{S}(c) - t_\mathcal{S}(c')|}{2} \rceil \quad (2)$$

Example 1 *Consider the IRV election of Table 1. The tallies of candidates c_1, c_2, c_3, and c_4, in the 1^{st} counting round are 26, 10, 9, and 15 votes. Candidate c_3 is eliminated, and 9 votes are distributed to c_4, who now has a tally of 24. Candidate c_2, on 10 votes, is eliminated next with 6 of their votes distributed to c_4 (the remainder have no subsequent preferences and are exhausted). Candidates c_1 and c_4 remain with tallies of 26 and 30. The last round margin is 2 votes. Candidate c_1 is eliminated from consideration and c_4 elected.*

4 A Fast Algorithm for Calculating Margins

We present a branch-and-bound algorithm for computing the MOV in IRV elections. This algorithm has the same basic structure as that of MRSW [16], being a traversal of the tree of possible orders of candidate elimination. Our algorithm incorporates a substantially improved pruning rule allowing us to dramatically reduce the portion of this tree we must traverse to compute the MOV. In this section, we describe our algorithm in detail and contrast its performance against MRSW on 29 IRV elections held in the United States between 2007 and 2014, and the 2015 NSW lower house election held in Australia. In the latter election, 93 IRV elections were held to elect a member of parliament in 93 electorates.

Magrino et al. [16] define an LP, DISTANCETO, shown below, for computing the minimum number of votes cast in an election $\mathcal{B}$ that, if

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

manipulated, realises a specific complete elimination order π (involving all candidates $\mathcal{C}$). When applied to a partial order π' ($\pi' \subset \mathcal{C}$), DISTANCETO computes a *lower bound* on the number of votes that, if manipulated, will realise an elimination order *ending* in π'. In this paper, we define a *bounding rule* that, when applied to a partial order π', computes an alternative, often tighter (higher), lower bound.

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 in the election with ranking $R \in \mathbf{R}$, and N the total number of votes cast. For each ranking $R \in \mathbf{R}$, we define variables:

q_R number of votes to be changed into R;

m_R 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 LP is:

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

$$N_R + q_R - m_R = y_R \qquad \forall R \in \mathbf{R} \quad (3)$$

$$\sum_{R \in \mathbf{R}} q_R = \sum_{R \in \mathbf{R}} m_R \quad (4)$$

$$\sum_{R \in \mathcal{R}_{i,i}} y_R \leq \sum_{R \in \mathcal{R}_{j,i}} y_R \qquad \forall c_i, c_j \in \pi \ . \ i < j \quad (5)$$

$$n \geq y_R \geq 0, \ \ N_R \geq m_R \geq 0, \ \ q_R \geq 0 \qquad \forall R \in \mathbf{R} \quad (6)$$

Constraint (3) 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 (5) defines a set of *special elimination constraints* which force the candidates in π to be eliminated in the stated order. $\mathcal{R}_{j,i}$ denotes the subset of rankings in $\mathbf{R}$ ($\mathcal{R}_{j,i} \subset \mathbf{R}$) in which c_j is the most preferred candidate still standing (i.e., that will count toward c_j's tally) at the end of round i (in which candidate c_i is eliminated). Constraint (4) ensures that the total number of votes cast in the election does not change as a result of the manipulation.

4.1 Two New Lower-Bounding Rules

Let us consider a partial elimination order $\pi' \subset \mathcal{C}$. Each candidate $e \in \mathcal{C} \setminus \pi'$ must be eliminated before every candidate $c \in \pi'$ (recall that we are computing a lower bound on the number of votes that must be manipulated to realise an elimination order *ending* in π'). We define $\Delta(c, e)$ as the number of votes $b \in \mathcal{B}$ for which c is ranked higher than e, or c appears and e does not. This is equal to the number of votes with rankings $[c, e]$ or $[c]$ when all candidates apart from c and e are removed. At any time e is eliminated before c, c has a tally of *at most* $\Delta(c, e)$ votes at the moment e is eliminated, with all other votes assigned to e, or another candidate. Recall that $p_{\mathcal{S}}(b)$ denotes the *projection* of b onto set $\mathcal{S}$ (i.e., the ranking of vote b with all candidates not in the set $\mathcal{S}$ removed).

$$\Delta(c, e) = \ | \ [b \mid b \in \mathcal{B}, p_{\{c,e\}}(b) \in \{[c, e], [c]\}] \ | \quad (7)$$

The *primary vote* of candidate $c \in \mathcal{C}$, denoted $f(c)$, is the number of votes $b \in \mathcal{B}$ for which c is ranked highest.

$$f(c) = \ | \ [b \mid b \in \mathcal{B}, c = \mathit{first}(b)] \ | \quad (8)$$

To ensure that candidate e is eliminated *before* candidate c, we require that $f(e) \leq \Delta(c, e)$. In other words, we require that the primary vote of e is less than or equal to the number of votes in which c is ranked higher than e, or c appears and e does not. If it is the case that $f(e) > \Delta(c, e)$, we need to change the relative counts by the amount $f(e) - \Delta(c, e)$ for this order of elimination to be feasible. Let $l_1(c, e)$ denote a lower bound on the number of votes that must be modified to achieve the elimination of e before c.

$$l_1(c, e) = \max(0, \lceil \frac{f(e) - \Delta(c, e)}{2} \rceil) \quad (9)$$

Example 2 *Consider the partial elimination order $\pi' = [c_2]$ in the election of Table 1. To realise an elimination order ending in c_2, all other candidates must be eliminated prior to c_2's election. To ensure that c_1 appears before c_2 in the elimination sequence, we count all votes that could possibly be in c_2's tally at the point at which c_1 is eliminated – this is denoted $\Delta(c_2, c_1)$ and defined in Equation 7. In this example, $\Delta(c_2, c_1) = 10$. The smallest number of votes that c_1 could have in their tally upon elimination is their initial tally (or primary vote) $f(c_1)$, defined in Equation 8. Here, $f(c_1) = 26$. For c_1 to appear before c_2 in the elimination sequence, we must change the votes so that, at the very least, the minimum number of votes that c_1 could have (upon elimination) is less than the maximum number of votes c_2 could have. Equation 8 computes this 'minimal number' of required vote changes, $l_1(c_2, c_1)$. Here, $l_1(c_2, c_1) = 8$.*

Since each candidate $e \in \mathcal{C} \setminus \pi'$ has been eliminated prior to each $c \in \pi'$, we can compute a lower bound on the number of votes that must be modified to realise an elimination order ending in π', $b_1(\pi')$, as shown in Equation 10. In contrast to the DISTANCETO LP, our lower bound does not consider the order in which candidates are eliminated in π', but computes a lower bound on the number of votes we must alter to ensure that the candidates in π' *are* the last candidates standing. The DISTANCETO LP, however, operates on a reduced election profile in which all candidates not in π' have been eliminated, and their votes redistributed. It computes the manipulation required to realise π' in this setting.

$$b_1(\pi') = \max\{l_1(c, e) \mid c \in \pi', e \in \mathcal{C} \setminus \pi'\} \quad (10)$$

Example 3 (Example 2 cont.) *For the partial order $\pi' = [c_2]$, we compute lower bounds on the smallest number of vote changes required to eliminate each candidate c_i ($i \neq 2$) prior to c_2's election, $l_1(c_2, c_i)$. The largest $l_1(c_2, c_i)$ becomes our lower bound, $b_1(\pi')$, on the number of votes we must change to realise an elimination order ending in $[c_2]$. In this example, $b_1(\pi') = 8$.* DISTANCETO *would assign a lower bound of 0 to π' as in a reduced election involving only c_2, no votes need be changed to ensure they are elected.*

The bound b_1 can be tightened. Consider the partial elimination order π', for which all candidates $e \in \mathcal{C} \setminus \pi'$ are eliminated before all $c \in \pi'$. We know that e has at least $f(e)$ votes in its tally. Candidate c may not have, in their tally, all votes which have been counted toward $\Delta(c, e)$ (those in which c appears before e, or c appears, but e does not). Some of these votes may lie in the tallies of other candidates in π', who have not yet been eliminated. We define $\Delta_{\mathcal{S}}(c, e)$ as the maximum number of votes that c can have in their tally at the time e is eliminated, where $\mathcal{S} = \{e\} \cup \pi'$ denotes the minimal set of candidates that must be 'still standing' at this time.

$$\Delta_{\mathcal{S}}(c, e) = \ | \ [b \mid b \in \mathcal{B}, c = \mathit{first}(p_{\mathcal{S}}(b)] \ | \quad (11)$$

To realise a situation in which candidate $e \in \mathcal{C} \setminus \pi'$ is eliminated prior to candidate $c \in \pi'$, we require that $f(e) \leq \Delta_{\mathcal{S}}(c, e)$. If $f(e) > \Delta_{\mathcal{S}}(c, e)$ then we must modify at least $l_2(c, e, \pi')$ votes.

$$l_2(c, e, \pi') = \max(0, \lceil \frac{f(e) - \Delta_{\mathcal{S}}(c, e)}{2} \rceil) \quad (12)$$

Equation 13 defines a tighter lower bound on the number of votes in $\mathcal{B}$ that must be changed to ensure that $\pi' \subset \mathcal{C}$ are the last remaining candidates, $b_2(\pi')$. Note that $l_2(c, e, \pi') \geq l_1(c, e)$, for all $\pi' \subset \mathcal{C}$, $c \in \pi'$, and $e \in \mathcal{C} \setminus \pi'$. Hence, $b_2(\pi') \geq b_1(\pi')$ for all $\pi' \subset \mathcal{C}$. Both $b_1(\pi')$ and $b_2(\pi')$ are independent of the order of candidates in π'.

$$b_2(\pi') = \max\{l_2(c, e, \pi') \mid c \in \pi', e \in \mathcal{C} \setminus \pi'\} \quad (13)$$

Example 4 *Consider the partial elimination order $\pi' = [c_3, c_1]$. Our first lower bound on the number of manipulations required to realise an elimination order ending in π', $b_1(\pi')$, equals 0. To compute this we evaluate $l_1(c_i, c_j)$ for $i = 1, 3$, and $j = 2, 4$, and take the maximum result. These values represent the smallest number of manipulations required to eliminate c_2 and c_4 before candidates c_3 and c_1. To compute our second lower bound, $b_2(\pi')$, we evaluate $l_2(c_i, c_j, \pi')$ for $i = 1, 3$, and $j = 2, 4$ (via Equation 12), and take the maximum result. We find that $l_2(c_i, c_j, \pi') = 0$ for all i and j with the exception of $l_2(c_3, c_2, \pi')$ which equals 1. Hence, $b_2(\pi') = 1$.* DISTANCETO *assigns π' a lower bound of 0.*

4.2 A Branch-and-Bound Algorithm: margin

Figure 2 outlines our algorithm, denoted margin, for computing exact margins in IRV elections. This algorithm shares the basic structure as MRSW [16], both being branch-and-bound algorithms.

An initially empty priority queue Q of partial elimination orders is maintained throughout the algorithm (step 1). An upper bound on the number of votes that must be modified to realise a winning candidate *other* than c_w is initialised to the last round margin of the election, $LRM_{\mathcal{B}}$ (step 2). A partial order $\pi' = \{c\}$ for all $c \in \mathcal{C} \setminus c_w$ (i.e., we do not consider orders that will end in the winning candidate) is inserted into Q with a score given by $b_2(\pi')$ of Equation 13 iff that score is lower than the current upper bound (steps 5–7). If this score is larger or equal to the upper bound, π' and all its descendents are pruned (will not be explored). MRSW adds these orders to Q with a score of 0 (as in a reduced election involving only one candidate c, no votes need to be altered to ensure that c wins).

We repeatedly select the partial order π' in Q with the smallest score (i.e., the smallest lower bound on the size of the manipulation required to realise an elimination order ending in π'). This order is removed from Q (step 10), and is expanded. It is at this point that we evaluate π' with the DISTANCETO LP (step 13). We have found, from experimentation, that DISTANCETO can provide a higher bound than b_2, albeit infrequently. If the revised bound from DISTANCETO is larger or equal to the current upper bound, π' is pruned from the tree in step 14–15 (i.e., it is not added to Q). Otherwise, we consider each candidate $c \in \mathcal{C} \setminus \pi'$ and create new order π in which c is eliminated just prior to the first candidate in π' (step 17). If π is a complete order, containing all candidates, we compute the exact number of vote changes required to realise π with DISTANCETO. If this number is lower than the current upper bound, U is replaced with the smaller number (step 19). If π is a partial order, we compute $b_2(\pi)$ as defined in Equation 13 (step 20). If this lower bound is lower than the current upper bound, π is inserted into Q (steps 21–22), otherwise it and its descendents are pruned from the search.

```
margin(C, B, c_w)
1   Q := ∅
2   U := LRM_B
3   for(c ∈ C \ {c_w})
4       π' := [c]
[5]     l := b_2(π')
[6]     if(l < U)
7           Q := Q ∪ {(l, π')}
8   while Q ≠ ∅
9       (l, π') := arg min Q
10      Q := Q \ {(l, π')}
11      U := expand(l, π', U, Q, C, B)
12  return U

expand(l, π', U, Q, C, B)
[13] l' := max{l, DISTANCETO(π', C,B)}
[14] if(l' ≥ U)
[15]    return U
16  for(c ∈ C \ π')
17      π := [c] ++π'
[18]    if(|π| = |C|)
[19]        return min{U,DISTANCETO(π,C, B)}
[20]    l'' = max{l', b_2(π)}
21      if(l'' < U)
22          Q := Q ∪ {(l'', π)}
23  return U
```

Figure 2. MOV computation for an IRV election $\mathcal{B}$ with candidates $\mathcal{C}$ and winner $c_w \in \mathcal{C}$; [#] denotes where our algorithm differs from MRSW [16].

The b_2 bound is not guaranteed to generate a tighter bound than DISTANCETO (although in practice we find that it *is* tighter in a majority of instances). In using our lower bounding rules to *select* partial orders for expansion, and evaluating DISTANCETO only on these selected orders, we reduce the number of LPs solved by our algorithm. MRSW evaluates the DISTANCETO LP for each child formed upon the expansion of a node.

When all elimination orders have been examined, or pruned, margin terminates (step 12), returning U, which now equals the smallest number of vote changes required to alter the outcome of election $\mathcal{B}$.

4.3 Comparing MRSW and margin: An Example

Consider the IRV election of Table 1. Figure 3a records the partial elimination orders considered by MRSW when computing the MOV. Each node denotes an elimination order that is traversed and evaluated by MRSW, with its score recorded. MRSW first considers the partial orders $[c_2]$, $[c_3]$, and $[c_4]$, assigning each a score of 0. The upper bound on the manipulation size required to change the election outcome is set to 2 votes (the last round margin). MRSW considers the children of node $[c_3]$ – $[c_2, c_3]$ with a score of 5, $[c_1, c_3]$ with a score of 11, and $[c_4, c_3]$ with a score of 0. These scores are obtained by solving DISTANCETO. Nodes $[c_2, c_3]$ and $[c_1, c_3]$ can be pruned as their scores are higher than the current upper bound. Node $[c_4, c_3]$ is expanded, creating children $[c_2, c_4, c_3]$ with a score of 0 and $[c_1, c_4, c_3]$ with a score of 6 (consequently pruned). The leaf node $[c_1, c_2, c_4, c_3]$ is then visited and assigned a score of 11 (also pruned). MRSW continues to expand nodes in this manner as shown

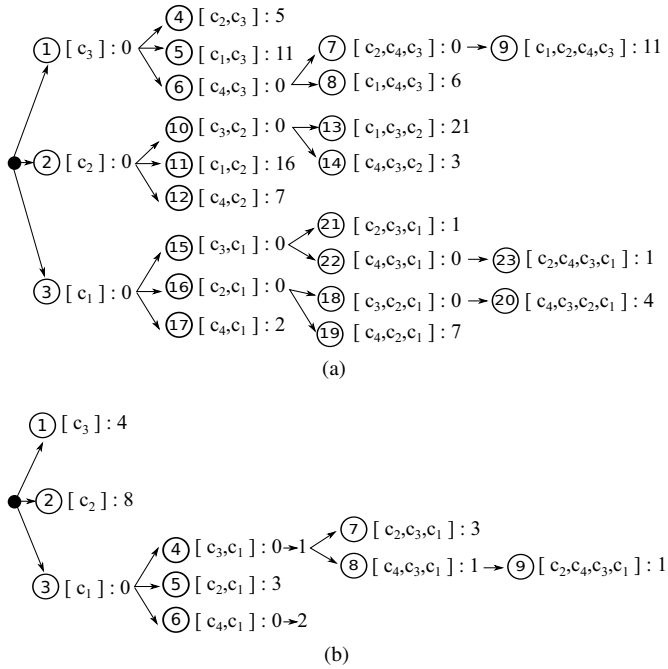

Figure 3. Traversal of elimination orders by (a) MRSW, and (b) margin, recording the sequence in which orders are evaluated (circled), and the score given to each order, for the election of Table 1.

in Figure 3a, visiting 23 nodes and solving 20 LPs (the children of the root node have a value of 0 as in a reduced election with one candidate, no votes need to be changed to ensure they are elected).

Our algorithm visits and evaluates the nodes shown in Figure 3b, reporting beside each node the score we assign to it. Nodes $[c_3]$, $[c_2]$, and $[c_1]$, are assigned scores of 4, 8, and 0, respectively. Nodes $[c_3]$ and $[c_2]$ are immediately pruned as their scores are larger than the current upper bound of 2. This allows us to concentrate on elimination orders ending in c_1. Nodes $[c_3, c_1]$, $[c_2, c_1]$, and $[c_4, c_1]$, are assigned scores of 0, 3, and 0 (when $[c_3, c_1]$ and $[c_4, c_1]$ are selected for expansion, DISTANCETO is solved for these nodes and their scores revised to 1 and 2, respectively). Node $[c_3, c_1]$ is expanded, visiting nodes $[c_2, c_3, c_1]$ and $[c_4, c_3, c_1]$ with scores of 3 and 1. The leaf $[c_2, c_4, c_3, c_1]$ is given a score of 1 by solving DISTANCETO. The upper bound is updated to 1. DISTANCETO is then solved for node $[c_4, c_1]$, assigned a score of 0 by our bounding rules, obtaining a revised lower bound of 2. As this is greater than the current upper bound, $[c_4, c_1]$ can be pruned immediately. Our algorithm assigns scores to 9 nodes, but solves only 4 LPs in the process – DISTANCETO is solved at a node only when it has been selected for expansion, or if it is a leaf node. Scores assigned by our lower bounding rules are used to determine *which* nodes to select for expansion.

5 Computational Results

We have evaluated our improved algorithm (Figure 2) on 29 IRV elections held in the United States, and 93 IRV elections involved in the 2015 NSW state election (lower house). We contrast the performance of our approach on this data set with that of MRSW. Execution was performed on a machine with four 2.10 GHz CPUs, 7.7 GB of memory, and with a 72 hour timeout. CPLEX 12.5.1 was used to solve all LPs. Table 2 reports, for each election considered, the number of candidates and votes cast, the number of calls to DISTANCETO made by MRSW and by our algorithm (denoted margin), the computation time (in milliseconds) of the two algorithms, the MOV and the last-round margin. Of the 93 IRV elections in the 2015 NSW state election, 4 reported a MOV that differed from its last round margin.

Our algorithm substantially reduces both the number of calls to DISTANCETO and computation time. For example, we are able to compute the MOV of the 2007 San Francisco Mayoral election, where MRSW timed out after 72 hours. In generating these results, our algorithm uses the tighter b_2 pruning rule of Equation 13. In the majority of instances considered, pruning with b_2 was either faster, or as fast, as pruning with b_1. The b_2 rule is more costly to compute, however, than b_1. In the 2007 San Francisco Mayoral election, for example, 1300 ms are used to compute the margin when pruning with b_2 (solving 94 LPs). In contrast, 1139 ms are used when pruning with b_1, even though 970 LPs are solved in the process. For the Aspen 2009 City Council race, however, 1241 ms are used when pruning with b_1 and 1039 ms when pruning with b_2. For the Pierce 2008 County Assessor instance, 17 ms and 9 ms are used when pruning with b_1 and b_2, respectively. The full table of results comparing the performance of b_1 and b_2 has been omitted for brevity.

In the 93 IRV elections of the 2015 NSW state election, MRSW computed the MOV in 14 to 354,007 ms, solving 21 to 23,768 LPs. Our margin algorithm computed margins in 1 to 35 ms, solving 1 to 13 LPs. Table 2 reports the results of 16 of these IRV elections – in 4 of which (Ballina, Maitland, Lismore, and Willoughby) the MOV differs from the last round margin. The number of candidates in each election range from 5 to 8. MRSW, in general, requires more time to compute margins in elections with more candidates.

We have additionally applied margin to all instances of the PrefLib data set[3] that can be interpreted as an election (261 instances). The number of candidates and votes cast in these instances range from 3 to 2819, and 4 to 15,101, respectively. Margin computation in all instances with more than 500 candidates is trivial, with only 4 votes cast. With a 30 minute time limit, margin computes margins in all but 10 instances, while MRSW fails in 30 instances. In the 10 instances for which margin fails to compute a margin in 30 minutes, it finds lower and upper bounds on the margin that differ by up to 9 votes in 7/10 instances and by 22 to 789 votes in the remainder.

6 Variations: Voter Control

Suppose some votes are lost during an election. We extend margin to determine the minimum number of votes that must be *added* to change an election outcome as follows. We first remove the division by two when calculating the last round margin (Definition 4), and l_1, b_1, l_2, and b_2 of Equations 9–13 (in this setting, manipulations can only add votes). We then modify the DISTANCETO LP to calculate the minimum number of vote additions required to enforce a certain elimination order. To do so, we interpret variable q_R as the number of votes with ranking $R \in \mathbf{R}$ added to the election. We set $m_R = 0$ for $R \in \mathbf{R}$, and remove Constraint (4) which forces the number of votes to remain constant. If the computed MOV is larger than the number of lost votes, then their inclusion could not have altered the election outcome. In the 2013 election of candidates to six seats in Western Australia's Senate a discrepancy of 1,375 initially verified votes was discovered during a recount (resulting from a lost ballot box) [18]. The election result was overturned, and a repeat election held in 2014. While Single Transferable Vote (STV) is used in Australian Senate elections – a more complex scheme than IRV – this case demonstrates the impact of such mistakes when they occur.

[3] http://www.preflib.org/

$\|\mathcal{C}\|$	# Votes Cast	MRSW LPs	margin LPs	MRSW Time (ms)	margin Time (ms)	MOV	LRM	Election Name
2	45,986	1	0	1	1	15,356	15,356	Berkeley 2010 Auditor
2	15,243	1	0	1	1	4,830	4,830	Oakland 2010 D2 School Board
2	14,040	1	0	1	1	4,826	4,826	Oakland 2010 D6 School Board
2	23,494	1	0	1	1	8,338	8,338	San Leandro 2010 D3 City Council
3	122,268	6	1	1	1	17,081	17,081	Oakland 2010 Auditor
3	15,243	6	1	1	1	2,175	2,175	Oakland 2010 D2 City Council
3	23,494	6	1	1	1	742	742	San Leandro 2010 D5 City Council
4	4,862	22	1	4	1	364	364	Berkeley 2010 D7 City Council
4	5,333	23	2	4	1	878	878	Berkeley 2010 D8 City Council
4	14,040	24	2	4	1	2,603	2,603	Oakland 2010 D6 City Council
4	43,661	19	1	3	1	2,007	2,007	Pierce 2008 City Council
4	159,987	19	1	3	1	8,396	8,396	Pierce 2008 County Auditor
5	312,771	49	4	9	1	2,027	2,027	Pierce 2008 County Executive
5	2,544	65	1	12	1	89	89	Aspen 2009 Mayor
5	6,426	85	1	17	1	1,174	1,174	Berkeley 2010 D1 City Council
5	5,708	64	1	12	1	517	517	Berkeley 2010 D4 City Council
5	13,482	49	1	9	1	486	486	Oakland 2012 D5 City Council
5	28,703	65	2	13	1	2,332	2,332	San Leandro 2012 D4 City Council
7	23,494	292	1	81	1	116	116	San Leandro 2010 Mayor
7	312,771	312	19	98	9	**1,111**	3,650	Pierce 2008 County Assessor
7	26,761	351	19	111	8	**386**	684	Oakland 2012 D3 City Council
8	23,884	4,989	2	3,905	2	2,329	2,329	Oakland 2010 D4 City Council
8	57,492	7,737	2	6,772	2	8,522	8,522	Berkeley 2012 Mayor
8	34,180	1,301	2	666	2	423	423	Oakland 2012 D1 City Council
11	122,268	26,195	4	90,988	18	1,013	1,013	Oakland 2010 Mayor
11	2,544	15,109	224	64,705	1,039	**35**	162	Aspen 2009 City Council
17	101,431	—	234	timeout	5,067	10,201	10,201	Oakland 2014 Mayor
18	149,465	—	94	timeout	1,300	50,837	50,837	San Francisco 2007 Mayor
36	79,415	—	2	timeout	1,173	6,949	6,949	Minneapolis 2013 Mayor
5	44797	130	1	37	2	8235	8235	Seat of Lakemba, NSW 2015 lower house
5	45467	1,071	1	2,326	2	8,495	8495	Seat of Liverpool, NSW 2015 lower house
5	47348	130	1	36	2	10,806	10,806	Seat of Manly, NSW 2015 lower house
6	47,933	222	13	161	12	**4,012**	5,446	Seat of Maitland, NSW 2015 lower house
6	47,208	173	8	107	9	**209**	1,173	Seat of Lismore, NSW 2015 lower house
6	47,370	655	12	502	11	**10,160**	10,247	Seat of Willoughby, NSW 2015 lower house
7	47,865	380	11	652	35	**1,130**	1,267	Seat of Ballina, NSW 2015 lower house
7	48,358	867	1	1,693	9	3,132	3,132	Seat of Newcastle, NSW 2015 lower house
7	45497	710	1	1,323	9	3,536	3,536	Seat of Newtown, NSW 2015 lower house
7	48065	1071	1	2,326	10	4253	4253	Seat of Lake Macquarie, NSW 2015 lower house
8	47,590	7,091	1	79,637	22	4,069	4,069	Seat of Clarence, NSW 2015 lower house
8	47,803	21,054	1	190,066	18	7,311	7,311	Seat of Hawkesbury, NSW 2015 lower house
8	48,571	9,923	1	100,143	21	4,974	4,974	Seat of Swansea, NSW 2015 lower house
8	46,756	23,768	2	354,007	34	8,574	8,574	Seat of Murray, NSW 2015 lower house
8	48,002	4,106	1	29,211	20	2,576	2,576	Seat of Penrith, NSW 2015 lower house
8	42,892	2,369	1	11,243	18	2,864	2,864	Seat of Sydney, NSW 2015 lower house

Table 2. Running times and margins computed for 29 IRV elections in the United States, and 16/93 IRV elections held in the 2015 NSW state election (lower house), using MRSW and margin. Cases where the margin of victory (MOV) differs from the last round margin (LRM) are in bold

In Australian state and federal elections, each polling station has a book containing the names and addresses of all voters in the region. As each voter casts their vote, their name is struck off by hand. This does not prevent a voter from voting more than once at multiple polling stations. In the 2013 Australian federal election, the Australian Electoral Commission (AEC) 'investigated almost 19,000 instances of multiple voting' [1]. In this situation we know the number of invalid votes, but not *which* votes are invalid. If this total exceeds the minimum number of votes that, if removed, change the result of the election, we know these invalid votes may have influenced the outcome. To determine this number, we extend margin as follows. The division by two in our definition of last round margin, and bounding rules, is removed. Variable m_R becomes the number of cast votes with ranking $R \in \mathbf{R}$ that we will delete. We set $q_R = 0$ for $R \in \mathbf{R}$, and remove Constraint (4). We replace q_R with m_R in the DISTANCETO objective, as we seek to minimise the number of deleted votes required to realise an alternate outcome.

Our margin algorithm, when applied to our suite of IRV instances, is able to find the MOV in both the *addition-* and *deletion-only* settings, with runtimes similar to those in Table 2. Table 3 reports, for each election in Table 2, the number of candidates and votes cast, the number of calls to DISTANCETO made by MRSW and margin, in the *addition-only* setting, together with the computation time (in milliseconds) of the two algorithms, the MOV and the last-round margin.

$\|\mathcal{C}\|$	# Votes Cast	MRSW LPs	margin LPs	MRSW Time (ms)	margin Time (ms)	MOV	LRM	Election Name
2	45,986	1	0	1	1	30,711	30,711	Berkeley 2010 Auditor
2	15,243	1	0	1	1	9,660	9,660	Oakland 2010 D2 School Board
2	14,040	1	0	1	1	9,651	9,651	Oakland 2010 D6 School Board
2	23,494	1	0	1	1	16,675	16,675	San Leandro 2010 D3 City Council
3	122,268	6	1	1	1	34,162	34,162	Oakland 2010 Auditor
3	15,243	6	1	1	1	4,349	4,349	Oakland 2010 D2 City Council
3	23,494	6	1	1	1	1,484	1,484	San Leandro 2010 D5 City Council
4	4,862	22	1	2	1	728	728	Berkeley 2010 D7 City Council
4	5,333	24	2	2	1	1,756	1,756	Berkeley 2010 D8 City Council
4	14,040	24	2	2	1	5,205	5,205	Oakland 2010 D6 City Council
4	43,661	19	1	2	1	4,014	4,014	Pierce 2008 City Council
4	159,987	19	1	2	2	16,792	16,792	Pierce 2008 County Auditor
5	312,771	49	4	4	1	4,054	4,054	Pierce 2008 County Executive
5	2,544	65	1	6	1	177	177	Aspen 2009 Mayor
5	6,426	79	1	7	1	2,348	2,348	Berkeley 2010 D1 City Council
5	5,708	62	1	5	1	1,033	1,033	Berkeley 2010 D4 City Council
5	13,482	49	1	4	1	972	972	Oakland 2012 D5 City Council
5	28,703	62	2	6	1	4,664	4,664	San Leandro 2012 D4 City Council
7	23,494	292	1	35	1	232	232	San Leandro 2010 Mayor
7	312,771	312	19	53	7	**2,221**	7,299	Pierce 2008 County Assessor
7	26,761	351	19	70	6	**771**	1367	Oakland 2012 D3 City Council
8	23,884	3,801	2	1,714	2	4,657	4,657	Oakland 2010 D4 City Council
8	57,492	5,693	2	2,465	2	17,044	17,044	Berkeley 2012 Mayor
8	34,180	1,186	2	315	2	845	845	Oakland 2012 D1 City Council
11	122,268	23,541	4	45,285	21	2,025	2,025	Oakland 2010 Mayor
11	2,544	13,943	220	50,117	862	**70**	323	Aspen 2009 City Council
17	101,431	—	224	timeout	4,812	20,402	20,402	Oakland 2014 Mayor
18	149,465	—	94	timeout	1,273	101,674	101,674	San Francisco 2007 Mayor
36	79,415	—	2	timeout	1,176	13,898	13,898	Minneapolis 2013 Mayor
5	44797	123	1	36	1	16,470	16,470	Seat of Lakemba, NSW 2015 lower house
5	45467	121	1	35	1	16,989	16,989	Seat of Liverpool, NSW 2015 lower house
5	47348	116	1	30	1	21,612	21,612	Seat of Manly, NSW 2015 lower house
6	47,933	213	13	194	15	**8,023**	10,892	Seat of Maitland, NSW 2015 lower house
6	47,208	173	8	134	11	**417**	2345	Seat of Lismore, NSW 2015 lower house
6	47,370	503	12	449	14	**20319**	20493	Seat of Willoughby, NSW 2015 lower house
7	47,865	385	11	945	50	**2,259**	2,534	Seat of Ballina, NSW 2015 lower house
7	48,358	819	1	2,182	8	6,264	6,264	Seat of Newcastle, NSW 2015 lower house
7	45,497	632	1	1,624	14	7,072	7,072	Seat of Newtown, NSW 2015 lower house
7	48,065	1,005	1	3,031	14	8,506	8,506	Seat of Lake Macquarie, NSW 2015 lower house
8	47,590	4,663	1	50,983	33	8,137	8,137	Seat of Clarence, NSW 2015 lower house
8	47,803	12,963	1	130,655	27	14,621	14,621	Seat of Hawkesbury, NSW 2015 lower house
8	48,571	6,679	1	62,190	31	9,948	9,948	Seat of Swansea, NSW 2015 lower house
8	46,756	11,624	2	156,357	38	17,147	17,147	Seat of Murray, NSW 2015 lower house
8	48,002	3,389	1	25,774	33	5,151	5,151	Seat of Penrith, NSW 2015 lower house
8	42,892	2,304	1	14,119	26	5,727	5,727	Seat of Sydney, NSW 2015 lower house

Table 3. Running times and margins computed for each of the IRV elections of Table 2, using MRSW and margin, under the restriction that votes can only be *added* (not deleted or modified). Cases where the margin of victory (MOV) differs from the last round margin (LRM) are in bold.

7 Concluding Remarks

We have presented an algorithm, denoted margin, for computing IRV margins that significantly outperforms the current state-of-the-art. Our algorithm can efficiently compute the MOV in all IRV instances for which we could obtain data. This includes a number of instances for which the current state-of-the-art approach could not compute the margin, in a reasonable time frame of 72 hours. The significance of this work is that automated margin computation is now practical for IRV elections with a large number of candidates. We have presented two easily computed lower bounds on the degree of manipulation required to realise an elimination order ending in a specific sequence. This allows us to prune large portions of the space of possible alternate elimination orders when computing IRV margins. Moreover, we have described how our margin algorithm can be used to determine whether lost votes, or invalid votes (e.g., from electors voting multiple times), could have influenced an election outcome.

IRV has several extensions, including various forms of the Single Transferable Vote (STV). STV is used to elect candidates to the Australian Senate, in all elections in Malta, and in most elections in the Republic of Ireland [14]. The extension of our algorithm for computing margins in IRV elections to STV elections, where candidates are elected to multiple seats, and the votes of elected candidates are redistributed at a fractional value, is a topic of future research. Preliminary results suggest that our margin algorithm can be adapted to apply to STV, using a non-linear version of DISTANCETO.

REFERENCES

[1] ABC. Thousands admit to multiple votes in 2013 federal election. www.abc.net.au/news/2014-02-26/thousands-admit-to-multiple-votes-in-2013-federal-election/5284230, 2014. Accessed: August 2015.

[2] J. J. Bartholdi III and J. B. Orlin. Single transferable vote resists strategic voting. *Social Choice and Welfare*, 8(4):341–354, 1991.

[3] J. J. Bartholdi III, C. A. Tovey, and M. A. Trick. How hard is it to control an election? *Mathematical and Computer Modelling*, 16(8):27–40, 1992.

[4] R. Bredereck, J. Chen, P. Faliszewski, A. Nichterlein, and R. Niedermeier. Prices matter for the parameterized complexity of shift bribery. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 1398–1404, 2014.

[5] R. Bredereck, P. Faliszewski, R. Niedermeier, and N. Talmon. Large-scale election campaigns: combinatorial shift bribery. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 67–75. International Foundation for Autonomous Agents and Multiagent Systems, 2015.

[6] R. Bredereck, P. Faliszewski, R. Niedermeier, and N. Talmon. Complexity of shift bribery in committee elections. *arXiv preprint arXiv:1601.01492*, 2016.

[7] D. Cary. Estimating the margin of victory for instant-runoff voting. In *USENIX Accurate Electronic Voting Technology Workshop: Workshop on Trustworthy Elections*, USENIX Association Berkeley, CA, USA, 2011.

[8] V. Conitzer, J. Lang, and T. Sandholm. How many candidates are needed to make elections hard to manipulate? In *Proceedings of the 9th Conference on Theoretical Aspects of Rationality and Knowledge*, pages 201–214, Bloomington, Indiana, USA, 2003. ACM.

[9] V. Conitzer, T. Sandholm, and J. Lang. When are elections with few candidates hard to manipulate? *Journal of the ACM*, 54(3): 14, 2007.

[10] E. Elkind, P. Faliszewski, and A. Slinko. Swap bribery. In *Algorithmic Game Theory*, pages 299–310. Springer, 2009.

[11] P. Faliszewski, E. Hemaspaandra, and L. A. Hemaspaandra. The complexity of bribery in elections. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, volume 6, pages 641–646, 2006.

[12] P. Faliszewski, E. Hemaspaandra, and L. Hemaspaandra. How Hard Is Bribery in Elections? *Journal of Artificial Intelligence Research*, 35:485–532, 2011.

[13] P. Faliszewski, Y. Reisch, J. Rothe, and L. Schend. Complexity of manipulation, bribery, and campaign management in Bucklin and fallback voting. *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 29(6):1091–1124, 2015.

[14] D. Farrell and I. McAllister. Australia: The Alternative Vote in a Compliant Political Culture. In M. Gallagher and P. Mitchell, editors, *The Politics of Electoral Systems*, pages 79–97. Oxford University Press, Oxford, 2005.

[15] M. Lindeman and P.B. Stark. A gentle introduction to risk-limiting audits. *IEEE Security and Privacy*, 10:42–49, 2012.

[16] 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.

[17] N. Narodytska and T. Walsh. The computational impact of partial votes on strategic voting. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, pages 657–662, 2014.

[18] Parliament of Australia. The disputed 2013 WA Senate election. www.aph.gov.au/About_Parliament/Parliamentary_Departments/Parliamentary_Library/FlagPost/2013/November/The_disputed_2013_WA_Senate_election, 2013. Accessed: April 2016.

[19] R. Richie. Instant Runoff Voting: What Mexico (and Others) Could Learn. *Election Law Journal*, 3:501–512, 2004.

[20] J. Rothe and L. Schend. Challenges to complexity shields that are supposed to protect elections against manipulation and control: a survey. *Annals of Mathematics and Artificial Intelligence*, 68(1-3):161–193, 2013. ISSN 1012-2443.

[21] A. Sarwate, S. Checkoway, and H. Shacham. Risk-limiting audits and the margin of victory in nonplurality elections. *Statistics, Politics, and Policy*, 4(1):29–64, January 2013.

[22] L. Xia. Computing the margin of victory for various voting rules. In *Proceedings of the ACM Conference on Electronic Commerce (EC)*, pages 982–999, 2012.