Shifting the Balance-of-Power in STV Elections

Michelle Blom[1], Andrew Conway[2], Peter J. Stuckey[3], and Vanessa J. Teague[4]

[1] School of Computing and Information Systems, University of Melbourne
michelle.blom@unimelb.edu.au
[2] Silicon Econometrics Pty. Ltd
andrewelections@greatcactus.org
[3] Department of Data Science & AI, Monash University
Peter.Stuckey@monash.edu
[4] Thinking Cybersecurity Pty. Ltd.
vanessa@thinkingcybersecurity.com

Abstract. In the context of increasing automation of Australian electoral processes, and accusations of deliberate interference in elections in Europe and the USA, it is worthwhile understanding how little a change in the recorded ballots could change an election result. In this paper we construct manipulations of the ballots in order to change the overall balance of power in an Australian Federal Senate election – the upper house of Parliament. This gives, hopefully tight, overestimations of the Margin of Victory (MOV) for the party or coalition winning the Senate. This is critical information for determining how well we can trust the reported results, and how much auditing should be applied to the election process to be certain that it reports the true result. The challenge arising in Australian Federal Senate elections is that they use a complicated Single Transferable Vote (STV) method for which it is intractable to compute the true MOV, hence we must rely on greedy methods to find small manipulations.

Keywords: Single Transferable Vote · Balance of Power · Margin of Victory

1 Introduction

In a climate of increasing public mistrust in all governmental activities, assurances that the results of elections are correct are critical for democracies to function well. One critical statistic that helps to define how trustworthy an election result is, is the so called Margin of Victory (MOV), which indicates the minimal number of ballots that need to be modified to change the election result. If the MOV is small, then we should invest considerable effort in auditing the election processes, since the true result may differ if inevitable errors lead to changes greater then that MOV. If the MOV is large we require less auditing to be assured that the election outcome is likely to be correct.

The Australian Federal Parliament consists of two houses: the House of Representatives, which defines the executive part of government responsible for making new laws; and the Senate, a house of review. For laws to be enacted they must pass both houses, so the controller of the Senate has significant influence on what legislation can be enacted. Australian politics is dominated by two “parties”: the Labor Party (progressive); and the Liberal/National Party Coalition (conservative), an enduring coalition of two parties. Historically, one or other party has formed government. The Senate is more complicated as there is a greater number of smaller parties and independents. In some cases no party has held the balance of power in the Senate, though usually one or other party, with perhaps some agreements with minor parties, does.

Existing work [2] has examined how to compute the MOV for Australian House of Representatives elections, which makes use of Instant Runoff Voting (IRV). In this paper, we examine the much more challenging problem of *estimating* the MOV for Australian Federal Senate elections. The difficulty arises because the election uses Single Transferable Vote (STV) which is a complicated election methodology. Determining the MOV of an STV election is NP-hard [8]. While we can determine MOVs for small individual STV elections [3], these methods do not scale to the size of the elections that actually occur for the Australian Federal Senate.

An Australian Federal Senate election consists of a separate STV election in each of the six Australian states, and the two Australian territories. There are 76 seats in the Senate, with 12 seats awarded to each of the six states, and 2 to each of the two territories. In a regular election, 6 of the available 12 seats for each state, and both of the 2 seats for each territory, are up for re-election. In a double-dissolution election, all 76 seats are vacated. The party, or coalition of parties, that occupies the majority of seats in the Senate chamber (39 or greater) significantly influences the legislation that the government is able to pass. Legislation has to pass through both houses of Parliament (the lower and upper house) before it can become law. In the 2016 and 2019 Australian Federal elections, conservative politicians have formed the majority in both houses. This has limited the power of more progressive parties to shape legislation.

In this paper, we consider estimating the number of ballot changes required to change the outcome of such an election to give a particular coalition of parties the majority in the Senate. In 2016, we would have had to shift four Senate seats away from conservatives, to progressive candidates, to change the nature of the majority. In 2019, only two seats were required to change hands to achieve a progressive majority. We present a heuristic, combined with an integer program (IP), to compute an upper bound on the number of ballot changes required in order to award an additional n seats to a coalition of parties, $\mathbb{C}$. In other words, if a coalition $d \in \mathbb{C}$ was originally awarded N_d seats, we are interested in manipulations that would result in d being awarded $n + N_d$ seats. This implies that n seats are taken away from candidates outside of d.

An Australian Federal Senate election consists of a number of separate STV elections. Our approach is based on finding small manipulations of each individual s-seat STV election, that awards an additional $j = 1, \ldots, k$ seats to our desired coalition, where $k = \min(s, n)$. A knapsack problem is then solved to determine the combination of these manipulations that results in a combined n-seat shift to our coalition with the least number of required ballot changes. Existing work [1] considers the use of local search for finding small manipulations of a STV election that elects a specific, favoured candidate c to a seat. This paper moves beyond this to find an upper bound on the manipulation required to elect n additional candidates from a coalition of parties, across a set of individual STV elections responsible for allocating seats in a Senate.

We apply our method to data from the 2016 and 2019 Australian Federal Senate elections. In both elections, candidates from conservative parties form the majority in the elected Senate. We consider a coalition of more centrist or left-leaning parties, forming our desired coalition d. We then use our approach to find an upper bound on the number of ballots we would have to change, across each of the state and territory STV elections that form part of the Senate election as a whole, to shift enough seats to candidates in d to change the nature of the majority. In the 2016 and 2019 elections, we want to shift $n = 4$, and $n = 2$, seats respectively. Our local search algorithm is used to compute candidate manipulations that shift $j = 1, \ldots, \min(s, n)$ seats in each individual s-seat STV election. A simple integer program (IP) is then applied to the results to select a least cost combination of manipulations to apply in each state and territory. We have found that we can give a progressive coalition d a majority by changing 40,008 ballots in the 2016 election, and 27,635 ballots in the 2019 election.

2 Preliminaries

STV is a preferential system of voting in which voters rank candidates or parties in order of preference, and candidates compete for s seats. In Australian Senate Elections voters may cast their ballot in two ways. First, they may vote *above the line*. At the top of each ballot is a sequence of boxes, one for each party and group of independent candidates. To vote above the line, a voter ranks at least 6 of these parties and grouped independents.Alternatively, a voter may vote *below the line*. Under each of the above the line party boxes is a list of candidates belonging to that party or group. To vote below the line, a voter ranks at least 14 individual candidates in order of preference. STV elections for the Australian Senate can involve over 100 candidates.

Definition 1 (STV Election) *An STV election $\mathcal{E}$ is a tuple $\mathcal{E} = (\mathcal{C}, \mathcal{P}, \mathcal{B}, Q, s)$ where $\mathcal{C}$ is a set of candidates, $\mathcal{P}$ is the set of parties or groups to which candidates belong, $\mathcal{B}$ the multiset of ballots cast, Q the election quota (the number of votes a candidate must attain to win a seat – the Droop quota – Eqn 1), and s the number of seats to be filled.*

$$Q = \left\lfloor \frac{|\mathcal{B}|}{s+1} \right\rfloor + 1 \tag{1}$$

We will use a small running example to describe the concepts in this section, and our n-seat shifting approach. In this example, two 3-seat STV elections, $\mathcal{E}_1$ and $\mathcal{E}_2$, are held to elect a total of 6 candidates to a small Senate. In each election, four parties (A, B, C, and D) field 2 candidates, resulting in a total of 8 candidates. The candidates of $\mathcal{E}_1$ are denoted a_{11}, a_{12}, . . ., d_{11} and d_{12}, and those of $\mathcal{E}_2$, a_{21}, a_{22}, . . ., d_{21} and d_{22}. Tables 1a and 2a define the ballot profiles of $\mathcal{E}_1$ and $\mathcal{E}_2$, listing the number of ballots cast with a range of different *above the line* party rankings. For each ranking, we state the equivalent *below the line* ranking, indicating how the ballot would pass from candidate to candidate if they were eliminated in that sequence. During the election counting process, valid above the line votes are treated exactly as their below the line equivalent.

The counting of ballots in an STV election starts by distributing each ballot to the tally pile of its first ranked candidate. An above the line vote with a first preference for party p is given to the first candidate of that party listed on the ballot. Candidates are awarded a seat if the number of votes in their tally reaches or exceeds a threshold,

Ranking ATL	Ranking BTL	Count
[A, B]	$[a_{11}, a_{12}, b_{11}, b_{12}]$	270
[B, A, D, C]	$[b_{11}, b_{12}, a_{11}, a_{12}, d_{11}, d_{12}, c_{11}, c_{12}]$	250
[C, D, A, B]	$[c_{11}, c_{12}, d_{11}, d_{12}, a_{11}, a_{12}, b_{11}, b_{12}]$	20
[D, C, A]	$[d_{11}, d_{12}, c_{11}, c_{12}, a_{11}, a_{12}]$	5

(a)

Seats: 3, Quota: 137

Candidate	Round 1	Rounds 2-3	Rounds 4-6	Rounds 7-8
		a_{11}, b_{11} elected 133 votes to a_{12} 113 votes to b_{12}	c_{12}, d_{12} eliminated d_{11} eliminated 50 votes to c_{11}	c_{11} eliminated 250 votes to a_{12} a_{12} elected
a_{11}	270	–	–	–
a_{12}	0	$0{+}133 = 133$	133	$133 + 25 = 158$
b_{11}	250	–	–	–
b_{12}	0	$0{+}113 = 113$	113	113
c_{11}	20	20	$20 + 5 = 25$	–
c_{12}	0	0	–	–
d_{11}	5	5	–	–
d_{12}	0	0	–	–

(b)

Table 1: STV election, $\mathcal{E}_1$, stating (a) the number of ballots cast with each listed above-the-line ranking over parties A to D, their equivalent below-the-line ranking over candidates a_{11} to d_{12}, and (b) the tallies after each round of election, and elimination.

called a *quota*. The value of the quota is based on the total number of ballots cast in the election, and the number of seats available (Eqn 1). The quotas of elections $\mathcal{E}_1$ and $\mathcal{E}_2$ are 137 votes (1 + $\lfloor$ 545 / (3 + 1) $\rfloor$ = 137) and 188 votes, respectively.

Counting proceeds by electing candidates whose *tallies* (Definition 2) reach or exceed the quota, and distributing their *surplus* to the candidates that remain standing. A candidate's surplus is equal to the difference between their tally value and the quota. The non-exhausted ballots in an elected candidates tally pile are distributed to eligible candidates at a *reduced value*. Each ballot, starting with a value of 1, is reduced in value so that the sum of the value of the transferred ballots is equal to the surplus.

Definition 2 (Tally $t_i(c)$) *The tally of a candidate $c \in \mathcal{C}$ in round i is the sum of the values of the ballots in c's tally pile. These are the ballots for which c is ranked first among the set of candidates still standing, S_i. Let $\mathcal{B}_{i,c}$ denote the subset of ballots sitting in c's tally pile, and $v_i(b)$ the value of ballot b, at the start of round i.*

$$t_i(c) = \sum_{b \in \mathcal{B}_{i,c}} v_i(b) \tag{2}$$

Ranking ATL	Ranking BTL	Count
[B, C, D, A]	$[b_{21}, b_{22}, c_{21}, c_{22}, d_{21}, d_{22}, a_{21}, a_{22}]$	2,000
[A, D, C, D]	$[a_{21}, a_{22}, d_{21}, d_{22}, c_{21}, c_{22}, d_{21}, d_{22}]$	2,100
[D, A, B, C]	$[d_{21}, d_{22}, a_{21}, a_{22}, b_{21}, b_{22}, c_{21}, c_{22}]$	1,700
[C, D, A, B]	$[c_{21}, c_{22}, d_{21}, d_{22}, a_{21}, a_{22}, b_{21}, b_{22}]$	1,700

(a)

Seats: 3, Quota: 188

Candidate	Round 1	Rounds 2-3	Rounds 4-6	Rounds 7-8
		a_{21}, b_{21} elected 22 votes to a_{22} 12 votes to b_{22}	d_{22}, c_{22} eliminated b_{22} eliminated 12 votes to c_{21}	a_{22} eliminated 22 votes to d_{21} d_{21} elected
a_{21}	200	–	–	–
a_{22}	0	$0+22 = 22$	22	–
b_{21}	210	–	–	–
b_{22}	0	$0+12 = 12$	–	–
c_{21}	170	170	$170 + 12 = 182$	182
c_{22}	0	0	–	–
d_{21}	170	170	170	$170 + 22 = 192$
d_{22}	0	0	–	–

(b)

Table 2: STV election, $\mathcal{E}_2$, stating (a) the number of ballots cast with each listed above-the-line ranking over parties A to D, their equivalent below-the-line ranking over candidates a_{21} to d_{22}, and (b) the tallies after each round of election, and elimination.

Table 1b shows that for $\mathcal{E}_1$, only the first listed candidate of each party have votes in their tallies after Round 1 of counting. This is because all voters have cast an above the line vote. The ballots sitting in a_{11}'s tally will pass to a_{12} when a_{11} is either elected or eliminated. Candidates a_{11} and b_{11} have a quota with tallies of 270 and 250 votes, and surpluses of 133 and 113 votes. They are elected to the first two available seats in Rounds 2-3 of counting. All 270 ballots in a_{11}'s tally pile are given to a_{12}, but they now have a combined value of 133 (each ballot now has a reduced value of 0.4926).

If no candidate has a quota, the candidate with the smallest tally is eliminated. In $\mathcal{E}_1$ (Table 1b), no candidate has a quota after the election of a_{11} and b_{11}. The candidates with the smallest tally, c_{12} and d_{12}, both with 0 votes, are eliminated in Rounds 4-5. In Round 6, d_{11}, with 5 votes, is eliminated. These 5 votes are transferred, at their current value, to c_{11}, as d_{12} is no longer standing. Candidate c_{11} now has 25 votes.

The STV counting process continues in rounds of electing candidates whose tallies have reached a quota, and elimination of candidates with the smallest tally. In $\mathcal{E}_1$, candidate c_{11} is eliminated in Round 7, with their votes distributed to a_{12}. In Round 8, a_{12} is elected to the final seat, with their tally having exceeded a quota's worth of votes.

Several STV variants exist, differing in the way that surpluses are distributed [7]. The method of reducing the value of transferred ballots described above is the Inclusive Gregory Method [5]. The precise rules used by the Australian Federal Senate for adjusting the values of transferred ballots are more complex, and outlined in legislation.

The approach we present in this paper searches for manipulations of the STV elections that form part of an Australian Federal Senate election that achieve a desired outcome. Given a favoured coalition of parties d, whose candidates have been awarded N_d seats in the un-manipulated election, we are interested in manipulations that award $N_d + n$ candidates in d a seat. We define a manipulation of an STV election as follows.

Definition 3 (Manipulation $\mathcal{M}$) *A manipulation for an election $\mathcal{E} = (\mathcal{C}, \mathcal{P}, \mathcal{B}, Q, s)$ is a tuple $\mathcal{M} = (\mathcal{B}^+, \mathcal{B}^-)$, where: $\mathcal{B}^+$ denotes a multiset of ballots to add to $\mathcal{B}$; $\mathcal{B}^-$ a multiset of ballots to remove from $\mathcal{B}$; and $|\mathcal{B}^+| \equiv |\mathcal{B}^-|$. The result of applying $\mathcal{M}$ to an election $\mathcal{E}$ is a modified election profile $\mathcal{E}' = (\mathcal{C}, \mathcal{P}, \hat{\mathcal{B}}, Q, s)$, where $\hat{\mathcal{B}}$ is the result of removing each ballot in $\mathcal{B}^-$ from $\mathcal{B}$, and then adding each ballot in $\mathcal{B}^+$ to $\mathcal{B}$.*

To assess whether a given manipulation $\mathcal{M}$ awards n additional seats to our favoured coalition d, we simulate the STV counting process on the manipulated election profile $\mathcal{E}'$, and count the number of seats awarded to d in the outcome. We use a simulator, denoted SIM-STV, that captures the intricate rules specific to the Australian Federal Senate election, as defined in legislation. All the manipulations we generate in our experiments are validated on the full federal rules [4] using SIM-STV, and ballot data published by the AEC, standardized at `https://vote.andrewconway.org/`. An example manipulation for the election of Table 1 is shown in Example 1.

Example 1. Consider election $\mathcal{E}_1$ of Table 1. If we replace 111 ballots with the above the line ranking [A, B] with the above the line ranking [D], we no longer elect candidate a_{12} but elect d_{11} in their place. Candidates a_{11} and b_{11} are elected in the first two rounds, as before. Candidates d_{12}, c_{12}, and c_{11}, are then eliminated. The reduced flow of votes from a_{11} to a_{12} leaves a_{12} on only 22 votes, compared to d_{11}'s 136 and b_{12}'s 135 votes.

The rules used for Australian Federal Senate elections [6] are close to that described above, with some idiosyncrasies. For example, when ballots are distributed from one candidate's tally pile to another, the total value of those ballots is rounded down to the nearest integer. This practice causes a number of votes to be lost over the course of the tallying process. For further details of SIM-STV, we refer the reader to [1].

3 Finding n-seat Senate Manipulations

We present an approach for computing a manipulation of one or more of the individual STV elections that form an Australian Federal Senate election, to shift the majority of seats away from an unfavoured coalition of parties to a favoured coalition. In the 2016 and 2019 Australian Federal Senate elections, a conservative coalition of parties had a 4 and 2 seat majority. Our approach looks for the best combination of manipulations to apply to the state and territory STV elections to realise a combined $n = 4$, and $n = 2$, seat shift to a progressive coalition. A shift of 2 seats could be realised by shifting 1 seat to our favoured coalition in Victoria, for example, and 1 seat in New South Wales.

Our approach consists of two stages. The first stage looks at each constituent s-seat STV election individually. We use a local search heuristic, described in Section 3.1, to find small manipulations that shift varying numbers of seats ($k = 1, \dots, \min(s, n)$) away from the undesired coalition $u \in \mathbb{C}$ to candidates that belong to a desired coalition of parties $d \in \mathbb{C}$. The local search method is not optimal – it may not find the smallest possible manipulation that shifts k seats to our favoured candidates.

As a result of this first stage, we have a series of manipulations, for each state and territory election, that shift varying numbers of seats to our desired coalition. The second stage of our approach solves a simple integer program (IP) to select the combination of manipulations that realises our n-seat shift with the smallest number of required ballot changes. To achieve a shift of 4 seats, for example, we may shift 1 seat in each of four state elections, or 2 seats in one state and 2 seats in another.

Example 2. In Example 1, we manipulated $\mathcal{E}_1$ by 111 ballots (shifted from party A to D), giving 1 seat to D at the expense of A. In order to give 2 seats to a coalition of parties C and D, we need to shift ballots away from A and B to C and D. Consider a manipulation that removes 116 ballots with ranking [A, B], replacing them with the ranking [C], and 131 ballots with ranking [B,A,D,C], replacing them with [D]. This manipulation results in both c_{11} and d_{11} being elected at the expense of a_{12} and b_{11}.

3.1 Finding Manipulations with Local Search

Given an s-seat STV election $\mathcal{E}$, we present a local search heuristic for finding manipulations that award an additional k seats to a candidates in a desired coalition d. In the original outcome of $\mathcal{E}$, N_d seats have been awarded to candidates from d. We seek a manipulation of $\mathcal{E}$ in which $N_d + k$ seats are awarded to candidates from d. The heuristic is provided as input k candidate pairs of unfavoured original winner w, and favoured original loser l. We then search for smaller and smaller manipulations that aim to rob each w of a seat, and elect l. We repeatedly apply this heuristic to different sets of k candidate pairs, returning the smallest found successful manipulation as a result.

To identify sets of k winner-loser pairs to consider, we start with: a set of unfavoured original winners $\mathcal{W}$; and a set of favoured original losers $\mathcal{L}_d$.

1. Let $\overline{\mathcal{W}}$ denote the set of all k-candidate subsets of $\mathcal{W}$, and $\overline{\mathcal{L}_d}$ the set of all k-candidate subsets of $\mathcal{L}_d$.
2. The k-candidate subsets in $\overline{\mathcal{W}}$ are sorted in order of the total tally of candidates upon their election, from smallest to largest. This is the sum of each candidates tally in the round in which they were elected to a seat. The first subset in the sorted $\overline{\mathcal{W}}$ consists of candidates who were elected to a seat with the smallest tallies.
3. The k-candidate subsets in $\overline{\mathcal{L}_d}$ are sorted in order of the total tally of candidates upon their elimination, from largest to smallest. This is the sum of each candidates tally in the round in which they were eliminated. The first subset in the sorted $\overline{\mathcal{L}_d}$ consists of candidates who were eliminated with the largest tallies.

4. To limit the complexity of our approach, we restrict our attention to the first M subsets in $\overline{\mathcal{W}}$ and $\overline{\mathcal{L}_d}$. For each $W \in \overline{\mathcal{W}}$, we consider each $L \in \overline{\mathcal{L}_d}$. We apply our local search heuristic with k winner-loser pairs formed by pairing the first winner in W with the first loser in L, the second winner in W with the second loser in L, and so on. We return the best (smallest) successful manipulation found by applying our local search heuristic to each set of the $M \times M$ generated k winner-loser pairs.

Given a list of k unfavoured winner, favoured loser, pairs, our method aims to replace each unfavoured winner with its paired loser. However, any manipulation that elects $N_d + k$ candidates from d is considered to be successful. Each application of the local search heuristic involves two phases.

Phase 1 The first stage finds an initial, but potentially quite large, successful manipulation that, upon simulation of the manipulated election profile, elects at least $k + N_d$ candidates from our desired coalition $d \in \mathbb{C}$. This manipulation is denoted M_0.

Phase 2 We then repeatedly search for a good 'size reducing' move to apply to M_0. These moves reduce the number of ballots shifted between candidates, while still ensuring that the manipulation successfully elects $k + N_d$ candidates from d. In each iteration, we examine the set of possible changes (moves) we could make to the current 'best found' manipulation, selecting the move that results in the largest reduction in the number of ballot changes. When no 'size reducing' move can be found, search terminates and returns the best (smallest) manipulation it has found.

A manipulation defines k sets of ballot shifts between pairs of candidates from W and L. For each such (w, l) pair, our goal is to find a manipulation that replaces a certain number of ballots that favour w – ballots that form part of w's tally at the point of their election – with ballots that favour l. We consider three different approaches for specifying the ranking of these l-favouring ballots, denoted BTL, ATL, and IW. The latter, IW, uses the set of original winners from our desired coalition d, denoted $\mathcal{W}_d$.

BTL A below the line vote that preferences l first, and each other loser in L subsequently, in the order they appear in L.
ATL An above the line vote that preferences l's party first, and the parties of all other candidates in L subsequently, in the order they appear in L.
IW A below the line vote that preferences l first, and each of the original winners from our desired coalition d, $\mathcal{W}_d$, subsequently.

Example 3. When seeking to elect $k = 2$ candidates from the coalition $d = \{C, D\}$ in $\mathcal{E}_1$, our list of original winners is $\mathcal{W} = \{a_{11}, a_{12}, b_{11}\}$ and favoured losers $\mathcal{L}_d = \{c_{11}, c_{12}, d_{11}, d_{12}\}$. Our winner subsets $\overline{\mathcal{W}}$, sorted in order of the total tally of the candidates, upon their election (smallest to largest), are $\overline{\mathcal{W}} = \{\{a_{12},b_{11}\}, \{a_{12},a_{11}\}, \{b_{11},a_{11}\}\}$. Our subsets of favoured losers, sorted in order of the total tally of candidates, upon their elimination (largest to smallest), are $\overline{\mathcal{L}_d} = \{\{c_{11},d_{11}\}, \{c_{11},d_{12}\}, \{c_{11},c_{12}\},\{d_{11},d_{12}\}, \{d_{11},c_{12}\}, \{d_{12},c_{12}\}\}$. Our approach will try to elect each pair of losers in $\overline{\mathcal{L}_d}$, at the expense of each pair of winners in $\overline{\mathcal{W}}$, starting with $\{c_{11},d_{11}\}$ and $\{a_{12},b_{11}\}$. For these winner-loser pairs, a_{12}-c_{11} and b_{11}-d_{11}, we find a M_0 that: replaces 226 ballots that sit

```
FINDINITIALMANIPULATION($k$, $N_d$, $W$, $L$, $t$)
1  $M_0 \leftarrow \emptyset$
2  for $i$ in $1..k$ do
3      $w \leftarrow W[i]$
4      $l \leftarrow L[i]$
5      $\Delta_{i,0} \leftarrow \lceil t[w] - t[l] \rceil$
6  verified $\leftarrow$ Verify $M_0$ with SIM-STV.
7  if verified then
8      return $M_0$ as our initial manipulation
9  else
10     $M_0' \leftarrow M_0$
11     while not verified do
12         $M_0' \leftarrow$ Increase each $\Delta_{i,0}$ by a factor of 2, capping each $\Delta_{i,0}$ by $t[i]$.
13         verified $\leftarrow$ Verify $M_0'$ with SIM-STV
14         if verified then
15             return $M_0'$ as our initial manipulation
16 return failure
```

Fig. 1: Phase 1: Find initial manipulation to achieve the election of $k + N_d$ candidates from a desired coalition, where: W denotes original winners that are not in our desired coalition; and L are original losers who are in our desired coalition. Note that $t[c]$ denotes the tally of candidate c upon their election (if $c \in W$) or elimination (if $c \in L$).

in a_{12}'s tally upon their election, with a ranking that favours c_{11}; and 249 ballots that sit in b_{11}'s tally upon their election, with a ranking that favours d_{11}. The total size of this manipulation is 475. The IW method would replace 226 of a_{12}'s ballots with the ranking [c_{11}]. The ATL method would replace these ballots with the ranking [C], where C is the party to which c_{11} belongs. The BTL method would form 226 ballots with ranking [$c_{11}, d_{11}, c_{12}, d_{12}$], adding the remaining favoured losers after c_{11}.

Additional Notation We use notation $\Delta_{i,j}$ to denote the number of ballots shifted between the i^{th} of our k winner-loser pairs, (w, l), in a manipulation M_j. A 'shift' of $\Delta_{i,j}$ ballots between w and l replaces $\Delta_{i,j}$ ballots that sit in w's tally at the time they are elected with $\Delta_{i,j}$ ballots whose ranking has been specified according to one of the above methods (BTL, ATL, or IW). The notation $t[c]$ denotes the tally of candidate c upon their election (if they are an original winner) or elimination (if they lost).

Phase 1: Finding an Initial Manipulation We define an initial manipulation M_0 by assigning a suitably high value to $\Delta_{i,0}$ for each winner-loser pair $i = 1, \ldots, k$ (see Fig 1). We verify M_0 by simulating it with SIM-STV, and verifying that $k + N_d$ candidates from our coalition are elected in the manipulated election.

Phase 2: Reduce size of Manipulation In the case where $k = 1$, we have one winner w that we want to replace with a loser l. Our initial manipulation M_0 is iteratively reduced by only one type of move (as shown in Fig 2). A 'step size', δ, controls how we reduce the size of our manipulation. We first reduce the number of ballots shifted between

```
MinimiseManipulation$_{k=1}$($M_0$, $k$, $N_d$, $\alpha$, $\gamma$)
1   $M_{best} \leftarrow M_0$
    ▷ Initialise step size $\delta$ based on size of initial manipulation
2   $\delta \leftarrow \lceil \frac{\Delta_{1,0}}{\gamma} \rceil$
3   while true do
4       $M_1 \leftarrow M_{best}$
5       $\Delta_{1,1} \leftarrow \Delta_{1,best} - \delta$
6       verified $\leftarrow$ Verify $M_1$ with SIM-STV
7       if verified then
8           $M_{best} \leftarrow M_1$
9       else
10          if $\delta \equiv 1$ then return $M_{best}$
11          $\delta \leftarrow \lceil \frac{\delta}{\alpha} \rceil$
```

Fig. 2: Phase 2 ($k = 1$): Reduce size of an initial manipulation M_0 by reducing the number of shifted votes Δ by a step size δ. The step size δ is reduced each time the manipulation becomes too small (i.e., fails to realise the election of $k + N_d$ candidates from our desired coalition). In the above, $\alpha \geq 2$ and $\gamma \geq 2$ are predefined constants.

our winner and loser by the step size δ. If that reduction does not lead to a successful manipulation, we reduce δ, and keep trying (until we fail to find a better manipulation by shifting 1 less ballot). If a successful, smaller manipulation is found, we increase δ.

In the case where $k > 1$, our heuristic applies one of three types of moves in each iteration: reduce the shift of votes between one pair unfavoured winner and favoured loser (Move$_1$); reduce the shift of votes between each unfavoured winner and favoured loser pair (Move$_2$); and reduce the number of ballots shifted between one winner-loser pair while increasing the shift of votes between each other winner-loser pair (Move$_3$).

We maintain a step size δ_i^m for each move type m and winner-loser pair i. When we first use a particular move type $m \in \{1, 2, 3\}$ to reduce the size of a shift of ballots between the i^{th} winner-loser pair, we reduce the number of ballots shifted by the step size δ_i^m. As in the $k = 1$ setting, if that reduction does not lead to a successful manipulation, we reduce δ. If a successful, smaller manipulation is found, we increase the step size for the next time this kind of move is applied. An interpretation of the steps sizes is that they are an estimate of how much we think we can reduce the size of a shift between two candidates, via each different move type, and achieve a successful manipulation.

We apply these moves iteratively, as follows. Pseudocode for each type of move is provided in Fig 3. The predefined constants $\gamma \geq 2$ and $\alpha \geq 2$ are used when initialising, and updating, step sizes. The constant γ is used to initialise our step size δ – the amount by which we reduce the size of a manipulation as we look for smaller and smaller successful manipulations. Given an initial, quite large, manipulation that shifts $\Delta_{i,0}$ ballots between winner-loser pair i, our step size δ_i is initialised to $\lceil \frac{\Delta_{i,0}}{\gamma} \rceil$. The constant α is used to reduce our step size as the algorithm progresses (Step 11 in Fig 2), allowing us to make more fine grained changes in the search for a minimal manipulation.

1. We maintain a running record of the best (smallest) manipulation found thus far, M_{best}, initialised to M_0.

2. Step sizes, δ_i^m, are first initialised to $\lceil \frac{\varDelta_{i,0}}{\gamma} \rceil$ for $i = 1, \ldots, k$.
3. As per Fig. 3, we apply move type 1 (MOVE$_1$) to find a smaller manipulation than M_{best}, using the current set of step sizes δ_i^1. The result is a new manipulation M_1.
4. If $M_1 \neq \emptyset$, we have been able to reduce the vote shift between one winner-loser pair. We then apply move type 2 (MOVE$_2$ in Fig. 3) to M_{best} to find a smaller manipulation than M_1, denoted M_2, using the step sizes δ_i^2.
5. If either move type 1 or 2 were successful, we update M_{best} to the smallest of the two manipulations, M_1 or M_2, and return to Step 3.
6. If neither moves 1 and 2 were successful, we apply MOVE$_3$ to find a smaller manipulation than M_{best}, denoted M_3, using the current set of step sizes δ_i^3.
7. If $M_3 \equiv \emptyset$, we have failed to improve upon M_{best}, and return M_{best} as our best found manipulation. If $M_3 \neq \emptyset$, we replace M_{best} with M_3, reset our step sizes for move types 1 and 2 to their initial values, and return to Step 3.

Example 4. After finding an initial manipulation of 475 ballots to award candidates c_{11} and d_{11} a seat at the expense of a_{12} and b_{11}, we move to Phase 2 and try to find a smaller manipulation. In our initial manipulation M_0, we have $\varDelta_{1,0} = 226$ and $\varDelta_{2,0} = 249$, where winner-loser pair 1 is a_{12}-c_{11} and winner-loser pair 2 is b_{11}-d_{11}.

Using the parameters $\alpha = 5$ and $\gamma = 2$, we initialise our step sizes for each move and winner-loser pair combination as follows:

$$\delta_1^1 = \delta_1^2 = \delta_1^3 = \lceil 226/\gamma \rceil = 113 \qquad \delta_2^1 = \delta_2^2 = \delta_2^3 = \lceil 249/\gamma \rceil = 125$$

As per Fig. 3, we first apply MOVE$_1$ to reduce one of the shifts $\varDelta_{1,0}$ and $\varDelta_{2,0}$. We consider each $\varDelta_{i,0}$ in turn. For pair 1, we can reduce $\varDelta_{1,0}$ by the step size, from 226 to 113, and maintain a successful manipulation. Similarly, we can reduce $\varDelta_{2,0}$ by its step size, from 249 to 124, leaving $\varDelta_{1,0} = 226$, and successfully manipulate $\mathcal{E}_1$ to elect 2 candidates from C and D. We choose the downward shift that results in the largest reduction in ballot changes, and reduce $\varDelta_{2,0}$ to 124. Our best found manipulation now shifts 350 ballots. We next consider MOVE$_2$ on our initial manipulation M_0. Here, we see if we can reduce both $\varDelta_{i,0}$ by their step sizes δ_i^2, and still maintain a successful manipulation. We find we cannot, the resulting manipulation of 237 ballots is too small. After the first iteration of local search, we accept the best manipulation found across the three move types, MOVE$_1$ (of 350 ballots) in this case, and increase δ_2^1 by a factor of γ.[5] Note that we only consider MOVE$_3$ when neither MOVE$_1$ and MOVE$_2$ is successful.

In the next two iterations, MOVE$_2$ yields the largest reduction in manipulation size, resulting in a manipulation of 282 ballots ($\varDelta_{1,best} = 193$ and $\varDelta_{2,best} = 89$). In the fourth iteration, MOVE$_1$ and MOVE$_2$ are not successful, and we consider MOVE$_3$. We start by reducing $\varDelta_{1,best}$ by δ_1^3, which is still 113 ballots, and increasing $\varDelta_{2,best}$ by 112 ballots. The manipulation with $\varDelta_{1,best} = 80$ and $\varDelta_{2,best} = 201$ is successful, resulting in a new best manipulation size of 281. Reducing $\varDelta_{2,best}$ and increasing $\varDelta_{1,best}$ does not lead to a smaller manipulation, and we accept the shift of 80 and 201 ballots as our

[5] Where $\delta_i^m > \varDelta_{i,j}$, we reset δ_i^m to $\lceil \varDelta_{i,j}/\gamma \rceil$.

```
Move1($S$, $M_{current}$, $k$, $N_d$, $\alpha$, $\gamma$)
1  $M_{best} \leftarrow \emptyset$, $S_{best} \leftarrow S$
2  for $i$ in $1..k$ do
3      $M_1 \leftarrow M_{current}$
4      while true do
5          $\Delta_{i,1} \leftarrow \Delta_{i,1} - \delta_i^1$
           ▷ Consider $M_1$ only if it is smaller than the size of the current best, $S_{best}$
6          if $|M_1| \geq S_{best}$ then break
7          if manipulation $M_1$ is verified by SIM-STV then
8              $M_{best} \leftarrow M_1$, $S_{best} \leftarrow |M_1|$
9              $\delta_i^1 \leftarrow \gamma\, \delta_i^1$
10             break
11         else if $\delta_i^1 \equiv 1$ then break else $\delta_i^1 \leftarrow \lceil \frac{\delta_i^1}{\alpha} \rceil$
12 return $M_{best}$

Move2($S$, $M_{current}$, $k$, $N_d$, $\alpha$, $\gamma$)
1  $M_{best} \leftarrow \emptyset$, $S_{best} \leftarrow S$, $M_2 \leftarrow M_{current}$
2  while true do
3      if $\sum_{i=1}^{k} \delta_i^2 \equiv 0$ then break
4      $\Delta_{i,2} \leftarrow \Delta_{i,2} - \delta_i^2$ for all $i \in \{1..k\}$
5      if $|M_2| \geq S_{best}$ then break
6      if manipulation $M_2$ is verified by SIM-STV then
7          $M_{best} \leftarrow M_2$, $S_{best} \leftarrow |M_2|$
8          $\delta_i^2 \leftarrow \gamma\, \delta_i^2$ for all $i \in \{1..k\}$
9          break
10     else Set all $\delta_i^2$ that are smaller than $\gamma$ to 0, and all remaining to $\lceil \frac{\delta_i^2}{\alpha} \rceil$
11 return $M_{best}$

Move3($S$, $M_{current}$, $k$, $N_d$, $\alpha$, $\gamma$)
1  $M_{best} \leftarrow \emptyset$, $S_{best} \leftarrow S$
2  for $i$ in $1..k$ do
3      $M_3 \leftarrow M_{current}$
4      while true do
5          $\Delta_{i,3} \leftarrow \Delta_{i,3} - \delta_i^3$
           ▷ Distribute decrease of $\delta_i^3$ across other pairwise shifts
6          $\Delta_{j,3} \leftarrow \Delta_{j,3} + \max\left(0, \lceil \frac{\delta_i^3}{k-1} \rceil - 1\right)$ for $j \in \{1..k\} \setminus \{i\}$
7          if manipulation $M_3$ is verified by SIM-STV then
8              $M_{best} \leftarrow M_3$, $S_{best} \leftarrow |M_3|$
9              break
10         else
11             if $\delta_i^3 \equiv 1$ then break
12             $\delta_i^3 \leftarrow \lceil \frac{\delta_i^3}{\alpha} \rceil$
13 return $M_{best}$
```

Fig. 3: Algorithms for move types one to three, where: S denotes the size of the best found manipulation in the current iteration of local search; $M_{current}$ is the best found manipulation at the start of the current iteration; k is the number of additional candidates we wish to elect from our desired coalition; N_d is the number of candidates from our desired coalition originally elected; and $\alpha \geq 2$, $\gamma \geq 2$ are predefined constants.

new ‘best found manipulation’. After applying $\textsc{Move}_3$, the step sizes associated with move types 1 and 2 are reset to their initial values.

After 9 iterations, we have reduced our overall manipulation size to 247 ballots with a $\textsc{Move}_3$. In the next iteration, we cannot reduce the size of this manipulation further and local search terminates. This process is repeated for different combinations of subsets in $\overline{\mathcal{W}}$ and $\overline{\mathcal{L}_d}$, returning the smallest found manipulation as our result. In this example, the smallest successful manipulation we can discover is 247 ballots.

3.2 Choosing a best combination of manipulations

Let $x_{i,k}$ denote a binary variable that takes on a value of 1 if we choose to apply a manipulation to election i that elects k additional candidates from our desired coalition d, and 0 otherwise. Let $|M_{i,k}|$ denote the size of the manipulation required to elect k additional candidates from d in election i. We formulate an integer program (IP), modelled as a knapsack problem, to select the best combination of manipulations that, when applied to their respective elections, realise a combined n-seat shift toward our coalition. Our objective is to minimise the total number of ballot changes required across all selected manipulations. We use s to denote the number of seats available in election i.

$$minimise \sum_{i} \sum_{k=1}^{min(s,n)} |M_{i,k}|\, x_{i,k} \tag{3}$$

subject to:

$$\sum_{i} \sum_{k=1}^{min(s,n)} k\, x_{i,k} = n \tag{4}$$

The constraint in Eqn 4 restricts the total number of seats shifted to our coalition, across the set of individual STV elections i, to n. Where our local search method was unable to find a manipulation that elects k additional favoured candidates to an election i, we fix $x_{i,k} = 0$. As we shall see in Section 4, there are a number of situations in which a k-seat shifting manipulation is not possible in a given election.

Example 5. For $\mathcal{E}_1$, the best found manipulation to award $k = 1$ extra seats to our coalition $d = \{\text{C, D}\}$ is 111 ballots in size. Awarding $k = 2$ extra seats to d requires 247 ballot changes, across all ballot replacement methods. For $\mathcal{E}_2$, 15 ballot changes are required to elect $k = 1$ more members from d (for IW, BTL, and ATL), and 121 ballots for $k = 2$ (using BTL). In the latter case, using IW and ATL result in a manipulation of 122 ballots. For our small 6-seat Senate, the best manipulation we can find to shift $n = 2$ seats to our coalition is to shift 2-seats in $\mathcal{E}_2$, with a cost of 121 ballots.

4 Case Studies

We use the 2016 and 2019 Australian Federal Senate elections as case studies. We have partitioned the set of parties taking part in these elections into two groups: conservative; and progressive. The conservative group includes parties such as the Liberal Party,

Table 3: For each of the individual STV elections forming part of the 2016 and 2019 Australian Federal Senate elections, we report the number of: seats available; candidates standing; and formal votes cast. We additionally state the quota, and number of candidates elected from our desired 'progressive' coalition d, for each election.

	2016 Senate Election					**2019 Senate Election**				
Region	**Seats**	$\lvert\mathcal{C}\rvert$	**Formal Votes Cast**	**Quota**	**Elected from d**	**Seats**	$\lvert\mathcal{C}\rvert$	**Formal Votes Cast**	**Quota**	**Elected from d**
ACT	2	22	254,767	84,923	1	2	17	270,231	90,078	1
NT	2	19	102,027	34,010	1	2	18	105,027	35,010	1
SA	12	64	1,061,165	81,629	7	6	42	1,094,823	156,404	3
VIC	12	116	3,500,237	269,250	6	6	82	3,739,443	534,207	3
QLD	12	122	2,723,166	209,475	5	6	83	2,901,464	414,495	2
NSW	12	151	4,492,197	345,554	5	6	105	4,695,326	670,761	3
WA	12	79	1,366,182	105,091	4	6	67	1,446,623	206,661	3
TAS	12	58	339,159	26,090	7	6	44	351,988	50,285	3

the Nationals, and One Nation. The progressive group contains parties such as the Australian Labor Party and the Greens. The conservative coalition attained a 4-seat majority in 2016, and a 2-seat majority in 2019. Consequently, we use the progressive group as our desired coalition d in our experiments, and seek to find as small as possible a manipulation to award 4, and 2 respectively, additional seats to candidates in d in the 2016 and 2019 elections. All experiments have been run with parameters $\gamma = 2$ and $\alpha = 4$.

Table 3 reports the number of candidates standing, seats available, and formal (valid) votes cast in each of the individual STV elections forming part of these two Senate elections. In addition, we report the quota and number of candidates elected from d.

We report in Table 4 the sizes of the smallest manipulations our local search approach was able to find to shift $k = 1, 2$ seats toward our favoured candidates in coalition d, in each state and territory STV election in 2019. A '–' indicates that no manipulation was found to achieve a given shift of seats. In the ACT and NT, for example, only 2 seats are available for election. In each case, 1 candidate from d has been elected to a seat in the original outcome. We can only award 1 additional seat to candidates in d. We report the number of ballot shifts required to shift 1, and 2, seats toward our favoured candidates when using the BTL, ATL, and IW ballot replacement methods. Overall, the IW method leads to smaller manipulations. Recall that the IW approach replaces ballots that favour an undesired winner with a below the line vote that preferences a favoured loser first, and each of the original winners from our desired coalition subsequently.

Table 5 states the sizes of the smallest manipulations our local search approach could find to shift $k = 1..4$ seats toward our favoured coalition d, in each state and territory STV election in 2016. We use the IW method of replacing ballots for each election. As in 2019, we can only award 1 additional seat to candidates in d in the ACT and NT. In SA and TAS, we were unable to find a manipulation that awarded 4 additional seats to candidates in d. Both Tables 4 and 5 show that the degree of manipulation required to shift k seats to desired candidates increases significantly as k increases.

We apply the IP of Section 3.2 to the available manipulations for 2019, listed in Table 4. The coefficients of our objective are obtained from reported manipula-

Table 4: Smallest manipulations found to elect 1 to 2 additional members of a centre-left leaning coalition of parties, in each state/territory for the 2019 Australian Federal Sentate election. For each region, the election quota, and number of ballot changes required to realise the desired change, are stated for each method of forming new ballots. We additionally state the number of ballot changes as a percentage of formal votes cast.

1 additional seat to desired coalition

Region	**Quota**	**Ballot Shifts Required**					
ACT	90,078	BTL	**12,938** (4.8%)	ATL	**12,938** (4.8%)	IW	**12,938** (4.8%)
NT	35,010	BTL	**14,697** (14%)	ATL	14,922 (14.2%)	IW	**14,697** (14%)
SA	156,404	BTL	**50,535** (4.6%)	ATL	50,695 (4.6%)	IW	**50,535** (4.6%)
VIC	534,207	BTL	**126,906** (3.4%)	ATL	127,068 (3.4%)	IW	**126,906** (3.4%)
QLD	414,495	BTL	**56,913** (2%)	ATL	**56,913** (2%)	IW	58,605 (2%)
NSW	670,761	BTL	**296,472** (6.3%)	ATL	297,389 (6.3%)	IW	**296,472** (6.3%)
WA	206,661	BTL	**108,915** (7.5%)	ATL	**108,915** (7.5%)	IW	**108,915** (7.5%)
TAS	50,285	BTL	**19,824** (5.6%)	ATL	20,399 (5.8%)	IW	**19,824** (5.6%)

2 additional seats to desired coalition

Region	**Quota**	**Ballot Shifts Required**					
ACT	90,078	BTL	–	ATL	–	IW	–
NT	35,010	BTL	–	ATL	–	IW	–
SA	156,404	BTL	177,554 (16.2%)	ATL	177,730 (16.2%)	IW	**177,504** (16.2%)
VIC	534,207	BTL	559,035 (14.9%)	ATL	558,734 (14.9%)	IW	**558,521** (14.9%)
QLD	414,495	BTL	370,091 (12.8%)	ATL	370,046 (12.8%)	IW	**353,692** (12.8%)
NSW	670,761	BTL	835,217 (17.8%)	ATL	835,180 (17.8%)	IW	**832,314** (17.8%)
WA	206,661	BTL	**294,005** (20.3%)	ATL	**294,005** (20.3%)	IW	**294,005** (20.3%)
TAS	50,285	BTL	**53,617** (15.2%)	ATL	55,275 (15.7%)	IW	54,295 (15.4%)

tion sizes. We use the smallest manipulation discovered for each state and territory, across the different ballot replacement methods. For example, $|M_{ACT,1}| = 12,938$ and $|M_{SA,2}| = 177,504$. The least cost way to shift 2 seats to our desired coalition is to shift 1 seat in ACT, with 12,938 ballot manipulations (4.8% of cast formal votes), and 1 seat in the NT, with 14,697 ballot changes (14% of the cast formal votes). The nature of the elected Senate in 2019 could have significantly changed with a change in 27,635 votes. If we chose to minimise the percentage of formal ballots cast in any manipulated election, in place of the total number of ballots changed, we would instead shift 1 seat in QLD (56,913 manipulations, 2% of formal votes) and 1 seat in VIC (126,906, 3.4% of formal votes). The total manipulation size is significantly larger, at 183,819 ballots, yet it involves a smaller percentage of changes (a maximum of 3.4%).

In 2016, the least cost combination of manipulations to shift 4 seats to our coalition d are: a 1 seat shift in SA, with 1,772 manipulations (0.17%); a 1 seat shift in the NT, with 11,245 manipulations (11%); a 1 seat shift in NSW, with 12,313 manipulations (0.27%); and a 1 seat shift in WA, with 14,678 manipulations (1.1%). The nature of the elected Senate in 2016 could have significantly changed with a change in 40,008 votes.

Table 5: Smallest manipulations found to elect 1 to 4 additional members of a centre-left leaning coalition of parties, in each state/territory for the 2016 Australian Federal Senate election. For each region, the election quota, and number of ballot changes required to realise the desired change (using the IW method of forming new ballots) are stated. We additionally state the number of ballot changes as a percentage of formal votes cast.

Region	**Quota**	**Ballot Shifts Required**			
		1 seat	2 seats	3 seats	4 seats
ACT	84,923	18,836 (7.4%)	–	–	–
NT	34,010	11,245 (11%)	–	–	–
SA	81,629	1,772 (0.17%)	57,607 (5.4%)	132,576 (12.5%)	–
VIC	269,250	45,046 (1.3%)	181,770 (5.2%)	420,880 (12%)	682,348 (19.5%)
QLD	209,475	49,829 (1.8%)	139,196 (5.1%)	354,475 (13%)	573,357 (21.1%)
NSW	345,554	12,313 (0.27%)	149,046 (3.3%)	386336 (8.6%)	731,280 (16.3%)
WA	105,091	14,678 (1.1%)	79,308 (5.8%)	161,963 (11.9%)	280,426 (20.5%)
TAS	26,090	21,692 (6.4%)	43,383 (12.8%)	65,698 (19.4%)	–

5 Conclusion

We have presented a local search heuristic that, in combination with an integer program, finds an upper bound on the number of ballot changes required to change the nature of the majority in an elected Senate. We have found that in two case study elections, a relatively small, but not insignificant, number of cast ballots need to be changed to shift the majority from a conservative coalition of parties to one that is more progressive. This number is a lot larger, however, than the number of ballot changes required to realise any change in outcome. For example, the 2016 results in Tasmania were very close, requiring only 71 ballot changes to change the result [1].

References

1. Blom, M., Conway, A., Stuckey, P.J., Teague, V.J.: Did that lost ballot box cost me a seat? computing manipulations of stv elections. In: IAAI (2020)
2. Blom, M., Stuckey, P.J., Teague, V.: Computing the margin of victory in preferential parliamentary elections. In: EVote-ID. LNCS, vol. 11143, pp. 1–16 (2018)
3. Blom, M., Stuckey, P.J., Teague, V.: Towards computing the margin of victory in STV elections. INFORMS Journal of Computing **31**(4), 636–653 (2019)
4. Conway, A.: Australian federal senate simulator. `https://github.com/SiliconEconometrics/PublicService` (2019), accessed: August 2019
5. Miragliotta, N.L.: Little differences, big effects: An example of the importance of choice of method for transferring surplus votes in PR-STV voting systems. Representation **41**, 15–24 (2004)
6. Australian federal senate election rules. `www.austlii.edu.au/au/legis/cth/consol_act/cea1918233/s273.html` (2019), accessed: Aug 2019
7. Weeks, L.: Tolerable Chance or Undesirable Arbitrariness? Distributing Surplus Votes Under PR-STV. Parliamentary Affairs **64**, 530–551 (2011)
8. Xia, L.: Computing the margin of victory for various voting rules. In: Proceedings of the 13th ACM Conference on Electronic Commerce. pp. 982–999. EC '12, ACM, New York, NY, USA (2012)