Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence

Eliminating the Weakest Link: Making Manipulation Intractable?

Jessica Davies
University of Toronto
Toronto, Canada
jdavies@cs.toronto.edu

Nina Narodytska
NICTA and UNSW
Sydney, Australia
ninan@cse.unsw.edu.au

Toby Walsh
NICTA and UNSW
Sydney, Australia
toby.walsh@nicta.com.au

Abstract

Successive elimination of candidates is often a route to making manipulation intractable to compute. We prove that eliminating candidates does not necessarily increase the computational complexity of manipulation. However, for many voting rules used in practice, the computational complexity increases. For example, it is already known that it is NP-hard to compute how a single voter can manipulate the result of single transferable voting(the elimination version of plurality voting). We show here that it is NP-hard to compute how a single voter can manipulate the result of the elimination version of veto voting, of the closely related Coombs' rule, and of the elimination versions of a general class of scoring rules.

Introduction

Voting is a general mechanism for combining the preferences together of multiple agents. Voting is, however, not without its problems. One such problem is that agents may vote strategically, mis-reporting their true preferences in order to improve the outcome for them. For instance, in each round of a popular TV game show, players vote on which other player to eliminate. The host, Anne Robinson then tells the unlucky player, "You are the *weakest* link, goodbye!". Players have an interesting strategic decision to make. On the one hand, they should vote to eliminate weak players (as weak players will tend to reduce the size of the jackpot). On the other hand, they should vote to eliminate strong players (as the overall winner takes the final jackpot and everyone else walks away empty-handed). Similarly, when the International Olympic Committee (IOC) meets to select a site for the next Olympics, there is an election in which the weakest city is successively eliminated. Strategic voting often appears to take place. For example, in the vote for the site of the 2012 Olympics, New York had 19 votes in the first round but only 16 in the second as several IOC members switched allegiances. In this paper, we study the computational resistance of elimination style voting rules to such strategic voting.

Results like those of Gibbard-Satterthwaite prove that most voting rules are manipulable. That is, it may pay for agents to mis-report their preferences. One potentially appealing escape from manipulation is computational complexity (Bartholdi, Tovey, and Trick 1989). Whilst manipulations might exist, what if they are too hard to find? Unfortunately, only a few voting rules used in practice are known to be NP-hard to manipulate with unweighted votes and a single manipulator: single transferable voting (STV) (Bartholdi and Orlin 1991), a variant of the Copeland rule (Bartholdi, Tovey, and Trick 1989), ranked pairs (Xia et al. 2009), and Nanson's and Baldwin's rules (Narodytska, Walsh, and Xia 2011). A feature common to a majority of these rules is that they successively eliminate candidates. We therefore explore in more detail whether such elimination style voting makes manipulation intractable to compute.

Background

We consider a general class of voting rules. A *scoring rule* over m candidates is defined by a vector $(s_1, \ldots, s_m)$ where for each vote ranking a candidate in position i, the candidate receives a score of s_i. The candidate with the highest total score wins. Plurality has the scoring vector $(1, 0, \ldots, 0)$, Borda has $(m-1, m-2, \ldots, 0)$, whilst veto has $(1, \ldots, 1, 0)$. For a scoring rule X with scoring vector $(s_1, \ldots, s_m)$, the *adjoint* of X, written X^* has the scoring vector $(s_1 - s_m, \ldots, s_1 - s_2, s_1 - s_1)$. For example, the adjoint of plurality is veto. Note that $(X^*)^* = X$.

Elimination versions of scoring rules can vary along a number of dimensions:

Base rule: STV is an elimination version of plurality voting, whilst Nanson's and Baldwin's rules are elimination versions of Borda voting. We consider here elimination versions of other scoring rules like veto voting.

Elimination criteria: Different criteria can be used to eliminate candidates. For instance, in STV and Baldwin's rule, we succcessively eliminate the last placed candidate. On the other hand, in Nanson's rule, we eliminate all candidates with less than the average Borda score.

Stopping criteria: Do we stop when all but one candidate has been eliminated, or as soon as one candidate has a majority of first placed votes? For example, Coombs' rule is an elimination version of veto voting which stops when one candidates has a majority.

Voting: Do agents vote just once, or in each round? For example, in STV voting, agents vote only once. On the other

hand, when selecting Olympic venues, IOC members can cast a new vote in each round. We shall show that this increases the opportunity for manipulation.

Given a voting rule X, $eliminate(X)$ is the rule that successively eliminates the candidate placed in last place by X. For a scoring rule X, $divide(X)$ is the rule that successively eliminates those candidates with the mean or smaller score. For non-scoring rules X, $divide(X)$ is the rule that successively eliminates candidates ranked by X in the bottom half. Finally, $sequential(X)$ is the voting rule which runs a sequence of elections using X to eliminate the last placed candidate from each successive election. In each round, voters can change their vote according to which candidates remain.

Example 1 *STV is $eliminate(plurality)$. Note that $eliminate(STV)$ is STV itself. Baldwin's rule is $eliminate(Borda)$. Nanson's rule is $divide(Borda)$. Exhaustive ballot is $sequential(plurality)$. The IOC uses $sequential(plurality)$ to select Olympic sites. The FIFA executive committee uses the same rule to select the location of the World Cup. The TV game shows, "Survivor" and "The Weakest Link" both use $sequential(veto)$ to eliminate players up to the final round. Finally, Coombs' rule is related to $eliminate(veto)$. Coombs' rule successively eliminates the candidate in last place in the most votes until there is a candidate with a majority of first place votes.*

Elimination style voting rules satisfy several desirable axiomatic properties. For example, consider Condorcet consistency, the property that a voting rule elects the candidate that beats all others in pairwise comparisons when such a candidate exists. Whilst the Borda rule is not Condorcet consistent, elimination versions of Borda voting like Nanson's and Baldwin's rule are Condorcet consistent. On the other hand, elimination rounds can also destroy a deriable axiomatic property. In particular, consider monotonicity, the property that raising the position of the winner in some ballots does not change the winner. Whilst Borda voting is monotonic, elimination style voting rules like STV, Nanson's and Baldwin's are not monotonic. The loss of monotonicity is one of the significant trade-offs involved in obtaining a voting rule that is, as we shall see, somewhat more resistant to manipulation.

Manipulation

Successively eliminating candidates can increase the complexity of computing a manipulation. For example, computing a manipulation of plurality is polynomial, but of $eliminate(plurality)$ is NP-hard (Bartholdi and Orlin 1991). Similarly, computing a manipulation of Borda by one manipulator is polynomial, but of $eliminate(Borda)$ and $divide(Borda)$ is NP-hard (Narodytska, Walsh, and Xia 2011). Elkind and Lipmaa (2005) conjectured that many elimination style voting rules will be intractable to manipulate. They argue that "*[such elimination style] protocols provide the most faithful manipulation-resistant approximation to the underlying protocols, which makes them compelling alternatives to the original protocols*".

We might wonder if elimination always increases tie computational complexity. The following result demonstrates that it does not always make computing a manipulation intractable.

Theorem 1 *There exists a non-dictatorial voting rule X for which computing a manipulation of X, $eliminate(X)$ and $divide(X)$ are polynomial.*

Proof: Consider the rule which orders candidates alphabetically unless there is unanimity when it returns the unanimous order. ■

Indeed there are even (admittedly artificial) voting rules where successively eliminating candidates reduces the computational complexity of computing a manipulation.

Theorem 2 *There exists a voting rule X for which computing a manipulation of X is NP-hard but of $eliminate(X)$ and of $divide(X)$ are polynomial.*

Proof: Let candidates be integers in $[0, m]$. X is a rule whose result decides a 1-in-3-SAT problem on positive clauses over m variables. A vote which starts with 0 is interpreted as a positive clause by taking the candidates in 2nd to 4th place as its literals. Any other vote is interpreted as a truth assignment: those candidates appearing before 0 are interpreted as true literals, and those after as false. With 2 candidates, X returns the majority winner. With 3 or more candidates, X returns 0 as winner if one of the votes represents a truth assignment which satisfies exactly 1 in 3 literals in each clause represented by a vote, otherwise 1 is winner. Other candidates are returned in numerical order. Computing a manipulation of X is NP-hard. However, computing a manipulation of $eliminate(X)$ or $divide(X)$ is polynomial. 0 and 1 always enter the final round, and the overall winner is simply the majority winner between 0 and 1. ■

Eliminate(veto)

Adding elimination rounds to plurality makes finding a manipulation intractable. Veto is essentially the opposite rule to plurality. This is reflected in the alternate name for veto of "anti-plurality". Computing a manipulation of veto is polynomial. We just veto the current winner until our chosen candidate wins. An interesting case to consider then is $eliminate(veto)$. With weighted votes, Coleman and Teague (2007) have proved that computing a manipulation of $eliminate(veto)$ is polynomial when the number of candidates is bounded[1]. They left "*the difficult[y] of WCM [weighted coalition manipulation] on Coombs for unlimited candidates as an open question*". We resolve this open question. Computing a manipulation of $eliminate(veto)$ and of the closely related Coombs' rule is NP-hard even with *unweighted* votes and an unbounded number of candidates.

Theorem 3 *Deciding if a single manipulator can make a candidate win for $eliminate(veto)$ is NP-complete.*

Proof: (Sketch, the full proof can be found online in a technical report). The proof is inspired by ideas from (Bartholdi and Orlin 1991). We reduce from the 3-COVER problem. We are given a set $S = \{d_1, \ldots, d_n\}$ with $|S| = n$ and

[1]Note that Coleman and Teague call the voting rule studied in their paper Coombs' rule but it is, in fact, $eliminate(veto)$.

#		# votes	Type of votes
			Block P_1
			'preferred candidate' and 'items'
1.		$X-1$	$p \prec \mathbf{g_p} \prec$ .. $\prec \mathbf{g'_p}$
2.		$X-f_4$	$d_0 \prec \mathbf{g_{d_0}} \prec$ .. $p \prec \mathbf{g'_{d_0}}$
3.	$i \in [1,n]$	$X-3$	$d_i \prec \mathbf{g_{d_i}} \prec$ .. $p \prec \mathbf{g'_{d_i}}$
			'First losers' and 'second line'
4.	$i \in [1,m]$	$X-6$	$b_i \prec \mathbf{g_{b_i}} \prec$ .. $p \prec \mathbf{g'_{b_i}}$
5.	$j \in S_i$	2	$b_i \prec d_j \prec \mathbf{g_{b_i d_j}} \prec$ .. $p \prec \mathbf{g'_{b_i d_j}}$
6.	$i \in [1,m]$	$X-2$	$\bar{b}_i \prec \mathbf{g_{\bar{b}_i}} \prec$ .. $p \prec \mathbf{g'_{\bar{b}_i}}$
7.		2	$\bar{b}_i \prec d_0 \prec \mathbf{g_{\bar{b}_i d_0}} \prec$ .. $p \prec \mathbf{g'_{\bar{b}_i d_0}}$
8.	$i \in [1,m)$	$X-f_{123}$	$a_i \prec \mathbf{g_{a_i}} \prec$ .. $p \prec \mathbf{g'_{a_i}}$
		$X-f_{12}$	$a_m \prec \mathbf{g_{a_m}} \prec$ .. $p \prec \mathbf{g'_{a_m}}$
9.	$i \in [1,m]$	f_1	$a_i \prec b_i \prec \mathbf{g_{a_i b_i}} \prec$ .. $p \prec \mathbf{g'_{a_i b_i}}$
10.		f_2	$a_i \prec \bar{a}_i \prec p_i \prec \mathbf{g_{a_i \bar{a}_i p_i}}$ .. $p \prec \mathbf{g'_{a_i \bar{a}_i p_i}}$
11.	$i \in [1,m)$	f_3	$a_i \prec \bar{a}_i \prec a_{i+1} \prec \mathbf{g_{a_i \bar{a}_i a_{i+1}}}$ .. $p \prec \mathbf{g'_{a_i \bar{a}_i a_{i+1}}}$
12.		f_3	$a_1 \prec \mathbf{g_{a_1}} \prec$ .. $p \prec \mathbf{g'_{a_1}}$
13.	$i \in [1,m)$	$X-f_{123}$	$\bar{a}_i \prec \mathbf{g_{\bar{a}_i}} \prec$ .. $p \prec \mathbf{g'_{\bar{a}_i}}$
		$X-f_{12}$	$\bar{a}_m \prec \mathbf{g_{\bar{a}_m}} \prec$ .. $p \prec \mathbf{g'_{\bar{a}_m}}$
14.	$i \in [1,m]$	f_1	$\bar{a}_i \prec \bar{b}_i \prec \mathbf{g_{\bar{a}_i \bar{b}_i}}$ .. $p \prec \mathbf{g'_{\bar{a}_i \bar{b}_i}}$
15.		f_2	$\bar{a}_i \prec a_i \prec p_i \prec \mathbf{g_{\bar{a}_i a_i p_i}}$ .. $p \prec \mathbf{g'_{\bar{a}_i a_i p_i}}$
16.	$i \in [1,m)$	f_3	$\bar{a}_i \prec a_i \prec \bar{a}_{i+1} \prec \mathbf{g_{\bar{a}_i a_i \bar{a}_{i+1}}}$ .. $p \prec \mathbf{g'_{\bar{a}_i a_i \bar{a}_{i+1}}}$
17.		f_3	$\bar{a}_1 \prec \mathbf{g_{\bar{a}_1}} \prec$ .. $p \prec \mathbf{g'_{\bar{a}_1}}$
			'Pumps'
18.	$i \in [1,m)$ $j \in (i,m]$	$2f_2$	$p_i \prec a_j \prec \mathbf{g_{p_i a_j}} \prec$ .. $p \prec \mathbf{g'_{p_i a_j}}$
19.		$2f_2$	$p_i \prec \bar{a}_j \prec \mathbf{g_{p_i \bar{a}_j}} \prec$ .. $p \prec \mathbf{g'_{p_i \bar{a}_j}}$
20.		$2f_2$	$p_i \prec b_j \prec \mathbf{g_{p_i b_j}} \prec$ .. $p \prec \mathbf{g'_{p_i b_j}}$
21.		$2f_2$	$p_i \prec \bar{b}_j \prec \mathbf{g_{p_i \bar{b}_j}} \prec$ .. $p \prec \mathbf{g'_{p_i \bar{b}_j}}$
22.		$2f_2$	$p_i \prec p_j \prec \mathbf{g_{p_i p_j}} \prec$ .. $p \prec \mathbf{g'_{p_i p_j}}$
23.		$2f_2$	$p_i \prec p \prec \mathbf{g_{p_i p}} \prec$ .. $\prec \mathbf{g'_{p_i p}}$
24.	$i \in [1,m]$	$2f_2$	$p_i \prec d_0 \prec \mathbf{g_{p_i d_0}} \prec$ .. $p \prec \mathbf{g'_{p_i d_0}}$
25.	$k \in [1,n]$	$2f_2$	$p_i \prec d_k \prec \mathbf{g_{p_i d_k}} \prec$ .. $p \prec \mathbf{g'_{p_i d_k}}$
26.		$2f_2$	$p_i \prec s_1 \prec \mathbf{g_{p_i s_1}} \prec$ .. $p \prec \mathbf{g'_{p_i s_1}}$
27.		f_1	$p_m \prec s_1 \prec \mathbf{g_{p_m s_1}} \prec$ .. $p \prec \mathbf{g'_{p_m s_1}}$
28.		$2f_2$	$p_i \prec s_2 \prec \mathbf{g_{p_i s_2}} \prec$ .. $p \prec \mathbf{g'_{p_i s_2}}$
			'Switches'
29.		$4f_2+f_1$	$s_1 \prec p \prec \mathbf{g_{s_1 p}} \prec$ .. $\prec \mathbf{g'_{s_1 p}}$
30.		$4f_2+f_1$	$s_1 \prec d_0 \prec \mathbf{g_{s_1 d_0}} \prec$ .. $p \prec \mathbf{g'_{s_1 d_0}}$
31.	$k \in [1,n]$	$4f_2+f_1$	$s_1 \prec d_k \prec \mathbf{g_{s_1 d_k}} \prec$ .. $p \prec \mathbf{g'_{s_1 d_k}}$
32.		$4f_2+f_1$	$s_1 \prec s_2 \prec \mathbf{g_{s_1 d_k}} \prec$ .. $p \prec \mathbf{g'_{s_1 d_k}}$
33.		1	$d_0 \prec s_2 \prec \mathbf{g_{d_0 s_2}} \prec$ .. $p \prec \mathbf{g'_{d_0 s_2}}$
34.	$k \in [1,n]$	1	$d_k \prec s_2 \prec \mathbf{g_{d_k s_2}} \prec$ .. $p \prec \mathbf{g'_{d_k s_2}}$
35.		$X-n-3$	$s_2 \prec g_{d_0} \prec g_{d_1}$ .. $\prec p$
			Block P_2
36.	$i \in [1,m]$	$X-\sigma^1_{P_1}(p_i)$	$p_i \prec \mathbf{g_{p_i}} \prec$ .. $p \prec \mathbf{g'_{p_i}}$
37.		$X-\sigma^1_{P_1}(s_1)$	$s_1 \prec \mathbf{g_{s_1}} \prec$ .. $p \prec \mathbf{g'_{s_1}}$

Table 1: The constructed election.

subsets $S_1, S_2, \ldots, S_m \subset S$ with $|S_i| = 3$ for $i \in [1,m]$. We ask if there exists an index set I with $|I| = n/3$ and $\bigcup_{i \in I} S_i = S$. This set of S_i is called a cover for S. We create an *eliminate(veto)* election such that a manipulator can make a given candidate win iff there exists a cover for S.

The set of all candidates is C where $|C| = c$, and consists of 7 groups: 'preferred candidate' p; 'items' $\{d_1, \ldots, d_n\}$ and an extra 'item' d_0; 'first losers' $\{a_1, \bar{a}_1 \ldots, a_m, \bar{a}_m\}$; 'second line' $\{b_1, \bar{b}_1, \ldots, b_m, \bar{b}_m\}$; 'pumps' $\{p_1, \ldots, p_m\}$; 'switches' s_1 and s_2; 'garbage collectors' $g_{d_0}, \ldots, g'_{s_1}$ (see Table 1 for the complete list of garbage collectors). The elimination has 4 phases: (1) cover selection, (2),(3) cover verification (4) garbage collection.

Let $\sigma^k(c')$, $c' \in C$ be the number of last place votes for candidate c' at round k. We call this the veto-score of c'. First, we explain the candidates. The candidates 'first losers', 'second line' and 'pumps' form a gadget to select a cover. There are $5m$ candidates of these types in total, logically partitioned into elimination groups $\langle a_i, \bar{a}_i, b_i, \bar{b}_i, p_i \rangle$, $i \in [1,m]$. The construction makes sure that 4 out of 5 elements of ith group are eliminated consecutively during 4 rounds starting at round $4i+1$, $i \in [0,m)$. Moreover, $\{a_i, \bar{a}_i, p_i\}$, $i \in [1,m]$ must be eliminated letting one of the $\{b_i, \bar{b}_i\}$ reach round $4m$. Eliminated candidates b_i determine selection sets S_i. The 'pump' p_i increases the veto-scores of all candidates except 'garbage collectors' and i running candidates from groups j, $j \leq i$. This allows us to remember i choices of the manipulator encoded in these i running candidates from $2i$ candidates in $\{b_j, \bar{b}_j\}$, $j = 1, .., i$. The 'items' candidates encode the set of items. The'switches' check the cover. The first 'switch' s_1 separates elimination of $p, d_0, \ldots, d_n, s_2$ from the elimination of other candidates. The second 'switch' s_2 is the *most* dangerous candidate that can be eliminated iff a valid cover is selected during the first $4m$ rounds. 'Garbage collectors' $g_{d_0}, \ldots, g_{s_1}$ control the veto-scores of non-garbage candidates. 'Garbage collectors' $g'_{d_0}, \ldots, g'_{s_1}$ prevent p from having a majority (which is needed later on to prove Theorem 6).

We partition votes into two sets: P_1 and P_2. Table 1 shows the votes. in *reverse* preference order. We refer to sets of votes in each line of the table by the number in the third column. For convenience, we introduce a new garbage candidate in each set of votes. Unspecified candidates are ordered in the same arbitrary order, starting with $g_{d_0}, \ldots, g_{d_n}$ in all votes. P_1 is the main construction. Lines 1–3 set up initial veto-scores for the preferred candidate and 'items'. Lines 4–22 encode the 1st phase. The important point to observe is how 'pumps' work (lines 18–28). The candidate p_i is eliminated last in its group and increases the veto-score of all other running candidates by a constant $2f_2$ except running candidates in $\cup_{j=1}^{i}\{b_j, \bar{b}_j\}$ and 'garbage' candidates. This allows m running candidates selected from $\cup_{j=1}^{m}\{b_j, \bar{b}_j\}$ to reach the 4th phase. Lines 23–28 make sure that veto-score of 'items' and 'switches', that are not eliminated during the 1st phase, grow the same way as veto-scores of a's, b's and p's. Line 27 is used to eliminate s_1 at round $4m+1$. Lines 29-34 are used to check a cover. In particular, lines 33–34 are used to count how many candidates d_i, $i \in [0,n]$ are eliminated by increasing veto-scores of s_2. Finally, line 35 is responsible for triggering the garbage collection procedure. P_2 ensures that all 'pumps' and the 'switch' s_1 have initial score of X, where X is sufficiently large number, e.g. $X > 16m^5$ and $\sigma^1_{P_1}(c')$ is the number of last place votes for candidate c' in the votes P_1 at the first round. The initial veto-score of a garbage collector equals 0 and stays less than X until the 4th phase. So we do not have to worry about the garbage collectors during the first three phases. We also define the following constants required to control elimination inside each group $\langle a_i, \bar{a}_i, b_i, \bar{b}_i, p_i \rangle$, $i \in [1,m]$. The f constants in Table 1 satisfy the following constraints: $f_{12} = f_1 + f_2$, $f_{123} = f_1 + f_2 + f_3$, $f_1 \geq f_2 + 2f_3 + 2$, $2f_2 \geq f_1 + 2$, $2f_2 \geq f_3 + 2$, $f_1 \geq f_3 + 2$, $f_i \geq 2m + 3$ for $i \in [1,3]$, and $f_4 = 2m - 2n/3 + 3$. For example, $f_1 = 11(2m+3)$,

$f_2 = 8(2m+3)$ and $f_3 = 3 + (2m+3)$. Overall, the construction ensures that initial veto-scores of all candidates a, b, p, s_1 equals X with an exception of a_1 and $\bar{a}_1$ that have $X+3$ veto-points. All the other candidates have veto-scores that are less than or equal to X. This forces the manipulator to make a choice between a_1 and $\bar{a}_1$ in the first round which triggers a selection of sets in a cover. We assume the tie-breaking rule: $s_2 \prec d_0 \prec p \prec d_1, \prec \ldots \prec d_n, \ldots$.

Phase 1. Cover selection. Rounds 1 **to** $4m$**.** The 1st phase eliminates m candidates, one from each pair $\{b_i, \bar{b}_i\}$, $i \in [1, m]$. If b_i is eliminated then the set S_i is selected in the cover. A manipulator will choose which candidate from each pair is eliminated. We claim the following holds in the first $4m$ rounds where $i \in [0, m-1]$:

▷ *Round* $4i{+}1$. The following invariant holds immediately before the $4i+1$st candidate is eliminated:

$$\sigma^{4i+1}(a_{i+1}) = \sigma^{4i+1}(\bar{a}_{i+1}) \geq \sigma^{4i+1}(c') + 3,$$

where $c' \in C \setminus \{a_1, \bar{a}_1, .., a_{i+1}, \bar{a}_{i+1}, b_1, \bar{b}_1, .., b_i, \bar{b}_i\}$. The manipulator can select which of a_{i+1} or $\bar{a}_{i+1}$ is eliminated.

The manipulator **cannot** change the outcome of the following three rounds.

▷ *Round* $4i+2$. b_{i+1} is eliminated at this round iff a_{i+1} is eliminated at the previous round. Similarly, $\bar{b}_{i+1}$ is eliminated iff $\bar{a}_{i+1}$ is eliminated at the previous round.

▷ *Round* $4i{+}3$. a_{i+1} is eliminated at this round iff $\bar{a}_{i+1}$ is eliminated at the $4i{+}1$st round. Similarly, $\bar{a}_{i+1}$ is eliminated at this round iff a_{i+1} is eliminated at the $4i+1$st round.

▷ *Round* $4i+4$. The candidate p_{i+1} is eliminated.

Hence, the manipulator select m candidates in $\cup_{i=1}^{m}\{b_i, \bar{b}_i\}$ to eliminate. The elimination of p_m at round $4m$ forces an increase of the veto-scores of $p, d_0, \ldots, d_n, s_1, s_2$ by $2f_2$ (lines 23–26,28) and an additional increase of f_1 in the veto-score of s_1 (line 27). This means s_1 is the next candidate to be eliminated.

Phase 2. Pump up of $p, s_2, d_0, \ldots, d_n$**. Round** $4m+1$**.** The elimination of s_1 increases the veto-scores of $p, s_2, d_0, \ldots, d_n$ by $4f_2 + f_1$.

Phase 3. Cover verification. Rounds $4m+2$ **to** $4m+2+(n{+}1)$**.** This phase ensures that p reaches the next phase iff the sets S_i that correspond to eliminated candidates b_i form a cover of $d_1, \ldots, d_n$ and there are exactly $n/3$ such candidates. Consider the candidates $p, s_2, d_0, \ldots, d_n$. We observe that at round $4m+2$:

$$\sigma^{4m+2}(p) = \sigma^{4m+2}(d_i) - 1 + 2 - y_i^{4m+2},$$

$$\sigma^{4m+2}(p) = \sigma^{4m+2}(d_0) - 1 + 2(m - n/3 + 1) - y_0^{4m+2},$$

$$\sigma^{4m+2}(p) = \sigma^{4m+2}(s_2) - 1 + (n+1) + 2,$$

where y_i^{4m+2}, $i \in [0, n]$ is the veto-score that candidate d_i gets during the first $4m+1$ rounds in addition to its initial veto-score, y_i^{4m+2} is even. As can be seen from these equations, s_2 can be eliminated before p iff s_2 gets $n+2$ extra veto-points. This is possible iff $d_0, \ldots, d_n$ are eliminated before s_2 so that s_2 gets $n+1$ veto-points from lines 33–34. Moreover, the manipulator must give an extra veto-point to s_2. Then, by the tie-breaking rule, s_2 is eliminated before p. Consider how to eliminate $d_0, \ldots, d_n$ before s_2 and p.

▷ *Candidates* d_i *for* $i = 1, \ldots, n$*:* Let d_k be the candidate with the highest value y_k^{4m+2}. If $y_k^{4m+2} \geq 2$ then d_k is eliminated. This only increases the veto-score of s_2 by 1 and does not affect the veto-scores of other running candidates. Suppose that there exists k such that $y_k^{4m+2} = 0$. In this case p has 1 veto-point extra compared to d_k. Moreover, the manipulator cannot save p from elimination due to the tie-breaking rule. Hence, $y_i^{4m+2} \geq 2$ for $i \in [1, n]$. This means that sets S_j that correspond to candidates b_j that are eliminated during the first phase cover all values. Next we show that exactly $n/3$ of b_j's are eliminated.

▷ *The candidate* d_0*:* This candidate has $2(m - n/3) + 1$ veto-points less than the veto-score of p. Hence, during the first phase d_0 needs to get at least $2(m - n/3)$ extra veto-points. This means that $m - n/3$ of the candidates $\bar{b}_j$ have to be eliminated during the first phase. Hence exactly $n/3$ b_j's can be eliminated during the first phase. Finally, the manipulator gives one extra veto-point to d_0 and, by the tie-breaking rule, d_0 is eliminated. Hence, s_2 is eliminated after d_is, and p reaches the next round.

Phase 4. Garbage collection. Rounds $4m+2+(n+1)+1$ **to** c**.** This phase ensures that p wins if it is not already eliminated. p is either the last, first or second candidate in all remaining votes at this round. Hence, its veto-score does not change until the penultimate round. The elimination of d_0, and then s_2, increases the veto-score of a candidate g_{d_0} by $2X - (n+1) - 2(m - n/3) - 5$ (lines 2 and 35), which triggers elimination of other running candidates up to round $c-2$. When only 2 candidates remain, p must win.

The reverse direction is trivial. Given a cover I, we construct the vote of a manipulator in the following way. If i is in cover, we put a_i at position $c - 2i$ and $\bar{a}_i$ at position $c - 2i - 1$. Otherwise, we invert their positions. Then we put d_0, s_2. Finally, we make p the most preferred candidate, and put the remaining candidates in an arbitrary order. ■

Coombs' Rule

Coombs' rule is a variant of $eliminate(veto)$ with the stopping criteria that a winner is declared when one candidate has a majority of first placed votes (instead of when one candidate remains). Although this is a small change, it can have a large impact on the result and on strategic voting. For instance, there are a family of elections where the number of manipulators required to achieve victory for a particular candidate is unbounded for $eliminate(veto)$ but bounded for Coombs', and vice versa.

Theorem 4 *There exists an election with* $n+3$ *candidates where a given candidate has already won with* $eliminate(veto)$ *but the number of manipulators with Coombs' rule is* $\Omega(n)$.

Proof: We have n votes: $(a, d_1, .., d_n, b, c)$, $(a, d_2, .., d_{n-1}, b, c)$, .., $(a, d_n, .., d_1, b, c)$. Note that positions $2 - (n+1)$ in these votes contain a cyclic permutation of candidates $d_1, \ldots, d_n$. Similarly, for the other two groups of n votes. We also have n votes: $(b, a, d_1, .., d_n, c)$, $(b, a, d_2, .., d_{n-1}, c)$, .., $(b, a, d_n, .., d_1, c)$. Finally we

have n votes: $(c, b, a, d_1, .., d_n)$, $(b, a, d_2, .., d_{n-1})$, .., $(c, b, a, d_n, .., d_1)$. The preferred candidate is a. As is common in the literature, ties are broken in favor of manipulators. For $eliminate(veto)$, c is eliminated in the first round and b in the second. a is now in first place in all votes so ultimately wins. For Coombs', c is eliminated in the first round. b is then in the first place in $2n$ votes and wins by the majority rule. There are two options for the manipulators. Either they add n votes to the elections to make sure that b does not have a majority after the first elimination round, or they prevent the elimination of the candidate c in the first round. With the exception of c, each candidate has has only one veto point. Therefore, the manipulators need at least $2n - 1$ votes to prevent the elimination of c. ■

Theorem 5 *There exists an election with $n + 2$ candidates which a single manipulator can manipulate with Coombs' rule but $eliminate(veto)$ requires $\Omega(n)$ manipulators.*

Proof: We have n votes: $(a, b, d_1, .., d_n)$, $(a, b, d_2, .., d_{n-1})$, .., $(a, b, d_n, .., d_1)$. Note that positions $2 - (n + 1)$ in these votes contain a cyclic permutation of candidates $d_1, \ldots, d_n$. Similarly, for the other group of n votes. We also have n votes: $(b, d_1, .., d_n, a)$, $(b, d_2, .., d_{n-1}, a)$, .., $(b, d_n, .., d_1, a)$. None of the candidates has a majority. For Coombs', if one manipulator puts a in first place then a wins. For $eliminate(veto)$, the manipulators must prevent the elimination of a in the first round. As candidates d_1 to d_n have only 1 veto point we need at least $n - 1$ manipulators to prevent the elimination of a. ■

Despite these differences between Coombs' rule and $eliminate(veto)$, it is intractable to compute a manipulation for Coombs' as it is with $eliminate(veto)$.

Theorem 6 *Deciding if a single manipulator can make a candidate win for the Coombs' rule is NP-complete.*

Proof: Follows from the proof of Theorem 3 as g'_i can be eliminated after a cover is verified. ■

Eliminate(scoring rule)

We next consider scoring rules in general. With weighted votes, Coleman and Teague (2007) have proved that manipulation by a coalition is NP-hard to compute for the elimination version of any scoring rule X that is not isomorphic to veto. With unweighted votes, we prove a general result that relates the computational complexity of manipulating a scoring rule and the elimination version of its adjoint.

Theorem 7 *Deciding whether k manipulators can make a candidate win for $eliminate(X)$ is NP-complete if it is NP-complete also for X^*.*

Proof: First, we argue that for votes V, k manipulators can make a candidate win with X^* iff for the reversed set of votes V^*, k manipulator can make a candidates come last with X. The proof is similar to Lemma 10 in (Coleman and Teague 2007). We simply reverse all the manipulating votes. Suppose V^* is an election over m candidates where $m \geq 3$, and the k manipulators want c_m to come last. Let U be $s_1(|V| + k + 1)$ copies of the following votes:

$$c_1 \succ c_2 \succ \ldots \succ c_{m-1} \succ c_m,$$
$$c_2 \succ c_3 \succ \ldots \succ c_m \succ c_1,$$
$$\vdots$$
$$c_m \succ c_1 \succ \ldots \succ c_{m-2} \succ c_{m-1}$$

Each candidate receives the same score in U irrespective of X. We argue that there is a manipulation making c_1 win in $V^* \cup U$ for $eliminate(X)$ if there is a manipulation making c_m last in V^* for X. By the same argument as in the proof of Theorem 13 in (Coleman and Teague 2007), if c_i is the first candidate eliminated in $V^* \cup U$, then no matter how the manipulators vote, the elimination order is $c_i, c_{i-1}, c_{i-2}, \ldots, c_{i+1}$ (where $c_{m+1} = c_1$) and c_{i+1} wins. Hence c_1 wins iff c_m is eliminated first. The manipulators can force c_m to be eliminated first in $V^* \cup U$ if they can force c_m to be last in V^* as the relative scores of the candidates are initially the same in V^* and in $V^* \cup U$. Hence manipulation of X^* reduces to manipulation of $eliminate(X)$. ■

Borda is NP-hard to manipulate with 2 manipulators (Betzler, Niedermeier, and Woeginger 2011; Davies et al. 2011). Since the adjoint of Borda is Borda itself, it follows from Theorem 7 that $eliminate(Borda)$, which is Baldwin's rule, is NP-hard to manipulate by 2 manipulators. This result is strengthened to NP-hard with just one manipulator in (Narodytska, Walsh, and Xia 2011). Note that the reverse of Theorem 7 does not hold. STV, which is $eliminate(plurality)$, is NP-hard to manipulate but $plurality$ is only polynomial to manipulate.

We next identify a large class of scoring rules which are intractable to manipulate. Given a fixed k, a *truncated scoring rule* ($score_t$) has a scoring vector $(s_1, \ldots, s_m)$ with $s_i = 0$ for all $i > k$. For example, plurality and k-approval voting are both truncated scoring rules. As a second example, the Heisman Trophy, which is awarded annually to the best player in collegiate football, uses the truncated scoring rule $(3, 2, 1, 0, \ldots, 0)$. As a third and final example, the Eurovision song contest uses the truncated scoring rule $(12, 10, 8, 7, 6, 5, 4, 3, 2, 1, 0, \ldots, 0)$. We now prove out next major results: computing a manipulation of $eliminate(score_t)$ or of $divide(score_t)$ is intractable. When candidates are eliminated, we suppose that the scoring vector is truncated to the first m positions where m is the number of candidates left after elimination.

Theorem 8 *Deciding if a single manipulator can make a candidate win for $eliminate(score_t)$ is NP-complete.*

Proof: (Sketch, the full proof can again be found online in a technical report). The proof is also inspired by ideas from (Bartholdi and Orlin 1991) and uses a reduction from 3-COVER. We block the first $k - 1$ positions in each vote with an additional set of $q(k - 1)$ dummy candidates, where the value q is computed taking into account the scoring vector. By this construction, only those scores at positions k to c, which are $(s_k, 0, \ldots, 0)$, determine the elimination order for the first $c - q(k-1) - 1$ rounds, where c is the total number of candidates. We thereby reduce our problem to one that resembles a multiple of the reduction used for STV. Only one non-dummy candidate reaches the $(c - q(k - 1) - 1)$th round. If the preferred candidate p

reaches this round then the remaining votes are such that p wins the election. Similar to the reduction used in the STV proof, this only happens if there is a 3-COVER. As we have a large number of additional dummy candidates, we can make sure that the individual score of each dummy candidate is greater than the score of any non-dummy candidate until the $(c - q(k-1) - 1)$th round and is smaller than the score of p at the $(c - q(k-1) - 1)$th round. ■

Theorem 9 *Deciding if a single manipulator can make a candidate win for $divide(score_t)$ is NP-complete.*

Proof: (Sketch, the full proof can again be found online in a technical report). The proof uses a reduction from the 3-COVER problem where $k = n/3$, n is the number of items. The first two rounds encode solving the 3-COVER problem and the remaining rounds are used to collect garbage. The main types of candidates are n 'items', m 'sets' and one 'preferred'. The rest of the candidates are dummy candidates that are used to control scores of non-dummy candidates and the average score. In the first round, we select k sets. Using a large number of dummy candidates we make sure that the score of 'sets' candidates equals the average score at the first round. Hence, manipulator can select k of them to pass to the second round. In the second round, we check that this forms a cover. If this is the case, all 'items' candidates in the covered set are eliminated. Otherwise, one of the 'items' candidates reaches the third round and wins the election. ■

Sequential Rules

When selecting the site of the next Olympics, IOC members can cast a new vote in each round. This increases the opportunity for manipulation. In fact, we can exhibit an election in which a manipulator can only change the result if the manipulator votes differently in each round.

Example 2 *Consider the following 21 votes:*

1: (a, h, p, ...),	*1: (c, a, h, p, ...),*	*1: (d, a, h, p, ...)*
3: (g, a, h, p, ...),	*2: (b, h, p, ...),*	*2: (e, b, h, p, ...)*
2: (f, b, h, p, ...),	*6: (h, p, ...),*	*5: (p, h, ...)*

The election uses $sequential(plurality)$, and the manipulator wants p to win. The tie-breaking rule is $p \prec g \prec c \prec d \prec a \prec e \prec f \prec b \prec h$.

We first argue that a single manipulator cannot make p win. Note that p cannot gain any points until h is eliminated. For p to win, the manipulator needs to give p one point so that it has 6 points (and beats h by the tie-breaking rule) and no other candidate receives more than 6 points. In order for h not to receive any more than the initial 6 points, a and b must not be eliminated. The manipulator must save a from elimination in the first round by voting for a. The first two rounds therefore eliminate c and d. Unfortunately, the manipulator cannot stop b being eliminated next. h now receives 2 more points and p cannot win the election. Therefore, a single manipulator cannot make p win.

On the other hand, if a single manipulator votes for a in the first two rounds, c and d are eliminated and a has 3 points and is safe for now. At this point, b is in danger with only 2 points. If the manipulator now votes for b, b is saved and both e and f are eliminated next. At this point, a and g are tied. If the manipulator now votes for a again, g is eliminated and the score of a increases to 6. At this point, all candidates except p have 6 points, and if the manipulator now votes for p, h is eliminated by tie-breaking and p wins the election. Hence, if the manipulator can change votes after each round, p can win.

In general, manipulating a sequential elimination election requires a strategy, which provides a manipulating response however the other agents vote. It is not hard to see that deciding if such a strategy exists is PSPACE-complete. In fact, strategic voting in a sequential elimination election invites a game theoretic analysis. We can view a sequential elimination election as a finite repeated sequential game. We could, for example, use backward induction to find the subgame perfect Nash equilibrium in which each agent makes the best strategic move in each round.

Other Related Work

Bag, Sabourian and Winter (2009) proved that many sequential elimination rules including $sequential(plurality)$ elect candidates in the top cycle (and are hence Condorcet consistent) supposing strategic voting. Contizer and Sandholm (2003) studied the impact on the tractability of manipulation of adding an initial round of the Cup rule to a voting rule. This initial round eliminates half the candidates and makes manipulation NP-hard to compute for several voting rule including plurality and Borda. Elkind and Lipmaa (2005) extended this idea to a general technique for combining two voting rules. The first voting rule is run for some number of rounds to eliminate some of the candidates, before the second voting rule is applied to the candidates that remain. They proved that many such combinations of voting rules are NP-hard to manipulate. However, they did not consider the veto or truncated scoring rules at the centre of our study here. They also considered the *closed protocol*, where a rule is combined with itself. In many cases, the closed protocol of X is $eliminate(X)$. They conjectured that such closed protocols will often be NP-hard to manipulate.

Conclusions

We have provided more evidence that successively eliminating candidates is often a route to making manipulation intractable to compute. In general, eliminating candidates does not necessarily increase the computational complexity of manipulation. Indeed, we exhibited an artificial voting rule where the computational complexity actually decreases. However, for many voting rules used in practice, the computational complexity increases. For example, it was known that it is NP-hard to compute how a single voter can manipulate the result of STV (the elimination verison of plurality voting), and Nanson's and Baldwin's rule (elimination versions of Borda voting). In this paper, we showed that it is NP-hard to compute how a single voter can manipulate the result of the elimination version of veto voting, of the closely related Coombs' rule, and of the elimination versions of a general class of truncated scoring rules. On the other hand, we showed that permitting voters to re-vote between elimination rounds can increase the opportunity for manipulation.

What general lessons can be learnt from these studies? First, elimination style voting does indeed appear to provide some computational resistance to manipulation. Second, these results have involved worst case complexity notions like NP-hardness. We need to treat these with care as there is theoretical evidence (for instance, (Xia and Conitzer 2008a; Friedgut, Kalai, and Nisan 2008; Xia and Conitzer 2008b)), as well as practical experience which suggests that elimination style rules like STV (Coleman and Teague 2007; Walsh 2010), as well as voting rules like veto (Walsh 2009) can be easy to manipulate on average. Third, if voters can re-vote between elimination rounds, new opportunities for manipulation arise. It would be interesting therefore to consider both game-theoretic and computational aspects of such strategic voting. For example, what are the possible equilibria and how difficult are they to compute? Fourth, manipulation is closely connected to questions about possible winners given uncertainty about the votes (Pini et al. 2007; Walsh 2007) and to elicitation (Walsh 2008). It would therefore be interesting to consider reasoning about possible winners and preference elicitation for elimination style voting rules.

Acknowledgments

The authors are supported by the Australian Government's Department of Broadband, Communications and the Digital Economy, the Australian Research Council and the Asian Office of Aerospace Research and Development through grants AOARD-104123 and 124056.

References

Bag, P.; Sabourian, H.; and Winter, E. 2009. Multi-stage voting, sequential elimination and condorcet consistency. *Journal of Economic Theory* 144(3):1278 – 1299.

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

Bartholdi, J.; Tovey, C.; and Trick, M. 1989. The computational difficulty of manipulating an election. *Social Choice and Welfare* 6(3):227–241.

Betzler, N.; Niedermeier, R.; and Woeginger, G. 2011. Unweighted coalitional manipulation under the Borda rule is NP-hard. In Walsh, T., ed., *Proc. of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011)*.

Coleman, T., and Teague, V. 2007. On the complexity of manipulating elections. In Gudmundsson, J., and Jay, B., eds., *Proc. of the 13th Australasian Symposium on Theory of Computing (CATS '07)*, 25–33. Australian Computer Society, Inc.

Conitzer, V., and Sandholm, T. 2003. Universal voting protocol tweaks to make manipulation hard. In *Proc. of 18th International Joint Conference on Artificial Intelligence (IJCAI 2003)*, 781–788.

Davies, J.; Katsirelos, G.; Narodytska, N.; and Walsh, T. 2011. Complexity of and algorithms for Borda manipulation. In Burgard, W., and Roth, D., eds., *Proc. of the Twenty-Fifth AAAI Conference on Artificial Intelligence (AAAI 2011)*. AAAI Press.

Elkind, E., and Lipmaa, H. 2005. Hybrid voting protocols and hardness of manipulation. In Deng, X., and Du, D.-Z., eds., *Proc. of 16th International Symposium on Algorithms and Computation (ISAAC 2005)*, vol. 3827 of *LNCS*, 206–215. Springer.

Friedgut, E.; Kalai, G.; and Nisan, N. 2008. Elections can be manipulated often. In *Proc. of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2008)*, 243–249. IEEE Computer Society Press.

Narodytska, N.; Walsh, T.; and Xia, L. 2011. Manipulation of Nanson's and Baldwin's rules. In Burgard, W., and Roth, D., eds., *Proc. of the Twenty-Fifth AAAI Conference on Artificial Intelligence (AAAI 2011)*. AAAI Press.

Pini, M.; Rossi, F.; Venable, B.; and Walsh, T. 2007. Incompleteness and incomparability in preference aggregation. In Veloso, M. M., ed., *Proc. of the 20th International Joint Conference on Artificial Intelligence (IJCAI-2007)*, 1464–1469.

Walsh, T. 2007. Uncertainty in preference elicitation and aggregation. In *Proc. of the 22nd National Conference on AI*, 3–8. Association for Advancement of Artificial Intelligence.

Walsh, T. 2008. Complexity of terminating preference elicitation. In Padgham, L.; Parkes, D. C.; Müller, J. P.; and Parsons, S., eds., *7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, 967–974. IFAAMAS.

Walsh, T. 2009. Where are the really hard manipulation problems? The phase transition in manipulating the veto rule. In Boutilier, C., ed., *Proc. of the 21st International Joint Conference on Artificial Intelligence (IJCAI-2009)*, 324–329.

Walsh, T. 2010. An empirical study of the manipulability of single transferable voting. In Coelho, H.; Studer, R.; and Wooldridge, M., eds., *Proc. of the 19th European Conference on Artificial Intelligence (ECAI-2010)*, 257–262. IOS Press.

Xia, L., and Conitzer, V. 2008a. Generalized scoring rules and the frequency of coalitional manipulability. In Fortnow, L.; Riedl, J.; and Sandholm, T., eds., *Proc. of the 9th ACM conference on Electronic Commerce (EC' 08)*, 109–118. ACM.

Xia, L., and Conitzer, V. 2008b. A sufficient condition for voting rules to be frequently manipulable. In Fortnow, L.; Riedl, J.; and Sandholm, T., eds., *Proc. of the 9th ACM conference on Electronic Commerce (EC' 08)*, 99–108. ACM.

Xia, L.; Zuckerman, M.; Procaccia, A.; Conitzer, V.; and Rosenschein, J. 2009. Complexity of unweighted coalitional manipulation under some common voting rules. In Boutilier, C., ed., *Proc. of the 21st International Joint Conference on Artificial Intelligence (IJCAI 2009)*, 348–353.