Complexity of and Algorithms for the Manipulation of Borda, Nanson and Baldwin's Voting Rules

Jessica Davies[1], George Katsirelos[2], Nina Narodytska[3]
Toby Walsh[3] and Lirong Xia[4]

[1] University of Toronto, Toronto, Canada, jdavies@cs.toronto.edu
[2] LRI, Université Paris Sud 11, Paris, France, gkatsi@gmail.com
[3] NICTA and UNSW, Sydney, Australia
{nina.narodytska,toby.walsh}@nicta.com.au
[4] SEAS, Harvard University, Cambridge, MA 02138, USA, lxia@seas.harvard.edu

October 4, 2011

Abstract

We investigate manipulation of the Borda voting rule, as well as two elimination style voting rules based on it, Nanson's and Baldwin's voting rules. We argue that these voting rules have a number of desirable computational properties. For Borda voting, we prove that it is NP-hard for a coalition of two manipulators to compute a manipulation. This resolves one of the last open problems in the computational complexity of manipulating common voting rules. For Baldwin's and Nanson's rules, we prove that manipulation is computationally more difficult as it is NP-hard for a *single* manipulator to compute a manipulation. In addition, for Baldwin's or Nanson's rules and weighted votes, we prove that it is NP-hard for a coalition of manipulators to compute a manipulation with a small number of candidates.

Because of this NP-hardness, we treat computing a manipulation as an approximation problem where we try to minimize the number of manipulators. We propose several new approximation methods. Experiments show that these methods significantly outperform the previous best known approximation method for the Borda rule. Our results suggest that, whilst computing a manipulation of the Borda rule is NP-hard, computational complexity may provide only a weak barrier against manipulation in practice. In contrast to the Borda rule, our experiments with Baldwin's and Nanson's rules demonstrate that both of them are often difficult to manipulate in practice. These results suggest that elimination style voting rules deserve further study.

1 Introduction

Voting is a simple mechanism to combine preferences in multi-agent systems. Unfortunately, results like those of Gibbrard-Sattertwhaite prove that most voting rules are manipulable. That is, it may pay for agents to mis-report their preferences. One appealing escape from manipulation is computational complexity [4]. Whilst a manipulation may exist, perhaps it is computationally too difficult to find? Unfortunately, few voting rules in common use are NP-hard to manipulate without the addition of weights to votes. The small set of voting rules that are known to be NP-hard to manipulate with unweighted votes includes single transferable voting, 2nd order Copeland, ranked pairs (all with a single manipulator), and maximin (with two manipulators) [3, 4, 23].

Borda is one of the few commonly used voting rules where the computational complexity of unweighted manipulation remains open. Xia, Conitzer and Procaccia (2010) observe that:

> *"The exact complexity of the problem [manipulation by a coalition with unweighted votes] is now known with respect to almost all of the prominent voting rules, with the glaring exception of Borda"*

It is known that computing a manipulation of Borda is NP-hard when votes are weighted [11], and polynomial when votes are unweighted and there is just a single manipulator [4]. With a coalition of manipulators and unweighted votes, it has been conjectured that the problem is NP-hard [26]. One of our most important contributions is to close this question. We prove that computing a manipulation of Borda with just two manipulators is NP-hard[1].

We also study two voting rules that are closely related to Borda's rule: Nanson's and Baldwin's rules. These are elimination-style rules that use Borda scoring to eliminate candidates in each round. There are several reasons to consider these two rules. Firstly, they have features that might appeal to the two opposing camps that support Borda and Condorcet. In particular, unlike the Borda rule itself, both Nanson's and Baldwin's rules are Condorcet consistent as they elect the candidate who beats all others in pairwise elections. Secondly, statistical analysis suggests that, whilst the Borda rule is often vulnerable to manipulation [9], Nanson's rule is particularly resistant [15]. We might expect Baldwin to be similarly resistant. Finally, the two rules have been used in real elections in the Universitiy of Melbourne (between 1926 and 1982), the University of Adelaide (since 1968), and the State of Michigan (in the 1920s).

We will show both theoretically and empirically that finding manipulations for Baldwin's and Nanson's rules is more computationally difficult than the Borda rule. In particular, we prove that unweighted coallitional manipulation of these two rules is NP-hard even with a *single* manipulator. We also consider the weighted coalition manipulation problem and show that Baldwin's and Nanson's rules are NP-hard to manipulate with 3 and 4 candidates, respectively

Our theoretical results suggest that all three rules are computationally difficult to manipulate. NP-hardness is only a worst case notion so we also investigate whether these rules are resistant to manipulation in practice. We propose several polynomial

[1] This result was proven independently in [5].

time approximation algorithms for Borda, Baldwin's and Nanson's rules that try to minimize the number of manipulators required to ensure a particular result. Our experiments suggest that the Borda rule is often easy to manipulate in practice. The heuristics that we study are able to find an optimal manipulation in 99% of the cases. Interestingly, these approximations were significantly less effective for Baldwin's and Nanson's rules. These results confirm our theoretical results that elimination-style voting rules are more computationally resistant to coalition manipulation.

The rest of the paper is organized as follows. In Section 2 we give the background material. Section 3 focuses on unweighted coalition manipulations and Section 4 on the weighted case. Section 5 presents four approximation algorithms that aim to find the minimum number of manipulators and Section 6 presents experimental evaluation of these algorithms. In Section 7 we present two interesting connections between unweighted coalition manipulation of the Borda rule and two problems from discrete mathematics. We mention other related work in Section 8 and conclude in Section 9.

2 Background

Let $\mathcal{C} = \{c_1, \ldots, c_m\}$ be the set of *candidates* (or *alternatives*). A linear order on $\mathcal{C}$ is a transitive, antisymmetric, and total relation on $\mathcal{C}$. The set of all linear orders on $\mathcal{C}$ is denoted by $L(\mathcal{C})$. An n-voter profile P on $\mathcal{C}$ consists of n linear orders on $\mathcal{C}$. That is, $P = (V_1, \ldots, V_n)$, where for every $j \leq n$, $V_j \in L(\mathcal{C})$. The set of all n-profiles is denoted by $\mathscr{F}_n$. We let m denote the number of candidates. A (deterministic) *voting rule* r is a function that maps any profile on $\mathcal{C}$ to a unique winning candidate, that is, $r : \mathscr{F}_1 \cup \mathscr{F}_2 \cup \ldots \rightarrow \mathcal{C}$. In this paper, if not mentioned otherwise, ties are broken in the fixed order $c_1 \succ c_2 \succ \cdots \succ c_m$.

(Positional) scoring rules are a common type of voting rule. Each positional scoring rule is identified by a *scoring vector* $\vec{s}_m = (\vec{s}_m(1), \ldots, \vec{s}_m(m))$ of m integers, for any vote $V \in L(\mathcal{C})$ and any candidate $c \in \mathcal{C}$, let $\vec{s}_m(c, V) = \vec{s}_m(j)$, where j is the rank of c in V. For any profile $P = (V_1, \ldots, V_n)$, let $\vec{s}_m(c, P) = \sum_{j=1}^{n} \vec{s}_m(c, V_j)$. The rule selects $c \in \mathcal{C}$ such that the total score $\vec{s}_m(c, P)$ is maximized. We assume scores are integers and decreasing. When voters are weighted (that is, each voter is associated with a positive real number as the weight), a positional scoring rule selects the candidate that maximizes the weighted total score.

The *unweighted (coalitional) manipulation* problem is defined as follows. An instance is a tuple (r, P^{NM}, c, M), where r is a voting rule, P^{NM} is the non-manipulators' profile, c is the candidate preferred by the manipulators, and M is the set of manipulators. We are asked whether there exists a profile P^M for the manipulators such that $r(P^{NM} \cup P^M) = c$. The *weighted (coalitional) manipulation* is defined similarly, where the weights of the voters (both non-manipulators and manipulators) are also given as inputs. As is common in the literature, we break ties in favour of the coalition of the manipulators where appropriate.

Borda rule The *Borda* rule, proposed by Jean-Charles de Borda in 1770 is the positional scoring rule that corresponds to the scoring vector $(m-1, m-2, \ldots, 0)$. We write $s(a, P)$ for the Borda score given to candidate a from the profile of votes P, and

$s(a)$ where P is obvious from the context. The Borda rule is used in parliamentary elections in Slovenia and, in modified form, in elections within the Pacific Island states of Kiribati and Nauru. The Borda rule or modifications of it are also used by many organizations and competitions including the Robocup autonomous robot soccer competition, the X.Org Foundation, the Eurovision song contest, and in the election of the Most Valuable Player in major league baseball. The Borda rule has many good features. For instance, it is monotonic as increasing the score for a candidate only helps them win. It never elects the Condorcet loser (a candidate that loses to all others in a majority of head to head elections). However, it may not elect the Condorcet winner (a candidate that beats all others in a majority of head to head elections).

Given a set of votes and n manipulators, we define the *gap* of candidate i as $g(i) = s(c) + n(m-1) - s(i)$. For c to win in a Borda election, we need additional votes that give to candidate i score which is less than or equal to $g(i)$. Note that if $g(i)$ is negative for any i, then c cannot win and Borda manipulation is impossible.

Nanson's and Baldwin's rules Nanson's and Baldwin's rules are derived from the Borda rule. Nanson's rule eliminates all those candidates with less than the average Borda score [19]. The rule is then repeated with the reduced set of candidates until there is a single candidate left. A closely related voting rule proposed by Baldwin successively eliminates the candidate with the lowest Borda score[2] until one candidate remains [2]. The two rules are closely related, and indeed are sometimes confused. One of the most appealing properties of Nanson's and Baldwin's rules is that they are Condorcet consistent, i.e. they elect the Condorcet winner. This follows from the fact that the Borda score of the Condorcet winner is never below the average Borda score. Both rules possess several other desirable properties including the majority criterion and the Condorcet loser criterion. There are also properties which distinguish them apart. For instance, Nanson's rule satisfies reversal symmetry (i.e. if there is a unique winner and all voters reverse their votes then the winner changes) but Baldwin's rule does not.

3 Unweighted coalition manipulation

We start by considering the computational complexity of manipulating all these rules with unweighted votes. We prove that the coalitional manipulation problem is NP-complete under the Borda rule with two manipulators. This settles an open problem in computational social choice. We show that Baldwin's and Nanson's rules are NP-complete to manipulate even with a single manipulator.

Computational intractability with a single manipulator is known only for a small number of other voting rules including the second order Copeland rule [4], STV [3] and ranked pairs [23]. In contrast, when there are two or more manipulators, unweighted coalitional manipulation is hard for some other common voting rules [13, 14, 24], as well as Borda's rule, as we show in section 3.1. Our results therefore significantly increase the size of the set of voting rules used in practice that are known to be NP-hard

[2]If multiple candidates have the lowest score, then we use a tie-breaking mechanism to eliminate one of them.

to manipulate with a single manipulator. This also contrasts to Borda where computing a manipulation with a single manipulator is polynomial [4]. Hence, adding elimination rounds to Borda to get Nanson's or Baldwin's rules increases the computational complexity of computing a manipulation with one manipulator from polynomial to NP-hard.

3.1 Borda rule

In this section we present one of our main results. We prove that computing a manipulation of the Borda rule is NP-hard. Our NP-hardness proof uses a reduction from a specialized permutation problem that is strongly NP-complete [25].

Definition 1 (Permutation Sum)**.** *Given n integers $X_1 \leq \ldots \leq X_n$ where $\sum_{i=1}^{n} X_i = n(n+1)$, do there exist two permutations σ and π of 1 to n such that $\sigma(i)+\pi(i) = X_i$?*

We first give a technical lemma that shows we can construct votes for the non-manipulators with a given target sum.

Lemma 1. *Given integers X_1 to X_m there exist votes over $m+1$ candidates and a constant C such that the final score of candidate i is $X_i + C$ for $1 \leq i \leq m$ and for candidate $m+1$ is y where $y \leq C$.*

Proof: This proof is inspired by Theorem 5.1 [24]. We show how to increase the score of a candidate by 1 more than the other candidates except for the last candidate whose score increases by 1 less. For instance, suppose we wish to increase the score of candidate 1 by 1 more than candidates 2 to m and by 2 more than candidate $m+1$. Consider the following pair of votes:

$$1 \succ m+1 \succ 2 \succ \ldots \succ m-1 \succ m$$
$$m \succ m-1 \succ \ldots \succ 2 \succ 1 \succ m+1$$

The score of candidate 1 increases by $m+1$, of candidates 2 to m by m, and of candidate $m+1$ by $m-1$. By repeated construction of such votes, we can achieve the desired result. □

Theorem 1. *Unweighted coalition manipulation for the Borda rule is NP-complete with two manipulators.*

Proof: Clearly the problem is in NP. A polynomial witness is simply the votes that the manipulators cast which make the chosen candidate win.

To show NP-hardness, we reduce a Permutation Sum problem over n integers, X_1 to X_n, to a manipulation problem with $n+3$ candidates. By Lemma 1, we can construct an election in which the non-manipulators cast votes to give the score vector:

$$\langle C, 2(n+2) - X_1 + C, \ldots, 2(n+2) - X_n + C, 2(n+2) + C, y \rangle$$

where C is a constant and $y \leq C$. We claim that two manipulators can make candidate 1 win such an election iff the Permutation Sum problem has a solution.

($\Rightarrow$) Suppose we have two permutations σ and π of 1 to n with $\sigma(i) + \pi(i) = X_i$. We construct two manipulating votes which have the scores:

$$\langle n+2, \sigma(1), \ldots, \sigma(n), 0, n+1 \rangle$$

$$\langle n+2, \pi(1), \ldots, \pi(n), 0, n+1 \rangle$$

Since $\sigma(i) + \pi(i) = X_i$, these give a total score vector:

$$\langle 2(n+2)+C, 2(n+2)+C, \ldots, 2(n+2)+C, 2(n+1)+y \rangle$$

As $y \leq C$ and we tie-break in favour of the manipulators, candidate 1 wins.

($\Leftarrow$) Suppose we have a successful manipulation. To ensure candidate 1 beats candidate $n+2$, both manipulators must put candidate 1 in first place. Similarly, both manipulators must put candidate $n+2$ in last place otherwise candidate $n+2$ will beat our preferred candidate. Hence the final score of candidate 1 is $2(n+2)+C$. The gap between the final score of candidate 1 and the current score of candidate $i+1$ (where $1 \leq i \leq n$) is X_i. The sum of these gaps is $n(n+1)$. If any candidate 2 to $n+1$ gets a score of $n+1$ then candidate 1 will be beaten. Hence, the two scores of $n+1$ have to go to the least dangerous candidate which is candidate $n+3$.

The votes of the manipulators are thus of the form:

$$\langle n+2, \sigma(1), \ldots, \sigma(n), 0, n+1 \rangle$$

$$\langle n+2, \pi(1), \ldots, \pi(n), 0, n+1 \rangle$$

Where σ and π are two permutations of 1 to n. To ensure candidate 1 beats candidate j for $j \in [1, n]$, we must have:

$$2(n+2) - X_j + C + \sigma(j) + \pi(j) \leq 2(n+2) + C$$

Rearranging this gives:

$$\sigma(j) + \pi(j) \leq X_j$$

Since $\sum_{i=1}^{n} X_i = n(n+1)$ and $\sum_{i=1}^{n} \sigma(i) = \sum_{i=1}^{n} \pi(i) = \frac{n(n+1)}{2}$, there can be no slack in any of these inequalities. Hence,

$$\sigma(j) + \pi(j) = X_j$$

That is, we have a solution of the Permutation Sum problem. □

As is usual in the literature, we have assumed that the manipulators have complete knowledge about the scores from the votes of the non-manipulators. The argument often put forward for such an assumption is that partial or probabilistic information about the votes of the non-manipulators will add to the computational complexity of computing a manipulation.

3.2 Baldwin's rule

Our result is proved by reductions from the EXACT 3-COVER (X3C) problem. An X3C instance contains two sets: $\mathcal{V} = \{v_1, \ldots, v_q\}$ and $\mathcal{S} = \{S_1, \ldots, S_t\}$, where $t \geq 2$ and for all $j \leq t$, $|S_j| = 3$ and $S_j \subseteq \mathcal{V}$. We are asked whether there exists a subset $\mathcal{S}'$ of $\mathcal{S}$ such that each element in $\mathcal{V}$ is in exactly one of the 3-sets in $\mathcal{S}'$.

Theorem 2. *Unweighted coalition manipulation for Baldwin's rule is NP-complete with one manipulator.*

Proof: We give a reduction from X3C. Given an X3C instance $\mathcal{V} = \{v_1, \ldots, v_q\}, \mathcal{S} = \{S_1, \ldots, S_t\}$, we let the set of candidates be $\mathcal{C} = \{c, d, b\} \cup \mathcal{V} \cup \mathcal{A}$, where c is the candidate that the manipulator wants to make the winner, $\mathcal{A} = \{a_1, \ldots, a_t\}$, and d and b are additional candidates. Members of $\mathcal{A}$ correspond to the 3-sets in $\mathcal{S}$. Let $m = |\mathcal{C}| = q + t + 3$.

The profile P contains two parts: P_1, which is used to control the changes in the score differences between candidates, after a set of candidates are removed, and P_2, which is used to balance the score differences between the candidates. We define the votes $W_{(u,v)} = \{u \succ v \succ \textit{Others}, \text{rev}(\textit{Others}) \succ u \succ v\}$ where *Others* is a total order in which the candidates in $\mathcal{C} \setminus \{u, v\}$ are in a pre-defined lexicographic order, and rev(*Others*) is the reverse.

We make the following observations on $W_{(c_1,c_2)}$. For any set of candidates $\mathcal{C}' \subseteq \mathcal{C}$ and any pair of candidates $e_1, e_2 \in \mathcal{C} \setminus \mathcal{C}'$,

$$\begin{aligned} & s(e_1, W_{(c_1,c_2)}|_{\mathcal{C}\setminus\mathcal{C}'}) - s(e_2, W_{(c_1,c_2)}|_{\mathcal{C}\setminus\mathcal{C}'}) \\ = & s(e_1, W_{(c_1,c_2)}) - s(e_2, W_{(c_1,c_2)}) + \begin{cases} 1 & \text{if } e_1 = c_2 \text{ and } c_1 \in \mathcal{C}' \\ -1 & \text{if } e_1 = c_1 \text{ and } c_2 \in \mathcal{C}' \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

Here $W_{(c_1,c_2)}|_{\mathcal{C}\setminus\mathcal{C}'}$ is the pair of votes obtained from W by removing all candidates in $\mathcal{C}'$. In words, the formula states that after $\mathcal{C}'$ is removed, the score difference between e_1 and e_2 is increased by 1 if and only if $e_1 = c_2$ and c_1 is removed; it is decreased by 1 if and only if $e_1 = c_1$ and c_2 is removed; for any other cases, the score difference does not change. It follows that for any $e \in \mathcal{C} \setminus \{c_1, c_2\}$, $s(c_1, W_{(c_1,c_2)}) - s(e, W_{(c_1,c_2)}) = 1$ and $s(c_2, W_{(c_1,c_2)}) - s(e, W_{(c_1,c_2)}) = -1$.

We next show how to use $W_{(c_1,c_2)}$ to construct the first part of the profile P_1. We recall that $m = |\mathcal{C}| = q + t + 3$. P_1 is composed of the following votes:

- for each $j \leq t$ and each $v_i \in S_j$, there are $2m$ copies of $W_{(v_i,a_j)}$;
- for each $i \leq q$, there are m copies of $W_{(b,v_i)}$;
- there are $m(t + 6)$ copies of $W_{(b,c)}$.

It is not hard to verify that $s(b, P_1) - s(c, P_1) \geq mq$, and for any $c' \in \mathcal{V} \cup \mathcal{A}$, $s(c', P_1) - s(c, P_1) \geq 2m$. P_2 is composed of the following votes:

- for each $i \leq q$, there are $s(v_i, P_1) - s(c, P_1) - m$ copies of $W_{(d,v_i)}$;

- for each $j \leq t$, there are $s(a_j, P_1) - s(c, P_1) - 1$ copies of $W_{(d,a_j)}$;
- there are $s(b, P_1) - s(c, P_1) - mq$ copies of $W_{(d,b)}$.

Let $P = P_1 \cup P_2$. We make the following observations on the Borda scores of the candidates in P.

- For any $i \leq q$, $s(v_i, P) - s(c, P) = m$.
- For any $j \leq t$, $s(a_j, P) - s(c, P) = 1$.
- $s(b, P) - s(c, P) = mq$.

Suppose the X3C instance has a solution, denoted by (after reordering the sets in $\mathcal{S}$) $S_1, \ldots, S_{q/3}$. Then, we let the manipulator vote for the following vote:

$$c \succ d \succ a_{q/3+1} \succ \cdots \succ a_t \succ b \succ \mathcal{V} \succ a_1 \succ \cdots \succ a_{q/3}$$

In the first $4q/3$ rounds, all candidates in $\mathcal{V}$ and $\{a_1, \ldots, a_{q/3}\}$ drop out. Then b drops out. In the following $t - q/3$ rounds the candidates in $\{a_{q/3+1}, \ldots, a_t\}$ drop out. Finally, d loses to c in their pairwise election, which means that c is the winner.

Suppose the manipulator can cast a vote to make c the winner. We first note that d must be eliminated in the final round since its score is higher than c in all previous rounds. In the round when b is eliminated, the score of b should be no more than the score of c. We note that $s(b, P) - s(c, P) = mq$ and the score difference can only be reduced by the manipulator ranking b below c, and by eliminating $v_1, \ldots, v_q$ before b. However, by ranking b below c, the score difference is reduced by no more than $m - 1$. Therefore, before b drops out, all candidates in $\mathcal{V}$ must have already dropped out. We note that for any $v_i \in \mathcal{V}$, $s(v_i, P) - s(c, P) = m$. Therefore, for each $v_i \in \mathcal{V}$, there exists a_j with $v_i \in S_j$ who is removed before v_i. For any such a_j, none of the candidates in S_j can drop out before a_j (otherwise the score of a_j cannot be less than c before b drops out), and in the next three rounds the candidates in S_j drop out. It follows that the set of candidates in $\mathcal{A}$ that drop out before any candidate in $\mathcal{V}$ corresponds to an exact cover of $\mathcal{V}$.

Therefore, the unweighted coalitional manipulation problem under Baldwin's rule is NP-complete, even when there is only one manipulator. □

3.3 Nanson's rule

We again use the EXACT 3-COVER (X3C) problem to reduce to a coalition manipulation problem under Nanson's rule.

Theorem 3. *Unweighted coalition manipulation for Nanson's rule is NP-complete with one manipulator.*

Proof: The idea of the proof is similar with the proof of Theorem 2. We prove the NP-completeness via a reduction from X3C. Given an X3C instance $\mathcal{V} = \{v_1, \ldots, v_q\}$, $\mathcal{S} = \{S_1, \ldots, S_t\}$, we let the set of alternatives be $\mathcal{C} = \{c, d, b_1, b_2\} \cup \mathcal{V} \cup \mathcal{A}$, where c is the alternative that the manipulation want to make to win, $\mathcal{V} = \{v_1, \ldots, v_q\}$, $\mathcal{A} =$

$\{a_1, \ldots, a_t\}$, and d, b_1, and b_2 are auxiliary alternatives. Without loss of generality, both q and t are even, and $t \geq 3q$. We will use the votes $W_{(c_1,c_2)}$ defined in the proof of Theorem 2 to construct the profile. For any $\mathcal{C}' \subsetneq \mathcal{C}$, we make the following observations on $W_{(c_1,c_2)}$.

$$s(c', W_{(c_1,c_2)}|_{\mathcal{C}\setminus\mathcal{C}'}) - |\mathcal{C} \setminus \mathcal{C}'| + 1 = \begin{cases} 1 & \text{if } c' = c_1 \text{ and } c_2 \notin \mathcal{C}' \\ -1 & \text{if } c' = c_2 \text{ and } c_1 \notin \mathcal{C}' \\ 0 & \text{otherwise} \end{cases}$$

We note that $|\mathcal{C} \setminus \mathcal{C}'| - 1$ is the average score of the alternatives in $W_{(c_1,c_2)}|_{\mathcal{C}\setminus\mathcal{C}'}$.

Let $m = q + t + 4$. Again, the profile has two parts: P_1, which is used to control the score differences between the alternatives and the average score, and P_2, which is used to set the initial scores. P_1 consists in the following votes:

- for every $j \leq t$ there are $7m/2 - q/3$ copies of $W_{(a_j,b_1)}$, and for every $v_i \in S_j$, there are m copies of $W_{(v_i,a_j)}$;
- for every $i \leq q$, there are m copy of $W_{(v_i,c)}$;
- there are mq copies of $W_{(c,b_1)}$ and $mq + t(7m/2 - q/3)$ copies of $W_{(b_1,b_2)}$.

That is, after removing a_j, for all $v_i \in S_j$, the relative score of v_i w.r.t. the average score will be reduced by m; after removing v_i, the relative score of c w.r.t. the average score will increase by m; after removing b_2, the relative score of b_1 w.r.t. the average score will be reduced by $mq + t(7m/2 - q/3)$; and after removing b_1, the relative score of c w.r.t. the average score will be reduced by mq, and the relative score of all a_j w.r.t. the average score will be reduced by $7m/2 - q/3$.

The second part of the profile P_2 consists in the following votes: for any $i \leq q$, there are $m \cdot \Delta(v_i)$ copies of $W_{(d,v_i)}$, where $\Delta(v_i)$ is the number of times that v_i is covered by the 3-sets in $\mathcal{S}$.

We make the following observations on $P = P_1 \cup P_2$.

- $s(c, P) - (m-1)|P|/2 = 0$, $s(d, P) - (m-1)|P|/2 = m(t + \sum_{i=1}^{q} \Delta(v_i))$, $s(b_1, P) - (m-1)|P|/2 = 0$, $s(b_2, P) - (m-1)|P|/2 = -(mq + t(7m/2 - q/3))$.
- For any $j \leq t$, $s(a_j, P) - (m-1)|P|/2 = m/2 - q/3$.
- For any $i \leq q$, $s(v_i, P) - (m-1)|P|/2 = m$.

Suppose the X3C instance has a solution, denoted by (without loss of generality) $\{S_1, \ldots, S_{q/3}\}$. Then, we let the manipulator vote for the following vote:

$$c \succ b_1 \succ b_2 \succ d \succ a_{q/3+1} \succ \cdots \succ a_t \succ \mathcal{V} \succ a_1 \succ \cdots \succ a_{q/3}$$

Then, in the first round, b_2 and $a_1, \ldots, a_{q/3}$ drop out; in the second round all alternatives in $\mathcal{V}$ drop out; in the following rounds $a_{q/3+1}, \ldots, a_t$ and d drop out, which means that c is the winner.

Next, we show that if the manipulator can cast a vote to make c to win, then there exists a solution to the X3C instance. It is easy to see that d will only drop out in the

final round. In the first round b_2 definitely drops out. This in turn makes the score of b_1 below the average score by $mq + t(3m/2 - q/3)$, which means that b_1 will definitely drop out in the second round. Then, in the third round the if any v_i still remains, then the score of c will be strictly lower than the average score. Therefore, all alternatives in $\mathcal{V}$ must drop out in the first and the second round. In fact, v_i can only drop out in the second round, and only when there exists j such that $v_i \in S_j$ and the alternative a_j drops out in the first round. Moreover, no more than $q/3$ alternatives in $\mathcal{A}$ can possibly drop out in the first round (since the only way for them to drop out is to rank them among the bottom $q/3$ positions). Therefore, in order for c to survive after the third round, the bottom $q/3$ alternatives in the manipulator's vote must be among $\mathcal{A}$ and they must correspond to an exact cover of $\mathcal{V}$, which means that the X3C instance has a solution.

Therefore, the unweighted coalitional manipulation problem under Nanson's rule is NP-complete, even when there is only one manipulator. □

4 Weighted coalition manipulation

In this section we show that weighted coalition manipulation under Baldwin's and Nanson's rules are NP-complete. Baldwin's rule is arguably more difficult computationally to manipulate since Nanson's rule is polynomial to manipulate with 3 candidates and requires at least 4 candidates to be NP-hard but Baldwin's is NP-hard already with 3 candidates. It follows that computing a manipulation is NP-hard for both rules when votes are unweighted, the number of candidates is small and there is uncertainty about how agents have voted in the form of a probability distribution [11]. Note that the coalition manipulation problem for Borda with weighted votes is NP-hard for 3 or more candidates [11]. Thus, somewhat surprisingly, adding an elimination round to Borda, which gives us Nanson's rule, decreases the computational complexity of computing a manipulation with 3 manipulators from NP-hard to polynomial.

Note that Coleman and Teague in Theorem 13 of [10] provide a NP-hardness result for the weighted coalition manipulation problem for voting rules like Baldwin's that eliminate candidates one by one Our result for Baldwin's rule is different in two aspects. First, Coleman and Teague use a different tie-breaking rule. They break ties against the manipulator whilst, as is more common in the literature, we suppose ties are broken in their favour. The second difference is that Coleman and Teague's do not specify a precise bound on the number of candidates, while we present a proof that weighted coalition manipulation under Baldwin's rule is NP-hard for just 3 candidates.

4.1 Baldwin's rule

Our result is proved by reduction from the PARTITION problem. A PARTITION instance contains a set of integers: $A = \{k_1, \ldots, k_n\}$ such that $\sum_{i=1}^{n} k_i = 2K$. The decision problem is to determine whether there exists a partition of these numbers into two sets the elements in each of which sum to K.

Theorem 4. *With Baldwin's rule and weighted votes, the coalition manipulation problem is NP-hard for 3 or more candidates.*

Proof: We reduce from a PARTITION problem. We construct a coalition manipulation problem with 3 candidates (a, b and p) in which the manipulators want to make p win. We suppose the non-manipulators have voted as follows: $11K$ for $a \succ b \succ p$, $5K$ for $a \succ p \succ b$, $14K$ for $b \succ p \succ a$, $2K-1$ for $b \succ a \succ p$, $5K$ for $p \succ a \succ b$ and $5K$ for $p \succ b \succ a$. At this point, $s(a) = 39K - 1$, $s(b) = 48K - 2$ and $s(p) = 39K$. For each integer k_i, we have a member of the manipulating coalition with weight $3k_i$.

Suppose there is a perfect partition. Let the manipulators corresponding to the integers in one half of the partition vote $a \succ p \succ b$, and the others vote $p \succ a \succ b$. The scores are now as follows: $s(a) = 48K - 1$, $s(b) = 48K - 2$, $s(p) = 48K$. Hence b will be eliminated. In the next round, p wins as $s(a) = 21K - 1$ but $s(p) = 27K$. Thus the manipulators can make p win if a perfect partition exists.

Conversely, suppose there is a manipulation in which p wins. Clearly, p cannot be eliminated in the first round. Suppose a is eliminated in the first round. Then the scores in the second round from the non-manipulators are: $s(b) = 27K - 1$, and $s(p) = 15K$. The manipulators cannot now prevent b from winning. Hence b must be eliminated in the first round. If any manipulator puts b above last place, b is unbeatable. Thus all the votes of the manipulators are $a \succ p \succ b$ or $p \succ a \succ b$. Consider the following partition constructed from any successful manipulation. In the first partition, we put all integer associated with weighted votes of the manipulators of the form $a \succ p \succ b$. In the second, we put all integers associated with weighted votes of the form $p \succ a \succ b$. Suppose the first partition sums up to $K - x$ and the second partition sums up to $K + x$. Then we have scores: $s(a) = 48K - 1 - 3x$, $s(b) = 48K - 2$ and $s(p) = 48K + 3x$. If $x \geq 1$ then a is eliminated. On the other hand, if $x \leq -1$ then p is eliminated. Hence $x = 0$ and we have a perfect partition. □

4.2 Nanson's rule

We show that coalition manipulation problem under Nanson's rule is NP-complete with 4 candidates. Removing one candidate decreases the computational complexity of computing a manipulation under Nanson's rule from NP-hard to polynomial.

Theorem 5. *With Nanson's rule and weighted votes, the coalition manipulation problem is NP-complete for just 4 candidates.*

Proof: We reduce from PARTITION. For any PARTITION instance, we construct a coalition manipulation problem with 4 candidates (a, b, c and p) where p is again the candidate that the manipulators wish to win. We suppose the non-manipulators have voted as follows: $2K + 1$ for each of $b \succ p \succ c \succ a$, $a \succ c \succ b \succ p$, $c \succ p \succ b \succ a$ and $a \succ b \succ c \succ p$, $K + 2$ for $p \succ a \succ b \succ c$ and $c \succ b \succ p \succ a$, and 1 each for $a \succ b \succ p \succ c$, $c \succ p \succ a \succ b$, $a \succ c \succ p \succ b$ and $b \succ p \succ a \succ c$. The total scores from non-manipulators are as follows: $s(a) = 14K + 18$, $s(b) = s(c) = 17K + 18$ and $s(p) = 12K + 18$. For each integer k_i, we have a member of the manipulating coalition with weight k_i.

Now, suppose there is a solution to the PARTITION instance. Let the manipulators corresponding to the integers in one half of the partition vote $p \succ a \succ b \succ c$, and let the others vote $p \succ a \succ c \succ b$. All scores are now $18K + 18$ (which is also the average). By the tie-breaking rule, p wins in the first round. Thus the manipulators can make p win if a perfect partition exists.

Conversely, suppose there is a successful manipulation. Clearly, p cannot be eliminated in the first round. To ensure this, all manipulators must put p in first place. Next, we show that if p is not a joint winner of the first round, p cannot win overall. We consider all possible sets of candidates that could be eliminated in the first round. There are 6 cases. In the first case, only a is eliminated in the first round. The scores from non-manipulators in the second round are as follows: $s(b) = s(c) = 12K + 13$, and $s(p) = 6K + 10$. The average score is $10K + 12$. Even with the maximum $4K$ possible score from the manipulators, p is eliminated. This contradicts the assumption that p wins. In the second case, only b is eliminated in the first round. As p and a are not eliminated in the first round, the manipulators have to cast votes that put p in first place and b in second place. With such manipulating votes, the scores in the second round are: $s(a) = 11K + 11$, $s(c) = 12K + 12$ and $s(p) = 13K + 13$. The average score is $12K + 12$. Hence, a is eliminated. In the next round, p is eliminated as $s(p) = 5K + 5$, $s(c) = 7K + 7$ and the average score is $6K + 6$. This contradicts the assumption that p wins. In the third case, only c is eliminated in the first round. This case is symmetric to the second case. In the fourth case, a and b are eliminated in the first round. The case when a and c are eliminated is symmetric. In the second round, the scores from non-manipulators are $s(c) = 7K + 7$ and $s(p) = 3K + 5$. The $2K$ score from the manipulators cannot prevent p being eliminated. This contradicts the assumption that p wins. In the fifth case, b and c are eliminated in the first round. However, in the first round, the score b and c receive from the non-manipulators is $17K + 18$. One of them will get at least K points from manipulators. This will give them greater than the average score of $18K + 8$. Hence, at least one of them is not eliminated. In the sixth and final case, a, b and c are all eliminated in the first round. This case is again impossible by the same argument as the last case.

The only way for p to win is to have a tie with all candidates in the first round. As we observed above, the manipulators have to put p in first place, and a in second place. In turn, both b and c have to get exactly K points from the manipulators. Hence, there exists a solution to the PARTITION instance. □

Clearly, it is polynomial to compute a manipulation of Baldwin's rule with 2 candidates (since this case degenerates to majority voting). With Nanson's rule, on the other hand, it is polynomial with up to 3 candidates.

Theorem 6. *With Nanson's rule and weighted votes, the coalition manipulation problem is polynomial for up to 3 candidates.*

Proof: Consider an election with 3 candidates (a, b and p) in which the manipulators want p to win. We prove that the optimal strategy is for the manipulators either all to vote $p \succ a \succ b$ or all to vote $p \succ b \succ a$. If p does not win using one of these two votes, then p cannot win. Therefore we simply try out the two votes and compute if p wins in either case.

Suppose the manipulators can make p win. We first note that there is no loss for them to raise p to the first position, while keeping the other parts of their preferences the same. By doing so, the score of p goes up and the scores of a and b go down. The only possible change in the elimination process is that now both a and b drop out in the first round, so that p still wins.

Now, suppose that all manipulators rank p in their top positions. Let P^M denote the manipulators' profile that makes p win. Because Nanson's rule never selects the Condorcet loser, p cannot be beaten by both a and b in pairwise elections. Without loss of generality, suppose p beats a. We argue that if all manipulators vote $p \succ a \succ b$, then p still wins. For the sake of contradiction, suppose all manipulators vote $p \succ a \succ b$ but p does not win. As the manipulators still rank p in their top positions, the score of p in the first round is the same as in P^M. Therefore, p must enter (and lose) the second round. Hence, only a is eliminated in the first round, and in the second round b beats p. However, having the manipulators vote $p \succ a \succ b$ only lowers b's score in the first round, compared to the case where they vote P^M. Hence, when the manipulators vote P^M, b also enters the second round and then beats p, which is a contradiction.

Therefore, if the manipulators can make p win, then they can make p win by all voting $p \succ a \succ b$, or all voting $p \succ b \succ a$. □

5 Approximation methods

NP-hardness only bounds the worst-case complexity of computing a manipulation. Given enough manipulators, we can easily make any candidate win. We consider next minimizing the number of manipulators required. For example, REVERSE is a simple approximation method proposed to compute Borda manipulations [26]. The method constructs the vote of each manipulator in turn: candidate c is put in first place, and the remaining candidates are put in reverse order of their current Borda scores. The method continues constructing manipulating votes until c wins. A long and intricate argument shows that REVERSE constructs a manipulation which uses at most one more manipulator than is optimal.

Example 1. *Suppose we have 4 candidates, and the 2 non-manipulators have cast votes:* $3 \succ 1 \succ 2 \succ 4$ *and* $2 \succ 3 \succ 1 \succ 4$*. Then we have the score vector* $\langle 3, 4, 5, 0 \rangle$*. We use* REVERSE *to construct a manipulation that makes candidate* 4 *win.* REVERSE *first constructs the vote:* $4 \succ 1 \succ 2 \succ 3$*. The score vector is now* $\langle 5, 5, 5, 3 \rangle$*.* REVERSE *next constructs the vote:* $4 \succ 1 \succ 2 \succ 3$*. (It will not matter how ties between* 1*,* 2 *and* 3 *are broken). The score vector is now* $\langle 7, 6, 5, 6 \rangle$*. Finally,* REVERSE *constructs the vote:* $4 \succ 3 \succ 2 \succ 1$*. The score vector is* $\langle 7, 7, 7, 9 \rangle$*. Hence,* REVERSE *requires 3 manipulating votes to make candidate* 4 *win. As we see later, this is one more vote than the optimal.*

We propose four new approximation methods. The first two algorithms work with all three voting rules. However, the last two algorithms are designed specifically for the elimination style of Baldwin's and Nanson's rule.

5.1 Manipulation matrices

In this section we prove some auxiliary results that will be helpful to construct our approximation algorithms.

We can view REVERSE as greedily constructing a manipulation matrix. A *manipulation matrix* is an n by m matrix, A where $A(i, j) = k$ iff the ith manipulator adds a

score of k to candidate j. A manipulation matrix has the properties that each of the n rows is a permutation of 0 to $m-1$, and the sum of the jth column is less than or equal to $g(j)$, the maximum score candidate j can receive without defeating c. REVERSE constructs this matrix row by row.

Our two new approximation methods break out of the straightjacket of constructing a manipulation matrix in row wise order. They take advantage of an interesting result that relaxes the constraint that each row is a permutation of 0 to $m-1$. This lets us construct a *relaxed manipulation matrix*. This is an n by m matrix that contains n copies of 0 to $m-1$ in which the sum of the jth column is again less than or equal to $g(j)$. In a relaxed manipulation matrix, a row can repeat an integer provided other rows compensate by not having the number at all.

Theorem 7. *Suppose there is an n by m relaxed manipulation matrix A. Then there is n by m manipulation matrix B with the same column sums.*

Proof: By induction on n. In the base case, $n = 1$ and we just set $B(1, j) = A(1, j)$ for $j = 1 \ldots, m$. In the inductive step, we assume the theorem holds for all relaxed manipulation matrices with $n-1$ rows. Let $h(i)$ be the sum of the ith column of A. We use a perfect matching in a suitable bipartite graph to construct the first row of B and then appeal to the induction hypothesis on an $n-1$ by m relaxed manipulation matrix constructed by removing the values in the first row from A.

We build a bipartite graph between the vertices V_i and W_j for $i \in [0, m-1]$ and $j \in [1, m]$. V_i represent the scores assigned to the first row of B, whilst W_j represent the columns of A from where these will be taken. We add the edge (V_i, W_j) to this bipartite graph for each $i \in [0, m-1]$,$j \in [1, m]$ and $k \in [1, n]$ where $A(k, j) = i$. Note that there can be multiple edges between any pair of vertices. By construction, the degree of each vertex is n.

Suppose we take any $U \subseteq \{V_i | i \in [0, m-1]\}$. By a simple counting argument, the neighborhood of U must be at least as large as U. Hence, the Hall condition holds and a perfect matching exists [17]. Consider an edge (V_i, W_j) in such a perfect matching. We construct the first row of B by setting $B(1, j) = i$. As this is a matching, each $i \in [0, m-1]$ occurs once, and each column is used exactly one time. We now construct an $n-1$ by m matrix from A by removing one element equal to $B(1, j)$ from each column j. By construction, each value $i \in [0, m-1]$ occurs $n-1$ times, and the column sums are now $h(j) - B(1, j)$. Hence it is a relaxed manipulation matrix. We can therefore appeal to the induction hypothesis. This gives us an n by m manipulation matrix B with the same column sums as A. □

We can extract from this proof a polynomial time method to convert a relaxed manipulation matrix into a manipulation matrix. Hence, it is enough to propose new approximation methods that construct *relaxed* manipulation matrices. This is advantageous for greedy methods like those proposed here as we have more flexibility in placing integers into good positions in a relaxed manipulation matrix.

5.2 Largest Fit

Our first approximation method, LARGEST FIT is inspired by bin packing and multiprocessor scheduling. Constructing an n by m relaxed manipulation matrix is similar

to packing n objects into m bins with a constraint on the capacity of the different bins. In fact, the problem is even similar to scheduling nm unit length jobs on n different processors with a constraint on the total memory footprint of the n different jobs running at every clock tick. Krause et al. have proposed a simple heuristic for this problem that schedules the unassigned job with the largest memory requirement to the time step with the maximum remaining available memory that has less than n jobs assigned [18]. If no time step exists that can accommodate this job, then the schedule is lengthened by one step.

LARGEST FIT works in a similar way to construct a relaxed manipulation matrix. It assigns the largest unallocated score to the largest gap. More precisely, it first assigns n instances of $m-1$ to column c of the matrix (since it is best for the manipulators to put c in first place in their vote). It then allocates the remaining $(n-1)m$ numbers in reverse order to the columns corresponding to the candidate with the current smallest score who has not yet received n votes from the manipulators. Unlike REVERSE, we do not necessarily fill the matrix in row wise order.

Example 2. *Consider again the last example. We start with the score vector* $\langle 3,4,5,0\rangle$. *One manipulator alone cannot increase the score of candidate* 4 *enough to beat* 2 *or* 3. *Therefore, we need at least two manipulators.* LARGEST FIT *first puts two* 3*s in column* 4 *of the relaxed manipulation matrix. This gives the score vector* $\langle 3,4,5,6\rangle$. *The next largest score is* 2. LARGEST FIT *puts this into column* 1 *as this has the larger gap. This gives the score vector* $\langle 5,4,5,6\rangle$. *The next largest scores is again* 2. LARGEST FIT *puts this into column* 2 *giving the score vector* $\langle 5,6,5,6\rangle$. *The two next largest scores are 1.* LARGEST FIT *puts them in columns* 1 *and* 3 *giving the score vector* $\langle 6,6,6,6\rangle$. *Finally, the two remaining scores of* 0 *are put in columns* 2 *and* 3 *so all columns contain two scores. This gives a relaxed manipulation matrix corresponding to the manipulating votes:* $4 \succ 2 \succ 1 \succ 3$ *and* $4 \succ 1 \succ 3 \succ 2$. *With these votes,* 4 *wins based on the tie-breaking rule. Unlike* REVERSE, LARGEST FIT *constructs the optimal manipulation with just two manipulators.*

5.3 Average Fit

Our second approximation method, AVERAGE FIT takes account of both the size of the gap and the number of scores still to be added to each column. If two columns have the same gap, we want to choose the column that contains the fewest scores. To achieve this, we look at the average score required to fill each gap: that is, the size of the gap divided by the number of scores still to be added to the column. AVERAGE FIT puts the largest unassigned score possible into the column which will accommodate the largest average score. AVERAGE FIT does not allocate the largest unassigned score but the largest such score that will fit into the gap. This avoids defeating c where it is not necessary. If two or more columns can accommodate the same largest average score, we tie-break either arbitrarily or on the column containing fewest scores. The latter is more constrained and worked best experimentally. However, it is possible to construct pathological instances on which the former is better.

5.4 Eliminate and Reverse Eliminate

Our third and fourth methods are designed to take into account the elimination style nature of Baldwin's and Nanson's rules.

ELIMINATE repeatedly construct votes in which the desired candidate is put in first place, and the other candidates in the reverse of the current elimination order. Thus, the first candidate eliminated is put in last place, the second candidate eliminated is put in the second from last place, and so on. For Nanson's rule, we order candidates eliminated in the same round by their Borda score in that round. The intuition behind ELIMINATE is to move the desired candidate up the elimination order whilst keeping the rest of the order unchanged.

REVELIMINATE repeatedly construct votes in which the desired candidate is put in first place, and the other candidates in the current elimination order. For instance, the first candidate eliminated is put in second place. For Nanson's rule, we order candidates eliminated in the same round by the inverse of their Borda score in that round. The intuition behind REVELIMINATE is to move the desired candidate up the elimination order, and to assign the largest Borda scores to the least dangerous candidates.

It is easy to show that all methods will eventually compute a manipulation of Nanson's or Baldwin's rule in which the desired candidate wins.

5.5 Theoretical properties

We show that LARGEST FIT is incomparable to REVERSE since there exists an infinite family of problems on which LARGEST FIT finds the optimal manipulation but REVERSE does not, and vice versa.

Theorem 8. *Let V be an election such that $m > 2$ is even, $s(m, V) = 0$ and $s(i, V) = \frac{m}{2} + i$ for all $1 \leq i \leq m - 1$. Then* LARGEST FIT *finds an optimal 2-manipulation, but* REVERSE *produces a 3-manipulation.*

First, note that two non-manipulator votes are always sufficient to create such an election. Let $\sigma =< 1, 2, ..., m - 1 >$ and let

$$\sigma' =< \frac{m}{2} + 1, \frac{m}{2} + 2, ..., \frac{m}{2} + \frac{m}{2} - 1, 1, 2, ..., \frac{m}{2} >$$

Then $\sigma + \sigma' =$

$$< \left(1 + \frac{m}{2} + 1\right), \left(2 + \frac{m}{2} + 2\right), ..., \left(\frac{m}{2} - 1 + \frac{m}{2} + \frac{m}{2} - 1\right), \left(\frac{m}{2} + 1\right), ..., \left(m - 1 + \frac{m}{2}\right) >$$

which gives us $\frac{m}{2} + 2x$ for $1 \leq x \leq \frac{m}{2} - 1$ and $\frac{m}{2} + 2x - 1$ for $1 \leq x \leq \frac{m}{2}$, or in other words, $\frac{m}{2} + i$ for all $1 \leq i \leq m - 1$.

The first vote generated by REVERSE is $r_1 = m \succ 1 \succ 2 \succ ... \succ m - 1$, after which $s(i, V \cup \{r_1\}) = \frac{m}{2} + m - 1$ for all competing candidates, which is larger than the score of the distinguished candidate $s(m, V \cup \{r_1\}) = m - 1$. Therefore another manipulator is added, without loss of generality its vote is $r_2 = m \succ 1 \succ 2 \succ .... \succ$

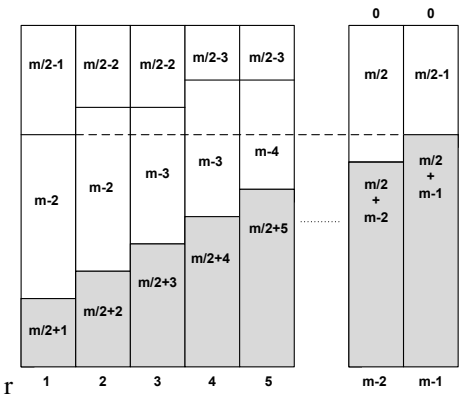

Figure 1: The 2-manipulation generated by LARGEST FIT for the election in Proposition 8

$m-1$. The resulting scores of the competing candidates are $s(i, V \cup \{r_1, r_2\}) = \frac{m}{2} + (m-1) + (m-i-1) = (5/2)m - 2 - i$. So candidate $i = 1$ still has larger score than $s(m, V \cup \{r_1, r_2\}) = 2m - 2$. Therefore, REVERSE does not find a 2-manipulation.

The first $m-1$ iterations of LARGEST FIT will place the k^{th} largest score from S_2 into the k^{th} column of matrix B for $1 \leq k \leq m-1$. Note that the k^{th} largest score is $m - 2 - \lfloor (k-1)/2 \rfloor$. Let B_{m-1} be the matrix at this point. Then $sum(B_{m-1}(i)) + s(i, V) = (m - 2 - \lfloor (i-1)/2 \rfloor) + \frac{m}{2} + i$ for all i. The next $m-1$ iterations of LARGEST FIT will place the k^{th} largest score from S_2 into the k^{th} column of matrix B for $m \leq k \leq 2(m-1)$. So column i will receive the element $\frac{m}{2} - 1 - \lceil (i-1)/2 \rceil$. Let $B_{2(m-1)}$ be the matrix when the loop terminates. Then $sum(B_{2(m-1)}(i)) + s(i, V) = (m - 2 - \lfloor (i-1)/2 \rfloor) + (\frac{m}{2} + i) + (\frac{m}{2} - 1 - \lceil (i-1)/2 \rceil) = 2(m-1)$ for all i, while the achievable score of m is also $2(m-1)$. Therefore, LARGEST FIT does find a 2-manipulation. Figure 1 shows the matrix generated by LARGEST FIT, where the shaded area represents the scores $s(i, V)$. □

Unfortunately, LARGEST FIT does not share the guarantee of REVERSE that in the worst case it requires one extra manipulator than is optimal. In fact the number of extra manipulators used by LARGEST FIT is not bounded.

Theorem 9. *Let k be positive integer greater than zero and divisible by 36. Let $s(V, 1) = 6k$, $s(V, 2) = 4k$, $s(V, 3) = 2k$, $s(V, 4) = 0$ be the scores of four candidates after some non-manipulators V vote. Then* REVERSE *will find the optimal manipulation, using $2k$ manipulators. However,* LARGEST FIT *requires at least $2k + k/9 - 3$ manipulators.*

Proof:

First, we should mention that for any k there is a set of votes V_k that gives the specified scores to the four candidates: V_k is simply $2k$ votes, all equal to $c_1 \succ c_2 \succ c_3 \succ c_4$. REVERSE will use $2k$ manipulators, all voting $c_4 \succ c_3 \succ c_2 \succ c_1$, to achieve a score of $6k$ for all candidates (the only optimal manipulation). It remains to argue that

LARGEST FIT requires more than $2k+k/9-4$ manipulators. Assume for contradiction that we find a manipulation using $n = 2k + k/9 - 4 = 19k/9 - 4$ manipulators. We will follow the execution of LARGEST FIT until a contradiction is obtained. Note that given our definition of n, since k is divisible by 4 and 9, $\frac{n-k}{2}$ is an integer.

First, the algorithm will place k 2's in B[3], at which point $sum(B[3]) = 2k+2k = 4k = s(2, V_k)$. Then it will begin to place 2's in columns B[2] and B[3] evenly, until all remaining $n - k$ 2's have been placed into B. At this point, B[2] contains $\frac{n-k}{2}$ 2's, and the number of 2's that B[3] contains is $k+\frac{n-k}{2} = k/2+n/2 = k/2+(19k/9-4)/2 = 14k/9 - 2 < 19k/9 - 4 = n$. So at this point, B[3] is not full yet and B[2] isn't either (it has fewer elements than B[3]). Both columns sum to $4k + 2(\frac{n-k}{2}) = 46k/9 - 4 = 5k + k/9 - 4 < 6k$. Therefore, the algorithm will start putting 1's in both B[2] and B[3] evenly, until either their column sums reach $6k$ or B[3] gets filled. In fact, B[3] will be filled before its sum reaches $6k$, since B[3] requires $\frac{n-k}{2}$ more elements to be filled, but at this point, $sum(B[2]) = sum(B[3]) = 46k/9-4+\frac{n-k}{2} = 51k/9-6 = 5k + 2k/3 - 6 < 6k$.

Now, the algorithm will continue by putting $k/3 + 6$ 1's into B[2], at which point $sum(B[2]) = 51k/9 - 6 + k/3 + 6 = 6k$. Then the algorithm will start putting 1's evenly in both B[1] and B[2], until either it runs out of 1's or B[2] is filled. In fact, the 1's will run out before B[2] is filled, since B[2] requires $n-(\frac{n-k}{2}+\frac{n-k}{2}+k/3+6) = 2k/3 - 6$ more elements, which is equal to the number of remaining 1's, but these are spread between B[1] and B[2]. So B[2] will get $(2k/3-6)/2 = k/3-3$ additional 1's, for a total of $sum(B[2]) = 4k+2(\frac{n-k}{2})+\frac{n-k}{2}+k/3+6+k/3-3 = 19k/3-3 > 19k/3 - 12 = 3n$. Since $sum(B[2]) > 3n$ there is no manipulation using $n = 19k/9-4$ manipulators. Therefore, LARGEST FIT requires at least $n+1 = 2k+k/9-3$ manipulators. □

AVERAGE FIT is also incomparable to LARGEST FIT. Like REVERSE, it finds the optimal manipulations on the elections in the last proof. So far we have not found any instances where REVERSE performs better than AVERAGE FIT. However, there exist examples on which LARGEST FIT finds the optimal manipulation but AVERAGE FIT does not.

Theorem 10. *There exists instances on which* LARGEST FIT *finds the optimal manipulation but* AVERAGE FIT *requires an additional vote.*

Proof: We failed to find a simple example but a computer search using randomly generated instances gave the following. Consider an election in which the non-manipulators wish the last candidate to win, given the score vector:

$$\langle 41, 34, 30, 27, 27, 26, 25, 14\rangle$$

On this problem, LARGEST FIT finds the optimal manipulation that makes the final candidate win but AVERAGE FIT requires an additional vote. □

Finally, we consider properties of approximation algorithms with respect to Baldwin's and Nanson's rules. These rules appear more difficult to approximate within such bounds. We can give examples where all five methods compute a manipulation that use several more manipulators than is optimal. The most intresting result is that although REVERSE was shown to never require more than one extra manipulator than optimal

[26] under Borda's rule, the result does not transfer to Baldwin's and Nanson's rules. Indeed, even with a fixed number of candidates, REVERSE can require an unbounded number of extra manipulators.

Theorem 11. *With Baldwin's rule, there exists an election with 7 candidates and $42n$ votes where* REVERSE *computes a manipulation with at least n more votes than is optimal.*

Proof: Consider an election over a, b, c, d, e, f and p where p is the candidate that the manipulators wish to win. We define $R(u, v)$ as the pair of votes: $u \succ v \succ$ *Others* $\succ p$, rev(*Others*) $\succ u \succ v \succ p$ where *Others* is some fixed ordering of the other candidates and rev(*Others*) is its reverse. The non-manipulators cast the following votes: $3n$ copies of $R(a, b)$, $R(b, c)$, $R(c, d)$, $R(d, e)$ and $R(e, f)$. In addition, there are $6n$ copies of the votes: $p \succ a \succ$ *Others* and rev(*Others*) $\succ p \succ a$. If $18n$ manipulators vote identically $p \succ a \succ \ldots \succ f$ then p wins.

This provides an upper bound on the size of the optimal manipulation. After the non-manipulators have voted, $s(a) = s(f) = 138n$, $s(b) = s(c) = s(d) = s(e) = 141n$ and $s(p) = 42n$. REVERSE will put p in first place. We suppose n is a multiple of 2, but more complex arguments can be given in other cases. After n manipulating votes have been constructed, the scores of candidates a to f are level at $142n + n/2$ and p is leveled at $48n$.

From then on, the manipulators put p in first place and alternate the order of the other candidates. At least $32n$ votes are therefore required for p to move out of last place. Hence, REVERSE requires an unbounded number of extra manipulators. □

Asymptotically this result is as bad as we could expect. Any election can be manipulated with $O(n)$ votes by simply reversing all previous votes, and this proof demonstrates that REVERSE may use $O(n)$ more votes than is optimal.

6 Experimental results

To test the performance of these approximation methods in practice, we ran some experiments. Our experimental setup is based on that in [21]. We generated either uniform random votes or votes drawn from a Polya Eggenberger urn model. In the urn model, votes are drawn from an urn at random, and are placed back into the urn along with b other votes of the same type. This captures varying degrees of social homogeneity. We set $b = m!$ so that there is a 50% chance that the second vote is the same as the first. In both models, we generated between 2^2 and 2^7 votes for varying m.

6.1 Borda rule

First we present our results for Borda rule. Manipulation under the Borda rule can be easily modeled as a constraint satisfaction problem. We used this property to obtain an optimum solution for our instances. We tested 1000 instances at each problem size and determined if the returned manipulations are optimal, using a constraint solver.

We were able to find the optimal manipulation in 32502 out of the 32679 distinct uniform elections within the 1 hour time-out. Results are shown in Figure 2. Both

m	# Inst.	REVERSE	LARGEST FIT	AVERAGE FIT	LARGEST FIT beat AVERAGE FIT
4	2771	2611	2573	2771	0
8	5893	5040	5171	5852	2
16	5966	4579	4889	5883	3
32	5968	4243	4817	5879	1
64	5962	3980	4772	5864	3
128	5942	3897	4747	5821	2
Total	32502	24350	26969	32070	11
%		75	83	99	<1

Figure 2: Number of uniform elections for which each method found an optimal manipulation.

m	# Inst.	REVERSE	LARGEST FIT	AVERAGE FIT	LARGEST FIT beat AVERAGE FIT
4	3929	3666	2604	3929	0
8	5501	4709	2755	5496	0
16	5502	4357	2264	5477	1
32	5532	4004	2008	5504	0
64	5494	3712	1815	5475	0
128	5571	3593	1704	5565	0
Total	31529	24041	13150	31446	1
%		76	42	99.7	<1

Figure 3: Number of urn elections for which each method found an optimal manipulation.

Table 1: Percentage of random uniform elections with 5 candidates where the heuristic finds the optimal manipulation.

Rules	REV	LAFIT	AVFIT	ELIM	REVELIM
Baldwin	74.4%	74.4%	**75.8%**	62.2%	75.2%
Nanson	74.6%	76.0%	**78.0%**	65.4%	66.9%
Borda	95.7%	98.8%	**99.8%**	95.7%	10.7%

Table 2: Percentage of urn elections with 5 candidates where the heuristic finds the optimal manipulation.

Rules	REV	LAFIT	AVFIT	ELIM	REVELIM
Baldwin	75.1%	75.4%	**77.3%**	68.9%	83.4%
Nanson	78.1%	79.0%	**79.8%**	72.2%	79.4%
Borda	96.1%	92.7%	**99.9%**	96.1%	4.4%

LARGEST FIT and AVERAGE FIT provide a significant improvement over REVERSE, solving 83% and 99% of instances to optimality. REVERSE solves fewer problems to optimality as the number of candidates increases, while AVERAGE FIT does not seem to suffer from this problem as much: AVERAGE FIT solved all of 4 candidate instances and 98% of the 128 candidate ones. We note that in every one of the 32502 instances, if REVERSE found an n vote manipulation either AVERAGE FIT did too, or AVERAGE FIT found an $(n-1)$ vote manipulation.

We were able to find the optimal manipulation for 31529 out of the 31530 unique urn elections within the 1 hour time-out. Figure 3 gives results. REVERSE solves about the same proportion of the urn instances as uniform instances, 76%. However, the performance of LARGEST FIT drops significantly. It is much worse than REVERSE solving only 42% of instances to optimality. We saw similar pathological behaviour with the correlated votes in the proof of Theorem 9. The good performance of AVERAGE FIT is maintained. It found the optimal manipulation on more than 99% of the instances. It never lost to REVERSE and was only beaten by LARGEST FIT on one instance in our experiments.

These results suggest that while Borda manipulation is NP-hard, in practice the simple approximation algorithms that we proposed can compute optimal manipulations in the vast majority of cases. Thus, it appears that Borda elections are vulnerable to manipulation.

6.2 Baldwin's and Nanson's rules

It is much more difficult to model the unweighted coalition manipulation problem under Baldwin's and Nanson's rules as a constraint satisfaction problem since the scores of the candidates in each vote change in each round. Hence, we partitioned our experiments into two parts: small problems where we can find an optimum solution in a brute-force manner and large problems that show how approximation algorithms scale.

Our first set of experiments used 3000 elections with 5 candidates and 5 non-

manipulating voters. This is small enough to find the optimal number of manipulators using brute force search, and thus to determine how often a heuristic computes the optimal solution. We threw out the 20% or so of problems generated in which the chosen candidate has already won before the manipulators vote. Results are given in Tables 1–2. Heuristics that are very effective at finding an optimal manipulation with the Borda rule do not perform as well with Baldwin's and Nanson's rules. For example, AVERAGEFIT almost always finds an optimal manipulation of the Borda rule but can only find an optimal solution about 3/4 of the time with Baldwin's or Nanson's rules.

In our second set of experiments, we eschew computation of an optimal manipulation in order to use larger problems. This amplifies the differences between the different approximation methods. Similar to section 6.1, we tested 1000 instances at each problem size, which gives 6000 instances in total.

Tables 3–6 show the results for the average number of manipulators. The results show that overall REVERSE works slightly better than LARGESTFIT and AVERAGEFIT, which themselves outperform the other two methods especially for problems with large number of candidates. We observe a similar picture with Nanson's rule. This contrasts with the Borda rule where LARGESTFIT and AVERAGEFIT do much better than REVERSE [12]. In most cases AVERAGEFIT is less effective than LARGESTFIT except urn elections with Nanson's rule.

These experimental results suggest that Baldwin's and Nanson's rules are harder to manipulate in practice than Borda. Approximation methods that work well on the Borda rule are significantly less effective on these rules. Overall, REVERSE, LARGESTFIT and AVERAGEFIT appear to offer the best performance, though no heuristic dominates.

7 Related problems

There exists an interesting connection between the problem of finding a coalition of two manipulators for the Borda voting rule and two other problems in discrete mathematics: the problem of finding a permutation matrix with restricted diagonals sums (PMRDS) [8] and the problem of finding multi Skolem sequences [20]. We consider this connection for two reasons. First, future advances in these adjacent areas may give insights into new manipulation algorithms or into the complexity of manipulation. Second, this connection reveals an interesting open case for Borda manipulation – Nordh has conjectured that it is polynomial when all gaps are distinct.

A permutation matrix is an n by n Boolean matrix which is obtained from an identity matrix by a permutation of its columns. Hence, the permutation matrix contains a single value 1 in each row and each column. Finding a permutation matrix such that the sums of its diagonal elements form a given sequence of numbers $(d_1, \ldots, d_{2n-1})$ is the permutation matrix with restricted diagonals sums problem. This problem occurs in discrete tomography, where we need to construct a permutation matrix from its X-rays for each row, column and diagonal. The X-ray values for each row and column are one, while the values for the diagonal are represented with the sequence $(d_1, \ldots, d_{2n-1})$.

We transform a manipulation problem with n candidates and 2 manipulators such that $\sum_{i=1}^{n} g_i = n(n-1)$ to a PMRDS problem. To illustrate the transformation we

Table 3: Uniform elections using Baldwin rule. This (and subsequent) tables give the average number of manipulators.

n	Rev	LaFit	AvgFit	Elim	RevElim
4	2.25	2.25	2.25	2.44	**2.21**
8	**2.99**	3.07	3.01	3.35	3.06
16	**4.31**	4.41	4.40	4.79	4.67
32	**5.93**	6.03	6.14	6.61	6.84
64	**8.56**	8.65	8.84	9.54	11.02
128	**12.13**	12.24	12.41	13.37	16.06

Table 4: Uniform elections using Nanson rule.

n	Rev	LaFit	AvgFit	Elim	RevElim
4	**2.15**	2.17	2.15	2.25	2.28
8	2.91	2.96	**2.84**	3.05	3.21
16	4.13	4.27	**4.05**	4.44	4.99
32	**5.80**	5.88	5.81	6.18	7.46
64	**8.51**	8.58	8.82	8.99	12.04
128	**12.07**	12.09	13.00	12.60	17.90

Table 5: Urn elections using Baldwin rule.

n	Rev	LaFit	AvgFit	Elim	RevElim
4	3.26	3.23	3.24	3.35	**3.14**
8	5.95	5.96	5.99	6.37	**5.82**
16	11.64	11.66	11.87	12.74	**11.52**
32	**21.70**	21.78	22.35	24.67	22.41
64	**43.09**	43.37	44.24	49.07	45.70
128	82.19	**81.82**	83.62	95.37	91.80

Table 6: Urn elections using Nanson rule.

n	Rev	LaFit	AvgFit	Elim	RevElim
4	3.20	**3.19**	3.20	3.28	3.22
8	**5.93**	5.98	5.95	6.13	6.09
16	**11.62**	11.93	11.64	12.16	12.37
32	**22.36**	22.78	22.53	24.00	24.39
64	**44.56**	45.50	44.77	48.81	49.69
128	87.18	87.55	**86.76**	97.02	99.43

use the following example with 5 candidates. Let $\langle 4, 4, 6, 6, 0\rangle$ be a score vector, where our favorite candidate has 0 score, and $\langle 4, 4, 2, 2\rangle$ be the corresponding gap vector. We label rows of a permutation matrix with scores of the first manipulator and columns of a permutation matrix with the reversed scores of the second manipulator. We label each element of the matrix with the sum of its row and column labels. Figure 4(a) shows the labelling for our example in gray.

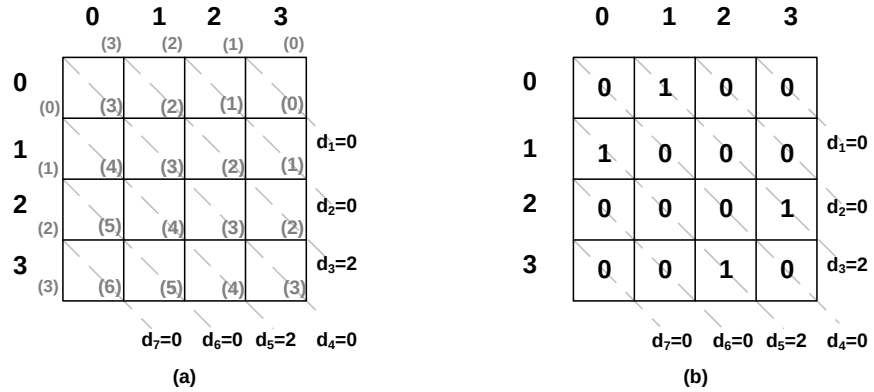

Figure 4: (a) A labelling of a permutation matrix; (b) a solution of PMRDS

Note that each element on a diagonal is labelled with the same value. Therefore, each diagonal labelled with value k represents the gap of size k in the manipulation problem. Hence, the sum of the diagonal d_i labelled with k encodes the number of occurrences of gaps of size k. For example, $d_3 = 2$ ensures that there are two gaps of size 2 and $d_5 = 2$ ensures that there are two gaps of size 4. The remaining diagonal sums, d_i, $i \in \{1, 2, 4, 6, 7\}$, are fixed to zero.

Consider a solution of PMRDS (Figure 4(b)). Cell $P(0, 1)$ contains the value one. Hence, we conclude that the first manipulator gives the score 0 and the second gives the score 2 to a candidate with the gap 2. Similarly, we obtain that the first manipulator gives the scores $\langle 1, 3, 0, 2\rangle$ and the second gives the scores $\langle 3, 1, 2, 0\rangle$ to fill gaps $\langle 4, 4, 2, 2\rangle$. As the number of ones in each diagonal is equal to the number of occurrences of the corresponding gap, the constructed two manipulator ballots make our candidate a co-winner.

Finding a coalition of two manipulators for the Borda voting rule is also connected to the problem of finding multi Skolem sequences used for the construction of Steiner triple system [20]. Given a multiset of positive integers G we need to decide whether there exists a partition of a set $H = \{1, \ldots, 2n\}$ into a set of pairs (h_i, h'_i), $i = 1, \ldots, n$ so that $G \equiv \{h_i - h'_i | h_i, h'_i \in H\}$. There is a reduction from a manipulation problem with n candidates and 2 manipulators such that $\sum_{i=1}^{n} g_i = n(n-1)$ to a special case of multi Skolem sequences with $\sum_{i=1}^{n} G_i = n^2$ similar to the reduction from a scheduling problem in [20] [3].

[3]The reduction implicitly assumes that $\sum_{i=1}^{n} G_i = n^2$ as the author confirmed in a private communication.

8 Related work

Xia *et al.* showed that unweigthed coalition manipulation problem is NP-hard for an artificially constructed positional scoring rule [24]. Brelsford et al. [7] showed that weighted (and thus unweighted) Borda manipulation has a FPTAS, which means that finding a very close to optimal manipulation can be done in polynomial time.

With respect to our investigation of Baldwin's and Nanson's rules, Bag *et al.* [1] prove that a class of voting rules which use repeated ballots and eliminate one candidate in each round are Condorcet consistent. They illustrate this class with the *weakest link* rule in which the candidate with the fewest ballots in each round is eliminated. Geller [16] has proposed a variant of single transferable vote where candidates are successively eliminated based on their *original* Borda score. Unlike Nanson's and Baldwin's rules, this method does not recalculate the Borda score based on the new reduced set of candidates. For any Condorcet consistent rule (and thus for Nanson's and Baldwin's rule), Brandt *et al.* [6] showed that many types of control and manipulation are polynomial to compute when votes are single peaked.

The manipulability of voting rules has also been studied empirically [22, 21]. For example, Walsh studied the Single Transferable Vote rule, which is theoretically NP-hard to manipulate. However, his experiments suggest that in practice, elections using this rule are easy to manipulate [21].

9 Conclusions

In this paper we investigated theoretically and empirically the coalition manipulation problem for the Borda voting rule and its generalizations, Baldwin's and Nanson's rules. We proved that it is NP-hard to manipulate Borda rule with just two manipulators. This resolves one of the last open questions regarding the computational complexity of unweighted coalition manipulation for common voting rules. We showed that two other scoring rules, Baldwin's and Nanson's rules, which share many of the benefits of Borda's rule are also NP-hard to manipulate. Due to these NP-hardness results, we proposed several simple approximation methods. We showed that they can compute optimal manipulations of the Borda rule in almost all the randomly generated elections. This suggests that the Borda rule is not resistant to manipulation in practice. In contrast, these approximation algorithms did not perform as well in either Baldwin or Nanson elections, suggesting that these elimination style rules are more resistant to manipulation than Borda's rule.

10 Acknowledgements

George Katsirelos is supported by the ANR UNLOC project ANR 08-BLAN-0289-01. Nina Narodytska and Toby Walsh are supported by the Australian Department of Broadband, Communications and the Digital Economy, the ARC, and the Asian Office of Aerospace Research and Development (AOARD-104123). Lirong Xia acknowledges a James B. Duke Fellowship and Vincent Conitzer's NSF CAREER 0953756

and IIS-0812113, and an Alfred P. Sloan fellowship for support.

References

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

[2] Baldwin, J. 1926. The technique of the Nanson preferential majority system of election. *Trans. and Proc. of the Royal Society of Victoria* 39:42–52.

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

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

[5] Betzler, N.; Niedermeier, R.; and Woeginger, G. 2011. Unweighted coalitional manipulation under the Borda rule is NP-hard. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI-11)*, p.55-60.

[6] Brandt, F.; Brill, M.; Hemaspaandra, E.; and Hemaspaandra, L. 2010. Bypassing combinatorial protections: Polynomial-time algorithms for single-peaked electorates. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI-10)*, 715–722.

[7] E. Brelsford, P. Faliszewski, E. Hemaspaandra, H. Schnoor and I. Schnoor. 2008. Approximability of Manipulating Elections. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI-2008)*, p.44-49.

[8] Brunetti, S.; Lungo, A.;A. Del; Gritzmann, P. and de Vries, S. 2008. On the reconstruction of binary and permutation matrices under (binary) tomographic constraints. *Theor. Comput. Sci.* 406(1–2):63–71.

[9] Chamberlin, J. 1985. An investigation into the relative manipulability of four voting systems. *Behavioural Science* 30:195–203.

[10] Coleman, T., and Teague, V. 2007. On the complexity of manipulating elections. In *Proceedings of the 13th Conference "Computing: The Australasian Theory Symposium" (CATS2007)*, 25–33.

[11] Conitzer, V.; Sandholm, T.; and Lang, J. 2007. When are elections with few candidates hard to manipulate? *Journal of the ACM* 54(3):1–33.

[12] Davies, J.; Katsirelos, G.; Narodytska, N.; and Walsh, T. 2010. An empirical study of Borda manipulation. In *Proceedings of the 3rd International Workshop on Computational Social Choice (COMSOC-10)*.

[13] Faliszewski, P.; Hemaspaandra, E.; and Schnoor, H. 2008. Copeland voting: ties matter. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-08)*, 983–990.

[14] Faliszewski, P.; Hemaspaandra, E.; and Schnoor, H. 2010. Manipulation of Copeland elections. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS-10)*, 367–374.

[15] Favardin, P., and Lepelley, D. 2006. Some further results on the manipulability of social choice rules. *Social Choice and Welfare* 26:485–509.

[16] Geller, C. 2005. Single transferable vote with Borda elimination: proportional representation, moderation, quasi-chaos and stability. *Electoral Studies* 24(2):265 – 280.

[17] Hall, P. 1935. On representatives of subsets. *Journal of the London Mathematical Society* 26–30.

[18] Krause, K. L.; Shen, V. Y.; and Schwetman, H. D. 1975. Analysis of several task-scheduling algorithms for a model of multiprogramming computer systems. *Journal of ACM* 22(4):522–550.

[19] Nanson, E. 1882. Methods of election. *Trans. and Proc. of the Royal Society of Victoria* 19:197 – 240.

[20] Nordh, G. 2010. A note on the hardness of Skolem-type sequences. *Discrete Appl. Math.* 158(8):63–71.

[21] Walsh, T. 2010. An empirical study of the manipulability of single transferable voting. In *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI-2010)*. IOS Press, p257–262.

[22] Walsh, T. 2009. Where are the Really Hard Manipulation Problems? The Phase Transition in Manipulating the Veto Rule. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI-2009)*, p324–329.

[23] Xia, L.; Zuckerman, M.; Procaccia, A.; Conitzer, V.; and Rosenschein, J. 2009. Complexity of unweighted coalitional manipulation under some common voting rules. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI-2009)*, p348–353.

[24] Xia, L.; Conitzer, V.; and Procaccia, A. 2010. A scheduling approach to coalitional manipulation. In Parkes, D.; Dellarocas, C.; and Tennenholtz, M., eds., *Proceedings of the 11th ACM Conference on Electronic Commerce (EC-2010)*, 275–284.

[25] Yu, W.; Hoogeveen, H.; and Lenstra J.K. 2004. Minimizing Makespan in a Two-Machine Flow Shop with Delays and Unit-Time Operations is NP-Hard. *Journal of Scheduling* 7(5): 333-348.

[26] Zuckerman, M.; Procaccia, A.; and Rosenschein, J. 2009. Algorithms for the coalitional manipulation problem. *Artificial Intelligence.* 173(2):392-412.