Artificial Intelligence 217 (2014) 20–42

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Complexity of and algorithms for the manipulation of Borda, Nanson's and Baldwin's voting rules[☆]

Jessica Davies[a], George Katsirelos[b], Nina Narodytska[a], Toby Walsh[c,*], Lirong Xia[d]

[a] University of Toronto, Toronto, Canada
[b] INRA, Toulouse, France
[c] NICTA and UNSW, Sydney, Australia
[d] Rensselaer Polytechnic Institute, Troy, USA

ARTICLE INFO

Article history:
Received 4 October 2011
Received in revised form 12 July 2014
Accepted 13 July 2014
Available online 23 July 2014

Keywords:
Social choice
Voting methods
Manipulation
Borda voting
Nanson's voting rule
Baldwin's voting rule

ABSTRACT

We investigate manipulation of the Borda voting rule, as well as two elimination style voting rules, Nanson's and Baldwin's voting rules, which are based on Borda voting. We argue that these rules have a number of desirable computational properties. For unweighted Borda voting, we prove that it is NP-hard for a coalition of two manipulators to compute a manipulation. This resolves a long-standing open problem in the computational complexity of manipulating common voting rules. We prove that manipulation of Baldwin's and Nanson's rules is computationally more difficult than manipulation of Borda, as it is NP-hard for a *single* manipulator to compute a manipulation. In addition, for Baldwin's and Nanson's rules with 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 these NP-hardness results, we compute manipulations using heuristic algorithms that attempt to minimise the number of manipulators. We propose several new heuristic methods. Experiments show that these methods significantly outperform the previously best known heuristic 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 more 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. Results like those of Gibbard and Satterthwaite demonstrate that it may often pay for agents to manipulate an election by misreporting their preferences [24,34]. One appealing escape from manipulation is computational complexity [3]. Whilst a manipulation may exist, perhaps it is computationally too difficult to find? With a single manipulator, there is only a small set of voting rules that are known to

[☆] This paper is an invited revision of the winner of the Outstanding Paper award at the Twenty-fifth AAAI Conference on Artificial Intelligence (AAAI-11). Parts of this paper also appear in [14].
[*] Corresponding author.
E-mail addresses: jdavies@cs.toronto.edu (J. Davies), gkatsi@gmail.com (G. Katsirelos), ninan@cs.toronto.edu (N. Narodytska), toby.walsh@nicta.com.au (T. Walsh), xial@cs.rpi.edu (L. Xia).

http://dx.doi.org/10.1016/j.artint.2014.07.005
0004-3702/Crown Copyright © 2014 Published by Elsevier B.V. All rights reserved.

be NP-hard to manipulate with unweighted votes: the second order Copeland rule [13,3], single transferable vote (STV) [2] and ranked pairs [35,40]. With two or more manipulators, computing a manipulation is NP-hard for some other common voting rules [19,40,20,41]. One case that remains open is Borda voting. Xia, Conitzer, and Procaccia [41] observe that:

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

Computing a manipulation of Borda is NP-hard when votes are weighted and we have a coalition of manipulators [12]. On the other hand, computing a manipulation of Borda is polynomial-time when votes are unweighted and there is just a single manipulator [3]. For a coalition of manipulators and unweighted votes, it has been conjectured that the problem is NP-hard [43]. Note that there exist other scoring rules besides Borda where computing a manipulation with unweighted votes has been show to be NP-hard [41]. One of the most important contributions of this paper is to close the question of the computational complexity of computing a coalitional manipulation for Borda with unweighted votes. We prove that computing a manipulation of Borda with just two manipulators is NP-hard. This result was proven independently in [7]. We will discuss the similarities and differences between the two proofs later in the paper.

We also study two voting rules that are closely related to the Borda rule: Nanson's and Baldwin's rules. These are elimination style rules that use Borda scoring to eliminate candidates over a number of rounds. The two rules have been used in real elections in the University of Melbourne (between 1926 and 1982), the University of Adelaide (since 1968), and the State of Michigan (in the 1920s). There are several reasons we consider Nanson's and Baldwin's 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 [10], Nanson's rule is particularly resistant [22]. We might expect Baldwin to be similarly resistant. Thirdly, for any Condorcet consistent rule (and thus for Nanson's and Baldwin's rules), Brandt et al. [8] have shown that many types of control and manipulation problems have polynomial-time algorithms when votes are single-peaked. It is an interesting question then if such manipulation problems remain polynomial when we drop the domain restriction.

Nanson's and Baldwin's rules are also interesting to study as they are elimination style rules, and elimination style rules are often computationally harder to manipulate than the base rule from which they are derived [2,15]. Elkind and Lipmaa have conjectured that computing a manipulation for the *closure* of many voting rules (where we successively use the rule to eliminate candidates) is NP-hard [16]. One of our contributions is to prove that computing a manipulation of Baldwin's rule, which is the closure of Borda voting, is NP-hard with a single manipulator. We also prove that manipulation of Nanson's rule is NP-hard, again with a single manipulator and unweighted votes. Finally, we consider the problem of computing a manipulation with weighted votes and a coalition of manipulators. We show that Baldwin's and Nanson's rules are NP-hard to manipulate in this setting with just three and four candidates respectively.

Our theoretical results suggest that all three rules are computationally difficult to manipulate in the worst case. We also investigate whether these rules are resistant to manipulation in practice [37–39]. We propose several polynomial-time heuristic algorithms for the three voting rules that try to minimise 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 heuristics were significantly less effective for Baldwin's and Nanson's rules. These empirical results, together with our theoretical results, provide further evidence for the recent claim that elimination style voting rules tend to be more computationally resistant to manipulation [15].

The focus in this paper is on manipulation problems. It would, however, be interesting in the future to consider also control and bribery problems [4,17]. Control is somewhat different from manipulation since in control problems we change the structure of the election (number of candidates, number of voters, etc.). Bribery, on the other hand, is very close to manipulation since we only change the votes. Manipulation is also related to the possible winner problem [26] and to dealing with uncertainty when eliciting and aggregating preferences [36,32,33]. See [21] for a longer discussion on the connections between these problems.

The rest of the paper is organised as follows. In Section 2 we provide background. Section 3 focuses on unweighted manipulation and Section 4 on the weighted case. Section 5 presents four heuristic algorithms that aim to find the minimum number of manipulators and Section 6 evaluates these algorithms experimentally. In Section 7 we present two interesting connections between unweighted coalitional manipulation of the Borda rule and two problems from discrete mathematics. We conclude in Section 8.

2. Background

Let $\mathcal{C} = \{c_1, \ldots, c_m\}$ be the set of m *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 $j \le n$, $V_j \in L(\mathcal{C})$. The set of all n-profiles is denoted by $\mathscr{F}_n$. A (deterministic) *voting rule* r is a function that maps any profile on $\mathcal{C}$ to a unique winning candidate, that is, $r : \mathscr{F}_1 \cup \mathscr{F}_2 \cup \ldots \to \mathcal{C}$. When voters are weighted, we have a function w that associates each voter j with a fixed positive integer $w(j)$. A voting rule treats weights as if we had $w(j)$ identical copies of the voter j.

Borda rule The *Borda* rule, proposed by Jean-Charles de Borda in 1770, is a positional scoring rule that gives a score of $m - i$ to candidate a for each vote that puts candidate a in ith place. The candidate with the highest total Borda score wins. We write $s(a, P)$ for the total Borda score given to candidate a from the profile of votes P, and $s(a)$ where P is obvious from the context. A *score vector* $\langle s_1, \dots, s_m \rangle$ indicates that the ith candidate receives the Borda score s_i. 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 similar scoring rules are also used by many organisations 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 monotone, 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 pairwise elections). However, it may fail to elect a Condorcet winner (a candidate that beats all others in pairwise elections) even if one exists.

Nanson's and Baldwin's rules These rules are derived from the Borda rule. Nanson's rule eliminates all candidates with less than the average Borda score [29]. This step 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 one of the candidates with the lowest Borda score[1] until one candidate remains [1]. The two rules are closely related. Indeed, they are sometimes confused in the literature. 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 whenever one exists. This follows from the fact that the Borda score of the Condorcet winner is never below the average Borda score. Both rules satisfy several other desirable criteria, including the majority criterion, i.e., a candidate that is preferred by a majority of voters always wins, and the Condorcet loser criterion. There are also properties which distinguish them. For instance, Nanson's rule satisfies reversal symmetry (i.e. if there is a unique winner under all tie breaking rules and all voters reverse their votes then the winner changes) but Baldwin's rule does not. Finally, there are also desirable properties that neither rule satisfies like monotonicity.

The manipulation problem We can now formally define the different manipulation problems we consider. The *unweighted coalitional manipulation* problem is defined as follows.

Definition 1 *(r-Coalitional-Manipulation).* Given a tuple (P^{NM}, p, M), where P^{NM} is the non-manipulators' profile, p is the candidate preferred by the manipulators, and M is the set of manipulators, does there exist a profile P^M for the manipulators such that $r(P^{NM} \cup P^M) = p$? In other words, does there exist a profile P^M for the manipulators such that candidate p wins an election under the voting rule r and the profile $P^{NM} \cup P^M$?

We drop the word "coalitional" when there is a single manipulator. The *weighted coalitional manipulation* is defined similarly, where the weights of the voters (both non-manipulators and manipulators) are also given as inputs.

Definition 2 *(r-Weighted-Coalitional-Manipulation).* Given a tuple (P^{NM}, p, w, M), where P^{NM} is the non-manipulators' profile, p is the candidate preferred by the manipulators, w is the weighting function and M is the set of manipulators, does there exist a profile P^M for the manipulators such that $r(w, P^{NM} \cup P^M) = p$? In words, does there exist a profile P^M for the manipulators such that candidate p wins an election under the voting rule r and the profile $P^{NM} \cup P^M$?

The corresponding optimisation versions of these problems seek to minimise $|M|$, the number of manipulators.

As is common in much of the literature, we break ties in favour of the coalition of the manipulators. This tie-breaking rule was originally used in [3]; see [18] for a discussion of why this has become a "tradition". We also assume that the manipulators have complete knowledge about the scores from the votes of the non-manipulators. Again, this has become the "tradition" within the literature from some of the earliest work. The argument often put forward for the assumption of complete information is that partial or probabilistic information about the votes of the non-manipulators would add to the computational complexity of computing a manipulation.

Given a set of votes and n manipulators, it is in the best interest of all manipulators to place the preferred candidate, p, first for the Borda rule. Hence, p will have Borda score $s(p) + n(m - 1)$. We define the *gap* of candidate i as $g(i) = s(p) + n(m-1) - s(i)$. For p to win under Borda voting, we need the manipulating votes to give to candidate i (where $i \neq p$) a total Borda score which is less than or equal to $g(i)$. Note that if $g(i)$ is negative for even one i, then p cannot win under Borda voting.

In the proofs of this paper, we will often need to refer to pairs of votes of a particular form. We define the pair of votes $W_{(u,v)} = \{u \succ v \succ \mathit{Others}, \mathrm{rev}(\mathit{Others}) \succ u \succ v\}$ 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 its reverse.

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

3. Unweighted coalitional manipulation

We start by considering the computational complexity of manipulating Borda, Nanson's and Baldwin's 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 also show that Baldwin's and Nanson's rules are NP-complete to manipulate even with a single manipulator.

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 for two manipulators. Our NP-hardness proof uses a reduction from a specialised permutation problem that is strongly NP-complete [42].

Definition 3 *(*Permutation Sum*).* Given q integers $X_1 \leq \cdots \leq X_q$ where $\sum_{i=1}^{q} X_i = q(q+1)$, do there exist two permutations σ and π of 1 to q 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, we can construct, in time polynomial in $\sum_{i=1}^{m} X_i$, votes over $m+1$ candidates such that the total Borda score of candidate c_i is $X_i + C$ for $1 \leq i \leq m$, and for candidate c_{m+1} is $y \leq C$, for some integer $C \geq 0$.*

Proof. This proof uses the construction of McGarvey [28], which has been used elsewhere in the computational social choice literature [41,6]. 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 c_1 by 1 more than candidates c_2 to c_m, and by 2 more than candidate c_{m+1}. Consider the pair of votes $W_{(c_1,c_{m+1})}$ defined in Section 2, and given below:

$$c_1 \succ c_{m+1} \succ c_2 \succ \ldots \succ c_{m-1} \succ c_m$$

$$c_m \succ c_{m-1} \succ \ldots \succ c_2 \succ c_1 \succ c_{m+1}.$$

The score of candidate c_1 increases by $m+1$, of candidates c_2 to c_m by m, and of candidate c_{m+1} by $m-1$. By repeated construction of such votes, we can achieve the desired result. For example, we may construct X_i copies of $W_{(c_i,c_{m+1})}$ for all $1 \leq i \leq m$.

As the number of votes is linear in $\sum_{i=1}^{m} X_i$, the time is polynomial in the sum of the given integers. □

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

Proof. The problem is clearly in NP, since a set of manipulator votes that make the preferred candidate win is a polynomial witness that a manipulation exists.

To show NP-hardness, we reduce from the Permutation Sum problem. Given an instance of Permutation Sum with q integers, X_1 to X_q, we assume, without loss of generality, that $2 \leq X_i \leq 2q$ for all $i \in \{1, \ldots, q\}$. Given such an instance of Permutation Sum, we create a manipulation problem with $m = q+3$ candidates $p, a_1 \ldots, a_{q+2}$ where the preferred candidate of the two manipulators is p. By Lemma 1, we can construct an election in which the non-manipulators cast votes to give the score vector for $\langle p, a_1, \ldots a_{q+2}\rangle$ of:

$$\langle C, 2(q+2) - X_1 + C, \ldots, 2(q+2) - X_q + C, 2(q+2) + C, y\rangle$$

for some $C \geq 0$ and $y \leq C$. We show next that two manipulators can make candidate p win such an election if and only if 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, in which the candidates get the following scores, respectively:

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

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

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

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

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

($\Leftarrow$) Suppose we have a successful manipulation. To ensure candidate p beats candidate a_{q+1}, both manipulators must put candidate p in first place. Similarly, both manipulators must put candidate a_{q+1} in last place, otherwise candidate a_{q+1} will beat our preferred candidate. Hence the final score of candidate p is $2(q+2)+C$. The gap between the final score of candidate p and the current score of candidate a_i (where $1 \le i \le q$) is X_i. The sum of these gaps is $q(q+1)$. Therefore, if any candidate a_1 to a_q gets a score of $q+1$ then candidate p will be beaten. Hence, the two scores of $q+1$ have to go to the least dangerous candidate which is candidate a_{q+2}.

The votes of the manipulators are thus of the form:

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

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

$$2(q+2) - X_j + C + \sigma(j) + \pi(j) \le 2(q+2) + C.$$

Rearranging this gives:

$$\sigma(j) + \pi(j) \le X_j.$$

Since $\sum_{i=1}^{q} X_i = q(q+1)$ and $\sum_{i=1}^{q} \sigma(i) = \sum_{i=1}^{q} \pi(i) = \frac{q(q+1)}{2}$, 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. □

The result of Theorem 1 was proved independently by Betzler et al. [7] using a different reduction from the same problem. Their proof relies on the same basic idea as ours – constructing a set of non-manipulating votes such that the candidates have specific gaps. In contrast to our proof which needs $\Theta(m)$ non-manipulators and a single dummy candidate using the construction of Lemma 1, Betzler et al. use a more complicated construction which introduces $\Theta(m)$ dummy candidates but needs only three non-manipulating votes. It follows therefore that the problem of computing a manipulation is not fixed parameter tractable in the number of voters.

3.2. Baldwin's rule

Our next result is proved by reduction from the exact 3-cover (X3C) problem.

Definition 4 *(X3C).* Given two sets $\mathcal{V} = \{v_1, \ldots, v_q\}$, $q = 3r$, and $\mathcal{S} = \{S_1, \ldots, S_t\}$, where $t \ge 2$ and for all $j \le t$, $|S_j| = 3$ and $S_j \subseteq \mathcal{V}$, does there exist 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 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 $\mathcal{C} = \{p, d, b\} \cup \mathcal{V} \cup \mathcal{A}$, where p is the manipulator's preferred candidate, $\mathcal{A} = \{a_1, \ldots, a_t\}$, 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 is removed, and P_2, which is used to balance the score differences between the candidates.

We make the following observations about the pair of votes $W_{(c_1,c_2)}$, which were defined in Section 2. First, these two votes give the following scores to each candidate

$$\begin{array}{lll} s(c_1): & m & \\ s(c_2): & m-2 & \\ s(e): & m-1 & \text{for } e \in \mathit{Others} \end{array}$$

Second, for any set of candidates $\mathcal{C}' \subseteq \mathcal{C}$ and any pair of candidates $e_1, e_2 \in \mathcal{C} \setminus \mathcal{C}'$,

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

Here $W_{(c_1,c_2)}|_{\mathcal{C}\setminus\mathcal{C}'}$ is the pair of votes obtained from $W_{(c_1,c_2)}$ 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. Additionally, 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 votes of the form $W_{(c_1,c_2)}$ 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 $j \le t$ and each $v_i \in S_j$, there are $2m$ copies of $W_{(v_i,a_j)}$;
- for each $i \le q$, there are m copies of $W_{(b,v_i)}$;
- there are $m(t+6)$ copies of $W_{(b,p)}$.

Let $avg(P_1)$ be the average score of candidates in P_1. That is, $avg(P_1) = s(d, P_1) = (m-1)(6mt + mq + m(t+6))$. Define $occ(i)$ to be the number of occurrences of the element v_i in sets in $\mathcal{S}$. The votes in P_1 give the scores

$$\begin{aligned}
&s(v_i, P_1): && avg(P_1) + 2m \cdot occ(i) - m\\
&s(a_j, P_1): && avg(P_1) - 6m\\
&s(p, P_1): && avg(P_1) - m(t+6)\\
&s(b, P_1): && avg(P_1) + mq + m(t+6)\\
&s(d, P_1): && avg(P_1)
\end{aligned}$$

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

- for each $i \le q$, there are $s(v_i, P_1) - s(p, P_1) - m = 2m \cdot occ(i) + mt + 4m$ copies of $W_{(d,v_i)}$;
- for each $j \le t$, there are $s(a_j, P_1) - s(p, P_1) - 1 = mt - 1$ copies of $W_{(d,a_j)}$;
- there are $s(b, P_1) - s(p, P_1) - mq = 2m(t+6)$ copies of $W_{(d,b)}$.

Let $avg(P_2)$ be the average score of candidates in P_2. That is, $avg(P_2) = s(p, P_2) = (m-1)(2m \cdot \sum_{i=1}^{q} occ(i) + mtq + 4mq + t(mt-1) + 2m(t+6))$. The votes in P_2 give the scores:

$$\begin{aligned}
&s(v_i, P_2): && avg(P_2) - (2m \cdot occ(i) + mt + 4m)\\
&s(a_j, P_2): && avg(P_2) - (mt - 1)\\
&s(p, P_2): && avg(P_2)\\
&s(b, P_2): && avg(P_2) - 2m(t+6)\\
&s(d, P_2): && avg(P_2) + 2m \cdot \textstyle\sum_{i=1}^{q} occ(i) + mtq + 4mq + t(mt-1) + 2m(t+6)
\end{aligned}$$

Let $P = P_1 \cup P_2$, $avg(P) = avg(P_1) + avg(P_2)$. The combined Borda scores are:

$$\begin{aligned}
&s(v_i, P): && avg(P) - m(t+5)\\
&s(a_j, P): && avg(P) - m(t+6) + 1\\
&s(p, P): && avg(P) - m(t+6)\\
&s(b, P): && avg(P) + mq - m(t+6)\\
&s(d, P): && avg(P) + qm(t+5) + t(m(t+6) - 1) + 2m(t+6) - mq
\end{aligned}$$

We make the following observations about the Borda scores of the candidates in P.

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

Suppose the X3C instance has a solution $S_1, \dots, S_{q/3}$ (this is without loss of generality since we can rename the subscripts of the 3-sets in the solution to $\{1, \dots, q/3\}$). Then, we let the manipulator vote as follows:

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

In the following, we use $\mathcal{C}_k$ to denote the set of candidates that have not yet been eliminated after round k.

The candidates with the lowest Borda score before the manipulator's vote are p followed by all a_j's, which all have 1 more, as explained above. With the manipulator's vote, p overtakes all a_j. Moreover, a_1 has the lowest Borda score, which means that a_1 is eliminated in the first round. We next show that for all $j = 1, \dots, q/3$, in round $4j-3$, candidate a_j is eliminated, and in round $4j-2, 4j-1, 4j$, S_j are eliminated.

Suppose this holds for all rounds before round $4j-2$. In round $4j-3$ candidate a_j is eliminated. By Eq. (1), for all $c' \in \mathcal{C}_{4j-3} \setminus (S_j \cup \{d, p\})$, we have

$$s\big(c', P_{|\mathcal{C}_{4j-3}}\big) - s(p, P_{|\mathcal{C}_{4j-3}}) = s\big(c', P_{|\mathcal{C}_{4j-4}}\big) - s(p, P_{|\mathcal{C}_{4j-4}})$$
$$s(d, P_{|\mathcal{C}_{4j-3}}) - s(p, P_{|\mathcal{C}_{4j-3}}) = s(d, P_{|\mathcal{C}_{4j-4}}) - s(p, P_{|\mathcal{C}_{4j-4}}) - (mt-1)$$

and for any $v \in S_j$, we have

$$s(v, P_{|\mathcal{C}_{4j-3}}) - s(p, P_{|\mathcal{C}_{4j-3}}) = s(v, P_{|\mathcal{C}_{4j-4}}) - s(p, P_{|\mathcal{C}_{4j-4}}) - 2m$$

Therefore, in round $4j-2$, the difference between p and the candidates in S_j is decreased by $2m$, which covers their initial difference of m and also the difference in the manipulator's vote (as this can be at most $m-2$). Meanwhile, d is still leading by a large margin. Therefore, in rounds $4j-2, 4j-1, 4j$, the candidates in S_j are eliminated (the order of elimination does not matter). Eliminating each $v \in S_j$ has three effects on score difference between p and other candidates:

1. the score difference between all a_k with $v \in S_k$ and p is increased by $2m$;
2. the score difference between b and p is decreased by m;
3. the score difference between d and p is decreased by the number of copies of $W_{(d,v)}$ in P.

These do not affect the fact that candidates in S_j are eliminated in rounds $4j-2, 4j-1, 4j$, and we also note that for all $j < k \le q/3$, the score difference between p and a_k does not change. Therefore, in round 4_j+1 candidate $j+1$ is eliminated.

Continuing, after the first $4q/3$ rounds, all candidates v_i are eliminated, each decreasing the score difference between b and p by m for a total of mq. Hence b and p are tied for the lowest total Borda score in P (it is not hard to verify that other a_k and d still have higher scores). Considering the manipulator's vote, b is eliminated next. This decreases the difference between d and p by $3m(t+6)$ (which is the cumulative effect of the third set of votes in P_1 and the third set of votes in P_2), and between $a_{q/3+1}, \dots, a_q$ and p by $m(t+6)$ (by the third set of votes in P_1). It follows that in the next $t-q/3$ rounds, all remaining candidates in $\mathcal{A}$ are eliminated. Finally, when only p and d remain, they are tied in P. But since the manipulator prefers p to d, p is the winner.

Suppose the manipulator can cast a vote to make p the winner. We first note that, from the votes in P_1, as long as not all of the candidates in $\mathcal{V}$, $\mathcal{A}$ and b are eliminated, the difference between the score of d and p is greater than m, so it cannot be covered by the vote of the manipulator. Hence, d must be eliminated after these other candidates, that is, in the last round. In the round when b is eliminated, the score of b can be no more than the score of p. We note that $s(b,P) - s(p,P) = mq$ and the score difference can only be reduced by the manipulator ranking b below p, and by eliminating $v_1, \dots, v_q$ before b. However, ranking b below p reduces the score difference by no more than $m-1$ and eliminating any single candidate in $\mathcal{V}$ reduces the difference by m. Therefore, before b drops out, all q candidates in $\mathcal{V}$ must have already dropped out. We note that for any $v_i \in \mathcal{V}$, $s(v_i, P) - s(p,P) = m$, so the manipulator's vote cannot cover this difference. Also the only other way to reduce this difference is by eliminating some a_j with $v_i \in S_j$. Therefore, for each $v_i \in \mathcal{V}$, there exists at least one a_j with $v_i \in S_j$ that is removed before v_i. For any such a_j, no candidate $v_i \in S_j$ can drop out before a_j. Otherwise the difference between a_j and p is increased by $2m$, reaching $2m+1$. Therefore, before b drops out, this difference cannot be covered by the manipulator's vote, which means that p drops out before a_j. This is a contradiction since we assume that p drops out after b. On the other hand, no other candidate a_k with $S_j \cap S_k \neq \emptyset$ can drop out before b, because when the candidates in the intersection of S_j and S_k drop out, the difference between a_k and p is increased by $2m$ and becomes positive. Finally, we note that after a_j drops out, 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 the candidates in $\mathcal{V}$ that they cover corresponds to an exact cover of $\mathcal{V}$. After all the candidates in $\mathcal{V}$ drop out, b drops out, followed by the rest of the candidates in $\mathcal{A}$ and then d, as above.

Therefore, the unweighted manipulation problem under Baldwin's rule is NP-complete with only a single manipulator. □

3.3. Nanson's rule

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

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

Proof. The idea of the proof is similar to that of the proof of Theorem 2. We prove NP-completeness by reduction from X3C with $t \ge 3q$ (this is without loss of generality because if $t < q$ then we can add dummy S_1's to $\mathcal{S}$). Given an X3C instance $\mathcal{V} = \{v_1, \dots, v_q\}$, $\mathcal{S} = \{S_1, \dots, S_t\}$, we let the set of alternatives be $\mathcal{C} = \{p, d, b_1, b_2\} \cup \mathcal{V} \cup \mathcal{A}$, where p is the manipulator's preferred candidate, $\mathcal{V} = \{v_1, \dots, v_q\}$, $\mathcal{A} = \{a_1, \dots, a_t\}$, and d, b_1, and b_2 are auxiliary alternatives. Without loss of generality, both q and t are even, and $t \ge 3q$. We will use the votes $W_{(c_1,c_2)}$ defined in Section 2 to construct the profile. For any $\mathcal{C}' \subsetneq \mathcal{C}$, we make the following observations about $W_{(c_1,c_2)}$.

$$s\big(c', W_{(c_1,c_2)}|_{\mathcal{C}\setminus\mathcal{C}'}\big) - |\mathcal{C}\setminus\mathcal{C}'| + 1 = \begin{cases} 1 & \text{if } c' = c_1 \text{ and } c_2 \notin \mathcal{C}' \\ -1 & \text{if } c' = c_2 \text{ and } c_1 \notin \mathcal{C}' \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

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 of the following votes:

- for every $j \le t$ there are $7m/2 - q/3$ copies of $W_{(a_j,b_1)}$;

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

The second part of the profile, P_2, consists of the following votes, where $occ(v_i)$ is the number of times that v_i is covered by the 3-sets in $\mathcal{S}$:

- for any $i \leq q$, there are $m \cdot occ(v_i)$ copies of $W_{(d,v_i)}$,

Let $P = P_1 \cup P_2$ and let $avg(P) = (m-1)|P|/2$. We make the following observations about P:

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

Suppose the X3C instance has a solution $\{S_1, \ldots, S_{q/3}\}$ (this is without loss of generality since we can rename the subscripts of the 3-sets in the solution to $\{1, \ldots, q/3\}$). Then, we let the manipulator vote as follows:

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

The manipulator's vote does not change the score of any candidate with respect to the average by more than $(m-1)/2$, therefore b_2 will be eliminated in the first round. The difference between the score of candidate a_j for $1 \leq j \leq q/3$ and the average will be $m/2 - q/3 - ((m-1)/2 - j + 1)$. Since $j \leq q/3$, this is seen to be less than or equal to $-1/2$, hence these candidates are also eliminated in the first round. No other candidates are eliminated in the first round: for all candidates in $\mathcal{V}$, the manipulator's vote is not enough to make their score less than the average, while all other candidates receive more than the average from the manipulator.

Let $\mathcal{C}_1 = \mathcal{C} \setminus \{a_1, \ldots, a_{q/3}\}$. In the second round, by Eq. (2), for any $v_i \in \mathcal{V}$, $s(v_i, P|_{\mathcal{C}_1}) - avg(P|_{\mathcal{C}_1}) = 0$. The reason is that each v_i gets $m(occ(i) - 1)$ points from the second part of P_1 because $\{S_1, \ldots, S_{q/3}\}$ covers $\mathcal{V}$, m points from the third part of P_1, and $-m \cdot occ(i)$ points from P_2. Counting in the manipulator's vote and recalling that we assumed $t \geq 2q$, we have that the scores of all candidates in $\mathcal{V}$ are below the average score. Moreover, because b_2 was eliminated in the first round, the score of b_1 is now below the average. Therefore, b_1 and all the candidates in $\mathcal{V}$ are the only candidates eliminated in the second round.

Let $\mathcal{C}_2 = \mathcal{C} \setminus (\{b_1, b_2, a_1, \ldots, a_{q/3}\} \cup \mathcal{V})$. In the beginning of the third round, because b_1, b_2, and all candidates in $\mathcal{V}$ were eliminated, we have that for each $W_{(c_1,c_2)}$ in P, at least one of c_1 and c_2 was eliminated. Therefore, the score of all remaining candidates is the same as the average score in $P|_{\mathcal{C}_2}$. Moreover, for the same reason, in any later round the score of all remaining candidates is the same as the average score in P (restricted to the remaining candidates). Therefore the manipulator's vote becomes decisive. It follows that in each of the following rounds, the candidates ranked below the mid-position in the manipulator's votes are eliminated. The final winner is the manipulator's top-ranked candidate, which is p.

Next, we show that if the manipulator can cast a vote to make p win, then there exists a solution to the X3C instance. In the first round b_2 definitely drops out. This makes the score of b_1 below the average score in the second round (from the $mq + t(7m/2 - q/3)$ copies of $W_{(b_1,b_2)}$ in P_1), which means that b_1 will definitely drop out in the second round. If any of the v_i candidates remain in the third round, then the score of p will be strictly lower than the average score (from m copies of $W_{(v_i,p)}$ in P_1). Therefore, all alternatives in $\mathcal{V}$ must drop out in the first and second rounds. In fact, v_i's 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 be ranked among the bottom $q/3$ positions). Therefore, in order for p to survive 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 manipulation problem under Nanson's rule is NP-complete when there is only one manipulator. □

Our results about the complexity of manipulating Baldwin's and Nanson's rules significantly increase the size of the set of voting rules used in practice that are known to be NP-hard to manipulate with a *single* manipulator. They also contrast to Borda where computing a manipulation with a single manipulator can be done in polynomial time [3]. 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-time to NP-hard.

4. Weighted coalitional manipulation

In this section we show that weighted coalitional manipulation under Baldwin's or Nanson's rules is NP-complete. It has already been shown that the weighted coalitional manipulation problem for Borda is NP-hard for three or more candidates [12].

4.1. Baldwin's rule

Similar to the case of Borda, we prove that the weighted coalitional manipulation problem for Baldwin's rule is NP-hard for three or more candidates. Our result is proved by reduction from the Partition problem.

Definition 5 *(Partition).* Given a set of integers $A = \{k_1, \dots, k_q\}$ such that $\sum_{i=1}^{q} k_i = 2K$, does there exist a partition of these numbers into two sets the elements in each of which sum to K?

A partition that witnesses the satisfiability of a Partition instance is called a *perfect* partition.

Theorem 4. *For Baldwin's rule and weighted votes, the coalitional manipulation problem is NP-hard with three or more candidates.*

Proof. We reduce from the partition problem. We construct a coalitional manipulation problem with three candidates (a, b, and p) in which the manipulators want to make p win.

We suppose the non-manipulators have voted as in the following table.

weights	votes
$11K$	$a \succ b \succ p$
$5K$	$a \succ p \succ b$
$14K$	$b \succ p \succ a$
$2K - 1$	$b \succ a \succ p$
$5K$	$p \succ a \succ b$
$5K$	$p \succ b \succ a$

At this point the scores of the candidates are

$s(a)$:	$39K - 1$
$s(b)$:	$48K - 2$
$s(p)$:	$39K$.

For each integer $k_i \in A$, 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. 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 will not be eliminated and will win. Thus all the votes of the manipulators are $a \succ p \succ b$ or $p \succ a \succ b$. Consider the following partition of A constructed from any successful manipulation. In the first half of the partition, we put all integers associated with weighted votes of the manipulators of the form $a \succ p \succ b$. In the second half, we put all integers associated with weighted votes of the form $p \succ a \succ b$. Suppose the first half of the partition sums up to $K - x$ and the second half 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. For more than three candidates, we add "harmless" candidates that are in the last places of every vote of the non-manipulators. □

Note that Coleman and Teague in Theorem 13 of [11] provide an NP-hardness result for the weighted coalitional 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 do not specify a precise bound on the number of candidates, while we present a proof that weighted coalitional manipulation under Baldwin's rule is NP-hard for just three candidates.

4.2. Nanson's rule

We show that the weighted coalitional manipulation problem under Nanson's rule is NP-complete with four or more candidates. However, if there are at most three candidates, the computational complexity of computing a manipulation under Nanson's rule is polynomial-time.

Theorem 5. *For Nanson's rule and weighted votes, the coalitional manipulation problem is NP-complete with four or more candidates.*

Proof. We reduce from PARTITION. For any PARTITION instance, we construct a coalitional manipulation problem with four candidates (a, b, c, and p) where p is the candidate that the manipulators wish to win. We suppose the non-manipulators have voted as in the following table.

weights	votes
$2(2K+1)$	$b \succ p \succ c \succ a$
$2(2K+1)$	$a \succ c \succ b \succ p$
$2(2K+1)$	$c \succ p \succ b \succ a$
$2(2K+1)$	$a \succ b \succ c \succ p$
$2(K+2)$	$p \succ a \succ b \succ c$
$2(K+2)$	$c \succ b \succ p \succ a$
1	$p \succ a \succ b \succ c$
1	$a \succ b \succ c \succ p$
1	$a \succ b \succ p \succ c$
2	$c \succ p \succ a \succ b$
2	$a \succ c \succ p \succ b$
2	$b \succ p \succ a \succ c$

The total scores from non-manipulators are as follows: $s(a) = 28K + 38$, $s(b) = 34K + 37$, $s(c) = 34K + 37$ and $s(p) = 24K + 38$. The average score is $30K + 37.5$. For each integer k_i, we have a member of the manipulating coalition with weight $2k_i$. 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 $p \succ a \succ c \succ b$.

The total scores are now as follows: $s(a) = 36K + 38$, $s(b) = 36K + 37$, $s(c) = 36K + 37$ and $s(p) = 36K + 38$. The average score is $36K + 37.5$.

The alternatives b and c are eliminated, and p wins in the second round. Thus the manipulators can make p win if a perfect partition exists.

Conversely, suppose there is a successful manipulation. Clearly, we need to ensure that p is not eliminated in the first round. To ensure this, all manipulators must rank p first. Otherwise, the score of p would be less than the average score $36K + 37.5$, so p would be eliminated. Next, we show that if b and c are not eliminated in the first round, p cannot win overall. We consider all possible sets of candidates besides b and c that could be eliminated in the first round. There are six cases.

1. only a is eliminated in the first round. The scores from non-manipulators in the second round are as follows: $s(b) = 24K + 27$, $s(c) = 24K + 27$, and $s(p) = 12K + 21$. The average score is $20K + 25$. Even with the maximum possible score of $8K$ from the manipulators, p is eliminated. This contradicts the assumption that p wins.
2. only b is eliminated in the first round. Note that if a is not ranked second in the votes of all manipulators, its score will be less than the average $36K + 37.5$ and it will be eliminated. This contradicts our assumption that p and a are not eliminated in the first round. Hence, all manipulators have to cast votes that rank p first and a second. The votes of the manipulators in the second round will then be $p \succ a \succ c$, giving scores $s(a) = 22K + 23$, $s(c) = 24K + 25$, and $s(p) = 26K + 27$. The average score is $24K + 25$. Hence, a is eliminated. In the next round, p is eliminated, as $s(p) = 10K + 10$, $s(c) = 14K + 15$, and the average score is $12K + 12.5$. This contradicts the assumption that p wins.
3. only c is eliminated in the first round. This case is symmetric to the second case.
4. a and b are eliminated in the first round. In the second round, the scores from non-manipulators are $s(c) = 14K + 15$ and $s(p) = 6K + 10$. The $4K$ points from the manipulators cannot now prevent p from being eliminated. This contradicts the assumption that p wins.
5. a and c are eliminated in the first round. This is symmetric to the fourth case.
6. a, b, and c are all eliminated in the first round. This case is impossible because if b and c are eliminated then a must get $8K$ points from the manipulators. Hence, a reaches the second round.

Thus, the only way for p to win is if b and c are eliminated in the first round. For this to occur, the manipulators have to put p in first place, and a in second place. If b gets more than a score of $2K$ from the manipulators in the first round, then its total score will be greater then the average of $36K + 37.5$ and it will not be eliminated in the first round. Similarly, if c gets more than a score of $2K$ from the manipulators, then it will not eliminated in the first round. However, as both the first and second place in the manipulators votes are fixed, there is exactly $4K$ points to divide between them. Hence, they must divide the $4K$ points equally. Hence, there exists a solution to the PARTITION instance. For more than four candidates, we add "harmless" candidates that are in last place in every vote of the non-manipulators. $\square$

Clearly, there is a polynomial-time algorithm to compute a manipulation of Baldwin's rule with two candidates (since this case degenerates to majority voting). For Nanson's rule, on the other hand, there is a polynomial-time algorithm for up to three candidates.

Theorem 6. *For Nanson's rule and weighted votes, the coalitional manipulation problem can be solved in polynomial time for up to three candidates.*

Proof. Consider an election with three candidates (a, b, and p) in which the manipulators want p to win. We prove that in a successful manipulation, either all manipulators vote $p \succ a \succ b$ or they all 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 harm in raising p to the first position while keeping the other parts of their preferences the same. By doing so, we ensure that 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 a profile for the manipulators that makes p win. Because Nanson's rule never selects the Condorcet loser, it cannot be the case that both a majority of voters prefer a to p, and b to p. Without loss of generality, suppose that a majority of voters prefer p to 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 a majority of voters prefer b to 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 a majority of voters prefer b to 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 $p \succ b \succ a$. $\square$

The reason that the above algorithm does not work for manipulating Baldwin's rule is that the algorithm requires that we can place p as the first choice in every manipulating vote. However, in a successful manipulation in the proof of Theorem 4, the manipulators are split between $p \succ a \succ b$ and $a \succ p \succ b$, and switching the votes of the latter group into $p \succ a \succ b$ spoils the manipulation.

The results in this section suggest that Baldwin's rule is arguably harder to manipulate because Nanson's rule is polynomial to manipulate with three candidates, and requires at least four candidates to be NP-hard, but Baldwin's is NP-hard already with three 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 [12]. Note that the coalitional manipulation problem for Borda with weighted votes is NP-hard for three or more candidates [12]. Thus, somewhat surprisingly, adding an elimination round to Borda, which gives us Nanson's rule, decreases the computational complexity of computing a manipulation with three manipulators from NP-hard to polynomial-time.

5. Heuristic methods

NP-hardness only characterises the worst-case complexity of computing a manipulation. Given enough manipulators, we can easily make any candidate win. We consider next minimising the number of manipulators required. For example, REVERSE is a simple heuristic method proposed to compute Borda manipulations [43]. The method constructs each manipulator's vote in turn: preferred candidate p 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 p 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 four candidates, c_1, c_2, c_3, p, and the two non-manipulators have cast votes: $c_3 \succ c_1 \succ c_2 \succ p$ and $c_2 \succ c_3 \succ c_1 \succ p$. Then we have the score vector $\langle 3, 4, 5, 0\rangle$. We use REVERSE to construct a manipulation that makes candidate p win. REVERSE first constructs the vote: $p \succ c_1 \succ c_2 \succ c_3$. The score vector is now $\langle 5, 5, 5, 3\rangle$. REVERSE next constructs the vote: $p \succ c_1 \succ c_2 \succ c_3$. (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: $p \succ c_3 \succ c_2 \succ c_1$. The score vector is $\langle 7, 7, 7, 9\rangle$. Hence, REVERSE requires three manipulating votes to make candidate p win. As we will see later, this is one more vote than in the optimal solution.

Following [43], we propose four new heuristic 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 rules. All algorithms attempt to construct a manipulation with a specific number of manipulators. Hence, in order to find the best possible number of manipulators using one of these algorithms, we run it for one manipulator, then two and so on until a manipulation is found. We refer to a manipulation with k manipulators as a k-manipulation.

5.1. Manipulation matrices

In this section we prove some auxiliary results that are needed to develop our heuristic algorithms.

We can view Reverse as greedily constructing a manipulation matrix. A *manipulation matrix* is an n by m matrix A, where n is the number of manipulators, m is the number of candidates, and $A(i, j) = k$ if and only if the ith manipulator adds a score of k to candidate c_j. The jth column of the matrix, $A(j)$, is the set of scores received by candidate c_j, and each of the n rows is a permutation of 0 to $m - 1$. We require that the sum of the jth column is less than or equal to $g(c_j)$, the maximum score candidate c_j can receive without defeating p. Reverse constructs this matrix row by row.

Our new methods break out of the straightjacket of constructing a manipulation matrix in row-wise order. To achieve this, we 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 $g(c_j)$. In a relaxed manipulation matrix, a row can repeat an integer provided other rows compensate by not having that integer at all.

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

Proof. The proof is by induction on n. In the base case, $n = 1$ and we just set $B(1, j) = A(1, j)$ for all $j \in \{1, \dots, m\}$. 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 as follows. The first half of the bipartite graph consists of m vertices $\{V_i\}|_{i=0}^{m-1}$, while the second half of the graph consists of m vertices $\{W_j\}|_{j=1}^{m}$. Each vertex V_i represents a score, i, that must appear in the first row of B. Each vertex W_j represents the jth column of A.

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 $A(k, j) = i$. An edge (V_i, W_j) therefore means that score i can be taken from the jth column of A.

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 $U \subseteq \{V_i \mid i \in [0, m - 1]\}$. Recall first that the Hall condition [25] states that a perfect matching exists if and only if $|V| \le |N(V)|$ for all sets of vertices V (where $N(V)$ is the neighbourhood of V). Since the degree of each vertex is n, there are $n|U|$ edges leaving U. For the same reason, each vertex in $N(U)$ can accommodate at most n incoming edges. Therefore $n|U| \le n|N(U)|$. Hence, the Hall condition holds and a perfect matching exists. 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 heuristic 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 heuristic method, Largest Fit is inspired by bin packing and multiprocessor scheduling. Constructing an n by m relaxed manipulation matrix is similar to packing nm objects into m bins with a constraint on the capacity of the different bins and an extra constraint on the number of items (n) that can be placed in each bin. The problem is also similar to scheduling nm unit length jobs with different memory requirements on n different processors with a constraint on the total memory footprint of the n different jobs running at every clock tick and schedule length fixed to m. Krause et al. [27] 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.

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 p of the matrix (since it is best for the manipulators to put p in first place in their votes). It then allocates the remaining $(n-1)m$ numbers in reverse order to the columns corresponding to the candidate with the currently 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 Example 1. We start with the score vector $\langle 3, 4, 5, 0\rangle$. One manipulator alone cannot increase the score of candidate p enough to beat c_2 or c_3. Therefore, we need at least two manipulators. LARGEST FIT first puts two 3s 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 score 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: $p \succ c_2 \succ c_1 \succ c_3$ and $p \succ c_1 \succ c_3 \succ c_2$. With these votes, p wins based on the tie-breaking rule. Unlike REVERSE, LARGEST FIT constructs an optimal manipulation with just two manipulators.

As we show in Section 5.5, LARGEST FIT and REVERSE, are, in fact, incomparable. There is an infinite family of problems on which LARGEST FIT finds an optimal manipulation but REVERSE does not, and vice versa.

5.3. AVERAGE FIT

Our second heuristic method, AVERAGE FIT takes into account 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 fewer 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. Each manipulator gives their largest score, $m-1$, to the preferred candidate p and then has to distribute their remaining scores among other candidates. Initially, a manipulator has $m-1$ scores to distribute. We call all manipulator scores that have not been distributed so far “unassigned scores”. At each step, AVERAGE FIT selects a column and a score to distribute to that column. First, the column is chosen, by selecting the column for which the size of the remaining gap divided by the number of scores still to be added to the column is largest. We tie-break by choosing the column containing the fewest scores. Then, an unassigned score is chosen to distribute to that column. We choose the largest unassigned score that will fit into that column's gap. If there is no unassigned score that will fit into the gap, then the largest unassigned score is chosen.

Example 3. Consider again Examples 1 and 2. We start with the score vector $\langle 3, 4, 5, 0\rangle$. This method computes, identically to LARGEST FIT, that two manipulators are needed. Like LARGEST FIT, AVERAGE FIT first puts two 3s 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. AVERAGE FIT puts this into column 1 as this has the largest average 3/2. This gives the score vector $\langle 5, 4, 5, 6\rangle$. The next largest score is again 2. AVERAGE FIT puts this into column 2, which has average $2/2 = 1$, giving the score vector $\langle 5, 6, 5, 6\rangle$. The two next largest scores are 1. AVERAGE FIT puts the first 1 in column 1, which has average 1/1 and the next 1 in column 3 which has average 1/2. This gives 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 is identical to the manipulation computed by LARGEST FIT, with votes $p \succ c_2 \succ c_1 \succ c_3$ and $p \succ c_1 \succ c_3 \succ c_2$.

For an example on which AVERAGE FIT beats LARGEST FIT, see Theorem 9. For an example on which LARGEST FIT beats AVERAGE FIT, see Theorem 10.

5.4. *Eliminate and Reverse Eliminate*

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

The first method, which we call ELIMINATE, repeatedly constructs 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 penultimate 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.

Our final method, which we call 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.

Example 4. We revisit Examples 1–3 and show the operation of ELIMINATE for Baldwin's rule. The initial score vector is $\langle 3, 4, 5, 0\rangle$, so p is eliminated in the first round. In the second round, the score vector is $\langle 1, 2, 3\rangle$, so c_1 gets eliminated, and in the last round c_3 and c_2 are in a tie with the score vector $\langle 1,1\rangle$. We assume the tie is broken in favour of c_2, so it wins the election. Therefore, ELIMINATE will have the first manipulator vote $p \succ c_2 \succ c_3 \succ c_1$. With one manipulator, this gives the score vector $\langle 3,6,6,3\rangle$. With tie breaking in favour of p, c_1 is eliminated in the first round, so the score vector in the second round is $\langle 4,3,2\rangle$, so p is eliminated in the second round. This means that we need at least one more manipulator. By construction, ELIMINATE can only change the step in which p is eliminated. The other candidates are eliminated in the same order amongst themselves. Hence, the vote of the second manipulator is also $p \succ c_2 \succ c_3 \succ c_1$. The initial score vector is now $\langle 3,8,7,6\rangle$. The candidate c_1 is again eliminated in the first round. In the second round, the score vector is $\langle 5,3,4\rangle$. Therefore, c_3 is eliminated. In the third round, c_2 and p are tied with the score vector $\langle 2,2\rangle$. Since we break ties in favour of the preferred candidate, ELIMINATE has computed a manipulation with two manipulators. This is optimal.

With REVELIMINATE, the first manipulator votes $p \succ c_1 \succ c_3 \succ c_2$. This gives the score vector $\langle 5,4,6,3\rangle$. Hence p is again eliminated in the first round. The score vector in the second round is $\langle 3,2,4\rangle$. Hence c_2 is eliminated. In the third round, The score vector in the second round is $\langle 1,2\rangle$. Hence c_1 is eliminated and c_3 wins. The vote of the second manipulator is therefore $p \succ c_2 \succ c_1 \succ c_3$. This gives the score vector $\langle 6,6,6,6\rangle$. We suppose tie breaking eliminates c_3. The score vector in the second round is $\langle 4,4,4\rangle$. We suppose tie breaking now eliminates c_2. The score vector in the third round is $\langle 2,2\rangle$. Since we break ties in favour of p, REVELIMINATE has also computed an optimal manipulation.

5.5. Theoretical properties

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

Theorem 8. *For Borda voting, there exists an election for which* LARGEST FIT *finds an optimal 2-manipulation, but* REVERSE *produces a 3-manipulation.*

Proof. We suppose there are just two non-manipulators with votes σ and σ', and the preferred candidate p is c_m. Let $\sigma = \langle 1,2,\dots,m-1,0\rangle$ and let

$$\sigma' = \left\langle \frac{m}{2}+1, \frac{m}{2}+2, \dots, \frac{m}{2}+\frac{m}{2}-1, 1, 2, \dots, \frac{m}{2}, 0\right\rangle.$$

Then

$$\sigma+\sigma' = \left\langle \left(1+\frac{m}{2}+1\right), \left(2+\frac{m}{2}+2\right), \dots, \left(\frac{m}{2}-1+\frac{m}{2}+\frac{m}{2}-1\right), \left(\frac{m}{2}+1\right), \dots, \left(m-1+\frac{m}{2}\right), 0\right\rangle.$$

This gives $\frac{m}{2}+2x$ for $1 \le x \le \frac{m}{2}-1$ and $\frac{m}{2}+2x-1$ for $1 \le x \le \frac{m}{2}$, or in other words, there exists a score $\frac{m}{2}+i$ for all $1 \le i \le m-1$. Before continuing, we rename the candidates so that $s(c_m, P) = 0$ and $s(c_i, P) = \frac{m}{2}+i$ for all $1 \le i \le m-1$.

Recall that the preferred candidate is $p = c_m$. The first vote generated by REVERSE is $v_1 = p \succ c_1 \succ c_2 \succ \dots \succ c_{m-1}$, after which $s(c_i, P \cup \{v_1\}) = \frac{m}{2}+m-1$ for all candidates $c_i \neq c_m$. This is larger than the score of the distinguished candidate $s(p, P \cup \{v_1\}) = m-1$. Therefore another manipulator is added. Without loss of generality, we suppose its vote is $v_2 = c_m \succ c_1 \succ c_2 \succ \dots \succ c_{m-1}$. The resulting scores of the competing candidates are $s(c_i, P \cup \{v_1, v_2\}) = \frac{m}{2}+(m-1)+(m-i-1) = (5/2)m-2-i$. So candidate c_1 still has larger score than $s(p, P \cup \{v_1, v_2\}) = 2m-2$. Therefore, REVERSE does not find a 2-manipulation. Note that, as REVERSE never uses more than one additional manipulator than is optimal, it will successfully find a 3-manipulation.

LARGEST FIT will first give the highest scores, $m-1$, from both manipulators to the preferred candidate. Then in each iteration, LARGEST FIT will place a score from the multi-set $S_2 = \{0,\dots,m-2\} \uplus \{0,\dots,m-2\}$ into the manipulation matrix B. The first $m-1$ iterations of LARGEST FIT will place the kth largest score from S_2 into the kth column of matrix B for $1 \le k \le m-1$. Note that the kth largest score is $m-2-\lfloor (k-1)/2 \rfloor$. Let B_{m-1} be the tentative manipulation matrix at this point and write $sum(B_{m-1}(i))$ for the sum of the elements of its ith column. Then, the score of candidate c_i under this partial manipulation is $sum(B_{m-1}(i)) + s(c_i, P) = (m-2-\lfloor (i-1)/2 \rfloor) + \frac{m}{2}+i$ for all i, hence $sum(B_{m-1}(i)) + s(c_i, P) \le sum(B_{m-1}(i+1)) + s(c_{i+1}, P)$ for all $1 \le i \le m-2$, and so the relative order of the candidates' scores does not change. The multi-set of scores available at this point is $S'_2 = \{0,\dots,\frac{m}{2}-1\} \uplus \{0,\dots,\frac{m}{2}-2\}$. The next $m-1$ iterations of LARGEST FIT will place the kth largest score from S'_2 into the kth column of matrix B for $1 \le k \le m-1$. 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. The score of candidate c_i under the manipulation $B_{2(m-1)}$ is $sum(B_{2(m-1)}(i)) + s(c_i, P) = (m-2-\lfloor (i-1)/2 \rfloor) + (\frac{m}{2}+i) + (\frac{m}{2}-1-\lceil (i-1)/2 \rceil) = 2(m-1)$ for all i, while the achievable score of candidate p is also $2(m-1)$. Therefore, LARGEST FIT finds a 2-manipulation. Fig. 1 illustrates how the scores are placed in matrix B by LARGEST FIT, where the shaded area represents the scores $s(c_i, P)$. $\square$

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

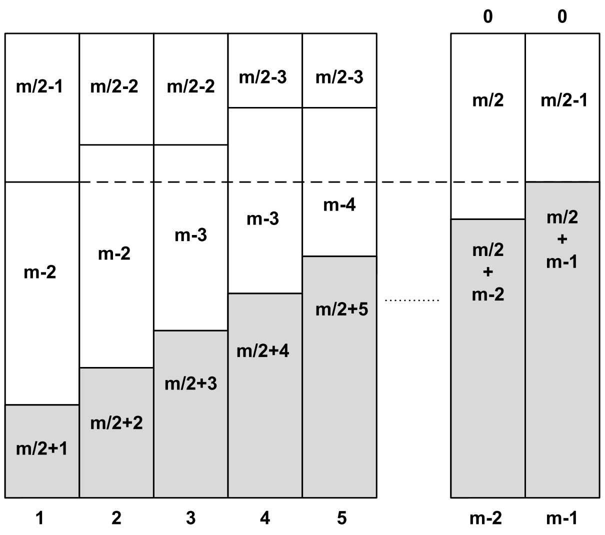


Fig. 1. The 2-manipulation generated by Largest Fit for the election in Theorem 8.

Theorem 9. *For Borda voting, there exists an election with 4 candidates and 2k votes (k divisible by 36) on which both* Reverse *and* Average Fit *will find an optimal manipulation but* Largest Fit *requires at least $k/9 - 3$ additional manipulators.*

Proof. Consider a Borda election in which the scores of four candidates after the non-manipulators P vote are $s(c_1, P) = 6k$, $s(c_2, P) = 4k$, $s(c_3, P) = 2k$, $s(p, P) = 0$. These scores can be achieved if all $2k$ votes are $c_1 \succ c_2 \succ c_3 \succ p$. Reverse will use $2k$ manipulators, all voting $p \succ c_3 \succ c_2 \succ c_1$, to achieve a score of $6k$ for all candidates. This is the only optimal manipulation. To see that Average Fit will find the optimal manipulation, note that the initial gaps are $0, 2k$, and $4k$ and the averages $0, 1$, and 2 for candidates c_1, c_2 and c_3, respectively. In the first step, Average Fit will assign a score of 2 to candidate c_3 and will continue to do that as long as the average of candidate c_3 is greater than that of candidate c_2. To find when that happens, we let x be the number of iterations and solve $\frac{4k-2x}{2k-x} = 1 \Rightarrow x = 2k$. This means that Average Fit will give all $2k$ scores of 2 to candidate c_3. Similarly, we see that it will give all scores of 1 to candidate c_2 and only scores of 0 to candidate c_1. This means that all manipulators will vote $p \succ c_3 \succ c_2 \succ c_1$, the only optimal manipulation.

It remains to argue that Largest Fit requires more than $2k + k/9 - 4$ manipulators. We exploit the fact that Largest Fit is monotonic, in the sense that if it finds a successful Borda manipulation with a given number of manipulators, it also succeeds with more. Additional manipulators only give Largest Fit more opportunity to increase the score of the preferred candidate over the other candidates. 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. By monotonicity, Largest Fit cannot use $2k + k/9 - 4$ or fewer manipulators. Note that given our definition of n, since k is divisible by 4 and 9, $\frac{n-k}{2}$ is an integer.

Let B denote the relaxed manipulation matrix constructed by Largest Fit, and let $B(i)$, $i \in \{1, \ldots, m\}$ denote its ith column. We write $sum(B(i))$ for the sum of the elements in $B(i)$. First, the algorithm will place k 2's in $B(3)$, at which point $sum(B(3)) = 2k + 2k = 4k = s(2, P)$. 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, neither $B(3)$ nor $B(2)$ is full yet ($B(2)$ 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, Average Fit finds optimal manipulations on the elections in the proof of Theorem 9. However, there exist examples on which Largest Fit finds an optimal manipulation but Average Fit does not.

Theorem 10. *For Borda voting, there exist an election on which* LARGEST FIT *finds an 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 manipulators wish candidate c_8 to win, and 8 non-manipulators have voted as follows:

#voters	vote
3	$c_1 \succ c_2 \succ c_3 \succ c_4 \succ c_5 \succ c_6 \succ c_7 \succ c_8$
1	$c_1 \succ c_2 \succ c_3 \succ c_4 \succ c_5 \succ c_7 \succ c_6 \succ c_8$
1	$c_1 \succ c_2 \succ c_3 \succ c_6 \succ c_5 \succ c_4 \succ c_7 \succ c_8$
1	$c_7 \succ c_1 \succ c_6 \succ c_5 \succ c_4 \succ c_2 \succ c_3 \succ c_8$
2	$c_8 \succ c_7 \succ c_6 \succ c_5 \succ c_4 \succ c_3 \succ c_2 \succ c_1$

This gives the score vector for $\langle c_1, \ldots, c_8\rangle$ of:

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

On this problem, LARGEST FIT finds an optimal manipulation that makes the final candidate win but AVERAGE FIT requires an additional vote. The calculations are shown in Appendix A. □

So far we have not found any instances where REVERSE performs better than AVERAGE FIT.

Finally, we consider properties of heuristic algorithms with respect to Baldwin's and Nanson's rules. It appears that it is harder to find an approximately optimal manipulation for these rules than for the Borda rule. For all our heuristic methods, we can give examples where the heuristic computes a manipulation that uses several more manipulators than is optimal. The most interesting result is that although REVERSE was shown to never require more than one extra manipulator than optimal under the Borda rule [43], 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. *For Baldwin's rule, there exists an election with 7 candidates and* $42n$ *votes (n even) where* REVERSE *computes a manipulation with at least* $10n$ *more votes than is optimal.*

Proof. Consider an election over candidates a, b, c, d, e, f, and p where p is the preferred candidate of the manipulators. We define $V_{(u,v)}$ as the pair of votes: $\{u \succ v \succ Others \succ p, \text{rev}(Others) \succ u \succ v \succ p\}$, where *Others* is some fixed ordering of the other candidates and rev(*Others*) is its reverse. Note that these votes differ from the pair of votes $W_{(u,v)}$ defined in Section 2. The non-manipulators cast the following votes: $3n$ copies of $V_{(a,b)}$, $V_{(b,c)}$, $V_{(c,d)}$, $V_{(d,e)}$, and $V_{(e,f)}$. In addition, there are $6n$ copies of the votes: $p \succ a \succ Others$ and $\text{rev}(Others) \succ p \succ a$. After the non-manipulators have voted, $s(a) = s(f) = 138n$, $s(b) = s(c) = s(d) = s(e) = 141n$, and $s(p) = 42n$.

If $18n$ manipulators vote identically $p \succ a \succ \ldots \succ f$ then p wins. The following table shows scores of all candidates in 6 rounds.

	$s(a)$	$s(b)$	$s(c)$	$s(d)$	$s(e)$	$s(f)$	$s(p)$
Round 1	$228n$	$213n$	$195n$	$177n$	$159n$	$138n$	$150n$
Round 2	$189n$	$174n$	$156n$	$138n$	$117n$	–	$126n$
Round 3	$150n$	$135n$	$117n$	$96n$	–	–	$102n$
Round 4	$111n$	$96n$	$73n$	–	–	–	$78n$
Round 5	$72n$	$54n$	–	–	–	–	$54n$
Round 6	$30n$	–	–	–	–	–	$30n$

By the tie-breaking rule, p wins in the last round.

This manipulation provides an upper bound on the size of an optimal manipulation for Baldwin's rule.

REVERSE will put p in the first place, then a and f in some order, and then the remaining candidates. Without loss of generality, we suppose REVERSE breaks ties by constructing the vote $p \succ a \succ f \succ b \succ c \succ d \succ e$. It alternates this vote with $p \succ a \succ f \succ e \succ d \succ c \succ b$. After n such manipulating votes have been constructed, the scores of candidates a to f are level at $142n + n/2$, and p is at $48n$. From then on, the manipulators put p in first place and alternate the order of the other candidates. Without loss of generality, we suppose REVERSE breaks ties by constructing the vote $p \succ a \succ b \succ c \succ d \succ e \succ f$. It alternates this vote with $p \succ f \succ e \succ d \succ c \succ b \succ a$. At least $28n$ votes are therefore required in total for p to move out of last place. Hence, REVERSE requires at least $10n$ extra manipulators compared to the optimum number for Baldwin's. □

Theorem 12. *For Nanson's rule, there exists an election with four candidates and* $110n$ *votes where* REVERSE *computes a manipulation with at least* $4n$ *more votes than is optimal.*

Table 1
Percentage of elections drawn from the impartial culture model for which each heuristic found an optimal manipulation with Borda voting.

m	# Inst.	Reverse	Largest Fit	Average Fit	Largest Fit beats Average Fit
4	2771	94.2%	92.9%	**100.0%**	0.00%
8	5893	85.5%	87.7%	**99.3%**	0.03%
16	5966	76.8%	81.9%	**98.6%**	0.05%
32	5968	71.1%	80.7%	**98.5%**	0.02%
64	5962	66.8%	80.0%	**98.4%**	0.05%
128	5942	65.3%	79.9%	**98.0%**	0.03%
Total	32 502	74.9%	83.0%	**98.7%**	0.03%

Proof. Consider an election over a, b, c, and p, where p is the preferred candidate of the manipulators. We use the votes $W_{(u,v)}$ defined in Section 2. Non-manipulators cast the following votes: $15n$ copies of $W_{(a,c)}$, $W_{(b,c)}$ and $W_{(b,p)}$, and $10n$ copies of $W_{(a,p)}$. After the non-manipulators have voted, $s(a) = 190n$, $s(b) = 195n$, $s(c) = 135n$, and $s(p) = 140n$. The average score is $165n$.

If $21n$ manipulators vote identically $p \succ c \succ a \succ b$ then p wins. The following table shows scores of all candidates in two rounds.

	average	$s(a)$	$s(b)$	$s(c)$	$s(p)$
Round 1	$196.5n$	$211n$	$195n$	$177n$	$203n$
Round 2	$65.5n$	$65n$	–	–	$66n$

Hence, p wins in the last round.

This manipulation provides an upper bound on the size of an optimal manipulation for Nanson's rule.

Reverse will construct the vote $p \succ c \succ a \succ b$. After $5n$ such manipulating votes, the scores of a and b will become level. Reverse will then alternate between $p \succ c \succ b \succ a$ and $p \succ c \succ a \succ b$. In total, Reverse will construct $25n$ such manipulating votes, $15n$ for $p \succ c \succ a \succ b$ and $10n$ for $p \succ c \succ b \succ a$. At this point, p wins under Nanson's rule as demonstrated in the following table.

	average	$s(a)$	$s(b)$	$s(c)$	$s(p)$
Round 1	$202.5n$	$205n$	$205n$	$185n$	$215n$
Round 2	$135n$	$135n$	$135n$	–	$135n$

Note that p wins in the second round by our tie-breaking assumption. Hence, Reverse uses $4n$ extra manipulators compared to the optimum number for Nanson's. □

These results demonstrate that, for Baldwin's and Nanson's rules, Reverse does not approximate the optimal number of manipulators by an additive constant (as it does for Borda).

6. Experimental results

To test the performance of these heuristic methods in practice, we ran some experiments. Our experimental setup is based on that in [38]. We generated votes drawn either from the impartial culture model, or the Polya–Eggenberger urn model [5]. In the urn model, votes are placed in an urn and drawn at random. Votes 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 an approximately 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 the Borda rule. Manipulation under the Borda rule can be easily modelled as a constraint satisfaction problem. We used this property to obtain optimal solutions for our instances. We tested 1000 instances at each problem size and determined if the returned manipulations are optimal, by modelling the problem of finding an optimal manipulation as a constraint satisfaction problem and solving it using the solver Gecode [23].

The constraint solver found an optimal manipulation in 32 502 out of the 32 679 distinct impartial culture elections within the 1 hour time-out. Results are shown in Table 1. Both 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

Table 2
Percentage of elections drawn from an urn model for which each heuristic found an optimal manipulation with Borda voting.

m	# Inst.	Reverse	Largest Fit	Average Fit	Largest Fit beats Average Fit
4	3929	93.3%	66.3%	**100.0%**	0.00%
8	5501	85.6%	50.1%	**99.9%**	0.00%
16	5502	79.2%	41.1%	**99.5%**	0.02%
32	5532	72.4%	36.3%	**99.5%**	0.00%
64	5494	67.6%	33.0%	**99.7%**	0.00%
128	5571	64.5%	30.6%	**99.9%**	0.00%
Total	31 529	76.3%	41.7%	**99.7%**	0.00%

Table 3
Average number of manipulators required by each heuristic in elections drawn from an impartial culture model using the Borda rule.

n	Opt.	Reverse	Largest Fit	Average Fit
4	9.30	9.36	9.37	**9.30**
8	6.33	6.48	6.45	**6.34**
16	7.08	7.31	7.26	**7.10**
32	7.86	8.15	8.06	**7.88**
64	8.62	8.95	8.82	**8.63**
128	9.29	9.63	9.49	**9.31**

Table 4
Average number of manipulators required by each heuristic in elections drawn from an urn model using the Borda rule.

n	Opt.	Reverse	Largest Fit	Average Fit
4	42.21	42.28	42.94	**42.21**
8	32.67	32.82	33.78	**32.67**
16	34.08	34.29	35.68	**34.09**
32	35.06	35.34	36.97	**35.07**
64	36.76	37.09	38.92	**36.77**
128	37.70	38.06	39.98	**37.71**

number of candidates increases, while Average Fit does not seem to suffer from this problem as much: Average Fit solved all of the four candidate instances and 98% of the 128 candidate ones. Table 3 shows the average number of manipulators used by each of the heuristics, compared to the average optimal number of manipulators. We also note that in every one of the 32 502 instances, if Reverse found a k vote manipulation either Average Fit did too, or Average Fit found a $(k-1)$ vote manipulation, i.e., Average Fit never found a worse solution than Reverse. Furthermore, Largest Fit used at most two more manipulators than the optimum.

With the urn model, we were able to find an optimal manipulation for 31 529 out of the 31 530 elections within the 1 hour time-out. Tables 2 and 4 give results. Reverse solves about the same proportion of the urn instances as impartial culture instances, 76%. However, the performance of Largest Fit drops significantly. It is much worse than Reverse solving only 42% of instances to optimality. Furthermore, in contrast to the impartial culture elections where Largest Fit used at most two extra manipulators, here Largest Fit used up to 14 more manipulators than the optimum. The reason for such behaviour is that the non-manipulators' profiles in urn instances are similar to the profiles in the proof of Theorem 9, where Largest Fit requires an unbounded number of additional manipulators. The good performance of Average Fit is maintained. It found an 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 heuristic 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 coalitional 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 heuristic algorithms scale.

Our first set of experiments used 3000 elections with five candidates and five 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 an optimal solution. We threw out the 20% or so of instances generated in which the preferred candidate has

Table 5
Percentage of elections drawn from an impartial culture model with five candidates where the heuristic finds an optimal manipulation.

Rules	Reverse	Largest Fit	Average Fit	Eliminate	RevEliminate
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 6
Percentage of elections drawn from an urn model with five candidates where the heuristic finds an optimal manipulation.

Rules	Reverse	Largest Fit	Average Fit	Eliminate	RevEliminate
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%

Table 7
Average number of manipulators required by each heuristic in elections drawn from an impartial culture model using Baldwin's rule.

n	Reverse	Largest Fit	Average Fit	Eliminate	RevEliminate
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 8
Average number of manipulators required by each heuristic in elections drawn from an impartial culture model using Nanson's rule.

n	Reverse	Largest Fit	Average Fit	Eliminate	RevEliminate
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 9
Average number of manipulators required by each heuristic in elections drawn from an urn model using Baldwin's rule.

n	Reverse	Largest Fit	Average Fit	Eliminate	RevEliminate
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

already won before the manipulators vote. Results are given in Tables 5–6. The tables demonstrate that heuristics that are very effective at finding an optimal manipulation for the Borda rule do not perform as well for Baldwin's and Nanson's rules. For example, AverageFit almost always finds an optimal manipulation of the Borda rule but can only find optimal solutions about three quarters of the time for Baldwin's or Nanson's rules. Note that Eliminate and RevEliminate are strictly speaking defined just for Nanson's and Baldwin's rules. With the Borda rule, we can simply use the same manipulating votes they construct with, say, Baldwin's rule. These put the preferred candidate in first place so eventually must be successful in constructing a successful Borda manipulation.

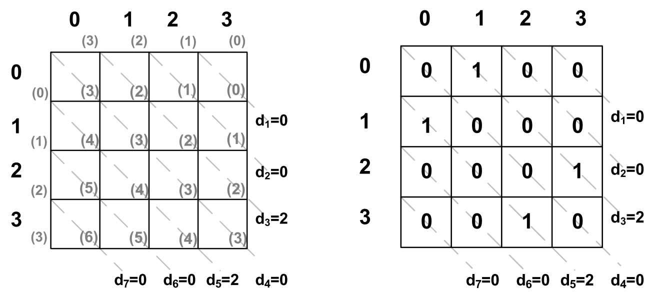
In our second set of experiments, we eschew computation of an optimal manipulation in order to use larger problems. This amplifies the differences between the different heuristic methods. Similarly to Section 6.1, we tested 1000 instances at each problem size, which gives 6000 instances in total.

Tables 7–10 show the results for the average number of manipulators. The results show that, with Nanson's and Baldwin's rules, Reverse works slightly better overall compared to LargestFit and AverageFit, which themselves outperform the other two methods especially for problems with large number of candidates. This contrasts with the results on the Borda rule in the previous section, where LargestFit and AverageFit do much better than Reverse. 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. Heuristic 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.

Table 10
Average number of manipulators required by each heuristic in elections drawn from an urn model using Nanson's rule.

n	Reverse	Largest Fit	Average Fit	Eliminate	RevEliminate
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



(a) labelling of a permutation matrix; (b) a possible solution.

Fig. 2. Permutation matrix with restricted diagonals sums (PMRDS).

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) [9] and the problem of finding multi Skolem sequences [30]. 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 when the gaps $g(i)$ of all candidates are distinct, then manipulation can be done in polynomial time [31].

A permutation matrix is an n by n Boolean matrix which is obtained from an identity matrix by a permutation of its columns. Hence, a permutation matrix contains a single value 1 in each row and each column. Consider the $2n-1$ diagonals of the matrix, numbering them from the top right to bottom left, and let d_i be the sum of the elements of the ith diagonal. Finding a permutation matrix such that its diagonal sums form a given sequence $(d_1, \ldots, d_{2n-1})$ 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 by the sequence $(d_1, \ldots, d_{2n-1})$.

We transform a Borda manipulation problem with $m+1$ candidates and 2 manipulators such that $\sum_{i=1}^{m} g(i) = m(m-1)$ to a PMRDS problem on an m by m matrix. Note that here we use m to denote the number of candidates excluding the preferred candidate. Note that such a manipulation problem is tight, i.e., all gaps will be matched exactly, and all candidates will have the same score after the manipulation. In parallel with the description of the transformation, we illustrate it with the following example with five candidates. Our preferred candidate is 4. Let $\langle 4, 4, 6, 6, 0\rangle$ be a score vector, where our preferred 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 given by the first manipulator, and columns of the permutation matrix with the reverse of the scores given by the second manipulator. We label each element of the matrix with the sum of its row and column labels. Fig. 2a shows the labelling for our example in gray.

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 (Fig. 2b). Suppose the cell $P(x, y)$ contains the value one. We conclude that the first manipulator gives the score x and the second gives the score $m-y-1$ to a candidate with the gap $x+(m-y-1)$. In our example, cell $P(0, 1)$ contains one, hence the first manipulator gives the score 0 and the second gives the score $m-y-1 = 4-1-1 = 2$ to a candidate with the gap 2. By examining all cells with the value one, we obtain the complete votes of the manipulators, which in our example are $(4 \succ 1 \succ 2 \succ 0 \succ 3)$ for the first manipulator and $(4 \succ 0 \succ 3 \succ 1 \succ 2)$ for the second, to fill the 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 votes make our candidate a winner. The total scores are $\langle 8, 8, 8, 8, 8\rangle$.

Finding a coalitional manipulation under the Borda rule using two manipulators is also connected to the problem of finding multi Skolem sequences used for the construction of Steiner triple system [30]. Given a multi-set of positive integers

Table 11
Operation of Largest Fit when trying to find a manipulation with four manipulators. The first seven columns show the gaps of the candidates. In the final column, we use the notation $x : y$ to indicate that Largest Fit assigns vote x to candidate y. The preferred candidate (c_8, not shown in the table) gets 7 from all four manipulators, thus its score is 42. We omit the 0 scores.

c_1	c_2	c_3	c_4	c_5	c_6	c_7	Vote
1	8	12	15	15	16	17	6:7
1	8	12	15	15	16	11	6:6
1	8	12	15	15	10	11	6:5
1	8	12	15	9	10	11	6:4
1	8	12	9	9	10	11	5:3
1	8	7	9	9	10	11	5:7
1	8	7	9	9	10	6	5:6
1	8	7	9	9	5	6	5:5
1	8	7	9	4	5	6	4:4
1	8	7	5	4	5	6	4:2
1	4	7	5	4	5	6	4:3
1	4	3	5	4	5	6	4:7
1	4	3	5	4	5	2	3:4
1	4	3	2	4	5	2	3:6
1	4	3	2	4	2	2	3:5
1	4	3	2	1	2	2	3:2
1	1	3	2	1	2	2	2:3
1	1	1	2	1	2	2	2:4
1	1	1	0	1	2	2	2:6
1	1	1	0	1	0	2	2:7
1	1	1	0	1	0	0	1:1
0	1	1	0	1	0	0	1:2
0	0	1	0	1	0	0	1:3
0	0	0	0	1	0	0	1:5
0	0	0	0	0	0	0	

$H = \{h_1, \dots, h_m\}$ we need to decide whether there exists a partition P of the set $\{1, \dots, 2m\}$ into pairs (p_i, p'_i), $i = 1, \dots, m$, so that $H \equiv \{p_i - p'_i \mid (p_i, p'_i) \in P\}$. There is a reduction from a manipulation problem with $m + 1$ candidates and 2 manipulators with gaps $G = \{g(1), \dots, g(m)\}$ such that $\sum_{i=1}^{m} g(i) = m(m-1)$ to a special case of multi Skolem sequences with $\sum_{i=1}^{m} h_i = m^2$ similar to the reduction from a scheduling problem in [30].[2] The multi Skolem sequence instance that corresponds to a manipulation instance is defined by $H = \{2m - g(1) - 1, \dots, 2m - g(m) - 1\}$. If a manipulation is given by the votes σ, π, then the partitions $(2m - \sigma(i), \pi(i) + 1)$ satisfy $2m - \sigma(i) - \pi(i) - 1 = 2m - g(i) - 1$. Conversely, suppose there exists partition P of the set $\{1, \dots, 2m\}$ into pairs (p_i, p'_i), $i = 1, \dots, m$, so that $h_i = p_i - p'_i$, $(p_i, p'_i) \in P, i = 1, \dots, m$. Then the votes are given by $\sigma(i) = 2m - p_i, \pi(i) = p'_i - 1$, which satisfy $\sigma(i) + \pi(i) = 2m - p_i + p'_i - 1 = 2m - (2m - g(i) - 1) - 1 = g(i)$.

8. Conclusions

In this paper we have investigated theoretically and empirically the computational complexity of manipulation problems for the Borda voting rule and two extensions of Borda voting, Baldwin's and Nanson's rules. We proved that it is NP-hard to compute a coalitional manipulation of the Borda rule with just two manipulators. This resolves a long-standing open question regarding the computational complexity of unweighted coalitional manipulation for common voting rules. We showed that two other rules, Baldwin's and Nanson's rules, which are derived from the Borda rule are also NP-hard to manipulate both with weighted and unweighted votes. Because of these NP-hardness results, we proposed several simple heuristic 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 heuristic algorithms did not perform as well in either Baldwin or Nanson elections, suggesting that these elimination style rules are more resistant to manipulation than the Borda rule.

Acknowledgements

George Katsirelos was partially 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 Australian Research Council, and the Asian Office of Aerospace Research and Development (AOARD-104123). Toby Walsh is currently supported by the German Federal Ministry for Education and Research through the Alexander von Humboldt Foundation. Lirong Xia acknowledges an RPI startup fund, NSF Grant #1136996 to the Computing Research Association for the CIFellows Project, and Vincent Conitzer's NSF CAREER 0953756 and IIS-0812113 for support.

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

Table 12
Operation of Average Fit when trying to find a manipulation with four manipulators. Since the algorithm works on averages, we show averages as fractions to convey both the actual gap, and the remaining number of votes.

c_1	c_2	c_3	c_4	c_5	c_6	c_7	Vote
1/4	8/4	12/4	15/4	15/4	16/4	17/4	6:7
1/4	8/4	12/4	15/4	15/4	16/4	11/3	6:6
1/4	8/4	12/4	15/4	15/4	10/3	11/3	6:5
1/4	8/4	12/4	15/4	9/4	10/3	11/3	6:4
1/4	8/4	12/4	9/3	9/3	10/3	11/3	5:7
1/4	8/4	12/4	9/3	9/3	10/3	6/2	5:6
1/4	8/4	12/4	9/3	9/3	5/2	6/2	5:3
1/4	8/4	7/3	9/3	9/3	5/2	6/2	5:5
1/4	8/4	7/3	9/3	4/2	5/2	6/2	4:4
1/4	8/4	7/3	5/2	4/2	5/2	6/2	4:7
1/4	8/4	7/3	5/2	4/2	5/2	2/1	4:6
1/4	8/4	7/3	5/2	4/2	1/1	2/1	4:4
1/4	8/4	7/3	1/1	4/2	1/1	2/1	3:3
1/4	8/4	4/2	1/1	4/2	1/1	2/1	3:2
1/4	5/3	4/2	1/1	4/2	1/1	2/1	3:3
1/4	5/3	1/1	1/1	4/2	1/1	2/1	3:5
1/4	5/3	1/1	1/1	1/1	1/1	2/1	2:7
1/4	5/3	1/1	1/1	1/1	1/1	0/0	2:2
1/4	3/2	1/1	1/1	1/1	1/1	0/0	2:2
1/4	1/1	1/1	1/1	1/1	1/1	0/0	

Appendix A. Proof of Theorem 10

In Table 11, we show the operation of Largest Fit when trying to find a manipulation with four manipulators. This is easily seen to be optimal because the maximum score of the preferred candidate that can be achieved with three manipulators is 35, which is not enough to defeat the 1st candidate. Based on these assignments, we find a perfect matching on the manipulation matrix (not shown), as described in Theorem 7. This gives the following votes for the manipulators:

$$\langle 0, 4, 2, 3, 1, 5, 6, 7\rangle$$

$$\langle 1, 0, 4, 2, 3, 6, 5, 7\rangle$$

$$\langle 0, 1, 5, 4, 6, 3, 2, 7\rangle$$

$$\langle 0, 3, 1, 6, 5, 2, 4, 7\rangle$$

In Table 12, we show the operation of Average Fit. At this point, the algorithm has yet to place a 2, but all candidates have gap of at most 1, so it has failed to find a manipulation. □

References

[1] J. Baldwin, The technique of the Nanson preferential majority system of election, Trans. Proc. R. Soc. Vic. 39 (1926) 42–52.
[2] J. Bartholdi, J. Orlin, Single transferable vote resists strategic voting, Soc. Choice Welf. 8 (4) (1991) 341–354.
[3] J. Bartholdi, C. Tovey, M. Trick, The computational difficulty of manipulating an election, Soc. Choice Welf. 6 (3) (1989) 227–241.
[4] J. Bartholdi, C. Tovey, M. Trick, How hard is it to control an election? Math. Comput. Model. 16 (8–9) (1992) 27–40.
[5] S. Berg, Paradox of voting under an urn model: the effect of homogeneity, Public Choice 47 (1985) 337–387.
[6] N. Betzler, B. Dorn, Towards a dichotomy of finding possible winners in elections based on scoring rules, J. Comput. Syst. Sci. 76 (8) (2010) 812–836.
[7] N. Betzler, R. Niedermeier, G. Woeginger, Unweighted coalitional manipulation under the Borda rule is NP-hard, in: Proceedings of the 22nd International Joint Conference on Artificial Intelligence, IJCAI-11, 2011, pp. 55–60.
[8] F. Brandt, M. Brill, E. Hemaspaandra, L. Hemaspaandra, Bypassing combinatorial protections: polynomial-time algorithms for single-peaked electorates, in: Proceedings of the 24th AAAI Conference on Artificial Intelligence, AAAI-10, 2010, pp. 715–722.
[9] S. Brunetti, A. Del Lungo, P. Gritzmann, S. de Vries, On the reconstruction of binary and permutation matrices under (binary) tomographic constraints, Theor. Comput. Sci. 406 (1–2) (2008) 63–71.
[10] J. Chamberlin, An investigation into the relative manipulability of four voting systems, Behav. Sci. 30 (1985) 195–203.
[11] T. Coleman, V. Teague, On the complexity of manipulating elections, in: Proceedings of the 13th Conference "Computing: The Australasian Theory Symposium", CATS2007, 2007, pp. 25–33.
[12] V. Conitzer, T. Sandholm, J. Lang, When are elections with few candidates hard to manipulate?, J. ACM 54 (3) (2007) 1–33.
[13] A.H. Copeland, A 'reasonable' social welfare function. Notes from a seminar on applications of mathematics to the social sciences, University of Michigan, 1951.
[14] J. Davies, G. Katsirelos, N. Narodytska, T. Walsh, An empirical study of Borda manipulation, in: Proceedings of the 3rd International Workshop on Computational Social Choice, COMSOC-10, 2010.
[15] J. Davies, N. Narodytska, T. Walsh, Eliminating the weakest link: making manipulation intractable?, in: Proceedings of the 26th AAAI Conference on Artificial Intelligence, AAAI-2012, 2012.
[16] E. Elkind, H. Lipmaa, Hybrid voting protocols and hardness of manipulation, in: Proceedings of 16th International Symposium on Algorithms and Computation, ISAAC 2005, 2005, pp. 206–215.
[17] P. Faliszewski, E. Hemaspaandra, L. Hemaspaandra, How hard is bribery in elections?, J. Artif. Intell. Res. 35 (1) (2009) 485–532.
[18] P. Faliszewski, E. Hemaspaandra, L. Hemaspaandra, J. Rothe, A Richer understanding of the complexity of election systems, in: Fundamental Problems in Computing: Essays in Honor of Professor Daniel J. Rosenkrantz, 2009, pp. 375–406.

[19] P. Faliszewski, E. Hemaspaandra, H. Schnoor, Copeland voting: ties matter, in: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS-08, 2008, pp. 983–990.
[20] P. Faliszewski, E. Hemaspaandra, H. Schnoor, Manipulation of Copeland elections, in: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems, AAMAS-10, 2010, pp. 367–374.
[21] P. Faliszewski, A. Procaccia, AI's war on manipulation: are we winning?, AI Mag. 31 (4) (2010) 53–64.
[22] P. Favardin, D. Lepelley, Some further results on the manipulability of social choice rules, Soc. Choice Welf. 26 (2006) 485–509.
[23] Gecode Team, Gecode: generic constraint development environment, 2006.
[24] A. Gibbard, Manipulation of voting schemes: a general result, Econometrica 41 (4) (1973) 587–601.
[25] P. Hall, On representatives of subsets, J. Lond. Math. Soc. (1935) 26–30.
[26] K. Konczak, J. Lang, Voting procedures with incomplete preferences, in: Proceedings of the IJCAI-2005 Workshop on Advances in Preference Handling, 2005.
[27] K.L. Krause, V.Y. Shen, H.D. Schwetman, Analysis of several task-scheduling algorithms for a model of multiprogramming computer systems, J. ACM 22 (4) (1975) 522–550.
[28] D. McGarvey, A theorem on the construction of voting paradoxes, Econometrica 21 (4) (1953) 608–610.
[29] E. Nanson, Methods of election, Trans. Proc. R. Soc. Vic. 19 (1882) 197–240.
[30] G. Nordh, A note on the hardness of Skolem-type sequences, Discrete Appl. Math. 158 (8) (2010) 63–71.
[31] G. Nordh, Personal communication, 2011.
[32] M. Pini, F. Rossi, B. Venable, T. Walsh, Incompleteness and incomparability in preference aggregation, in: Proceedings of 20th International Joint Conference on Artificial Intelligence, IJCAI-2007, 2007, pp. 1464–1469.
[33] M. Pini, F. Rossi, B. Venable, T. Walsh, Incompleteness and incomparability in preference aggregation, Artif. Intell. 175 (7–8) (2011) 1272–1289.
[34] M.A. Satterthwaite, Strategy-proofness and Arrow's conditions: existence and correspondence theorems for voting procedures and social welfare functions, J. Econ. Theory 10 (2) (1975) 187–217.
[35] T.N. Tideman, Independence of clones as a criterion for voting rules, Soc. Choice Welf. 4 (1987) 185–206.
[36] T. Walsh, Uncertainty in preference elicitation and aggregation, in: Proceedings of the 22nd AAAI Conference on Artificial Intelligence, AAAI-2007, 2007, pp. 3–8.
[37] T. Walsh, 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, 2009, pp. 324–329.
[38] T. Walsh, An empirical study of the manipulability of single transferable voting, in: Proceedings of the 19th European Conference on Artificial Intelligence, ECAI-2010, 2010, pp. 257–262.
[39] T. Walsh, Where are the really hard manipulation problems?, J. Artif. Intell. Res. 42 (2011) 1–29.
[40] L. Xia, M. Zuckerman, A. Procaccia, V. Conitzer, J. Rosenschein, Complexity of unweighted coalitional manipulation under some common voting rules, in: Proceedings of the 21st International Joint Conference on Artificial Intelligence, IJCAI-2009, 2009, pp. 348–353.
[41] L. Xia, V. Conitzer, A. Procaccia, A scheduling approach to coalitional manipulation, in: Proceedings of the 11th ACM Conference on Electronic Commerce, EC-2010, 2010, pp. 275–284.
[42] W. Yu, H. Hoogeveen, J.K. Lenstra, Minimizing makespan in a two-machine flow shop with delays and unit-time operations is NP-hard, J. Sched. 7 (5) (2004) 333–348.
[43] M. Zuckerman, A. Procaccia, J. Rosenschein, Algorithms for the coalitional manipulation problem, Artif. Intell. 173 (2) (2009) 392–412.