Computational Aspects of Multi-Winner Approval Voting

Haris Aziz Serge Gaspers
NICTA and UNSW
Sydney, Australia
{haris.aziz, serge.gaspers}@nicta.com.au

Joachim Gudmundsson
NICTA and University of Sydney
Sydney, Australia
joachim.gudmundsson@sydney.edu.au

Simon Mackenzie Nicholas Mattei Toby Walsh
NICTA and UNSW
Sydney, Australia
{simon.mackenzie, nicholas.mattei, toby.walsh}@nicta.com.au

ABSTRACT

We study computational aspects of three prominent voting rules that use approval ballots to select multiple winners. These rules are *proportional approval voting*, *reweighted approval voting*, and *satisfaction approval voting*. Each rule is designed with the intention to compute a representative winning set. We show that computing the winner for proportional approval voting is NP-hard, closing an open problem (Kilgour, 2010). As none of the rules we examine are strategyproof, we study various strategic aspects of the rules. In particular, we examine the computational complexity of computing a best response for both a single agent and a group of agents. In many settings, we show that it is NP-hard for an agent or agents to compute how best to vote given a fixed set of approval ballots of the other agents.

Categories and Subject Descriptors

F.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity; I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems; J.4 [**Computer Applications**]: Social and Behavioral Sciences - Economics

General Terms

Economics, Theory and Algorithms

Keywords

Multi-winner voting; approval voting; computational social choice.

1. INTRODUCTION

The aggregation of possibly conflicting preferences is a central problem in artificial intelligence [9]. Agents express preferences over candidates and a voting rule selects a winner or winners based on these preferences. We focus here on rules that select k winners where k is fixed in advance. This covers a variety of important settings including parliamentary elections [25]; selecting committees [33, 26]; the hiring of faculty members [13]; understanding heredity and signals in DNA [20]; and movie recommendation systems [32, 14].

Generally, in approval-based voting rules, an agent approves of (votes for) a subset of the candidates. The most straightforward way to aggregate these votes is to have every approval for a candidate contribute one point to that candidate; yielding the rule known as Approval Voting (AV). The same principle of selecting the candidate with the highest number of approvals can be extended to select multiple candidates the highest number of approvals. Unlike plurality voting, where agents only vote for their most preferred candidate, approval ballots permit agents to identify multiple candidates that they wish to win. Approval voting has many desirable properties in the single winner case [17, 6], including its 'simplicity, propensity to select Condorcet winners (when they exist), its robustness to manipulation and its monotonicity' [25]. However for the case of multiple winners, the merits of AV are 'less clear' [25]. In particular, for the multi-winner case, AV does not address more egalitarian concerns such as proportional representation.

Various methods for counting approvals have been introduced in the literature, each attempting to address the fairness and representation concerns when using AV for multiple winners (a *winning set* of candidates) [22]. In this paper we investigate three popular ways to count approvals that are designed with the intention of selecting a more representative set of winners than AV. First, under *Proportional Approval Voting (PAV)*, each agent's contribution to the committee's total score depends on how many candidates from the agent's approval set have been selected. A second way to promote fairness is to proceed across a set of rounds. In each round, the candidate with the most approvals wins. However, in each subsequent round we decrease the weight of agents who have already had a candidate selected in earlier rounds; this method is implemented in *Reweighted Approval Voting (RAV)*. Finally, *Satisfaction Approval Voting (SAV)* modulates the weight of approvals with a satisfaction score for each agent, based on the ratio of approved candidates appearing in the winning set to the agent's total number of approved candidates.

The three approaches we consider for generalizing approval voting to the case of multiple winners each have their own benefits and drawbacks. It is therefore necessary to understand the computational and axiomatic properties of these voting rules for the variety of domains for which they are typically deployed. In terms of computational properties,

Appears in: *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Bordini, Elkind, Weiss, Yolum (eds.), May 4–8, 2015, Istanbul, Turkey.*

it is beneficial to have a voting rule which admits a polynomial algorithm to determine the winners. Otherwise, we cannot hope to use the rule in settings with large numbers of candidates or voters [21]. Another key problem studied in computational social choice is the complexity of computing a beneficial misreport (i.e., manipulation) [3, 15, 16]. If computing a beneficial misreport is computationally hard for a given rule, the rule is said to be resistant to manipulation. If it is computationally hard to compute a misreport, agents may decide to be truthful, since they cannot always easily manipulate. Connections also exist between manipulation and other important questions in social choice such as deciding when to terminate preference elicitation and determining possible winners [24, 37].

From the axiomatic perspective, studying the positive or negative properties of these multi-winner rules can help us make informed, objective decisions about which rule is better suited for the particular situation to which we are applying a multi-winner rule [14]. Though AV is the most widely known among these rules, RAV has been used, for example, in elections in Sweden. Rules other than AV may have better axiomatic properties in the multi-winner setting and thus, motivate our study. For example, each of PAV, RAV, and SAV have a more egalitarian objective than AV [2]. Brams, a proponent of AV when selecting a single winner, has argued that SAV is more suitable for equitable representation when multiple winners are to be selected [5].

We undertake a detailed study of the computational aspects of PAV, RAV, and SAV. We first consider the computational complexity of computing the winner. Although PAV was introduced over a decade ago, the computational complexity of determining the winners is an open question, having only been referred to as "computationally demanding" before [22]. We close this outstanding open problem, showing that winner determination for PAV is NP-hard. We then consider strategic voting for these rules. We show that, even with dichotomous preferences, PAV, RAV and SAV are not strategyproof. That is, it may be beneficial for agents to misreport their true preferences. We therefore consider computational aspects of manipulation. We prove that finding the best response given the preferences of other agents is NP-hard under a number of conditions for PAV, RAV, and SAV. In particular, we examine the complexity of checking whether an agent or a set of agents can make a given candidate or a set of candidates win. These results offer support for RAV over PAV or SAV as it is the only one of these three rules for which winner determination is computationally easy but manipulation is hard.

We stress here that our results are not restricted to AV, PAV, RAV and SAV. Our NP-hardness result for any rule implies NP-hardness for a more general class of rules that includes that rule. In certain cases, our precise reduction can be adapted to prove similar results for other variants of the rule. For example, our reduction to show that winner determination for PAV is NP-hard can be adapted to prove NP-hardness for winner determination for any variant of PAV in which the coefficient weight for the second approved candidate is less than the coefficient weight for the first approved candidate.

2. RELATED WORK

Surprisingly, there has only been limited consideration of computational aspects of selecting multiple-winners. Exceptions include work by Meir et al. [29] who consider single non-transferable voting, approval voting, k-approval, cumulative voting and the proportional schemes of Monroe, and of Chamberlin and Courant. Most relevant to our study is that for approval voting, Meir et al. prove that manipulation with general utilities and control by adding/deleting candidates are both polynomial to compute, but control by adding/deleting agents is NP-hard. Another work that considers computational aspects of selecting multiple-winners is Obraztsova et al. [32], but their study is limited to k-approval and scoring rules. Finally, the control and bribery problems for AV and two other approval voting variants are well catalogued by Baumeister et al. [4].

The Handbook of Approval Voting discusses various approval-based multi-winner rules including PAV, RAV and SAV. Another prominent multi-winner rule in the Handbook is *minimax approval voting* [7]. Each agent's approval ballot and the winning set can be seen as a binary vector. Minimax approval voting selects the set of k candidates that minimizes the maximum Hamming distance from the submitted ballots. Although minimax approval voting is a natural and elegant rule, LeGrand et al. [26] showed that computing the winning set is NP-hard. Strategic issues and approximation questions for minimax approval voting are covered by Caragiannis et al. [8] and Gramm et al. [20].

The area of multi-winner approval voting is closely related to the study of proportional representation when selecting a committee [34, 35]. Understanding approval voting schemes which select multiple winners, as the rules we consider do, is an important area in social choice with applications in a variety of settings from committee selection to multi-product recommendation [14].

3. FORMAL BACKGROUND

We consider the social choice setting (N, C) where $N = \{1, \ldots, n\}$ is the set of agents and $C = \{c_1, \ldots, c_m\}$ is the set of candidates. Each agent expresses an approval ballot $A_i \subset C$ that represents the subset of candidates that he approves of, yielding a set of approval ballots $A = \{A_1, \ldots, A_n\}$. We will consider approval-based multi-winner rules that take as input (C, A, k) and return the subset $W \subseteq C$ of size k that is the winning set. We will assume a lexicographic tie-breaking rule over the candidate set.

Approval Voting (AV).

AV finds a set $W \subseteq C$ of size k that maximizes the total score $App(W) = \sum_{i \in N} |W \cap A_i|$. That is, the set of AV winners are those candidates that are approved by the largest number of agents. AV has been adopted by several academic and professional societies such as the American Mathematical Society (AMS), the Institute of Electrical and Electronics Engineers (IEEE), and the International Joint Conference on Artificial Intelligence (IJCAI).

Proportional Approval Voting (PAV).

In PAV, an agent's satisfaction score is $1 + 1/2 + 1/3 + \cdots + 1/j$ where j is the number of his or her approved candidates that are selected in W. Formally, PAV finds a set $W \subseteq C$ of size k that maximizes the total score $PAV(W) = \sum_{i \in N} r(|W \cap A_i|)$ where $r(p) = \sum_{j=1}^{p} \frac{1}{j}$. PAV was proposed by the mathematician Forest Simmons in 2001 [22].

We observe that PAV is only one way to apply scores in a decreasing manner. Kilgour and Marshall [23] introduce the notion of Generalized Approval Voting (GAV) where the harmonic weights $1+1/2+1/3+\cdots+1/j$ are replaced by an arbitrary vector of weights $w(1), w(2), w(3), \ldots, w(m)$. We will focus on PAV in our results but show how they directly extend to the case of GAV. Moreover, we are interested in the harmonic weights as these are the *only* set of weights for GAV which satisfy a strong egalitarian axiomatic property called extended justified representation [2].

Reweighted Approval Voting (RAV).

RAV converts *AV* into a multi-round rule, selecting a candidate in each round and then reweighing the approvals for the subsequent rounds. In each of the k rounds, we select an unselected candidate to add to the winning set W with the highest "weight" of approvals. In each round we reweight each agents approvals, assigning the weight $\frac{1}{1+|W \cap A_i|}$ to each agent $i \in N$. *RAV* was invented by the Danish polymath Thorvald Thiele in the early 1900's. *RAV* has also been referred to as "*sequential proportional AV*" [5].

Satisfaction Approval Voting (SAV).

An agent's satisfaction is the fraction of his or her approved candidates that are selected. *SAV* maximizes the sum of such scores. Formally, *SAV* finds a set $W \subseteq C$ of size k that maximizes $Sat(W) = \sum_{i \in N} \frac{|W \cap A_i|}{|A_i|}$. The rule was proposed by Brams and Kilgour [5] with the aim of representing more diverse interests than *AV*.

Kilgour and Marshall [23] discuss several variants of SAV. We focus on SAV as it was the first rule in this family to be introduced and many open problems about strategic aspects of SAV remain open. As there is no consensus as to the best rule, our study will help understand the relative merits of different rules.

Tie-breaking is an important issue to consider when investigating the complexity of manipulation and winner determination problems as it can have a significant impact on the complexity of reasoning tasks [31, 30, 1]. We make the assumption that ties are broken lexicographically (both for individual elements and sets) with $a \succ b \succ c$, e.g., $\{a, b\}$ is preferred to $\{a, c\}$. We also note that many of our proofs are independent of the tie-breaking rule, in which case the hardness results transfer to any arbitrary tie-breaking rule.

EXAMPLE 1. *Consider a setting with 7 agents, a candidate set $C = \{a, b, c, d\}$, $k = 2$, and the following ballots: $A_1 = A_2 = A_3 = \{a, b\}$; $A_4 = A_5 = \{c\}$; $A_6 = A_7 = \{d\}$.*

Approval Voting (AV): *The approval scores of the candidates are $App(\{a\}) = 3$, $App(\{b\}) = 3$, $App(\{c\}) = 2$, and $App(\{d\}) = 2$. Hence, the set $\{a, b\}$ is selected.*

Proportional Approval Voting (PAV): *PAV scores are not additive and we can sum the satisfaction for each agent for each potential set of winners to determine the winner. For example, $PAV(\{a, b\}) = 3 \cdot 1.5 = 4.5$. Computing $PAV(W)$ for all W leaves us with a tie between the sets $\{a, c\}, \{a, d\}, \{b, c\}, \{b, d\}$, which is broken in favor of $\{a, c\}$.*

Reweighted Approval Voting (RAV): *In the first round, a is selected. This causes the weight of A_1, A_2, and A_3 to drop to 0.5 for the second round. Consequently, c is selected next, which gives the winning set $\{a, c\}$.*

Satisfaction Approval Voting (SAV): *As SAV scores are additive we can greedily add the agent which provides the most satisfaction, in order, to compute the result. Observe that the satisfaction obtained by adding c or d to the winning set is 2, the maximum of any candidate. Thus, the SAV rule selects $\{c, d\}$.*

4. WINNER DETERMINATION

We first examine one of the most basic computational questions, computing the winners of a voting rule. This problem should be computationally easy if we want to use a voting rule in practice [21].

Name: $\mathcal{R}$–WINNER DETERMINATION ($\mathcal{R}$–WD).
Input: A a set of approval ballots A over the set C of candidates and a winning set size $k \in \{1, \ldots, |C|\}$.
Output: Compute the winning set of size k under $\mathcal{R}$.

A related problem is the decision problem which tests whether a given set is the winning set.

Name: $\mathcal{R}$–TEST WINNING SET ($\mathcal{R}$–TESTWS).
Input: A a set of approval ballots A over the set C of candidates, a winning set size $k \in \{1, \ldots, |C|\}$, and a set $W \subseteq C$ with $|W| = k$.
Question: Is W the winning set under $\mathcal{R}$?

Since we consider resolute multi-winner rules, $\mathcal{R}$–WD is at least as hard as $\mathcal{R}$–TESTWS. Note that if there is a polynomial-time algorithm for $\mathcal{R}$–WD, then $\mathcal{R}$–TESTWS is in P as well: compute the unique winning set and check whether W coincides with the winning set. This means that if $\mathcal{R}$–TESTWS is coNP-complete, then there exists no polynomial-time algorithm for $\mathcal{R}$–WD unless $P = NP$.

We observe that *AV*–WD, *RAV*–WD, *SAV*–WD are all polynomial-time computable. We give a formal argument for *SAV*–WD as it is not formally stated in the literature whether the winning set can be computed in polynomial time.

THEOREM 1. *SAV*–WD *can be solved in linear time.*

PROOF. For $|W| = k$, $Sat(W) = \sum_{i \in N} \frac{|W \cap A_i|}{|A_i|} = \sum_{c \in W} \sum_{i \in N, c \in A_i} \frac{1}{|A_i|}$ where we say that $\sum_{i \in N, c \in A_i} \frac{1}{|A_i|}$ is the *Sat score of an individual candidate*. Hence computing a winning set of size k is equivalent to selecting k candidates with the highest Sat scores. □

Although *RAV*–WD is polynomial-time computable, it has been termed "computationally difficult" to analyze in [22]. We provide support for this claim by showing that computing a best response for *RAV* is NP-hard (Theorem 5). We close the computational complexity of *PAV*–TESTWS in this section.[1]

[1] Thm. 9 and 11 in Skowron et al. [36] independently show a similar result to Thm. 2 and Cor. 2. The framework of Skowron et al. [36] uses *order weighted averages* combined with voting [19] to select subsets of objects that have highest utility to the agents. Their results show that finding the set of winners is NP-hard for any weight vector that is non-increasing and non-constant. Our reduction *additionally* shows that it is computationally hard to even *verify* that a given set is a winning set, holds even when agents approve of only two objects, and shows Cor. 3.

THEOREM 2. *PAV–*TESTWS *is coNP-complete, even if each agent approves of 2 candidates.*

PROOF. Consider the complement problem. A polynomial time checkable witness is simply a set of candidates of size k that has greater *PAV* score than W. For NP-hardness, we give a polynomial-time reduction from the NP-hard INDEPENDENT SET problem [18], which is defined as follows. Given (G', t), where G' is a graph and t an integer, determine whether G' has an independent set of size t. An independent set is a subset of vertices S such that no edge of G' has both endpoints in S. For an instance (G', t) of INDEPENDENT SET, we build a *PAV* instance such that G' has an independent set of size t if and only if the winning set for the *PAV* is not W.

Consider a graph G'. Obtain a new graph $G = (V, E)$ from G by adding a set I of t vertices to G', making them adjacent to all vertices in G', and adding exactly one edge between two arbitrary vertices of I. Note that a set of vertices $S \subseteq V$ of size at least t is independent in G if and only if it is independent in G'. Define the following *PAV* instance, $pav(G) = (N, C, A, k)$: We have a set of agents N and a set of candidates C. For each vertex $v \in V$, we create $deg(G) - deg(v)$ 'dummy' candidates in C, where $deg(G)$ is the maximum degree of G, $deg(G) > 1$, and $deg(v)$ the degree of vertex v. For each $v \in V$, we also create another candidate in C, labeled C_v. We create an agent in N for each edge $e \in E$. For each vertex we also create $deg(G) - deg(v)$ agents. Each of the "edge" agents we create approves of the two candidates corresponding to the two vertex candidates connected by the edge. Each vertex agent associated with vertex v approves of C_v and one of the dummy candidates associated with v, such that each dummy candidate has exactly one agent approving him. We also set $k = t$ and $W = \{C_v : v \in I\}$. The tie-breaking rule prefers the set W to any other set of k candidates. Observe that the satisfaction score for W is $deg(G) \cdot t - 1/2$, and a winning set distinct from W has satisfaction score at least $deg(G) \cdot t$.

We will show that $pav(G)$ has a winning set of size $k = t$ scoring a total approval of at least $s = deg(G) \cdot t$ if and only if G has an independent set of size t.

First, note that adding a candidate to a winning set increases its total score by at most $deg(G)$, since at most $deg(G)$ agents see their satisfaction score rise by at most one. Also, if adding a candidate c to a winning set increases the total score of the winning set by exactly $deg(G)$, then c corresponds to a vertex in G, since each dummy vertex is approved by only one agent, and the vertex corresponding to c is not adjacent to a vertex corresponding to any other candidate in the winning set. Thus, the candidates in a winning set of size $k = t$ scoring a total approval of s correspond to an independent set of size t in G and vice-versa. □

COROLLARY 1. *PAV–*WD *is NP-hard, even if each agent approves of 2 candidates.*

The proof of Theorem 2 does not rely on the particular score assigned to candidates assigned subsequently to the first one, as long as the second candidate contributes strictly less satisfaction score than the first. Thus, *PAV–*TESTWS remains hard as long as the satisfaction score satisfies this strictly decreasing condition. Thus, we can state the following corollary.

COROLLARY 2. *GAV–*TESTWS *is coNP-complete for any set of weights where $w(1) > w(2)$, even if each agent approves of 2 candidates.*

Since the reduction for Theorem 2 is a parameterized reduction from the W[1]-hard [12] INDEPENDENT SET problem with parameter t to the complement of *PAV–*TESTWS with parameter k, we have that *PAV–*TESTWS is coW[1]-hard with parameter k. Therefore it is unlikely that the problem can be solved in time $f(k) \cdot m^{O(1)}$ for any function f. Thus, a factor m^k in the running time seems unavoidable.

COROLLARY 3. *PAV–*TESTWS *is coW[1]-hard for parameter k, even if each agent approves of 2 candidates.*

5. STRATEGIC VOTING

As in the single winner case, agents may benefit from misreporting their true preferences when selecting multiple winners. Formally, we will assume that each agent $i \in N$ is equipped with a utility function $u_i : C \to \mathbb{R}^+$ and an agent i's utility for a winning set W is $\sum_{c \in W} u_i(c)$. We will mostly assume that agents have *Boolean* utilities: $u_i(c) \in \{0, 1\}$ for all $c \in C$. For agents with Boolean utilities, we say that a multi-winner approval-based voting rule is *strategyproof* if and only if there is no agent who will receive strictly greater utility by either approving candidates for whom he has utility 0 or not approving candidates for whom he has utility 1.

We begin by showing some results that are likely folklore but provide a useful starting point for our main results

THEOREM 3. *PAV, RAV and SAV are not strategyproof with Boolean utilities.*

PROOF. We treat each case separately. Ties are always broken lexicographically and $k = 2$. We show that we can construct instances where agent 1 has an opportunity to misreport his preference and get a more preferred set selected with respect to his true preferences.

For *PAV*, if we have $C = \{a, b, c\}$ and the votes:

$$A_1 = \{a, c\}, A_2 = A_3 = A_4 = \{a, b\}, A_5 = \{c\}, A_6 = \{a\}.$$

Then $\{a, b\}$ is the winning set due to tie-breaking (over the set $\{a, c\}$). However, if agent 1 only approves c, then we get $\{a, c\}$ as the winning set.

For *RAV*, consider the following votes:

$$A_1 = \{a, b\}, A_2 = A_3 = A_4 = \{a\}, A_5 = \{c\}, A_6 = \{b, c\}.$$

The outcome is $\{a, c\}$ for all reported preferences of agent 1 except if $A_1 = \{b\}$, in which case the outcome is $\{a, b\}$.

For *SAV*, if we have $C = \{a, b, c, d\}$ and the votes:

$$A_1 = \{a, b\}, A_2 = A_3 = \{a, c, d\}, A_4 = \{a\}.$$

Thus, $\{a, c\}$ and $\{a, d\}$ are the sets with highest SAV scores. Agent 1 is indifferent between them. If agent 1 only approves b, then we get $\{a, b\}$ as the winning set without tie-breaking. □

With *PAV*, *RAV* and *SAV*, it can therefore be beneficial for agents to vote strategically. Next, we consider the computational complexity of computing such strategic votes. Throughout the paper, we will denote by j the number of additional ballots submitted to achieve strategic goals such

as getting a candidate, subset of candidates, or the exact set of candidates selected.

Name: $\mathcal{R}$–Winner Manipulation ($\mathcal{R}$–WM)
Input: A set of approval ballots A over the set C of candidates, a winning size k, a number of agents j still to vote, and a preferred candidate c^*.
Question: Are there j additional approval ballots so that c^* is in the winning set W of size k (i.e., $c^* \in W, |W| = k$) under $\mathcal{R}$?

Name: $\mathcal{R}$–Winning Set Manipulation ($\mathcal{R}$–WSM).
Input: A set of approval ballots A over the set C of candidates, a winning set size k, a number of agents j still to vote, and a set of preferred candidates $C^* \subseteq C$.
Question: Are there j additional approval ballots such that C^* is *exactly* the winning set of size k (i.e., $W = C^*, |W| = k$) under $\mathcal{R}$?

Note that our definitions for $\mathcal{R}$–WM and $\mathcal{R}$–WSM are for the basic computational problems. These problem definitions do not rely on our use of a particular utility model for the agents or lifting this model to sets of candidates. We also introduce the manipulation problem based on agent utilities.

Name: $\mathcal{R}$–Utility Manipulation ($\mathcal{R}$–UM).
Input: A set of approval ballots A over the set C of candidates, a winning set size k, a manipulating agent i with utility function $u : C \to \mathbb{R}^+$, a number of votes j still to be cast, and a target utility $t \in \mathbb{R}^+$
Question: Can the manipulator i submit j additional approval ballots so that he gets utility $u_i(W) \geq t$ from the winning set W of size k under rule R?

The problem of utility manipulation has been considered previously by Meir et al. [29]. We note that the problem of utility manipulation is at least as hard as $\mathcal{R}$–WM or $\mathcal{R}$–WSM. Hence, if $\mathcal{R}$–WM or $\mathcal{R}$–WSM is NP-hard for a single agent ($j = 1$), then the more general problem of maximizing the utility of an agent ($\mathcal{R}$–UM) is also NP-hard even for Boolean utilities. The reductions for WM and WSM to utility maximization involve giving desired candidates utility one and the rest utility zero. For AV, the utility maximizing best response of a single agent can be computed in polynomial time for both Boolean and general utilities [29, 28]. Note that AV is strategyproof for Boolean utilities and therefore an agent being able to compute a utility maximising vote is not a negative result. These single agent results for AV can be extended in a straightforward way to settings with multiple agents ($j \geq 1$).

Since j is encoded in binary in $\mathcal{R}$–WM, $\mathcal{R}$–WSM, and $\mathcal{R}$–UM, it is not entirely obvious that the problems are in NP as the the number of ballots of the manipulator could be exponential in the input size. However, notice that for $\mathcal{R} \in \{PAV, RAV, SAV\}$, $\mathcal{R}$–WM and $\mathcal{R}$–WSM instances are Yes-instances if $j > (n+1)(m+1)$, and likewise a utility-maximizing manipulation can be constructed for $\mathcal{R}$–UM if $j > (n+1)(m+1)$. Therefore, we may assume that $j \leq (n+1)(m+1)$, and as a result the problems are in NP since the manipulator ballots are polynomial-size certificates.

A summary of our results about strategic voting and winner determination can be found in Table 1. In the rest of this section we will investigate each of the rules in turn, examining the relationships between the complexity of computing various strategic behaviors for each.

5.1 Proportional Approval Voting (PAV)

Since PAV–TestWS is coNP-complete, deciding if a coalition of agents who have yet to vote can ensure a given candidate or set of candidates wins is coNP-hard. Hence we can state the following corollary.

Corollary 4. *PAV–WM and PAV–WSM are coNP-hard.*

However the hardness of manipulation here comes from the hardness of winner determination even if the agents vote sincerely. This seems rather unsatisfying and motivates us to investigate the situation where a "real" manipulation is necessary, that is, whether a single manipulator can find a vote that will maximise his utility. The proof of Theorem 2 can be modified to show that a manipulator can compute a vote which maximises his utility if and only if he can decide the existence of an independent set of size t in G. We define an even more constrained problem:

Name: $\mathcal{R}$-Binary Choice Manipulation ($\mathcal{R}$–BCM)
Input: A set of approval ballots A over the set C of candidates, a winning set size k, a set of preferred candidates $C^* \subseteq C$, and 2 ballots v_1 and v_2 such that the manipulator is guaranteed at least one of the two will ensure all his preferred candidates get selected.
Question: Will the manipulator choosing v_1 ensure all his preferred candidates get selected?

Theorem 4. *PAV–BCM is NP-hard.*

Proof. To show NP-hardness we leverage the construction from the proof of Theorem 2. Given an instance (G', t) of Independent Set we create an instance of PAV–BCM, (C, A, k, C^*, v_1, v_2) where $k = t+5$ is the size of the committee. We first modify the provided graph G' by augmenting it in several ways. The resulting graph G will then be converted into a PAV instance through a modified version of the reduction described in Theorem 2. In the resulting instance, the manipulator will be able to decide on the vote which maximises his utility if and only if he can decide the existence of an independent set of size t in G'.

First, we obtain G from G' by adding three connected components: a simple 2-clique, and two bipartite graphs $K_{2,2}$. Let us label the vertices in the 2-clique q_1 and q_2. Let us label the vertices of the first $K_{2,2}$ graph $b_{1,1}$, $b_{1,2}$, $b_{2,1}$, $b_{2,2}$ and those of the second $d_{1,1}$, $d_{1,2}$, $d_{2,1}$ and $d_{2,2}$. The labeling is such that $b_{1,1}$ and $b_{1,2}$ are not connected by an edge, and neither are $d_{1,1}$ and $d_{1,2}$. Note that G' has an independent set of size t if and only if G has an independent set of size $t + 5$. Now, we obtain a PAV instance $pav(G)$ from G as in Theorem 2. Then, we multiply the number of agents by 20, to ensure that the manipulator is not capable of having a meaningful impact on the candidates chosen in the graph G (by meaningful we mean that he cannot force two adjacent candidates to be chosen, which would invalidate the construction in Theorem 2). Note that multiplying the number of agents does not change the winning set.

In the resulting PAV instance, we delete 10 of the agents that correspond to the edges in the 2-clique and exactly 4 agents corresponding to the edges between q_2 and its dummy candidates. We now add 2 agents approving both q_2 and $d_{2,1}$ and 2 agents approving q_2 and $b_{1,2}$. We add 2 agents approving q_1 and one of its dummy candidates.

It only remains to specify the preferred candidates and the two approval ballots between which the manipulator needs to decide: $C^* = \{b_{1,1}, d_{1,1}\}$, $v_1 = \{d_{1,1}, d_{1,2}\}$ and $v_2 = \{b_{1,1}, b_{1,2}\}$.

If there is no independent set of size $t+5$ in G then q_1 and q_2 will be selected since adjacency is penalized less on these components. Else, if there is an independent set of size $t+5$ in G then only q_1 will be a part of the winning set, since q_1 gives slightly higher score than q_2 and other candidates.

Without the manipulator's vote, either q_2 is selected as part of the winning set – along with $b_{2,1}$, $b_{2,2}$, $d_{1,1}$ and $d_{1,2}$, or q_2 is not selected and $b_{1,1}$, $b_{1,2}$, $d_{2,1}$ and $d_{2,2}$ are selected. In either case the manipulator only gets one of his approved candidates.

It is possible for the manipulator to ensure the selection of both his approved candidates by either casting the approval vote $v_1 = \{d_{1,1}, d_{1,2}\}$ (if G' has an independent set of size t) or $v_2 = \{b_{1,1}, b_{1,2}\}$ (if G' has no independent set of size t). Therefore, the PAV–BCM is a YES-instance if and only if the INDEPENDENT SET instance is a YES-instance. □

5.2 Reweighted Approval Voting (RAV)

Though RAV–WD is polynomial-time computable, "computationally difficult" to analyze. This is because the decision for a single agent of whom to vote for in order to maximize his utility is not straightforward, as we show in the following observations.

OBSERVATION 1. *Under RAV an agent i who wants to include a single candidate c^* in the winning set may have incentive to approve other candidates.*

Suppose we are selecting a winning set of size $k = 2$ and $C = \{a, b, c, d\}$:

$$A_2 = \{b, d\}, A_3 = \{c, d\}, A_4 = \{a, b, c, d\}$$
$$A_5 = A_6 = \{b, c, d\}, A_7 = \{a, b\}, A_8 = \{c\}, A_9 = \{a\}.$$

If agent 1 is only interested in getting a to the winning set, then he may need to approve candidates other than a. If agent 1 casts the ballot $A_1 = \{a\}$ then in round 1, b is selected, in round 2, c is selected. However, if the agent casts the ballot $\{a, d\}$ then in round 1, d is selected, and in round 2, a is selected.

OBSERVATION 2. *Under RAV an agent i who wants to make C^* the winning set may need to approve a strict subset of C^*.*

Suppose we are selecting a winning set of size $k = 3$ with $C = \{a, b, c, d\}$, using lexicographic tie-breaking:

$$A_2 = \{b, d\}, A_3 = \{c, d\}, A_4 = A_5 = A_6 = \{b, c, d\}$$
$$A_7 = \{b\}, A_8 = \{c\}, A_9 = A_{10} = \{a\}.$$

If agent 1 has favored set $C^* = \{a, b, d\}$ and he approves all of them, then in round 1, b is selected, in round 2, c is selected, and in round 3, a is selected. However, if the agent casts the ballot $\{a, d\}$ then in round 1, d is selected, in round 2, a is selected, and in round 3, b is selected, exactly the favored set.

With these observations we are able to show RAV–WM is NP-complete.

THEOREM 5. *RAV–WM is NP-complete even for a single manipulator ($j = 1$).*

PROOF. To show that RAV is NP-complete to manipulate we reduce from the 3SAT decision problem [18]. Given an instance of 3SAT with w variables $\Phi = \{\phi_1, \ldots, \phi_w\}$, t clauses $\Psi = \{\psi_1, \ldots, \psi_t\}$, inducing $2w$ literals $\{l_1, \ldots, l_{2w}\}$, we construct an instance of RAV, (C, A, k) where a manipulator's preferred candidate c^* is in the winning set if and only if there is an assignment to the variables in Φ such that all clauses are satisfied.

For each variable ϕ_i introduce 2 candidates in C, corresponding to the positive and negative literal of that variable, and $2n - i$ agents approving the 2 candidates; note that $n \gg w + t$. For each clause ψ_j introduce two additional new candidates, corresponding to the clause being satisfied or unsatisfied, along with $2n - w - j$ new agents approving both new candidates. Additionally for each clause ψ_j, we add an agent in N approving each of the candidates that correspond to the positive and negative literals in ψ_j; this ensures that both the positive and negative literal have the same weight of approval in the set of agents. We also need to add 2 agents approving the candidate corresponding to the negation of the clause to maintain the weighting. Finally, add an extra 2 candidates to C, a and b. We add 2 agents approving the candidate corresponding to a clause being unsatisfied, and 2 agents approving the the candidate corresponding to each clause being satisfied and approving b.

Add t agents approving a. The size of the winning set k is equal to $|\Phi| + |\Psi| + 1$. Intuitively, the manipulator must approve of a setting of all the variables in the original 3SAT instance that satisfies all the clauses, plus the preferred candidate. We can now see that the manipulating agent is only capable of ensuring candidate a is selected by computing a solution to the initial 3SAT instance. □

The above proof also shows it is NP-complete to determine if C^* can be made a subset of the winning set, i.e. $C^* \subset W$. From the construction above, and setting $t = 1$, $u(c^*) = 1$, $u(c) = 0$ for all $c \in C \setminus \{c^*\}$, we can state the following corollary.

COROLLARY 5. *RAV–UM is NP-complete.*

Finally, the proof can also be modified to show the following:

THEOREM 6. *RAV–WSM is NP-complete even for a single manipulator ($j = 1$).*

The important observation is that for RAV, the order in which the candidates are included in the committee matters. Therefore we can modify the reduction so that all literals end up in the committee, but the manipulator will impact which ones are selected first. We can then ensure that the committee will consist of exactly all literal agents, the satisfied clause agents, and the preferred candidate if and only if the manipulator ensures that the literals corresponding to a satisfying assignment were chosen first.

5.3 Satisfaction Approval Voting (SAV)

We start this section with the following straightforward observation, namely to select a given candidate c^*, the manipulators need only approve c^*.

OBSERVATION 3. *SAV–WM is polynomial-time solvable.*

It follows that we can also construct the set of candidates that can possibly win in polynomial time.

It is more difficult to decide if a given k-set of candidates can possibly win. With certain voting rules, this problem simplifies if there exists an optimal strategy of j manipulating agents where they all approve the same set of candidates. This is not the case with SAV. Suppose $k = 3$ and $C = \{a, b, c, d, e, f, g\}$, one agent approves both a and b, and three agents approve d, e, f and g. If there are two more agents who want a, b and c to be selected, then one agent needs to approve c and the other both a and b, or one agent needs to approve a and b, and the other a and c. This makes it difficult to decide how a coalition of agents must vote. In fact, it is intractable in general to decide if a given set of candidates can be made winners.

THEOREM 7. *SAV–*WSM *is NP-complete.*

PROOF. The proof is by reduction from the permutation sum problem, inspired by the NP-hardness proof for Borda manipulation with two agents [11]. In the permutation sum problem, we are given integers $X_1 \leq \cdots \leq X_n$ with $\sum_i X_i = n(n+1)$ and the question is if there exist two permutations σ and π of 1 to n such that $\sigma(i) + \pi(i) = X_i$. We construct an instance such there is a strategic vote for $2n$ manipulating agents which ensures a given result if and only if there exists a solution to an instance of the permutation sum problem. In this instance, we want all the candidates alphabetically before e to win.

Our proof exploits the fact that for large m, an approval vote for $m+x$ candidates, all of whom win, results in a satisfaction score of $\frac{1}{m} - \frac{x}{m^2} + O(\frac{1}{m^3})$ for each of these candidates. For large m, terms in $O(\frac{1}{m^3})$ will only ensure we do not need to tie-break in favour of our preferred candidates. We will therefore ignore such terms. Note that the exact choice of m is not very critical other than to ensure such higher order terms are unimportant. For example, we could set $m = n^2$.

We construct an instance with the following candidates: b_y for $1 \leq y \leq n$, $c_{x,y}$ and $d_{x,y}$ for $1 \leq x \leq n$ and $1 \leq y \leq m + x - 1$, the danger candidate e, and some dummy candidates $f_{x,y,z}$, $g_{x,y,z}$ and $h_{y,z}$ for $1 \leq x \leq n$, $1 \leq y \leq m + x - 1$ and $1 \leq z \leq m^2 - 1$. We want all the candidates alphabetically before the danger candidate e to be selected. That is, $k = 2mn + n^2$ and we want b_y, $c_{x,y}$ and $d_{x,y}$ to be the winners.

We construct the votes in our instance as follows: there is one approval ballot for e, giving e a satisfaction score of 1. Hence, each candidate in the winning set will need to have a satisfaction score greater than or equal to 1. Next, for $x = 1$ to n and $y = 1$ to $m+x-1$, we have m^2-m+i votes, each of m^2 approvals for $c_{x,y}$ and $m^2 - 1$ dummy candidates $f_{x,y,z}$ where $z = 1$ to m^2-1. Similarly, for $x = 1$ to n and $y = 1$ to $m + x - 1$, we have $m^2 - m + x$ votes, each of m^2 approvals for $d_{x,y}$ and $m^2 - 1$ dummy candidates $g_{x,y,z}$ where $z = 1$ to $m^2 - 1$. Finally, for $y = 1$ to n, for each partial sum X_y, we have $m^2 - 2m + X_y$ votes, each of m^2 approvals for b_y and $m^2 - 1$ dummy candidates $h_{y,z}$ where $z = 1$ to $m^2 - 1$.

We now ask if we can cast $j = 2n$ additional votes so that the preferred candidates win. The candidates $c_{x,y}$ and $d_{x,y}$ each have a satisfaction score before the final agents vote of $1 - \frac{1}{m} + \frac{x}{m^2}$ ignoring higher order terms. For $c_{x,y}$ and $d_{x,y}$ to win, by a pigeonhole argument, they must each receive an approval vote from the $2n$ agents still to vote with between $m + x - 1$ and $m + x$ approvals. Similarly, the candidate b_y will have a satisfaction score before the agents vote of $1 - \frac{2}{m} + \frac{X_y}{m^2}$ ignoring higher order terms. Again, by a pigeonhole argument, each b_y must receive two approval ballots from the agents containing $m + \sigma(y)$ and $m + \pi(y)$ approvals respectively where $\sigma(y) + \pi(y) = X_y$. This also forces the approval ballots for $c_{x,y}$ and $d_{x,y}$ to win to contain exactly $m + x$ approvals. The winning ballots are thus of the form $\{c_{x,1}, \ldots, c_{x,m+y-1}, b_{\sigma(y)}\}$ and $\{d_{x,1}, \ldots, d_{x,m+y-1}, b_{\pi(y)}\}$. Note, we can swap $c_{x,l}$ and $d_{x,l'}$ between two such ballots without changing the result as long as the number of approvals in each ballot remains fixed. Hence, there is a manipulation if and only if there are two permutations with the appropriate sums. □

The proof requires neither the number of manipulating agents nor the size of the winning set to be constant. Thus, we turn to the special cases of a single agent and a pair of agents. Winning set manipulation is polynomial-time solvable with either one or two agents left to vote. This result holds even if the size of the winning set is not bounded (e.g. $k = m/2$).

On the other hand, SAV–WSM is polynomial-time solvable for $j = 1$ or 2. The main idea for the case for one manipulator is to identify the lexicographically minimum candidate $c \in W \setminus C^*$ with the highest satisfaction score when the manipulator does not vote. In order to ensure that C^* is the winning set (if possible), the manipulator approves all those candidates in C^* that either have smaller satisfaction score than c or the same score but lesser tie-breaking priority than c.

THEOREM 8. *If $j = 2$, then SAV–*WSM *can be solved in polynomial time.*

PROOF. Consider the satisfaction scores of the candidates in the winning set before agents 1 and 2 vote. Let s be the highest score of any candidate in $C^* \setminus W$. Let c be the lexicographically smallest candidate from $C^* \setminus W$ with score s.

Note that A_1 and A_2 will only include candidates in C^* since a vote on a candidate in $C \setminus C^*$ will never be beneficial. We show a simple algorithm such that if C^* can be made a winning set with two approval ballots of size at most j_1 and j_2 then the algorithm will produce two such ballots. Note that the manipulator ballots will add $1/j_1$ to the scores of the candidates approved by A_1 and $1/j_2$ to the scores of the candidates approved by A_2.

For every two integers j_1 and j_2, $k \geq j_1 \geq j_2 \geq 1$, the algorithm performs the following steps.

If there is a candidate $c' \in C^*$ such that c' has score less than $s - 1/j_2 - 1/j_1$ or c' has score equal to $s - 1/j_2 - 1/j_1$ and c is lexicographically smaller than c, then $C^* \neq W$ for any A_1 and A_2 with $|A_1| = j_1$ and $|A_2| = j_2$.

Otherwise, let $C^*_{\perp}$ be the set of all candidates $c' \in C^*$ such that c' has score less than $s - 1/j_2$ or c' has score equal to $s - 1/j_2$ and c is lexicographically smaller than c'. The candidates in $C^*_{\perp}$ must be in both A_1 and A_2 otherwise they will not reach a score of at least s. The candidates in C^* with score less than $s - 1/j_1$ (or equal to $s - 1/j_1$ if they are lexicographically larger than c) must be in A_2 and, finally, the remaining candidates in C^* with score smaller than s (or equal to s if they are lexicographically larger than c) must be in either A_1 or A_2. In the final step, one can distribute these candidates among A_1 and A_2 by guaranteeing, if possible, that $|A_1| \leq j_1$ and $|A_2| \leq j_2$.

	$\mathcal{R}$–WD	$\mathcal{R}$–TestWS	$\mathcal{R}$–WM	$\mathcal{R}$–WSM	$\mathcal{R}$–UM
AV	in P	in P	in P	in P	in P
PAV	**NP-hard** (Cor. 1)	**coNP-c** (Thm. 2)	**coNP-c** ($j = 1$) (Cor. 4)	**coNP-c** ($j = 1$) (Cor. 4)	**coNP-c** (Cor. 4)
RAV	in P	in P	**NP-c** ($j = 1$) (Thm. 5)	**NP-c** ($j = 1$) (Thm. 6)	**NP-c** ($j = 1$) (Cor. 5)
SAV	**in P** (Thm. 1)	**in P** (Thm. 1)	**in P** (Obs. 3)	**NP-c** (Thm. 7) **in P** ($j \in \{1, 2\}$) (Thm. 8)	**NP-c** (Thm. 10)

Table 1: Summary of computational results for approval-based multi-winner rules for **W**inner **D**etermination, **Test** **W**inning **S**et, **W**inner **M**anipulation, **W**inning **S**et **M**anipulation, and **U**tility **M**anipulation. Value j denotes the number of additional strategic ballots. Results in **bold** are from this paper; other results are from Kilgour [22] and Meir et al. [29].

If the size of A_1 is at most j_1 and the size of A_2 is at most j_2 then C^* is a winning set, otherwise there are no ballots with at most j_1 and j_2 approved candidates such that C^* is a winning set. □

In contrast to the complexity of SAV–WM, ensuring that a candidate c^* is *not* in the winning set of size k under rule SAV is intractable.

THEOREM 9. *The following problem is NP-complete: are there j additional approval ballots so that candidate c^* is* not *in the winning set of size k under rule SAV.*

Hence, in the case of multi-winner voting rules, destructive manipulation can be computationally harder than constructive manipulation. This contrasts to the single winner case where destructive manipulation is often easier than constructive manipulation [10]. It also follows from Theorem 9, simply by adding Boolean utilities to the winning set, that it is intractable for an agent (who can cast multiple ballots) to manipulate SAV to ensure at least a given utility.

THEOREM 10. *SAV–UM is NP-complete.*

6. CONCLUSIONS

We have studied some basic computational questions regarding three prominent rules for multi-winner approval voting: PAV, RAV and SAV. These three rules are designed with the intention of selecting a more representative set of winners than the more popular AV method. Our results are summarized in Table 1. We closed the computational complexity of computing the winner for PAV and studied the computational complexity of computing a best response for PAV, RAV and SAV. In many settings, we proved that it is NP-hard for an agent or agents to compute how best to vote given the other approval ballots.

Our most important conclusion is that among the three approval-based rules considered, RAV is the only rule for which winner determination is polynomial-time solvable but winner manipulation is NP-hard. From this computational perspective, RAV is more attractive than AV, PAV, and SAV. To complement this complexity study, it would be interesting to undertake further axiomatic and empirical analyses of prominent approval-based rules as well as considering other variants of the rules [23]. Such analyses would provide further insight into the relative merits of the voting methods. In addition, as several of our results are worst case, it would be interesting to undertake an empirical study that looks at the actual computational hardness in practice using, for instance, data from PrefLib.org [27].

Acknowledgments

NICTA is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program. Serge Gaspers is the recipient of an Australian Research Council Discovery Early Career Researcher Award (project number DE120101761). Joachim Gudmundsson is funded by the Australian Research Council (project number FT100100755).

REFERENCES

[1] H. Aziz, S. Gaspers, N. Mattei, N. Narodytska, and T. Walsh. Ties matter: Complexity of manipulation when tie-breaking with a random vote. In *Proc. of the 27th AAAI Conference*, pages 74–80, 2013.

[2] H. Aziz, M. Brill, V. Conitzer, E. Elkind, R. Freeman, and T. Walsh. Justified representation in approval-based committee voting. In *Proc. of the 29th AAAI Conference*, 2015.

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

[4] D. Baumeister, G. Erdélyi, E. Hemaspaandra, L. A. Hemaspaandra, and J. Rothe. Computational aspects of approval voting. In *Handbook on Approval Voting*, chapter 10, pages 199–251. Springer, 2010.

[5] S. J. Brams and D. M. Kilgour. Satisfaction approval voting. In *Voting Power and Procedures*, Studies in Choice and Welfare, pages 323–346. Springer, 2014.

[6] S. J. Brams, D. M. Kilgour, and M. R. Sanver. Mathematics and democracy: Designing better voting and fair-division procedures. In *How to elect a representative committee using approval balloting*, pages 83–96. Springer, 2006.

[7] S. J. Brams, D. M. Kilgour, and M. R. Sanver. A minimax procedure for electing committees. *Public Choice*, 132:401–420, 2007.

[8] I. Caragiannis, D. Kalaitzis, and E. Markakis. Approximation algorithms and mechanism design for minimax approval voting. In *Proc. of the 24th AAAI Conference*, pages 737–742, 2010.

[9] V. Conitzer. Making decisions based on the preferences of multiple agents. *CACM*, 53(3):84–94, 2010.

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

[11] J. Davies, G. Katsirelos, N. Narodytska, and T. Walsh. Complexity of and algorithms for Borda manipulation. In *Proc. of the 25th AAAI Conference*, pages 657–662, 2011.

[12] R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

[13] E. Elkind, J. Lang, and A. Saffidine. Choosing collectively optimal sets of alternatives based on the condorcet criterion. In *Proc. of the 22nd IJCAI*, pages 186–191, 2011.

[14] E. Elkind, P., Faliszewski, P. Skowron, and A. Slinko. Properties of multiwinner voting rules. In *Proc. of the 13th AAMAS Conference*, pages 53–60, 2014.

[15] P. Faliszewski and A. D. Procaccia. AI's war on manipulation: Are we winning? *AI Magazine*, 31(4):53–64, 2010.

[16] P. Faliszewski, E. Hemaspaandra, and L. A. Hemaspaandra. Using complexity to protect elections. *JACM*, 53(11):74–82, 2010.

[17] P. C. Fishburn. Axioms for approval voting: Direct proof. *Journal of Economic Theory*, 19(1):180–185, 1978.

[18] M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 1979.

[19] J. Goldsmith, J. Lang, N. Mattei, and P. Perny. Voting with rank dependent scoring rules. In *Proc. of the 28th AAAI Conference*, pages 698–704, 2014.

[20] J. Gramm, R. Niedermeier, and P. Rossmanith. Fixed-parameter algorithms for closest string and related problems. *Algorithmica*, 37(1):25–42, 2003.

[21] E. Hemaspaandra, L. Hemaspaandra, and J. Rothe. Exact analysis of Dodgson elections: Lewis Carroll's 1876 voting system is complete for parallel access to NP. *JACM*, 44(6):806–825, 1997.

[22] D. M. Kilgour. Approval balloting for multi-winner elections. In *Handbook on Approval Voting*, chapter 6. Springer, 2010.

[23] D. M. Kilgour and E. Marshall. Approval balloting for fixed-size committees. In *Electoral Systems*, Studies in Choice and Welfare, chapter 12, pages 305–326. Springer, 2012.

[24] K. Konczak and J. Lang. Voting procedures with incomplete preferences. In *Proc. of the 3rd MPREF Workshop*, 2005.

[25] J.-F. Laslier and M. R. Sanver, editors. *Handbook on Approval Voting*. Studies in Choice and Welfare. Springer, 2010.

[26] R. LeGrand, E. Markakis, and A. Mehta. Some results on approximating the minimax solution in approval voting. In *Proc. of the 6th AAMAS Conference*, pages 1193–1195, 2007.

[27] N. Mattei and T. Walsh. Preflib: A library for preferences, http://www.preflib.org. In *Proc. of the 3rd ADT Conference*, pages 259–270, 2013.

[28] R. Meir, A. D. Procaccia, and J. S. Rosenschein. A broader picture of the complexity of strategic behavior in multi-winner elections. In *Proc. of the 7th AAMAS Conference*, pages 991–998, 2008.

[29] R. Meir, A. D. Procaccia, J. S. Rosenschein, and A. Zohar. Complexity of strategic behavior in multi-winner elections. *JAIR*, 33:149–178, 2008.

[30] S. Obraztsova and E. Elkind. On the complexity of voting manipulation under randomized tie-breaking. In *Proc. of the 22nd IJCAI*, pages 319–324, 2011.

[31] S. Obraztsova, E. Elkind, and N. Hazon. Ties matter: Complexity of voting manipulation revisited. In *Proc. of the 10th AAMAS Conference*, pages 71–78, 2011.

[32] S. Obraztsova, Y. Zick, and E. Elkind. On manipulation in multi-winner elections based on scoring rules. In *Proc. of the 12th AAMAS Conference*, pages 359–366, 2013.

[33] T. C. Ratliff. Selecting committees. *Public Choice*, 126 (3/4):343–355, 2006.

[34] P. Skowron, P. Faliszewski, and A. Slinko. Fully proportional representation as resource allocation: Approximability results. In *Proc. of the 23rd IJCAI*, pages 353–359, 2013.

[35] P. Skowron, P. Faliszewski, and A. Slinko. Achieving fully proportional representation is easy in practice. In *Proc. of the 12th AAMAS Conference*, pages 399–406, 2013.

[36] P. Skowron, P. Faliszewski, and J. Lang. Finding a collective set of items: From proportional multirepresentation to group recommendation. In *Proc. of the 4th COMSOC Workshop*, 2014.

[37] T. Walsh. Uncertainty in preference elicitation and aggregation. In *Proc. of the 22nd AAAI Conference*, pages 3–8, 2007.