Bribery as a Measure of Candidate Success: Complexity Results for Approval-Based Multiwinner Rules

Piotr Faliszewski
AGH University
Krakow, Poland
faliszew@agh.edu.pl

Piotr Skowron
TU Berlin
Berlin, Germany
p.k.skowron@gmail.com

Nimrod Talmon
Weizmann Institute of Science
Rehovot, Israel
nimrodtalmon77@gmail.com

ABSTRACT

We study the problem of bribery in multiwinner elections, for the case where the voters cast approval ballots (i.e., sets of candidates they approve) and the bribery actions are limited to: adding an approval to a vote, deleting an approval from a vote, or moving an approval within a vote from one candidate to the other. We consider a number of approval-based multiwinner rules (AV, SAV, GAV, RAV, approval-based Chamberlin–Courant, and PAV). We find the landscape of complexity results quite rich, going from polynomial-time algorithms through NP-hardness with constant-factor approximations, to outright inapproximability. Moreover, in general, our problems tend to be easier when we limit out bribery actions on increasing the number of approvals of the candidate that we want to be in a winning committee (i.e., adding approvals only for this preferred candidate, or moving approvals only to him or her). We also study parameterized complexity of our problems, with a focus on parameterizations by the numbers of voters or candidates.

Keywords

Multiwinner elections, Bribery, Approval-Based voting

1. INTRODUCTION

No one enjoys losing an election. Nonetheless, it is a natural part of life and instead of drowning in sorrow, a skillful candidate (or, a rational agent in a multiagent environment) should rather focus on learning as much as possible from the defeat. In particular, such a candidate deserves to know how well he or she did in the election and how close he or she was to winning. In single-winner elections the candidates typically receive some scores (e.g., in Plurality elections, the most popular type of single-winner elections, these scores are the numbers of voters that consider a given candidate as the best one) and the highest-scoring candidate is a winner. Then reporting the scores for all the candidates gives them some idea of their performance.

This score-reporting approach, however, has a number of drawbacks. First, for some rules either there are no natural notions of the score or ones that exist do not necessarily give a very good idea of a candidate's level of success. For example, under the single-winner variant of the STV rule, the voters rank candidates from the best one to the worst one and we keep on deleting the candidates with the lowest Plurality score until there is only one left, the winner. On the surface, the rule does not assign scores to the candidates. We could, of course, define the STV score as the round number in which the candidate is eliminated, but it would not be very useful: Even a tiny change in the votes can radically change the elimination order (see, e.g., the work of Woodall [34]; the effect is also used in the hardness proofs of manipulation for STV [4, 33]).

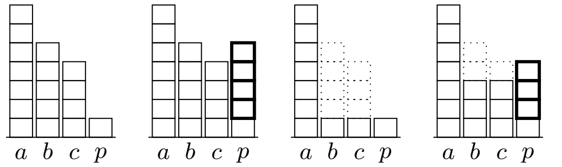
Second, this approach is quite problematic to use within multiwinner elections, where whole committees of candidates are selected. If there were m candidates and the committee size were k, then one would have to list $\binom{m}{k}$ scores, one for each possible committee. One possible remedy would be to list for each candidate p the score and the contents of the best committee that included p. Unfortunately, this would not address the first issue, which in multiwinner elections is even more pressing than in single-winner ones and, more importantly, would not really tell the candidate what *this candidate's* performance was, but rather would bind him or her to some committee.

Third, the scores used by some rules may not be sufficiently informative. For example, in Copeland elections the score of candidate c is the number of candidates d for whom a majority of voters ranks c higher than d. Yet, no one would claim that two candidates with the same Copeland score, where one loses his or her pairwise contests by just a few votes each and the other loses them by a huge margin, performed similarly.

Finally, the notion of a score may be quite arbitrary. Going back to the previous example, the Copeland score can be defined so that a candidate receives 1 point for winning a pairwise contest, -1 point for losing it, and 0 points for a tie, but one may as well define it to give 1 point for a victory, 0 points for a loss, and 0.5 points for a tie. Both approaches are perfectly appropriate and both are used in the literature, but when used as measures of a candidate's success, they need to be interpreted quite differently.

To address the issues mentioned above, we propose an approach based on the BRIBERY family of problems, introduced by Faliszewski et al. [17] and then studied by a number of other authors (see the works of Elkind, Faliszewski, and Slinko [15], Dorn and Schlotter [12], Bredereck et al. [8], and Xia [35] as some examples; we give a more detailed discussion in Section 4). In these problems we are allowed to per-

Appears in: *Proc. of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017), S. Das, E. Durfee, K. Larson, M. Winikoff (eds.), May 8–12, 2017, São Paulo, Brazil.*



(a) Original (b) Adding (c) Deleting (d) Swapping

Figure 1: Approval scores in the election from the introduction, together with an illustration show how d can join the size-2 committee by, respectively, adding, deleting, or swapping approvals.

form some actions that modify the votes and we ask what is the smallest number of such actions that ensure that a given candidate is a winner of the election. The fewer actions are necessary for a particular candidate, the better he or she did in the election (e.g., the winners require no actions at all).

To present our ideas, we focus on approval-based multiwinner elections. We are interested in multiwinner elections because for them measuring the performance of losing candidates is far less obvious than in the single-winner case, and we focus on approval-based rules (where each voter provides a set of candidates that he or she approves), as opposed to rules based on preference orders (where each voter ranks candidates from best to worst), because BRIBERY-style problems for preference-order-based rules are already quite well-studied [19] (even in the multiwinner setting [8]).

Let us now describe our setting more precisely. We assume that we are given an election, i.e., a set of candidates and a collection of voters (each voter with a set of candidates that he or she approves of), and a multiwinner voting rule. Multiwinner rules take as input an election and a committee size k, and output a set of k candidates that form a winning committee (formally, we assume that they output a set of tied committees, but for now we disregard this issue). Let us consider one of the simplest approval-based multiwinner rules, namely APPROVAL VOTING (AV for short; we also consider a number of other rules throughout the paper). Under AV, a candidate receives a point for each voter that approves him or her, and the winning committee consists of the k highest-scoring candidates. Let us assume that we have four candidates, a, b, c, and p, and nine voters, $v_1, \ldots, v_9$, who approve the following candidates:

$$v_1 = \{a, b, c\}, \quad v_2 = \{b, c\}, \quad v_3 = \{a\},$$
$$v_4 = \{a, b\}, \quad v_5 = \{a, b\}, \quad v_6 = \{a, c\},$$
$$v_7 = \{b, c, p\}, \quad v_8 = \{a\}, \quad v_9 = \{a\}.$$

The scores of a, b, c, and p, are, respectively, 7, 5, 4, and 1. (See Figure 1 for a graphical presentation.) For size two, the winning committee is $\{a, b\}$. We analyze the performance of p by considering the following three types of bribery actions:

Adding Approvals. In this case, we are allowed to add candidates to the voters' sets of approved candidates, paying a unit price for each addition. In our example it suffices to add four approvals for p (let us assume that we break ties in favor of p). Indeed, for the case of AV, this number is the difference between the scores of p and the lowest scoring committee member.

Deleting Approvals. In this case we are allowed to remove approvals. In our example one has to remove seven approvals. While deleting approvals may not seem an intuitively good measure of candidate's performance, in fact it behaves quite interestingly. As opposed to adding approvals, not only does it measure how many points a candidate is missing to join the committee, but also it accounts for the number of committee non-members that did better than p.

Swapping Approvals. Here we are allowed to move approvals between candidates within each vote. In our example it suffices to move three approvals (two from b to p, e.g., in the votes v_1 and v_2, and one from c to p, e.g., in v_6). This measure seems to be somewhere between the previous two. It takes into account the score difference between the lowest-scoring committee member and p, the number of candidates with scores in between, and how the approvals are distributed between the votes (within a single vote, we can swap only one approval to p).

Indeed, the above interpretations are particularly easy and natural for AV, but the numbers of approvals that one has to add, delete, or swap to ensure a particular candidate's victory are useful measures for other rules as well.

Unfortunately, many bribery problems are known to be NP-hard. In this paper we study the complexity of our three variants of bribery under approval-based rules (ADDAPPROVALS-BRIBERY, DELETEAPPROVALS-BRIBERY, and SWAPAPPROVALS-BRIBERY) in the following settings: Either each bribery action comes at unit price (as in the examples above) or each bribery action has a separate price (this can be used, e.g., to model certain knowledge about some voters, such as the fact that some voters would never approve our candidate, or would never delete any approvals), and either we allow all possible actions, or only those that increase the number of approvals of our preferred candidate (this restriction does not apply to the case of deleting approvals). We obtain the following results:

1. Most of our problems turn out to be NP-hard for most of our rules (the exceptions include AV in most settings, and GAV and RAV when adding unit-cost approvals for the preferred candidate).

2. Problems where bribery actions are focused on the preferred candidate tend to be easier than the unrestricted ones (e.g., we sometimes obtain 2-approximation algorithms instead of inapproximability results, or FPT algorithms instead of XP ones). Focusing on unit prices has a similar effect.

3. Most of our problems are in FPT parameterized either by the number of candidates or the number of voters.

Due to space restrictions, we omit many of the proofs (available upon request). We included proofs that we felt were most illustrative of the techniques used, and were not based on those already in the literature. We discuss related literature in Section 4.

2. PRELIMINARIES

An approval-based election (C, V) consists of a set C of candidates and a collection $V = (v_1, \ldots, v_n)$ of voters. Each voter has a set of candidates that he or she approves, and—by a slight abuse of notation—we refer to these sets through the voters' names (e.g., we write $v_1 = \{a, b\}$ to indicate that voter v_1 approves candidates a and b).

A multiwinner voting rule is a function $\mathcal{R}$ that given an election $E = (C, V)$ and a number k ($1 \le k \le |C|$) returns a nonempty family of committees (i.e., size-k subsets of C). We treat each committee in $\mathcal{R}(E, k)$ as tied for winning. (Tie-breaking can sometimes affect the complexity of voting problems [11, 29, 28]; our approach is frequently taken in the literature as a simplifying assumption [19]).

Approval-Based Rules. Let (C, V) be an election and let k be the desired committee size. Following Aziz et al. [3, 2], we consider the following six rules (unless we mention otherwise, for each rule described below there is a simple, natural polynomial-time winner determination algorithm):

Approval Voting (AV). Under the AV rule, the score of each candidate is the number of voters that approve him or her. Winning committees are those that contain k candidates with highest scores.

Satisfaction Approval Voting (SAV). Under the SAV rule, each voter v_i gives $\frac{1}{|v_i|}$ points to each of his or her approved candidates (i.e., each voter is given one point that he or she distributes equally among the approved candidates). Winning committees consist of k candidates with highest total scores.

Chamberlin–Courant Approval Voting (CCAV). We say that a voter approves a given committee if he or she approves at least one member of this committee. CCAV selects those committees that are approved by the largest number of voters. (One interpretation is that voters get representatives from the committee; a voter who approves the committee can be represented well). Unfortunately, computing a winning committee under CCAV is NP-hard [30, 5].

Greedy Approval Voting (GAV). GAV was considered by Lu and Boutlier [25] as an approximation algorithm for CCAV, but it turned out to be an interesting rule on its own [14, 2].[1] GAV starts with an empty committee W and executes k rounds, where in each round it adds to W a candidate that maximizes the number of voters who approve at least one member of W (in case of a tie, we assume that there is a fixed tie-breaking rule; thus GAV always returns a single committee). If a winning committee under CCAV is approved by OPT voters, then the committee produced by GAV is approved by at least $(1 - {}^{1}\!/_{e})$OPT voters [25].

Proportional Approval Voting (PAV). Under PAV, voter v_i assigns to committee W score $\sum_{t=1}^{|W \cap v_i|} \frac{1}{t}$. PAV outputs those committees that receive the highest total score from the voters. The rule is NP-hard to compute [31, 3], but—as shown by Aziz et al. [2]—satisfies strong axiomatic properties, making it well-suited for choosing representative bodies (e.g., parliaments, university senates, etc.; GAV and CCAV satisfy weaker variants of these properties).

[1]Strictly speaking, Lu and Boutilier [25] and Elkind et al. [14] discuss a variant of the algorithm that uses preference orders and Borda scores. Nonetheless, their main conclusions transfer to the approval-based case.

Reweighted Approval Voting (RAV). RAV relates to PAV in the same way as GAV relates to CCAV. It starts with an empty committee W and proceeds in k rounds, in each adding to the committee a candidate that maximizes the PAV score of the committee. It guarantees finding a committee whose score is at least a $(1-{}^{1}\!/_{e})$ fraction of that of a PAV winning committee.

Naturally, the above rules have different strengths and weaknesses, and should be applied in different settings [2, 14, 31]. We provide more pointers regarding their properties and history in Section 4.

Bribery Problems. We are interested in bribery problems where we can perform the following types of operations:

AddApprovals: A single operation means adding an approval for a given candidate in a given vote.

DeleteApprovals: A single operation means removing an approval from a given candidate in a given vote.

SwapApprovals: A single operation means moving an approval from a given candidate in a given vote to another candidate—originally not approved—within the same vote.

In the basic variant of our problems, each operation comes with the same, unit price.

Definition 1. *Let $\mathcal{R}$ be an approval-based multiwinner rule and let* Op *be one of* AddApprovals, DeleteApprovals, *or* SwapApprovals. *In the* $\mathcal{R}$-Op-Bribery *problem, we are given an election (C, V), a preferred candidate $p \in C$, and two integers, the committee size k and the budget b. We ask whether it is possible to ensure that p belongs to at least one $\mathcal{R}$-winning committee of size k by applying at most b operations of type* Op *to election (C, V).*

We follow Bredereck et al. [8] in that it suffices for p to belong to just one of the winning committees. (The approach where p should belong to every winning committee—as in the work of Meir et al. [27]—would be as natural.)

While Definition 1 gives the baseline variants of our problems, we also consider two modifications. In the priced variant (denoted by operations \$AddApprovals, \$DeleteApprovals, and \$SwapApprovals), we assume that each possible operation comes with a distinct price (that depends both on the voter and on the candidate(s) to which it applies; e.g., adding an approval for p to some vote v could cost 10 units, whereas adding an approval for some other candidate c to the same vote v could cost 2 units). That is, our problems are closer to Swap Bribery and Shift Bribery of Elkind et al. [15, 13] than to Bribery and \$Bribery of Faliszewski et al. [17] (where upon paying a voter's price, one can modify the vote arbitrarily).

We also distinguish variants of the (\$)AddApprovals and (\$)SwapApprovals operations where one is limited to, respectively, adding approvals only for p or swapping approvals only to p. These problems model natural, positive scenarios, where we want to find out what support candidate p should have garnered to win the election.

3. RESULTS

For all our rules and settings, we seek results of three kinds. First, we check whether the problem is polynomial-time solvable (few rare cases) or is NP-hard (typical). Then,

<table>
<tr><th>operation</th><th colspan="4">Adding Approvals</th><th colspan="2">Deleting Approvals</th><th colspan="4">Swapping Approvals</th></tr>
<tr><th>restriction</th><th colspan="2">(for p)</th><th colspan="2">(none)</th><th colspan="2"></th><th colspan="2">(for p)</th><th colspan="2">(none)</th></tr>
<tr><th>prices</th><th>(unit)</th><th>(any)</th><th>(unit)</th><th>(any)</th><th>(unit)</th><th>(any)</th><th>(unit)</th><th>(any)</th><th>(unit)</th><th>(any)</th></tr>
<tr><td rowspan="4">AV</td><td colspan="2" rowspan="4">P</td><td colspan="2" rowspan="4">P</td><td colspan="2" rowspan="4">P</td><td rowspan="4">P</td><td>NP-hard</td><td rowspan="4">P</td><td>NP-hard</td></tr>
<tr><td>inapprox.</td><td>inapprox.</td></tr>
<tr><td>FPT(m)</td><td>FPT(m)</td></tr>
<tr><td>FPT(n)</td><td>XP(n)</td></tr>
<tr><td rowspan="4">SAV</td><td colspan="2">NP-hard</td><td colspan="2">NP-hard</td><td colspan="2">NP-hard</td><td colspan="2">NP-hard</td><td colspan="2">NP-hard</td></tr>
<tr><td colspan="2">2-approx.</td><td>2-approx.</td><td>inapprox.</td><td colspan="2">inapprox.</td><td colspan="2">inapprox.</td><td>?</td><td>inapprox.</td></tr>
<tr><td colspan="2">FPT(m)</td><td colspan="2">FPT(m)</td><td colspan="2">FPT(m)</td><td>FPT(m)</td><td>XP(m)</td><td>FPT(m)</td><td>XP(m)</td></tr>
<tr><td colspan="2">FPT(n)</td><td>FPT(n)</td><td>XP(n)</td><td colspan="2">XP(n)</td><td colspan="2">FPT(n)</td><td>FPT(n)</td><td>XP(n)</td></tr>
<tr><td rowspan="4">GAV</td><td colspan="2" rowspan="4">P</td><td colspan="2">NP-hard</td><td colspan="2">NP-hard</td><td colspan="2">NP-hard</td><td colspan="2">NP-hard</td></tr>
<tr><td colspan="2">inapprox.</td><td colspan="2">inapprox.</td><td colspan="2">inapprox.</td><td colspan="2">inapprox.</td></tr>
<tr><td colspan="2">FPT(m)</td><td colspan="2">FPT(m)</td><td>FPT(m)</td><td>XP(m)</td><td>FPT(m)</td><td>XP(m)</td></tr>
<tr><td colspan="2">FPT(n)</td><td colspan="2">FPT(n)</td><td colspan="2">FPT(n)</td><td>FPT(n)</td><td>XP(n)</td></tr>
<tr><td rowspan="4">RAV</td><td rowspan="4">P</td><td>?</td><td colspan="2">NP-hard</td><td colspan="2">NP-hard</td><td colspan="2">NP-hard</td><td colspan="2">NP-hard</td></tr>
<tr><td>PTAS</td><td colspan="2">inapprox.</td><td colspan="2">inapprox.</td><td colspan="2">inapprox.</td><td colspan="2">inapprox.</td></tr>
<tr><td>FPT(m)</td><td colspan="2">FPT(m)</td><td colspan="2">FPT(m)</td><td>FPT(m)</td><td>XP(m)</td><td>FPT(m)</td><td>XP(m)</td></tr>
<tr><td>FPT(n)</td><td>FPT(n)</td><td>XP(n)</td><td colspan="2">XP(n)</td><td colspan="2">FPT(n)</td><td>FPT(n)</td><td>XP(n)</td></tr>
<tr><td rowspan="2">CCAV</td><td colspan="2">FPT(m)</td><td colspan="2">FPT(m)</td><td colspan="2">FPT(m)</td><td>FPT(m)</td><td>XP(m)</td><td>FPT(m)</td><td>XP(m)</td></tr>
<tr><td colspan="2">FPT(n)</td><td colspan="2">FPT(n)</td><td colspan="2">FPT(n)</td><td colspan="2">FPT(n)</td><td>FPT(n)</td><td>XP(n)</td></tr>
<tr><td rowspan="2">PAV</td><td colspan="2">FPT(m)</td><td colspan="2">FPT(m)</td><td colspan="2">FPT(m)</td><td>FPT(m)</td><td>XP(m)</td><td>FPT(m)</td><td>XP(m)</td></tr>
<tr><td colspan="2">FPT(n)</td><td>FPT(n)</td><td>XP(n)</td><td colspan="2">XP(n)</td><td colspan="2">FPT(n)</td><td>FPT(n)</td><td>XP(n)</td></tr>
</table>

Table 1: Results for all our rules and all variants of the problems. For each rule and each scenario we report four entries: (1) is the problem in P or is it NP-hard, (2) what is the best known approximation algorithm, (3 and 4) what are the best known parameterized algorithm for parameterization by the number of candidates (m) and the number of voters (n), respectively. Each cell in the table is divided into two columns, one for the unpriced variant of the problem and one for the the priced variant. When a result for both columns is the same, we write it in the middle of the cell.

to deal with NP-hardness, we seek approximation and FPT algorithms. Unfortunately, in most cases we show that our problems are hard to approximate in polynomial time within any constant factor. For the case of parameterized complexity, we show that all of our problems are fixed-parameter tractable (in FPT), provided that we consider unit prices and take as the parameter either the number of candidates or the number of voters. For the case of priced elections, we still get a fairly comprehensive set of FPT algorithms, but we do miss some cases (and we resort to XP algorithms then; nonetheless, we strongly believe that new proof techniques would lead to FPT results for all our cases).

We summarize our results in Table 1. Below we first study our polynomial-time computable rules (AV, SAV, GAV, and RAV), for which we prove P-membership, NP-hardness, and (in)approximability results, and then move on to parameterized complexity, where we consider all the rules.

3.1 The Easy Case: Approval Voting

For AV, almost all our problems can be solved in polynomial time using simple greedy algorithms.

THEOREM 1. *Let* OP *be one of* (\$)ADDAPPROVALS*,* (\$)DELETEAPPROVALS*, and* SWAPAPPROVALS*.* AV-OP-BRIBERY *is in* P *(also for the cases where we can add/swap approvals only to p).*

PROOF SKETCH. Let (C, V) be the input election, p be the preferred candidate, k be the committee size, and b be the budget. For the case of (priced) bribery by adding approvals, it suffices to keep on adding approvals for p in the order of nondecreasing price of this operation, until either p becomes a member of some winning committee or we exceed the budget (adding approvals for others is never beneficial).

For the case of (priced) bribery by deleting approvals, if p is not a winner already then we proceed as follows. Let C' be the set of candidates that have more approvals than p. By "bringing a candidate $c \in C'$ down" we mean the cheapest sequence of approval-deletions that ensures that c has the same number of approvals as p has (we refer to the total cost of this sequence as the cost of bringing c down). We keep on bringing candidates from C' down (in the order of non-decreasing cost of this operation) until p becomes a member of some winning committee or we exceed the budget.

For the case of (unpriced) bribery by swapping approvals, we first guess a threshold T $(0 \leq T \leq |V|)$ and then repeat the following steps until either p belongs to some winning committee or we exceed the budget (if we exceed the budget for every choice of T, then we reject): We let C' be the set of candidates who have more approvals than p, except the $k-1$ candidates approved by most voters (with ties broken arbitrarily, but in the same way in each iteration; this works since we consider unit prices). We remove from C' those candidates who are approved by at most T voters. Then, if C' is nonempty, we move an approval from some $c \in C'$ to p (there is a vote where it is possible because c has more approvals than p). If C' is empty, then we move an approval to p from some arbitrarily chosen candidate in some arbitrarily chosen vote. Intuitively, in this algorithm we guess the score T that we promise p will have upon entering the winning committee, and we keep on moving approvals from "the most fragile" opponents to p, so their scores drop to T, whereas p's score increases to T. $\square$

Unfortunately, AV-\$SWAPAPPROVALS-BRIBERY is NP-hard and hard to approximate within any constant factor. This hardness comes from the fact that when swaps have prices, then it does not suffice to simply know that there will be *some swap* to perform (as in the algorithm above) and one cheap swap may prevent another, more useful, one.

THEOREM 2. AV-\$SWAPAPPROVALS-BRIBERY *is* NP-*hard, even if we are allowed to swap approvals to the preferred candidate only.*

PROOF. We reduce from the INDEPENDENT SET problem, where we are given a graph G and an integer h, and we ask if there is a set of h pairwise non-adjacent vertices in G. INDEPENDENT SET is known to be NP-hard even on cubic graphs, i.e., graphs with vertices of degree three [21].

Let (G, h) be an instance of INDEPENDENT SET, where G is a cubic graph with n vertices. We construct an instance for AV-\$SWAPAPPROVALS-BRIBERY, as follows. We let the candidate set be $C = \{p\} \cup \{c_v \mid v \text{ is a vertex of } G\}$, where p is the preferred candidate. For each edge $e = \{u, v\}$ in G, we introduce a voter v_e who approves the candidates c_u and c_v. For each of these edge voters, each approval swap has unit cost. We introduce further $3h$ voters, each approving all the vertex candidates; all the swaps for these voters cost $3h + 1$. Finally, we set the committee size to $k = n - h + 1$ and the budget to $b = 3h$. This completes the construction which can be computed in polynomial time.

Prior to any approval swaps, p has score zero and every other candidate has score $3 + 3h$ (each vertex touches three edges, and we get $3h$ points from the second group of voters).

If there is a set IS of h pairwise non-adjacent vertices of G, then we can ensure that p belongs to some winning committee: It suffices that for each vertex $v \in IS$, we move the approval from c_v to p for the three edge voters that correspond to the edges touching v (this is possible because IS is an independent set). As a result, p's score increases to $3h$, the scores of the h candidates corresponding to the vertices from IS drop to $3h$, and so $C \setminus \{c_v \mid v \in IS\}$ is a winning committee (and contains p).

For the other direction, note that (1) the score needed for p is $3h$, (2) this score is achieved only if we swap for p in each swap operation, and (3) if p is to be a member of some winning committee then at least h candidates have to lose at least three approvals each. It follows that these h candidates have to form an independent set because otherwise we would not be able to perform all the approval swaps. □

Inapproximability results follow by similar proofs.

THEOREM 3. *For each* $\alpha > 1$*, if* P $\neq$ NP *then there is no polynomial-time* α*-approximation algorithm for* AV-\$SWAPAPPROVALS-BRIBERY *(even if we focus on swapping approvals to* p *only).*

3.2 Chance for Approximation: SAV

On the surface, SAV is very similar to AV. Yet, the fact that adding or deleting a single approval can affect many candidates at the same time (by decreasing or increasing their share of a voter's point) can be leveraged to show NP-hardness of all our problems.

THEOREM 4. *Let* OP *be one of* (\$)ADDAPPROVALS*,* (\$)DELETEAPPROVALS*, and* (\$)SWAPAPPROVALS. SAV-OP-BRIBERY *is* NP*-hard (also for the cases where we can add/swap approvals only to* p*).*

Fortunately, not all is lost. Using the general technique of Elkind et al. [13, 15], we obtain a 2-approximation algorithm for the (priced) variant of adding approvals for p only. To employ the approach of Elkind et al. [13, 15], it must be the case that (1) after each bribery action, each non-preferred candidate c loses at most as many points as the preferred one gains, (2) there is a pseudo-polynomial time algorithm that computes a bribery action maximizing the score of p for a given budget, and (c) if X and Y are two sets of legal bribery actions (i.e., all bribery actions from X can be executed jointly, and all actions from Y can be executed jointly,) then $X \cup Y$ also is a legal set of bribery actions. These conditions hold for SAV-(\$)ADDAPPROVALS-BRIBERY (for adding approvals to p only) and we get the following result.

THEOREM 5. *There is a* 2*-approximation polynomial-time algorithm for* SAV-(\$)ADDAPPROVALS-BRIBERY *for the case where we add approvals to* p *only.*

The theorem also works for SAV-ADDAPPROVALS-BRIBERY (i.e., for the unrestricted, unpriced case) because if there is a solution that adds approvals to some candidates other than p, then there is also one with the same cost or lower that adds approvals to p only. (If we add an approval for some candidate c, $c \neq p$, in a vote where p is not approved, then it is better to add the approval to p. If we add an approval in a vote where p already is approved, then it is better to not make this addition.)

On the other hand, the above technique does not apply to SAV-\$ADDAPPROVALS-BRIBERY (e.g., there are bribery actions that do not increase the score of the preferred candidate but decrease the scores of others, which breaks condition (1) above) and, indeed, we obtain inapproximability.

THEOREM 6. *For each* $\alpha > 1$*, if* P $\neq$ NP *then there is no polynomial-time* α*-approximation algorithm for* SAV-\$ADDAPPROVALS-BRIBERY.

The proof follows by noting that the classic SETCOVER problem (which is not approximable within any constant factor when P $\neq$ NP) can be embedded within SAV-\$ADDAPPROVALS-BRIBERY. The key idea is to model each set from a SETCOVER instance as a voter. Due to the nature of SAV, as soon as we add an approval to a vote, the scores of all the previously approved candidates (who correspond to elements) decrease. Our construction guarantees that to make p winner, one needs to decrease the score of all element-candidates and, thus, adding an approval to a "set voter" can be viewed as covering the elements from the corresponding set. It is possible to provide such construction which preserves the inapproximability bound of SETCOVER.

The proof for SAV-(\$)DELETEAPPROVALS-BRIBERY relies on similar tricks, but is far more involved (again, we cannot use the 2-approximation technique because deleting an approval for a candidate decreases his or her score more than it increases the score of the preferred candidate).

THEOREM 7. *For each* $\alpha > 1$*, if* P $\neq$ NP *then there is no polynomial-time* α*-approximation algorithm for* SAV-\$DELETEAPPROVALS-BRIBERY.

The case of swapping approvals is more tricky. We cannot use the 2-approximation trick, because if X and Y are two sets of approval-swaps to perform (each possible to execute) then $X \cup Y$ may be impossible to perform (e.g., it may require to move an approval to the preferred candidate within some vote from two different candidates). In fact, for the case where we only move approvals to the preferred candidate, we obtain outright inapproximability result (which immediately translates to the unrestricted, priced setting; with high prices we can enforce approval-swaps to p only). The general result for unit-price swaps remains elusive.

THEOREM 8. *For each $\alpha > 1$, if* P $\neq$ NP *then there is no polynomial-time α-approximation algorithm for* SAV-(\$)SWAPAPPROVALS-BRIBERY *for the case where we only move approvals to p.*

PROOF. Let us fix α to be a positive integer, $\alpha \geq 1$. We will give a reduction f from a restricted variant of the X3C problem to SAV-SWAPAPPROVALS-BRIBERY (for the case where we can move approvals to p only) and argue that an α-approximation algorithm for the latter would have to decide the former. In our RESTRICTED X3C we are given a set $X = \{x_1, \ldots, x_{3n}\}$ of elements and a family $\mathcal{S} = \{S_1, \ldots, S_{3n}\}$ of sets, such that (a) each set contains exactly three elements, and (b) each element belongs to exactly three sets. We ask if there is a family of n sets from $\mathcal{S}$ whose union is exactly X. This variant remains NP-hard [22].

Let I be an instance of RESTRICTED X3C (with input as described above). We set $N = 27(\alpha n + 1)$ (intuitively, N is simply a value much larger than n) and we form an instance of our problem as follows. We let the candidate set be $C = \mathcal{S} \cup D \cup \{p\}$, where $D = \{d_1, \ldots, d_N\}$ is a set of dummy candidates needed for our construction, and we introduce the following voters:

1. For each element $x_i \in X$, we introduce one voter v_i that approves the three set-candidates $S_{j'}$, $S_{j''}$, $S_{j'''}$ that correspond to the sets that contain x_i. We refer to these voters as element voters.

2. We introduce $n \cdot (N + 3n) - 1$ voters, each approving all the candidates from $\mathcal{S}$ and D. We write V' to denote the set of these voters.

3. We introduce $10nN$ voters, each approving all the candidates in D. We denote the set of these voters by V''.

Prior to bribery, p has score 0, each set candidate has score $1 + n - \frac{1}{N+3n}$, and each dummy candidate has score at least $10n$. We set the committee size to $k = N + 2n + 1$, and the budget to $b = 3n$.

If I is a "yes"-instance, then it is possible to ensure that p belongs to some winning committee using at most $3n$ approval swaps: For each set S_j from the exact cover we take all voters corresponding to elements covered by S_j and for these voters we move approvals from S_j to p. Consequently, p is approved by all the voters corresponding to elements of X and obtains $1/3 \cdot 3n = n$ points. Since the score of each of the sets from the exact cover drops to $n - 1/N+3n < n$, there are n candidates with score lower than p. In effect, p belongs to a winning committee.

Now, consider what happens if I is a "no"-instance. After $3\alpha n$ swaps, the score of p can be at most $n + (3\alpha n - 3n)/N < n + 1/9$ (at best, we can get n points from the element voters using $3n$ swaps, and use the remaining $3\alpha n - 3n$ swaps for voters in V''). Since there is no exact cover, after executing all the swaps there are at most $n - 1$ set candidates such that no element voter approves them. Every other set candidate is approved by at least one element voter and at least $n \cdot (N + 3n) - 1 - 3\alpha n$ voters from V'. The score of such candidate is, thus, at least:

$$
\begin{aligned}
&1/3 + \big(n \cdot (N + 3n) - 1 - 3\alpha n\big) \cdot \tfrac{1}{N+3n} \geq \\
&1/3 + n - \tfrac{3\alpha n + 1}{N} \geq n + 1/3 - 1/9 > n + 1/9.
\end{aligned}
$$

The candidates from D have even higher scores. Consequently, at most $n - 1$ candidates have scores lower than p and so p cannot be a member of a winning committee.

Thus, if there were a polynomial-time α-approximation algorithm for our problem, then we could use it to decide the NP-hard RESTRICTED X3C problem. $\square$

3.3 Mostly Hard Cases: GAV and RAV

Unfortunately, for GAV and RAV we obtain an almost uniform set of NP-hardness and inapproximability results. The only exception regards (priced) adding approvals for the preferred candidate.

THEOREM 9. *Let $\mathcal{R}$ be one of* GAV *and* RAV*, and let* OP *be one of* (\$)ADDAPPROVALS*,* (\$)DELETEAPPROVALS*, and* (\$)SWAPAPPROVALS*. For each $\alpha > 1$, if* P $\neq$ NP *then there is no polynomial-time α-approximation algorithm for $\mathcal{R}$-*OP-BRIBERY*. This also holds for* (\$)SWAPAPPROVALS *when we can move approvals only to the preferred candidate.*

The somewhat involved proof of this theorem is inspired by a related result of Bredereck et al [8], for the case of ordinal elections.

Nonetheless, the case of adding approvals for the preferred candidate only is easy for both GAV and PAV (although for PAV in the priced variant we only obtain a PTAS, i.e., a polynomial-time approximation scheme).

THEOREM 10. *When restricted to adding approvals to the preferred candidate only,* {GAV,RAV}-ADDAPPROVALS-BRIBERY *is in* P*. For* GAV*, the priced variant of this problem is also in* P*, whereas for* RAV *there is a PTAS for it.*

PROOF SKETCH. Let (C, V) be an election, let p be the approved candidate, let k be the committee size, and let b be the budget. We consider GAV first. Since it proceeds in k iterations, to ensure that p is selected, we first guess the iteration ℓ in which we plan for p to be added to the committee. We execute GAV until the ℓ'th round. Then we execute the following operation until either p is to be selected in the ℓ'th round[2] or we exceed the budget (in which case, we try a different guess for ℓ, or reject, if we ran out of possible guesses): We find a voter who does not approve any candidate in the so-far-selected committee and for whom the price for adding approval for p is lowest; we add approval for p for this voter. Simple analysis confirms the running time and correctness of the algorithm.

The algorithm for RAV is very similar: We also guess a round number where we plan for p to be selected, and after simulating the algorithm until this round, we add the cheapest set of approvals guaranteeing that p would be selected in this (or earlier) round. The only difference is that for the priced variant, this involves solving an instance of the KNAPSACK problem (each voter has a price for adding approval for p and the number of points that we obtain by this

[2]Technically, it is possible that by our actions p would be selected in an earlier round, but it does not affect the correctness of the algorithm.

approval, which is of the form $1/t$, for some $t \in \{1, \dots, \ell\}$). We can use a classic KNAPSACK PTAS for this task. $\square$

Whether RAV-\$ADDAPPROVALS-BRIBERY is NP-hard when we can add approvals for the preferred candidate only remains open (however, we suspect that it does).

3.4 FPT Algorithms

While for several important special cases we either obtained direct polynomial-time algorithms or polynomial-time approximation algorithms, most of our problems are NP-hard and hard to approximate within any constant factor. Fortunately, if either the number of candidates or the number of voters is considered as the parameter (i.e., can be assumed to be small), we have many FPT algorithms.

Indeed, for the unpriced setting and the parameterization by the number of candidates all our problems are in FPT. This follows by the classic approach of formulating problems as integer linear programs (ILPs) and applying Lenstra's algorithm [24]. Using the approach of Bredereck et al. [7] that combines Lenstra's algorithm with mixed integer linear programming, we also obtain FPT algorithms for the priced cases of adding and deleting approvals.

THEOREM 11. *For each $\mathcal{R}$ in* {AV, SAV, GAV, RAV, CCAV, PAV}*, $\mathcal{R}$-*(\$)ADDAPPROVALS-BRIBERY *(also when we only add approvals for the preferred candidate), $\mathcal{R}$-*(\$)DELETEAPPROVALS-BRIBERY*, and $\mathcal{R}$-*SWAPAPPROVALS-BRIBERY *are in* FPT *when parameterized by the number of candidates.*

The reason why we do not obtain the result for \$SWAPAPPROVALS is that the technique of Bredereck et al. [7] requires that for each set of candidates A, and for each possible set of bribery actions that can be applied to votes approving exactly A—denote such votes as V_A—we have to be able to precompute the cheapest cost of applying these actions to exactly one vote from V_A, to exactly two votes from V_A, etc. This is easy to do for (priced) adding and deleting approvals because bribery actions are independent from each other. Yet, this is impossible for priced approval swaps as the lowest cost of moving an approval from some candidate c to some candidate d, within a vote from V_A may depend on what other swaps were performed before on votes from V_A. Nonetheless, we can handle AV-\$SWAPAPPROVALS-BRIBERY: In this case it suffices to guess the winning committee and score T of its lowest-scoring member; then computing a bribery that ensure that each member of the committee has score at least T and each non-member has score at most T is easy though a min-cost/max-flow argument.

PROPOSITION 12. AV-\$SWAPAPPROVALS-BRIBERY *is in* FPT *when parameterized by the number of candidates.*

For the parameterization by the number of voters, we use a more varied set of approaches. For the case where we add approvals for the preferred candidate only, a simple exhaustive search algorithm is sufficient, even for arbitrary prices. Specifically, it suffices to guess for which voters we add an approval for p, check that it is within the budget, and that p is then selected for some winning committee. Recall that for the parameterization by the number of voters, winner determination is in FPT for all our rules; for PAV and CCAV this follows from the proof of Theorem 15 of Faliszewski et al. [20]. To simplify notation, we will say that a rule has FPT(n) winner determination if there is an FPT algorithm (parameterized by the number of voters) that checks if a given candidate belongs to some winning committee.

THEOREM 13. *For each rule $\mathcal{R}$ with* FPT(n) *winner determination, $\mathcal{R}$-*(\$)ADDAPPROVALS-BRIBERY *for the case where we add approvals for the preferred candidate only is in* FPT *when parameterized by the number of voters.*

There are also general algorithms for the case of unpriced adding or swapping approvals (not necessarily for p). A *unanimous* voting rule is a voting rule for which if there is a candidate which is approved by all the voters, then this candidate is in some winning committee. (Note that all our rules are unanimous.) A rule is *symmetric* if it treats all candidates and voters in a uniform way (i.e., the results do not change if we permute the collection of voters, and if we permute the set of candidates, then this analogously permutes the candidates in the winning committees). We say that two candidates are of the same *type* if they are approved by the same voters; there are at most 2^n candidate types in an election with n voters (this idea of candidate types was previously used by Chen et al. [10]).

THEOREM 14. *For each symmetric, unanimous rule $\mathcal{R}$ with* FPT(n) *winner determination, $\mathcal{R}$-*ADDAPPROVALS-BRIBERY *and $\mathcal{R}$-*SWAPAPPROVALS-BRIBERY *are in* FPT *when parameterized by the number of voters.*

PROOF. Consider an instance of our problem with n voters, where p is the preferred candidate. If the budget is at least n, then we accept because we can ensure that every voter approves p, and p is selected for some winning committee by unanimity. So we assume that the budget is less than n. We (arbitrarily) select n candidates of each candidate type present in the election (or all candidates of a given type, if there are fewer than n of them). These at most $n \cdot 2^n$ candidates are the only ones which we allow to add (for $\mathcal{R}$-ADDAPPROVALS-BRIBERY) or to swap between (for $\mathcal{R}$-SWAPAPPROVALS-BRIBERY). We can now check all possibilities of adding or swapping these candidates, and we accept if at least one leads to p belonging to a winning committee and is within the budget. $\square$

A similar technique works for \$SWAPAPPROVALS, for the case where we are allowed to move approvals to the preferred candidate only. However, this time we cannot arbitrarily choose n candidates of each type, because we might choose candidates for whom moving the approvals is too expensive.

THEOREM 15. *For each symmetric, unanimous rule $\mathcal{R}$ with* FPT(n) *winner determination, $\mathcal{R}$-*\$SWAPAPPROVALS-BRIBERY *(for the case where we are allowed to move approvals to the preferred candidate only) is in* FPT *when parameterized by the number of voters.*

PROOF. Consider an instance of our problem with n voters, where p is the preferred candidate. Since we can move approvals to p only, it follows that we cannot operate twice on the same voter and the number of operations in every solution is at most n. Further, in a solution we might change at most n candidates from their original types to some other types. Consider two types, σ and σ', and note that, as a corollary to the above observation, we have that at most n

candidates of type σ might change to type σ' in the solution. Therefore, for each pair of types σ and σ', we select at most n candidates of type σ which are the cheapest to change to type σ'. These at most $n \cdot 2^{2n}$ candidates are the only ones which we allow to operate on. We can now check all possible sets of move-approval-to-p bribery actions on these candidates, and we accept if at least one leads to p belonging to a winning committee and is within the budget. $\square$

For CCAV and GAV, we use the fact that they operate on candidate types. Then we provide a general XP result.

Theorem 16. *If $\mathcal{R}$ is* CCAV *or* GAV, *then* $\mathcal{R}$-($)AddApprovals-Bribery *and* $\mathcal{R}$-($)DeleteApprovals-Bribery *are in* FPT *(parameterized by the number of voters).*

Proof. Consider some election with n voters, committee size k, and where p is the preferred candidate. The crucial observation is that both for CCAV and GAV, the set of winning committees is fully determined by the candidate types present in the election (irrespective of the number of candidates of each type). This holds because whenever a candidate of some type is included in the committee, then adding another candidate of the same type will not change the committee's score. In consequence, we can think of a winning committee as of a set of (at most k) candidate types. Candidate p belongs to some winning committee if and only if its type belongs to some winning committee.

If $k \geq n$, then we accept because for each candidate type there is a winning committee that includes it (it is always possible to choose at most n candidate types so that the maximum number of voters approve the committee, and then we can add further, arbitrary candidate types).

For the case where $k < n$, we proceed as follows. First, we guess candidate types that should be present in the election after the bribery (we also guess the type that p shall have). Second, we compute the lowest cost of obtaining an election where exactly these candidate types are present (see below for an algorithm). Finally, we check if this cost does not exceed the budget and if there is a winning committee that includes p's type. If so, we accept, and otherwise we try different guesses (and reject if no guess leads to acceptance).

To compute the cost of transforming an election to one with exactly the guessed candidate types, we solve the following instance of the min-cost/max-flow problem [1]. For each candidate c_i from the election, we create a node c_i. We have a source node s and we connect s to each c_i with an arc of capacity 1 and cost 0. For each type σ_j that we guessed to appear in the post-bribery election, we create a node σ_j. We have a target node t and we connect each σ_j to t with an arc of cost 0, infinite capacity, and the requirement that at least 1 unit of flow passes through this arc (this ensures that each of the guessed types actually appears in the election after the bribery). We connect each c_i to each σ_j by an arc with capacity 1 and cost equal to the price of changing the type of c_i to the type σ_j. (For $AddApprovals and $DeleteApprovals we can indeed compute these costs independently for each candidate and each candidate type, using infinite costs to model impossible transformations). This network contains at most $O(m + 2^n)$ nodes, where m is the number of candidates and n is the number of voters. Since there is a polynomial-time algorithm that solves the min-cost/max-flow problem [1] (i.e., that finds a minimum-cost flow that satisfies all the arc requirements and moves a given number of units of flow, m in our case, from the source to the sink), this algorithm computes the desired cost in FPT time with respect to the number of voters. $\square$

Theorem 17. *For each rule and bribery problem studied in this paper, the problem is in* XP *both for the parameterization by the number of candidates and voters.*

4. RELATED WORK

The bribery family of problems was introduced by Faliszewski et al. [17], but in that work the authors mostly (but not only) focused on the case where after "buying" a vote it is possible to change it arbitrarily. Bribery problems where each local change in a vote is accounted for separately, were first studied by Faliszewski et al. [18] (for irrational votes) and then by Faliszewski [16] (in particular, for the single-winner approval setting, by allowing different costs of moving approvals between candidates) and by Elkind et al. [15, 13] (for the standard ordinal model, in the Swap Bribery problem by assigning different costs for swapping adjacent candidates in a preference order, and in the Shift-Bribery problem by assigning different costs for shifting the preferred candidate forward in preference orders). Swap Bribery and Shift Bribery were then studied by a number of authors, including Dorn and Schlotter [12] and Bredereck et al. [6]. Bredereck et al. [8] studied the complexity of Shift Bribery for multiwinner elections (their paper is very close to ours).

Our work was inspired by that of Aziz et al. [3] on the complexity of winner determination and strategic voting in approval-based elections. In addition, Aziz et al. [2] introduced the notion of justified representation and argued why rules such as PAV, CCAV, and GAV should be very effective for achieving proportional representation or, at least, diversity within the committee. For more details regarding AV, SAV, PAV, and RAV, we point the reader to the work of Kilgour [23]. PAV, RAV, CCAV, and GAV were introduced in the 19th century by Thiele [32] (GAV-style rules attracted attention after Lu and Boutilier [25] considered them in the ordinal setting). CCAV is a variant of the Chamberlin–Courant rule [9], but for the approval setting; studied, e.g., by Procaccia et al. [30] and Betzler et al. [5].

Other closely related papers include that of Meir et al. [27] (on the complexity of manipulation and control for multiwinner rules) and those of Magrino et al. [26] and Xia [35] (on using bribery to quantify chances of election fraud; we also use bribery for post-election analysis).

5. OUTLOOK

We believe that our most important contribution is conceptual: We propose to use bribery problems to measure how well each candidate performed in an election. While many (yet, not all) of our results are negative, we show an extensive set of FPT results. It is important to verify how efficiently can our problems be solved in practice.

Acknowledgments

Piotr Faliszewski was supported by the National Science Centre, Poland, under project 2016/21/B/ST6/01509. Piotr Skowron was supported by the ERC grant 639945 (ACCORD) and by a Humboldt Research Fellowship for Postdoctoral Researchers. Nimrod Talmon was supported by a postdoctoral fellowship from I-CORE ALGO.

REFERENCES

[1] R. Ahuja, T. Magnanti, and J. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, 1993.

[2] H. Aziz, M. Brill, V. Conitzer, E. Elkind, R. Freeman, and T. Walsh. Justified representation in approval-based committee voting. *Social Choice and Welfare*, 48(2):461–485, 2017.

[3] H. Aziz, S. Gaspers, J. Gudmundsson, S. Mackenzie, N. Mattei, and T. Walsh. Computational aspects of multi-winner approval voting. In *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS '15)*, pages 107–115, 2015.

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

[5] N. Betzler, A. Slinko, and J. Uhlmann. On the computation of fully proportional representation. *Journal of Artificial Intelligence Research*, 47:475–519, 2013.

[6] R. Bredereck, J. Chen, P. Faliszewski, A. Nichterlein, and R. Niedermeier. Prices matter for the parameterized complexity of shift bribery. *Information and Computation*, 251:140–164, 2016.

[7] R. Bredereck, P. Faliszewski, R. Niedermeier, P. Skowron, and N. Talmon. Elections with few candidates: Prices, weights, and covering problems. In *Proceedings of the 4nd International Conference on Algorithmic Decision Theory (ADT '15)*, pages 414–431, 2015.

[8] R. Bredereck, P. Faliszewski, R. Niedermeier, and N. Talmon. Complexity of shift bribery in committee elections. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI '16)*, pages 2452–2458, 2016.

[9] B. Chamberlin and P. Courant. Representative deliberations and representative decisions: Proportional representation and the borda rule. *American Political Science Review*, 77(3):718–733, 1983.

[10] J. Chen, P. Faliszewski, R. Niedermeier, and N. Talmon. Elections with few voters: Candidate control can be easy. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI '15)*, pages 2045–2051, 2015.

[11] V. Conitzer, M. Rognlie, and L. Xia. Preference functions that score rankings and maximum likelihood estimation. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI '09)*, pages 109–115, July 2009.

[12] B. Dorn and I. Schlotter. Multivariate complexity analysis of swap bribery. *Algorithmica*, 64(1):126–151, 2012.

[13] E. Elkind and P. Faliszewski. Approximation algorithms for campaign management. In *Proceedings of the 6th International Workshop on Internet and Network Economics (WINE '10)*, pages 473–482, 2010.

[14] E. Elkind, P. Faliszewski, P. Skowron, and A. Slinko. Properties of multiwinner voting rules. *Social Choice and Welfare*, 2017. To appear.

[15] E. Elkind, P. Faliszewski, and A. Slinko. Swap bribery. In *Proceedings of the 2nd International Symposium on Algorithmic Game Theory (SAGT '09)*, pages 299–310, 2009.

[16] P. Faliszewski. Nonuniform bribery (short paper). In *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems (AAMAS '08)*, pages 1569–1572, 2008.

[17] P. Faliszewski, E. Hemaspaandra, and L. Hemaspaandra. How hard is bribery in elections? *Journal of Artificial Intelligence Research*, 35:485–532, 2009.

[18] P. Faliszewski, E. Hemaspaandra, L. Hemaspaandra, and J. Rothe. Llull and Copeland voting computationally resist bribery and constructive control. *Journal of Artificial Intelligence Research*, 35:275–341, 2009.

[19] P. Faliszewski and J. Rothe. Control and bribery in voting. In F. Brandt, V. Conitzer, U. Endriss, J. Lang, and A. D. Procaccia, editors, *Handbook of Computational Social Choice*, chapter 7. Cambridge University Press, 2015.

[20] P. Faliszewski, P. Skowron, A. Slinko, and N. Talmon. Multiwinner analogues of the plurality rule: Axiomatic and algorithmic views. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI '16)*, pages 482–488, 2016.

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

[22] T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.

[23] D. Kilgour. Approval balloting for multi-winner elections. In J. Laslier and R. Sanver, editors, *Handbook on Approval Voting*, pages 105–124. Springer, 2010.

[24] H. Lenstra, Jr. Integer programming with a fixed number of variables. *Mathematics of Operations Research*, 8(4):538–548, 1983.

[25] T. Lu and C. Boutilier. Budgeted social choice: From consensus to personalized decision making. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI '11)*, pages 280–286, 2011.

[26] T. Magrino, R. Rivest, E. Shen, and D. Wagner. Computing the margin of victory in IRV elections. Presented at 2011 Electronic Voting Technology Workshop/Workshop on Trushworthy Elections, 2011.

[27] R. Meir, A. Procaccia, J. Rosenschein, and A. Zohar. The complexity of strategic behavior in multi-winner elections. *Journal of Artificial Intelligence Research*, 33:149–178, 2008.

[28] S. Obraztsova and E. Elkind. On the complexity of voting manipulation under randomized tie-breaking. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI '11)*, pages 319–324, July 2011.

[29] S. Obraztsova, E. Elkind, and N. Hazon. Ties matter: Complexity of voting manipulation revisited. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS '11)*, pages 71–78, May 2011.

[30] A. Procaccia, J. Rosenschein, and A. Zohar. On the complexity of achieving proportional representation. *Social Choice and Welfare*, 30(3):353–362, 2008.

[31] P. Skowron, P. Faliszewski, and J. Lang. Finding a collective set of items: From proportional multirepresentation to group recommendation. *Artificial Intelligence*, 241:191–216, 2016.

[32] T. N. Thiele. Om flerfoldsvalg. In *Oversigt over det Kongelige Danske Videnskabernes Selskabs Forhandlinger*, pages 415–441. 1895.

[33] T. Walsh. Uncertainty in preference elicitation and aggregation. In *Proceedings of the 22nd Conference on Artificial Intelligence (AAAI '07)*, pages 3–8, 2007.

[34] D. Woodall. Properties of preferential election rules. *Voting Matters*, 3:Paper 4, 1994.

[35] L. Xia. Computing the margin of victory for various voting rules. In *Proceedings of the 13th ACM Conference on Electronic Commerce (EC '12)*, pages 982–999, June 2012.