The Complexity of Control and Bribery in Majority Judgment

Yongjie Yang
[1]School of Information Science and Engineering, Central South University, Changsha, China
[2]Chair of Economics, Saarland University, Saarbrücken, Germany
yyongjiecs@gmail.com

ABSTRACT

We study strategic voting problems for majority judgment, in which each voter assigns to every candidate a grade and the winners are determined by their majority-grades. We first study the constructive/destructive control by adding/deleting votes/candidates problems. Then we study the bribery problem, where an external agent wants to change the result by asking a limited number of voters to change their votes. In addition, we study the variant of the bribery problem where each voter has a price for changing her grade assigned to a candidate and the external agent has a limited budget. Finally, we propose and study the constructive/destructive control by adding & deleting grades problem where an external agent aims to change the result by adding and deleting some grades simultaneously. We show that majority judgment is immune to constructive control by adding candidates and destructive control by deleting candidates. Moreover, for each other problem, we either derive a polynomial-time algorithm or show it is NP-hard.

Keywords

majority judgment; control; bribery; complexity

1. INTRODUCTION

Voting has significant applications in multi-agent systems, political elections, web spam reduction, pattern recognition, etc. For instance, in multiagent systems, it is often necessary for a group of agents to make a collective decision by means of voting in order to reach a joint goal. However, according to numerous impossibility theorems, there are no voting systems which simultaneously satisfy a set of desirable properties when more than two candidates are involved, see, e.g., [1, 22, 33]. This mainly attributes to the fashion of voting that asks voters to rank all candidates in linear orders.

Recently, Balinski and Laraki [4] introduced a new voting system named majority judgment, which completely discards the fashion of traditional voting where voters are asked to rank candidates in linear orders, and consequently escapes many impossibility theorems [4]. In particular, in majority judgment, each voter measures each candidate by assigning a grade from a common language of grades (e.g., {bad, good, excellent} or $\{0, 1, ..., 100\}$). The winners are then selected according to their majority-grades. In general, the majority-grade of a candidate is the grade assigned by the middlemost voter to the candidate, where the middlemost voter is with respect to an order of the voters according to the grades they assign to the candidate. For instance, if five voters respectively assign the grades average, average, good, good, classic to a wine, then the majority-grade of the wine is good. In a series of papers [3, 4, 5, 6], Balinski and Laraki showed that majority judgment not only satisfies many desirable properties and escapes many impossibility theorems in theory, but also performs well in practice. As a matter of fact, when the number of candidates is not small, seldom real-world voting asks voters to rank candidates in linear orders. Instead, letting voters assign grades to candidates seems more efficient. For instance, many websites offer the opportunity for internet users to grade hotels, online shops, movies, etc. Since the publication of the first paper on majority judgment, there have been many papers investigating problems pertaining to majority judgment [3, 20, 30, 40]. It should be pointed out that there are other non rank-based voting systems, such as the range voting and majority voting (see [34] for further details). However, as argued by Balinski and Laraki [4], majority judgment performs well enough in real-world elections to distinguish itself from those voting systems.

Unfortunately, no matter how perfect a voting system is, it unavoidably suffers from strategic behavior. For instance, some voters may cast votes that do not reflect their real preferences, a powerful agent may add some new voters or candidates, some voters may change their votes if they are bribed. Fortunately, there exist prominent approaches to address such issues. In particular, using complexity as a barrier against strategic behavior has been suggested by many researchers, see, e.g., [2, 8, 10, 11, 16, 26, 28, 29, 35, 36]. The key point is that if it is NP-hard for the strategic agent to successfully find out how to change the results, he may refrain from attacking the voting. Therefore, whether a voting system is resistant to attacks has been recognized by many researchers as a significant property to measure the voting systems [8]. In addition, complexity analysis helps practitioners decide what kind of solution method is appropriate. For polynomial-time solvability results, we directly provide efficient algorithms. On the other hand, hardness results suggest that finding an exact solution is apt to be costly or impractical, and resorting to approximation or heuristic algorithms may be a necessary choice.

In this paper, we are mainly concerned with the complexity of various strategic voting problems for majority judgment, where an external agent wants to change the result by reconstructing the election. There would be two goals that the external agent wants to reach: making a non-winning candidate p win the election, or making a winning candidate p lose the election. The former case is indicated by the term *constructive* and the latter case by the term *destructive*. The candidate p is often referred to as the *distinguished candidate* in the literature. We first study several stan-

Appears in: *Proceedings 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.*

dard control problems, namely constructive/destructive control by adding/deleting votes/candidates. In these problems, a constructive/destructive external agent aims to reach his goal by either adding some votes (CCAV/DCAV), deleting some votes (CCDV/DCDV), adding some candidates (CCAC/DCAC) or deleting some candidates (CCDC/DCDC). In particular, we show that CCAV and CCDV for majority judgment are NP-hard even when there are only two grades. In addition, we show that majority judgment is immune to CCAC and DCDC, i.e., it is impossible for the external agent to reach his goal by carrying out corresponding strategic operations. On the other hand, we show that all the remaining problems are polynomial-time solvable.

In addition, we study the bribery problem where an external agent (briber) has an incentive to change the results by bribing some voters. A bribed voter is asked to recast a new vote. We show that the problem is NP-hard if the briber has a constructive goal, and is polynomial-time solvable if the briber has a destructive goal. We also study a variant of the bribery problem where it costs 1 dollar for a voter to change a grade she assigned to some candidate, and the briber has a limited budget. We show that this problem is polynomial-time solvable no matter whether the briber has a constructive or a destructive goal.

Finally, we propose and study a problem where an external agent aims to change the results by adding and deleting some grades. In this problem, we assume that each voter has a linear order preference over the grades for each candidate, and the voter assigns the most preferred grade existing in the current voting to the candidate. In particular, we assume that each voter has an ideal grade to each candidate, and the further a grade is from the ideal grade the less it is preferred. We show that this problem is polynomial-time solvable, no matter whether the external agent has a constructive or a destructive goal.

2. PRELIMINARIES

Majority Judgment. Throughout this paper, we interchangeably use the terms "voter" and "vote". In *majority judgment*, we have a set $\mathcal{C}$ of *candidates*, a multiset $\mathcal{V}$ of votes (voters), and a common language of grades G associated with a linear order $\lhd$ used to compare the grades. Such a 4-tuple $(\mathcal{C}, G, \lhd, \mathcal{V})$ is called an *election*. For $g, g' \in G$, $g \trianglelefteq g'$ means $g \lhd g'$ or $g = g'$. Each voter assigns to each candidate a grade from G. We denote by $v(c)$ the grade a voter $v \in \mathcal{V}$ assigns to a candidate $c \in \mathcal{C}$. For a vote v, a subset C of candidates and a grade g, $v(C) = g$ means $v(c) = g$ for every $c \in C$. The *majority-grade* of a candidate $c \in \mathcal{C}$ is the middlemost grade in $(v_1(c), v_2(c), ..., v_n(c))$, where $(v_1(c), v_2(c), ..., v_n(c))$ is an order of all votes such that for every $1 \leq x < y \leq n$ it holds that $v_x(c) \trianglelefteq v_y(c)$. More precisely, the majority-grade of c is $v_j(c)$, where $j = \lfloor \frac{n}{2} \rfloor$. For a $g \in G$, if a candidate c has a majority-grade g' such that $g' \trianglelefteq g$ (resp. $g \trianglelefteq g'$), we say c has a majority-grade at most g (at least g). Moreover, if c has a majority-grade g' such that $g' \lhd g$ (resp. $g \lhd g'$), we say c has a majority-grade less than g (more than g). The candidates with the highest majority-grade are called the *winners* of $(\mathcal{C}, G, \lhd, \mathcal{V})$. If there is only one winner, we call the winner the *unique winner*; otherwise, we call them *co-winners*.

Example. Consider the scenario where a website wants to select the most popular movie among Zootopia, Finding Dory and Weiner, and invites movie fans to vote. Assume that there are 100 voters and each voter evaluates the movies by assigning a grade ranging from 0 to 5 to the movies. The larger the grade a voter assigns to a movie, the more the voter prefers it. The assignments are summarized in Table 1.

Now we formulate the problems studied in the paper.

	0	1	2	3	4	5
Zootopia	15	13	10	32	23	7
Finding Dory	30	19	21	15	15	0
Weiner	5	10	0	25	50	10

Table 1: The majority-grades of Zootopia, Finding Dory and Weiner are 3, 2 and 4, respectively.

Control. We study 8 control problems, each of which is a special case of the following problem.

Constructive/Destructive Multimode Control

Input: A set $\mathcal{C}$ of candidates, a common language of grades G associated with a linear order $\lhd$, a multiset $\mathcal{V}$ of votes, a subset $\mathcal{D} \subseteq \mathcal{C}$, a distinguished candidate $p \in \mathcal{C} \setminus \mathcal{D}$, a submultiset $\mathcal{U} \subseteq \mathcal{V}$, and four nonnegative integers $k_{\text{AV}} \leq |\mathcal{U}|, k_{\text{DV}} \leq |\mathcal{V} \setminus \mathcal{U}|, k_{\text{AC}} \leq |\mathcal{D}|$ and $k_{\text{DC}} \leq |\mathcal{C} \setminus \mathcal{D}|$.

Question: Are there $D \subseteq \mathcal{D}, C \subseteq \mathcal{C} \setminus (\mathcal{D} \cup \{p\}), V \subseteq \mathcal{V} \setminus \mathcal{U}, U \subseteq \mathcal{U}$ such that $|D| \leq k_{\text{AC}}, |C| \leq k_{\text{DC}}, |U| \leq k_{\text{AV}}, |V| \leq k_{\text{DV}}$ and p wins/loses $(B, G, \lhd, W)$, where $B = ((\mathcal{C} \setminus \mathcal{D}) \setminus C) \cup D$ and $W = ((\mathcal{V} \setminus \mathcal{U}) \setminus V) \cup U$?

In the definition, candidates in $\mathcal{D}$ and votes in $\mathcal{U}$ are called *unregistered candidates* and *unregistered votes*, respectively. Accordingly, candidates in $\mathcal{C} \setminus \mathcal{D}$ and votes in $\mathcal{V} \setminus \mathcal{U}$ are called *registered candidates* and *registered votes*, respectively. We study 8 special cases of the above problem. Precisely, we study CCAV, CCDV, CCAC, CCDC, DCAV, DCDV, DCAC and DCDC, where the first two characters "CC|DC" stand for "constructive control|destructive control", and the last two characters "AV|DV|AC|DC" stand for "adding votes|deleting votes|adding candidates|deleting candidates". For $X \in \{\text{AV, DV, AC, DC}\}$, CCX (DCX) is the special case of Constructive (Destructive) Multimode Control, such that $k_Y = 0$ for every $Y \in \{\text{AV, DV, AC, DC}\} \setminus \{X\}$. In addition, for $X \in \{\text{AV, DV, DC}\}$, $\mathcal{D} = \emptyset$ and for $X \in \{\text{AC, DC, DV}\}$, $\mathcal{U} = \emptyset$. For simplicity, when we mention an instance of a problem, we ignore the components with values 0 or $\emptyset$.

Bribery. In the bribery problem, an external agent bribes voters and asks them to change their votes. The external agent has a limited budget. The formal definition is as follows.

Constructive/Destructive Bribery (CB/DB)

Input: A set $\mathcal{C}$ of candidates, a common language of grades G associated with a linear order $\lhd$, a multiset $\mathcal{V}$ of votes, a distinguished candidate $p \in \mathcal{C}$, and a nonnegative integer $k \leq |\mathcal{V}|$.

Question: Can we change at most k votes in $\mathcal{V}$ to make p win/lose $(\mathcal{C}, G, \lhd, \mathcal{V})$?

In addition, we study a variant of the bribery problem, where it costs 1 dollar for a voter to change a grade, and the briber has k dollars to pay.

Constructive/Destructive Grade Bribery (CGB/DGB)

Input: A set $\mathcal{C}$ of candidates, a common language of grades G associated with a linear order $\lhd$, a multiset $\mathcal{V}$ of votes, a distinguished candidate $p \in \mathcal{C}$, and an integer $0 \leq k \leq |\mathcal{C}| \cdot |\mathcal{V}|$.

Question: Can we change at most k grades assigned to the candidates to make p win/lose $(\mathcal{C}, G, \lhd, \mathcal{V})$?

Grade Control. We study the scenario where an external agent wants to change the result by adding and deleting some grades. Before giving the formal definition of the problem, we discuss how the

grade of a candidate assigned by a voter changes if some grades are deleted or added. Assume that voter v assigned a grade g to a candidate c. If g is deleted, then a very natural stipulation is that v will assign to c a new grade that is close to g. On the other hand, if some new grade is added, only voters who assigned grades very close to the new grade have an incentive to change their grades. To capture both cases, we assume that for each voter v and each candidate c, the voter v has a strict preference over the grades and v assigns to c the most-preferred grade existing in the current voting. More importantly, we assume that each voter has an ideal grade to every candidate, and the further a grade to the ideal grade with respect to $\lhd$, the less it is preferred. This is reminiscent of single-peaked preferences (more precisely, 1-dimensional Euclidean preferences), where $\lhd$ serves as the societal axis. Clearly, the definition of vote here is compatible with the previous definition. For a set A, let $\mathcal{L}(A)$ be the set of all linear orders over A.

Constructive/Destructive Control by Adding & Deleting Grades (CCADG/DCADG)

Input: A set $\mathcal{C}$ of candidates, a common language of grades G associated with a linear order $\lhd$, a multiset of votes $\mathcal{V}$ where each $v \in \mathcal{V}$ is a function $v : \mathcal{C} \mapsto \mathcal{L}(G)$ such that for every $c \in \mathcal{C}$ and every three grades $g_x, g_y, g_z \in G$ with $g_x \lhd g_y \lhd g_z$, $g_z\, v(c)\, g_y$ implies $g_y\, v(c)\, g_x$, a distinguished candidate $p \in \mathcal{C}$, a subset $G' \subseteq G$, and two nonnegative integers $k_{\text{AG}} \leq |G'|$ and $k_{\text{DG}} \leq |G \setminus G'|$.

Question: Are there $H \subseteq G \setminus G'$ and $H' \subseteq G'$ such that $|H| \leq k_{\text{DG}}$, $|H'| \leq k_{\text{AG}}$, and p wins/loses the election

$$(\mathcal{C}, (G \setminus (G' \cup H)) \cup H', \lhd, \mathcal{V})?$$

Example. The following example shows how adding and deleting grades affects the majority-grade of a candidate. Imagine that 7 experts are invited to evaluate a wine. There are five potential grades namely, extremely flawed, below average, average, good and classic. The preferences of the experts are shown in Figure 1. If the organizer asks the experts to evaluate the wine only with the grades extremely flawed, average and classic, then the majority-grade of the wine is average. If the organizer adds the grade below average, then the majority-grade of the wine will be below average. Moreover, if the given grades are extremely flawed, good, classic, the majority-grade of the wine is good. See the table below for the details.

	{E, A, C}	{E, B, A, C}	{E, G, C}
3 : (blue)	E	B	E
3 : (red)	A	A	G
1 : (green)	A	B	G
majority-grade:	A	B	G

In all problems with the constructive goal defined above, we assume that the distinguished candidate p does not win the election in advance, i.e., each instance with all integers in the input being 0 is a NO-instance. Moreover, in all problems with the destructive goal, we assume that p wins the election in advance, i.e., each instance with all integers in the input being 0 is a NO-instance. Throughout this paper, $m = |\mathcal{C}|, n = |\mathcal{V}|$ and $t = |G|$.

Our NP-hardness results are mainly based on reductions from the following NP-hard problem [21].

Exact 3-Set Cover (X3C)
Input: A universal set $H = \{c_1, c_2, ..., c_{3\kappa}\}$ and a collection $S = \{s_1, s_2, ..., s_\lambda\}$ of 3-subsets of H.
Question: Is there an $S' \subseteq S$ such that $|S'| = \kappa$ and each $c_i \in H$ appears in exactly one set of S'?

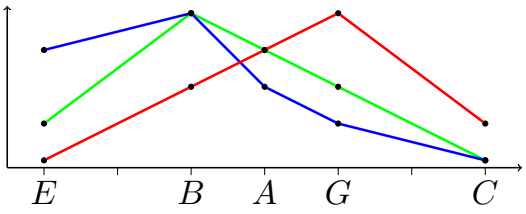

Figure 1: An illustration of how adding and deleting grades affects the majority-grade of a candidate. Here, "E, B, A, G, C" stand for "extremely flawed, below average, average, good, classic**", respectively. There are three experts with the preference B $\succ$ E $\succ$ A $\succ$ G $\succ$ C (blue line), three experts with the preference G $\succ$ A $\succ$ B $\succ$ C $\succ$ E (red line), and one expert with the preference B $\succ$ A $\succ$ G $\succ$ E $\succ$ C (green line).**

We assume that each element $c_i \in H$ occurs in exactly 3 subsets of S. This assumption does not affect the NP-hardness [23]. Observe that under this assumption, we have that $\lambda = 3\kappa$.

Remarks. Following the convention [19, 37], for each problem we distinguish between the *unique-winner model* and the *nonunique-winner model*. In the unique-winner model, winning an election means to be the unique winner, while in the nonunique-winner model, winning an election means to be the unique winner or to be a co-winner. All our results hold for both the unique-winner model and the nonunique-winner model, and we will not state this explicitly in the theorems. Due to space limitations, unless stated otherwise, our proofs are solely based on the unique-winner model.

It should be pointed out that majority judgment was proposed as a single-winner voting system. In particular, Balinski and Laraki [4] put forward two schemes to break the tie when several candidates have the same highest majority-grade. However, as shown in [20, 40], both schemes lead to undesirable results. This is one of the reasons that we discard tie-breaking schemes in our discussions, and instead study both the unique-winner model and the nonunique-winner model of each problem.

Finally, we would like to point out that in the analysis of the running times of all our polynomial-time algorithms for the problems studied in this paper, we ignore the time to read the input. Nevertheless, all results in Table 2 do not change if we take into account the time to read the input. The reason is that the sizes of the inputs of all problems in this table are asymptotically smaller than the running times of the corresponding algorithms.

3. MAJORITY CONTROL

In this section, we study the complexity of constructive/destructive control by adding/deleting votes/candidates for majority judgment. A motivation is that the issues of adding/deleting votes/candidates occur in many electoral settings, see, e.g., [12, 18] for some concrete examples. In addition, as argued in [18], adding votes pertains to simply encouraging some agents to vote, multiplying the existing agents, or performing false-name attacks. We first show that CCAV and CCDV for majority judgment are NP-hard, even when there are only two grades.

THEOREM 1. *CCAV and CCDV for majority judgment are NP-hard even when there are only two grades.*

Control										Bribery			
Constructive (CC)					Destructive (DC)					Constructive		Destructive	
AV	DV	AC	DC	ADG	AV	DV	AC	DC	ADG	CB	CGB	DB	DGB
NP-h		I	$O(m^2 n \log t)$	$O(mnt^2)$	$O(n^3 mt)$		$O(mn \log t)$	I	$O(mnt^2)$	NP-h	$O(mnt)$	$O(mnt)$	
Thm 1		Thm 2	Thm 4	Thm 11	Thm 5		Thm 7	Thm 2	Thm 11	Thm 8	Thm 9	Thm 10	

Table 2: A summary of our results. Here, "NP-h" stands for "NP-hard" and "I" stands for "immune". The running times of the polynomial-time algorithms ignore the time to read the input. See our remarks in the end of Preliminaries for further details.

PROOF FOR CCAV. Let $\mathcal{I} = (H, S)$ be an instance of X3C. We create an instance $\mathcal{E}_{\mathcal{I}} = (\mathcal{C}, G, \lhd, \mathcal{V}, p \in \mathcal{C}, \mathcal{U} \subseteq \mathcal{V}, k_{\text{AV}} = \kappa)$ as follows. For each $c_i \in H$, we create a candidate. For simplicity, we still use c_i to denote the corresponding candidate. In addition, we have a distinguished candidate p. Hence, in total there are $3\kappa+1$ candidates. We have two grades denoted by bad and $good$ such that $bad \lhd good$. We have in total $\kappa - 2$ registered votes v such that $v(p) = bad$ and $v(H) = good$. The unregistered votes are created as follows. For each $s \in S$, we create a vote v_s such that (1) $v_s(p) = good$; and (2) for every $c \in H$, $v_s(c) = good$ if and only if $c \in s$. It remains to prove the correctness.

($\Rightarrow$:) Assume that there is an exact 3-set cover S' of $\mathcal{I}$. Let $U = \{v_s \mid s \in S'\}$ be the set of unregistered votes corresponding to S'. Consider the election $(\mathcal{C}, G, \lhd, (\mathcal{V} \setminus \mathcal{U}) \cup U)$. Due to the construction, there are in total $\kappa - 2$ votes v such that $v(p) = bad$ and κ votes v such that $v(p) = good$. Hence, the majority-grade of p is $good$. For each $c \in H$, there are $\kappa - 1$ votes v such that $v(c) = bad$ and $\kappa - 2 + 1 = \kappa - 1$ votes v such that $v(c) = good$. Hence, the majority-grade of each $c \in H$ is bad. Clearly, p uniquely wins the final election.

($:\Leftarrow$) Assume that there is a solution U of $\mathcal{E}_{\mathcal{I}}$. Observe first that U must contain exactly κ votes . Therefore, there are κ votes v such that $v(p) = good$ and $\kappa - 2$ votes v such that $v(p) = bad$ in the final election, implying p has a majority-grade $good$. If for some $c \in H$, there are at least two votes $v \in U$ such that $v(c) = good$, then c has the majority-grade $good$ as well. Therefore, for each $c \in H$ there must be at most one $v \in U$ such that $v(c) = good$. Due to the construction, this means that $S' = \{s \in S \mid v_s \in U\}$ is an exact 3-set cover of $\mathcal{I}$. □

Majority judgment with two grades is similar to Approval where each voter either approves (assigns the grade $good$ to) or disapproves (assigns the grade bad to) each candidate and candidates with the most approvals win. However, they are different. Consider three candidates a, b, c and three voters v_1, v_2, v_3, where v_1 approves all candidates, v_2 approves a and b, and v_3 approves a. Then, a is the unique Approval winner. However, both a and b are the winners with respect to majority judgment.

Now we study CCAC and DCDC. The Weak Axiom of Revealed Preference (WARP) means that a winner among a subset of candidates always remains a winner among every subset of candidates that includes her. The "unique" version of the Weak Axiom of Revealed Preference (Unique-WARP) means that the unique winner among a subset of candidates always remains the unique winner among every subset of candidates that includes her. Hemaspaandra et al. [24] studied the unique-winner model of several constructive control and destructive control problems, and showed that if a voting system satisfies Unique-WARP, then it is immune to the unique-winner model of CCAC and DCDC. Recall that a voting system is immune to CCAC if it is impossible to make some candidate who does not win the election in advance wins the election by adding at most k_{AC} candidates, and is immune to DCDC if it is impossible to prevent a winner from winning by deleting at most k_{DC} candidates. For the nonunique-winner model, it is easy to see that if a voting system satisfies WARP, then the voting system is immune to the nonunique-winner model of CCAC and DCDC. It is fairly easy to verify that majority judgment satisfies both WARP and Unique-WARP. Then, we have the following theorem.

THEOREM 2. *Majority judgment is immune to CCAC and DCDC.*

CCDC has been shown to be polynomial-time solvable for numerous voting systems such as Approval and Condorcet in the literature. We prove that CCDC is polynomial-time solvable for all voting systems that satisfy some properties. Our result generalizes the polynomial-time algorithms for CCDC for Approval and Condorcet studied in [8, 17, 24], and implies that CCDC for majority judgment is polynomial-time solvable.

The WINNER DETERMINATION problem (WD) for a voting system ϕ is to calculate all winners with respect to ϕ and a given election.

THEOREM 3. *Let ϕ be a voting system that satisfies WARP. The unique-winner model of CCDC for ϕ can be solved in $O(m \cdot f(\ell))$ time, where f is a computable function, ℓ is the size of an instance of WD for ϕ and $O(f(\ell))$ is the complexity of WD for ϕ.*

PROOF. Let p be the distinguished candidate. Assume that there is a candidate $q \neq p$ which is a current winner. If q is still in the final election, then due to the definition of WARP, q is still a winner in the final election. Hence, to make p uniquely win the election, q has to be deleted. Due to this observation, the following algorithm correctly solves the unique-winner model of CCDC for a voting system satisfying WARP:

```
for each i = 1 to k_DC
    if p does not uniquely win the current election, then
        delete one arbitrary current winner except p;
    else return "YES";
return "NO";
```

Clearly, there are at most $k_{\text{DC}} \leq m$ loops, and the running time of each loop is dominated by the complexity of WD for ϕ. Hence, if the winners can be calculated in $O(f(\ell))$ time, the running time of the above algorithm is bounded by $O(m \cdot f(\ell))$. □

The winners with respect to many voting systems can be calculated in polynomial time. Due to Theorem 3, the unique-winner model of CCDC for these voting systems is polynomial-time solvable. Regarding majority judgment, it is easy to see that the winners can be calculated in polynomial-time. Nevertheless, the exact running times of algorithms to solve WD for majority judgment have not been mentioned in the literature so far. As several of our polynomial-time algorithms call an algorithm to solve WD for majority judgment, we study two algorithms for WD for majority judgment. The first algorithm runs in $O(mn \log n)$ time and the second one runs in $O(mn \log t)$ time. In particular, in the first algorithm, we calculate the majority-grades of all candidates first, based

on some sorting algorithms, and then, based on the majority-grades we calculate the winners. In the second algorithm, we guess the highest majority-grade g of the candidates and determine whether each candidate has majority-grade at least g. Let $(\mathcal{C}, G, \lhd, \mathcal{V})$ be an election with respect to which we want to calculate the winners. We first study the complexity of comparing the majority-grade of a candidate and a given grade g. Several of our polynomial-time algorithms also call this procedure.

LEMMA 1. *It takes $O(n)$ time to compare the majority-grade of a candidate c and a grade $g \in G$.*

PROOF. Let y be the number of voters who assign a grade at most g to c, with respect to $\lhd$. Clearly, y can be calculated in $O(n)$ time. Then, due to the definition of majority judgment, if $y < \lfloor n/2 \rfloor$, the majority-grade of c is at least g; otherwise, the majority-grade of c is at most g. Moreover, if $y \geq \lfloor n/2 \rfloor$ and there is at least one voter who assigns the grade g to c, then the majority-grade of c is g. □

Now we are ready to give the algorithms for WD for majority judgment.

LEMMA 2. *WD for majority judgment is solvable in $O(mn \log n)$ time or in $O(mn \log t)$ time.*

PROOF. Consider first the following algorithm. To calculate the majority-grade of a candidate c, we sort the grades assigned by the voters to c (based on $\lhd$). This can be done in $O(n \log n)$ time by many sorting algorithms such as the Merge sorting algorithm [27]. As we have in total m candidates, it takes $O(mn \log n)$ time to calculate the majority-grades of all candidates. Given these majority-grades, it takes $O(m)$ time to calculate the candidates with the highest majority-grade, i.e., the winners. This algorithm terminates in $O(mn \log n) + O(m) = O(mn \log n)$ time.

Consider now another algorithm. Let $(g_1, ..., g_t)$ be the order of G such that $g_x \lhd g_{x+1}$ for every $1 \leq x \leq t-1$. For each $\ell \in \{1, 2, ..., t\}$, let S_ℓ be the set of all candidates whose majority-grades are at least g_ℓ. The algorithm aims to find the largest ℓ such that $S_\ell \neq \emptyset$. Clearly, candidates in such an S_ℓ are exactly the winners. By using binary search with respect to $(g_1, ..., g_t)$, we need only to calculate at most $O(\log t)$ many S_ℓ. Due to Lemma 1, given g_ℓ for some $\ell \in \{1, 2, ..., t\}$ we can calculate S_ℓ in $m \cdot O(n) = O(mn)$ time. As we consider at most $O(\log t)$ different values of ℓ, the whole running time of the algorithm is $O(mn \log t)$. □

Notice that if we take into account the time to read the input, the running time of the first algorithm shown in the proof of Lemma 2 is $O(mn \log n + t)$. The reason that the number of grades does not occur in the running time of the first algorithm is that no matter how many grades we have, we need only the n grades assigned by the voters to calculate the majority-grade of a candidate. Lemma 2 suggests that when t is considerably larger than n, we should use the first algorithm to determine the winners; otherwise, we should turn to the second algorithm. The following analysis is purely based on the second algorithm presented in the proof of Lemma 2.

According to Theorem 3 and Lemma 2, the unique-winner model of CCDC for majority judgment can be solved in $O(m^2 n \log t)$ time. For the nonunique-winner model of CCDC for majority judgment, observe that to make a candidate p a winner by deleting candidates, all candidates having a greater majority-grade than that of p have to be deleted. Hence, we can solve it in polynomial-time with an algorithm similar to the one in the proof of Theorem 3. The following theorem summarizes our results for CCDC.

THEOREM 4. *CCDC for majority judgment can be solved in $O(m^2 n \log t)$ time.*

Now we study DCAV and DCDV for majority judgment. In particular, we show that both problems are polynomial-time solvable.

THEOREM 5. *DCAV and DCDV for majority judgment can be solved in $O(n^3 mt)$ time.*

PROOF. We prove the theorem by deriving polynomial-time algorithms for DCAV and DCDV for majority judgment as follows.

DCAV. Let $I = (\mathcal{C}, G, \lhd, \mathcal{V}, p \in \mathcal{C}, \mathcal{U} \subseteq \mathcal{V}, k_{\text{AV}})$ be a given instance. The algorithm breaks down I into $(m-1) \cdot t$ subinstances, each of which takes I, a candidate $q \in \mathcal{C} \setminus \{p\}$ and a grade $g \in G$ as the input, and asks if we can add at most k_{AV} unregistered votes such that the majority-grade of p is at most g and the majority-grade of q is at least g. Clearly, I is a YES-instance if and only if at least one of the subinstances is a YES-instance. Let $I' = (I, q, g)$ be a subinstance. Now, we illustrate how to solve I'. We first calculate the following submultisets.

$\mathcal{U}_{\downarrow,\uparrow}$: all votes $v \in \mathcal{U}$ such that $v(p) \trianglelefteq g \trianglelefteq v(q)$.

$\mathcal{U}_{\uparrow,\uparrow}$: all votes $v \in \mathcal{U}$ such that $g \lhd v(p)$ and $g \trianglelefteq v(q)$.

$\mathcal{U}_{\downarrow,\downarrow}$: all votes $v \in \mathcal{U}$ such that $v(p) \trianglelefteq g$ and $v(q) \lhd g$.

Clearly, these three sets can be calculated in $O(n)$ time. Let $m_{*,\star} = |\mathcal{U}_{*,\star}|$ for $*, \star \in \{\uparrow, \downarrow\}$. Then, we add arbitrary $\min\{k_{\text{AV}}, m_{\downarrow,\uparrow}\}$ votes v in $\mathcal{U}_{\downarrow,\uparrow}$ and decrease k_{AV} by $\min\{k_{\text{AV}}, m_{\downarrow,\uparrow}\}$. Clearly, this can be done in $O(k_{\text{AV}}) = O(n)$ time. If after doing so p has majority-grade at most g and q has majority-grade at least g, I' is a YES-instance; otherwise, if $k_{\text{AV}} = 0$, I' is a NO-instance. Due to Lemma 1, this can be done in $O(n) + O(n) = O(n)$ time. Assume that none of the above cases occurs. Then, if there are two nonnegative integers k_1 and k_2 such that $k_1 + k_2 \leq k_{\text{AV}}$, and after adding arbitrary $\min\{k_1, m_{\uparrow,\uparrow}\}$ votes in $\mathcal{U}_{\uparrow,\uparrow}$ and arbitrary $\min\{k_2, m_{\downarrow,\downarrow}\}$ votes in $\mathcal{U}_{\downarrow,\downarrow}$, p has majority-grade at most g and q has majority-grade at least g, I' is a YES-instance; otherwise, I' is a NO-instance. As there are $O(k_{\text{AV}}^2) = O(n^2)$ combinations of k_1 and k_2 to consider, the above procedure can be done in $O(n^2) \cdot O(n) = O(n^3)$ time, where $O(n)$ is the time to compare the majority grades of p and q with g (the complexity is due to Lemma 1). In summary, the algorithm to solve I' terminates in $O(n^3)$ time, as its running time is dominated by the complexity of the last case. As we have $(m-1) \cdot t$ subinstances, we can solve I in $(m-1) \cdot t \cdot O(n^3) = O(n^3 mt)$ time.

DCDV. Let $I = (\mathcal{C}, G, \lhd, \mathcal{V}, p \in \mathcal{C}, k_{\text{DV}})$ be a given instance. The algorithm breaks down I into $(m-1) \cdot t$ subinstances, each of which takes I, a candidate $q \in \mathcal{C} \setminus \{p\}$ and a grade $g \in G$ as the input, and asks if we can delete at most k_{DV} votes such that the majority-grade of p is at most g and the majority-grade of q is at least g. Clearly, I is a YES-instance if and only if at least one of the subinstances is a YES-instance. Let $I' = (I, q, g)$ be a subinstance. To solve I', we calculate the following submultisets:

$\mathcal{V}_{\uparrow,\downarrow}$: all votes $v \in \mathcal{V}$ such that $v(q) \lhd g \lhd v(p)$.

$\mathcal{V}_{\uparrow,\uparrow}$: all votes $v \in \mathcal{V}$ such that $g \lhd v(p)$ and $g \trianglelefteq v(q)$.

$\mathcal{V}_{\downarrow,\downarrow}$: all votes $v \in \mathcal{V}$ such that $v(p) \trianglelefteq g$ and $v(q) \lhd g$.

Let $m_{*,\star} = |\mathcal{V}_{*,\star}|$ for $*, \star \in \{\uparrow, \downarrow\}$. We first delete arbitrary $\min\{k_{\text{DV}}, m_{\uparrow,\downarrow}\}$ votes v in $\mathcal{V}_{\uparrow,\downarrow}$ and decrease k_{DV} by $\min\{k_{\text{DV}}, m_{\uparrow,\downarrow}\}$. If after doing so p has majority-grade at most g and q has majority-grade at least g, I' is a YES-instance; otherwise, if $k_{\text{DV}} = 0$, I' is a NO-instance. Assume that none of the above cases occurs. Then, if there are two nonnegative integers k_1 and k_2 such that $k_1 + k_2 \leq k_{\text{DV}}$, and after deleting arbitrary $\min\{k_1, m_{\uparrow,\uparrow}\}$ votes

in $\mathcal{V}_{\uparrow,\uparrow}$ and arbitrary $\min\{k_2, m_{\downarrow,\downarrow}\}$ votes in $\mathcal{V}_{\downarrow,\downarrow}$, p has majority-grade at most g and q has majority-grade at least g, I' is a YES-instance; otherwise, I' is a NO-instance. Analogous to the analysis of the algorithm for DCAV, we can conclude that the algorithm to solve DCDV is bounded by $O(n^3mt)$. □

Finally, we study DCAC. We first develop a polynomial-time algorithm for DCAC for voting systems ϕ satisfying some properties.

THEOREM 6. *The unique-winner model of DCAC for ϕ can be solved in $O(m \cdot f(\ell))$ time if ϕ satisfies Unique-WARP, and the nonunique winner model of DCAC for ϕ can be solved in $O(m \cdot f(\ell))$ time if ϕ satisfies WARP, where ℓ is the size of an instance of WD for ϕ, f is a computable function in ℓ, and $O(f(\ell))$ is the complexity of WD for ϕ in elections consisting of only two candidates.*

PROOF. In an instance of DCAC for ϕ satisfying Unique-WARP, to make the distinguished candidate p not the unique winner, there must be another candidate $q \neq p$ such that q wins the election restricted to p and q. Hence, the following algorithm correctly solves the unique-winner model of DCAC: Enumerate all candidates $q \neq p$ and check if q is a winner in the election restricted to only q and p. If there is such a q, return "YES"; otherwise, return "NO". If the winners in elections with only two candidates with respect to ϕ can be calculated in $O(f(\ell))$ time, the above algorithm terminates in $O(m \cdot f(\ell))$ time. □

There are voting systems such as Kemeny, Dodgson and Young for which WD is not polynomial-time solvable [7, 25, 32]. Nevertheless, for almost all widely-studied voting systems the winners between two candidates are polynomial-time determinable.

It is easy to see that majority judgment satisfies all conditions stated in Theorem 6. Moreover, if there are only two candidates, the winners can be calculated in $O(n \log t)$ time, according to Lemma 2. As a result, we have the following theorem.

THEOREM 7. *DCAC for majority judgment can be solved in $O(mn \log t)$ time.*

4. MAJORITY BRIBERY

In this section, we study the complexity of the bribery problems. The motivation of the study of bribery problems is that the phenomena of vote changing occurs often in real-world voting. An example of scenario is when candidate attempts to change the preferences of voters by running a campaign, or in more extreme cases where this strategy involves paying voters to change their votes, or bribing election officials to get access to already submitted votes in order to modify them. We refer to [9, 15, 19, 39] for further discussions on the bribery problems.

We first prove that the constructive bribery problem for majority judgment is NP-hard even in a very special case.

THEOREM 8. *CB for majority judgment is NP-hard, even when there are only two grades.*

PROOF. Let $\mathcal{I} = (H = \{c_1, c_2, ..., c_{3\kappa}\}, S = \{s_1, s_2, ..., s_{3\kappa}\})$ be an instance of X3C. We create an instance $\mathcal{E}_{\mathcal{I}} = (\mathcal{C}, G, \lhd, \mathcal{V}, p \in \mathcal{C}, k = \kappa)$ as follows.

Candidates. For each $c_i \in H$, we create a candidate. For simplicity, we still use c_i to denote the corresponding candidate. In addition, we have a distinguished candidate p. Hence, we have in total $3\kappa + 1$ candidates.

Grades. We have two grades denoted by *bad* and *good* respectively such that $bad \lhd good$.

Votes. We have in total $4\kappa + 1$ votes. In particular, for each $s \in S$, we create a vote v_s such that (1) $v_s(p) = bad$; and (2) for every $c \in H$, $v_s(c) = bad$ if and only if $c \in s$. In addition, we create two votes v such that $v(H \cup \{p\}) = good$. Finally, we create $\kappa - 1$ votes v such that $v(p) = good$ and $v(H) = bad$.

It remains to prove the correctness.

($\Rightarrow$:) Let $S' \subset S$ be an exact 3-set cover of $\mathcal{I}$. For each $s \in S'$, change v_s so that $v_s(p) = good$ and $v_s(H) = bad$. After these changes, there are 2κ votes v such that $v(p) = bad$ and $2\kappa + 1$ votes v such that $v(p) = good$. So, the majority-grade of p is *good*. For each $c \in H$, there are $3 + (\kappa - 1) + (\kappa - 1) = 2\kappa + 1$ votes v such that $v(c) = bad$ and 2κ votes v such that $v(c) = good$. So, the majority-grade of each $c \in H$ is *bad*. Clearly, p uniquely wins the final election.

($\Leftarrow$:) Assume that there is a $V \subseteq \mathcal{V}$ such that $|V| \leq k$ and we can change votes in V to make p uniquely win. Observe that in the original election, there are $\kappa + 1$ votes v such that $v(p) = good$. Hence, V must contain κ votes v with $v(p) = bad$ which are changed so that $v(p) = good$. This implies that V contains exactly k votes corresponding to S. Let $S' = \{s \mid v_s \in V\}$. To make every $c \in H$ has the majority-grade *bad* in the final election, there must be at least $\kappa - 1$ votes $v_s \in V$, $c \in s$, with $v_s(c) = good$ which are changed so that $v(c) = bad$. As $|V| = \kappa$, this implies that S' is an exact 3-set cover of $\mathcal{I}$. □

Now we study the destructive bribery problem. In contrast to the NP-hardness of the constructive bribery problem, we show that the destructive bribery problem for majority judgment is polynomial-time solvable, regardless of the number of grades.

THEOREM 9. *DB for majority judgment can be solved in $O(mnt)$ time.*

PROOF (UNIQUE-WINNER MODEL). We prove the theorem by giving a polynomial-time algorithm for the problem stated in the theorem. Let $I = (\mathcal{C}, G, \lhd, \mathcal{V}, p \in \mathcal{C}, k)$ be a given instance. The algorithm breaks down I into $(m-1) \cdot t$ subinstances, each of which takes as input I, a candidate $q \in \mathcal{C} \setminus \{p\}$ and a grade $g \in G$, and asks whether we can change at most k votes so that q has majority-grade at least g and p has majority-grade at most g. It is clear that I is a YES-instance if and only if at least one of the subinstances is a YES-instance. In the following, we show how to solve each subinstance. Let $I' = (I, q, g)$ be a subinstance. To solve I', we calculate the following three submultisets.

$\mathcal{V}_{\uparrow,\downarrow}$: all votes $v \in \mathcal{V}$ such that $v(q) \lhd g \lhd v(p)$.

$\mathcal{V}_{\uparrow,\uparrow}$: all votes $v \in \mathcal{V}$ such that $g \lhd v(p)$ and $g \trianglelefteq v(q)$.

$\mathcal{V}_{\downarrow,\downarrow}$: all votes $v \in \mathcal{V}$ such that $v(p) \trianglelefteq g$ and $v(q) \lhd g$.

It is easy to verify that the above three sets can be calculated in $O(n)$ time. Let $m_{*,\star} = |\mathcal{V}_{*,\star}|$ for $*, \star \in \{\uparrow, \downarrow\}$. We first change arbitrary $\min\{k, m_{\uparrow,\downarrow}\}$ votes v in $\mathcal{V}_{\uparrow,\downarrow}$ so that after these changes $v(\{p, q\}) = g$, and decrease k by $\min\{k, m_{\uparrow,\downarrow}\}$. This can be done in $O(n)$ time. If after doing so p has majority-grade at most g and q has majority-grade at least g, I' is a YES-instance; otherwise, if $k = 0$, I' is a NO-instance. According to Lemma 1, this can be checked in $O(n)$ time. If none of the above cases occurs, we do the following. If p has majority-grade more than g (this can be checked in $O(n)$ time based on Lemma 1), let k' be the number of all votes in $\mathcal{V}$ which assign some grade at most g to p. Then, if $k < \lfloor \frac{n}{2} \rfloor - k'$, I' is a NO-instance; otherwise, change arbitrary $\lfloor \frac{n}{2} \rfloor - k'$ votes in $\mathcal{V}_{\uparrow,\uparrow}$ so that after the change each of the changed vote assigns the grade g to p, and decrease k by $\lfloor \frac{n}{2} \rfloor - k'$. Clearly, this can be done in $O(n)$ time. If after doing so q has majority-grade at least

g (this can be checked in $O(n)$ time according to Lemma 1), I' is a YES-instance; otherwise, we need to further change some votes in $\mathcal{V}_{\downarrow,\downarrow}$. In particular, let k' be the number of all votes in $\mathcal{V}$ which assign to q a grade less than g. Then, if $k \geq k' - (\lfloor \frac{n}{2} \rfloor - 1)$, I' is a YES-instance, since we can change arbitrary $k' - (\lfloor \frac{n}{2} \rfloor - 1)$ votes in $\mathcal{V}_{\downarrow,\downarrow}$ so that each of the changed votes assigns q the grade g to make q has a majority-grade at least g; otherwise, I' is a NO-instance. It is fairly easy to check that this can be done in $O(n)$ time. Due to the above analysis, the algorithm to solve I' terminates in $O(n)$ time. As we have $(m-1) \cdot t$ subinstances, the running time of the algorithm to solve I is bounded by $(m-1) \cdot t \cdot O(n) = O(mnt)$. □

Now, we study the variant of the bribery problem where it costs 1 dollar for each voter to change a grade she assigned to a candidate, and the briber wants to pay at most k dollars in total, i.e., only at most k grades assigned by the voters can be changed.

THEOREM 10. *CGB and DGB for majority judgment can be solved in $O(mnt)$ time.*

PROOF. We first give a polynomial-time algorithm for the unique-winner model of CGB. Let $I = (\mathcal{C}, G, \lhd, \mathcal{V}, p \in \mathcal{C}, k)$ be a given instance. Moreover, let $G = \{g_1, ..., g_t\}$ such that $g_x \lhd g_y$ for every $1 \leq x < y \leq t$ and $\mathcal{V} = \{v_1, ..., v_n\}$. The algorithm breaks down I into t subinstances, each of which takes I and a grade $g_u \in G$ with $2 \leq u \leq t$ as the input, and asks whether it is possible to change in total at most k grades of the candidates assigned by the votes so that p has the majority-grade g_u and every candidate in $\mathcal{C} \setminus \{p\}$ has majority-grade at most g_{u-1}. Clearly, I is a YES-instance if and only if at least one of the subinstances is a YES-instance. Let $I' = (I, g_u)$ be a subinstance.

We first study some useful properties. Let c be a candidate with majority-grade g and g' another grade such that $g \lhd g'$. Moreover, let V' be the set of all voters who assign a grade less than g' to c and $y = |V'|$. Then, $y - (\lfloor \frac{n}{2} \rfloor - 1)$ is the minimum number of grades needed to be changed to increase the majority-grade of c to g'. In fact, we need to change arbitrary $y - (\lfloor \frac{n}{2} \rfloor - 1)$ votes in V' so that after the changes each changed vote assigns c the grade g'. For each grade g' and each candidate c, if c has majority-grade less than g', let $k_{c,g',\uparrow} = y - (\lfloor \frac{n}{2} \rfloor - 1)$, where y is as discussed above; otherwise, if c has majority at least g', let $k_{c,g',\uparrow} = 0$. Consider now another case where we want to decrease the majority-grade of a candidate c from her current majority-grade g to some grade g' with $g' \lhd g$. Then, we need to change at least $\lfloor \frac{n}{2} \rfloor - z$ grades, where z is the maximum number of voters who assign a grade at most g' to c. For a grade g' and a candidate c who has majority grade more than g', let $k_{c,g',\downarrow} = \lfloor \frac{n}{2} \rfloor - z$, where z is as discussed above.

Due to the above properties, we solve I' as follows. Let C be the set of candidates in $\mathcal{C} \setminus \{p\}$ who have majority-grades at least g_u. If $k \geq k_{p,g_u,\uparrow} + \sum_{c \in C} k_{c,g_{u-1},\downarrow}$, I' is a YES-instance; otherwise, I' is a NO-instance.

It remains to analyze the running time of the algorithm. Due to Lemma 1, it takes $O(n)$ time to determine if a candidate has majority-grade at least g_u. Therefore, the set C can be calculated in $O(mn)$ time. Due to the above analysis, it takes $O(n)$ time to calculate $k_{c',g',\uparrow}$ and $k_{c',g',\downarrow}$ for a candidate c and grade g'. Hence, it takes $O(mn)$ time to calculate $\sum_{c \in C} k_{c,g_{u-1},\downarrow}$. In summary, the running time of the algorithm to solve I' is bounded by $O(mn) + O(mn) = O(mn)$. As we have $t-1$ subinstances, the running time of the algorithm to solve I is bounded by $O(mnt)$.

A polynomial-time algorithm with the same running time for the nonunique-winner model can be derived from the above algorithm with a slight modification.

Consider now the unique-winner model (nonunique-winner model) of DGB. To prevent the distinguished candidate p from being the unique winner (a winner), there must be another candidate q who has the majority-grade no less than (greater than) that of p, after changing at most k grades. Due to this, we guess such a candidate q and a grade $g_u \in G$, and ask if we can change at most k grades so that the majority-grade of p is at most g_u and the majority-grade of q is at least g_u (g_{u+1}). Notice that for the nonunique-winner model, the guessed grade g_u should be strictly less than g_t, i.e., $1 \leq u < t$. This leads to at most mt subinstances. Then, if $k \geq k_{q,g_u,\uparrow} + k_{p,g_u,\downarrow}$ ($k \geq k_{q,g_{u+1},\uparrow} + k_{p,g_u,\downarrow}$), the subinstance under consideration is a YES-instance; otherwise, it is a NO-instance. This can be done in $O(n)$ time, as calculating $k_{c',g',\uparrow}$ and $k_{c',g',\downarrow}$ for $c' \in \mathcal{C}$ and $g' \in G$ can be done in $O(n)$ time. As we have mt subinstances, we can solve DGB in $O(mnt)$ time. □

5. CONTROL ON GRADES

In this section, we study the problem where an external agent wants to make a distinguished candidate win/lose the election by adding and deleting some grades. This problem captures many real-world scenarios. For instance, the committee chair of a conference or workshop may choose the grade range $\{-3, -2, -1, 0, 1, 2, 3\}$ or $\{1, ..., 10\}$ for reviewers to evaluate papers. We show that the problem is polynomial-time solvable regardless whether the external agent has a constructive goal or a destructive goal.

Before presenting our main results, let's first study two properties. In general, the first property says that if we want to decrease the majority-grade g' of a candidate to a grade $g \lhd g'$, then we have to delete g' and all grades between g and g'. A formal description of the property is summarized in Lemma 3. For $v \in \mathcal{V}, c \in \mathcal{C}$ and $H \subseteq G$, let $v(c, H)$ be the most-preferred grade of v to c among the grades in H, i.e., $v(c, H)$ is the grade the voter v assigns to c if the existing grades in the current voting are exactly the ones in H. In the following, let $I = (\mathcal{C}, G, \lhd, \mathcal{V}, p \in \mathcal{C}, G', k_{\text{AG}}, k_{\text{DG}})$ be a given instance, and $(g_1, ..., g_t)$ the order over G such that $g_x \lhd g_{x+1}$ for every $1 \leq x < t$.

LEMMA 3. *Let c be a candidate with majority-grade g_z where $2 \leq z \leq t$ in $(\mathcal{C}, G \setminus G', \lhd, \mathcal{V})$. Then, to decrease the majority-grade of c to some grade $g_x \lhd g_z$, all grades $g_y \in G \setminus G'$ such that $g_x \lhd g_y \trianglelefteq g_z$ have to be deleted.*

PROOF. Let $(v_1, ..., v_n)$ be an order of the votes in $\mathcal{V}$ such that $v_i(c, G \setminus G') \trianglelefteq v_{i+1}(c, G \setminus G')$ for every $1 \leq i < t$. Let $j = \lfloor n/2 \rfloor$. So, $v_j(c, G \setminus G') = g_z$ and $g_z \trianglelefteq v_h(c, G \setminus G')$ for every $h \geq j$. Let $H = \{g_y \in G \setminus G' \mid g_x \lhd g_y \trianglelefteq g_z\}$. To decrease the majority-grade of c to some grade $g_x \lhd g_z$, there has to be at least one vote v_h with $h \geq j$ which turns to assign to c a grade at most g_x after adding and deleting some grades. However, if some $g_y \in H$ is not deleted, then according to the definition of the votes (precisely, according to the single-peakedness of the votes over the grades), every vote v_h with $h \geq j$ must assign to c a grade at least g_y. □

The second property states that to increase the majority-grade g_x of a candidate to a grade g_z, the grade g_x and all grades between g_x and g_z have to be deleted.

LEMMA 4. *Let c be a candidate with majority-grade g_x where $1 \leq x \leq t-1$ in $(\mathcal{C}, G \setminus G', \lhd, \mathcal{V})$. Let $g_z \in G \setminus G'$ be a grade such that $g_x \lhd g_z$. Then, to increase the majority-grade of c to g_z, all grades $g_y \in G \setminus G'$ such that $g_x \trianglelefteq g_y \lhd g_z$ have to be deleted.*

PROOF. Let $(v_1, ..., v_n)$ and j be as defined in the proof of Lemma 3. So, $v_j(c, G \setminus G') = g_x$ and $v_h(c, G \setminus G') \trianglelefteq g_x$ for

every $h \leq j$. Let $H = \{g_y \in G \setminus G' \mid g_x \trianglelefteq g_y \lhd g_z\}$. To increase the majority-grade of c to g_z, there has to be at least one vote v_h with $h \leq j$ which turns to assign to c a grade at least g_z after adding and deleting some grades. However, if some grade $g_y \in H$ is not deleted, then according to the definition of the votes, every vote v_h with $h \leq j$ must assign to c a grade at most $g_y \lhd g_z$. □

Now we are ready to show our polynomial-time algorithms for CCADG and DCADG based on the above lemmas.

THEOREM 11. *CCADG and DCADG can be solved in $O(mnt^2)$ time.*

PROOF. We first give a polynomial-time algorithm for the unique-winner model of CCADG. In the first step, we check if we can add at most 1 grade in G' to make p the winner. If this is the case, we solve the instance in polynomial time. This can be done in $t \cdot O(mn \log t) = O(mnt \log t)$ time, where $O(mn \log t)$ is the time to calculate the winners (see Lemma 2). Assume now that this is not the case. Then, we break down I into t subinstances, each of which takes I together with a grade $g_z \in G$ where $2 \leq z \leq t$ as the input, and asks if we can add at most k_{AG} grades in G' and delete at most k_{DG} grades in $G \setminus (G' \cup \{g_z\})$ so that the majority-grade of p is at least g_z and the majority-grade of every other candidate is less than g_z. Clearly, I is a YES-instance if and only if at least one of the subinstances is a YES-instance. Let $I' = (I, g_z)$ be a subinstance. If $g_z \in G'$, we remove g_z from G' (i.e., we add g_z) and decrease k_{AG} by one. If after doing so, $k_{\text{AG}} < 0$, we immediately conclude that I' is a NO-instance. Now we distinguish between the following cases. Notice that at this moment, it cannot be the case that p has majority-grade at least g_z and everyone else has majority-grade less than g_z, since this case has been dealt with in the first step of the algorithm. Hence, we have only the following two cases to consider.

Case 1. There is a candidate $q \neq p$ whose majority-grade is at least g_z at the moment.

In this case, according to Lemma 3, g_z has to be deleted to decrease the majority-grade of q to a grade less than g_z. Therefore, we immediately conclude that I' is a NO-instance.

Case 2. All candidates have majority-grades less than g_z.

Let $g_x \lhd g_z$ be the majority-grade of p at the moment. Then, due to Lemma 4, to increase the majority-grade of p to g_z, we delete g_x, and accordingly decrease k_{DG} by one (if $k_{\text{DG}} < 0$ after this, we conclude that I' is a NO-instance). Then, either we are back in Case 1, or stay in Case 2, or p has majority-grade g_z and every other candidate has majority-grade less than g_z. If we are in Case 1 or Case 2, we call the procedure as described above in the corresponding Case; otherwise, we conclude that I' is a YES-instance. Note that, due to Lemma 4, after deleting g_x in Case 2, no candidate can have a majority-grade greater than g_z.

The running time of the algorithm to solve I' is $O(tmn)$ (hint: As we delete in total at most $k_{\text{DG}} \leq t$ grades, we call the procedure in Case 2 at most t times. Moreover, after each deletion we use $O(mn)$ time to compare the majority-grades of all candidates with g_z. See Lemma 1 for the time to compare the grades).

As we have t subinstances, the algorithm to solve I terminates in $O(mnt \log t) + t \cdot O(mnt) = O(mnt^2)$ time, where $O(mnt \log t)$ is the time used in the first step of the algorithm.

A polynomial-time algorithm for the nonunique-winner model of CCADG with the same time complexity can be obtained from the above algorithm by a slight modification.

Consider now the unique-winner (nonunique-winner) model of DCADG. To prevent p from being the unique-winner (a winner), there must be a candidate $q \in \mathcal{C} \setminus \{p\}$ who has a majority-grade at least the same as (more than) that of p. Hence, we guess such a candidate q and a grade $g_z \in G$. Analogous to the algorithm above, if $g_z \in G'$, we remove g_z from G' and decrease k_{AG} by one. This leads to $(m-1) \cdot t$ subinstances. Let $I' = (I, q, g_z)$ be a subinstance. Observe that the majority-grade of a candidate in an election is completely independent of the majority-grades of other candidates. As a result, we can remove all candidates in $\mathcal{C} \setminus \{p, q\}$ without changing the answer to I'. After removing all these candidates, we reduce I' to a subinstance of the nonunique-winner (unique-winner) model of CCADG discussed above, by resetting q as the distinguished candidate (p is not the distinguished candidate now). As we have only two candidates now, the subinstance of CCADG in our case can be solved in $O(nt)$ time. As we have in total $(m-1) \cdot t$ subinstances, the whole running time of the algorithm is bounded by $(m-1) \cdot t \cdot O(nt) = O(mnt^2)$. □

6. CONCLUSION

Since the proposal of majority judgment there have been both endorsements and criticisms appearing in the literature [3, 4, 20, 30]. For instance, on the positive side, majority judgment allows voters to express their opinions more accurate. In addition, majority judgment escapes many traditional impossibilities. Moreover, majority judgment encourages voters to be honest, in the sense that if a voter wants some candidate to have a majority-grade g, then a best strategy for the voter is to assign g to the candidate. On the negative side, the two tie-breaking schemes proposed in [4] may both lead to undesirable results [20]. However, to use majority judgment as a single-winner voting system, a tie-breaking scheme is necessary. In addition, majority judgment is vulnerable to manipulation, i.e., voters can efficiently figure out a best strategy in order to make someone win even without the evaluations of other voters on the candidates: assign to the candidate who they want to win the best grade and assign to other candidates the worst grade [4].

In this paper, we studied the complexity of several strategic voting problems for majority judgment. In all these problems, we assume that voters honestly assign grades to candidates. However, an external agent has an incentive to change the result by reconstructing the election such as adding/deleting votes/candidates/grades, or bribing voters to change their votes or grades. Our results reveal that majority judgment is vulnerable to many of these strategic behavior (see Theorems 4, 5, 7, 9, 10, 11). Nevertheless, it is important to point out that almost all of our polynomial-time algorithms require a full access to the grades assigned by all voters. In practice, however, it is not always the case that an external agent is able to access the full information of the election. It is an intriguing topic to investigate the complexity of these strategic voting problems when only partial information of the election is known. Recently, the complexity of voting problems with partial information has received a considerable attention, see, e.g., [13, 14, 31, 38]. On the other hand, we prove that some strategic voting problems for majority judgment are NP-hard (see Theorems 1 and 8). Again, it should be pointed out that these results are only worst-case based, and whether these NP-hard strategic voting problems for majority judgment are difficult to solve in practice deserves further investigations.

Acknowledgments

The author would like to thank the anonymous AAMAS referees for their helpful comments and Laura Kasper for her careful proofreading of the paper.

REFERENCES

[1] K. J. Arrow. A difficulty in the concept of social welfare. *J. Polit. Econ.*, 58(4):328–346, 1950.

[2] H. Aziz, S. Gaspers, J. Gudmundsson, S. Mackenzie, N. Mattei, and T. Walsh. Computational aspects of multi-winner approval voting. In *AAMAS*, pages 107–115, 2015.

[3] M. Balinski and R. Laraki. Election by majority judgment: Experimental evidence. In Bernard Dolez, Bernard Grofman, and Annie Laurent, editors, *In Situ and Laboratory Experiments on Electoral Law Reform: French Presidential Elections*, chapter 2, pages 13–54. Springer New York, New York, NY, 2011.

[4] M. Balinski and R. Laraki. *Majority Judgment: Measuring, Ranking, and Electing*. MIT Press, Cambridge, MA, 2011.

[5] M. Balinski and R. Laraki. How best to rank wines: Majority judgment. In Eric Giraud-Héraud and Marie-Claude Pichery, editors, *Wine Economics: Quantitative Studies and Empirical Applications*, chapter 8, pages 149–172. Palgrave Macmillan UK, London, 2013.

[6] M. Balinski and R. Laraki. Judge: Don't vote! *Oper. Res.*, 62(3):483–511, 2014.

[7] J. J. Bartholdi III, C. A. Tovey, and M. A. Trick. Voting schemes for which it can be difficult to tell who won the election. *Soc. Choice. Welfare.*, 6(2):157–165, 1989.

[8] J. J. Bartholdi III, C. A. Tovey, and M. A. Trick. How hard is it to control an election? *Math. Comput. Model.*, 16(8-9):27–40, 1992.

[9] D. Baumeister, G. Erdélyi, O. J. Erdélyi, and J. Rothe. Complexity of manipulation and bribery in judgment aggregation for uniform premise-based quota rules. *Math. Social. Sci.*, 76:19–30, 2015.

[10] D. Baumeister, G. Erdélyi, E. Hemaspaandra, L. A. Hemaspaandra, and J. Rothe. *Computational Aspects of Approval Voting*, chapter 10, pages 199–251. Handbook on Approval Voting. Springer Berlin Heidelberg, 2010.

[11] F. Brandt, M. Brill, E. Hemaspaandra, and L. A. Hemaspaandra. Bypassing combinatorial protections: Polynomial-time algorithms for single-peaked electorates. *J. Artif. Intell. Res.*, 53:439–496, 07 2015.

[12] F. Brandt, V. Conitzer, U. Endriss, J. Lang, and A. Procaccia, editors. *Handbook of Computational Social Choice*. Cambridge University Press, 2016.

[13] D. Briskorn, G. Erdélyi, and C. Reger. Bribery in k-Approval and k-Veto under partial information: (extended abstract). In *AAMAS*, pages 1299–1300, 2016.

[14] P. Dey, N. Misra, and Y. Narahari. Complexity of manipulation with partial information in voting. In *IJCAI*, pages 229–235, 2016.

[15] E. Elkind, P. Faliszewski, and A. M. Slinko. Swap bribery. In *SAGT*, pages 299–310, 2009.

[16] P. Faliszewski, E. Hemaspaandra, and L. A. Hemaspaandra. How hard is bribery in elections? *J. Artif. Intell. Res. (JAIR)*, 35:485–532, 2009.

[17] P. Faliszewski, E. Hemaspaandra, and L. A. Hemaspaandra. Multimode control attacks on elections. *J. Artif. Intell. Res. (JAIR)*, 40:305–351, 2011.

[18] P. Faliszewski, E. Hemaspaandra, and L. A. Hemaspaandra. Weighted electoral control. *J. Artif. Intell. Res. (JAIR)*, 52:507–542, 2015.

[19] P. Faliszewski, E. Hemaspaandra, L. A. Hemaspaandra, and J. Rothe. Llull and Copeland voting computationally resist bribery and constructive control. *J. Artif. Intell. Res. (JAIR)*, 35:275–341, 2009.

[20] D. S. Felsenthal and M. Machover. The majority judgement voting procedure: A critical evaluation. *Homo Oeconomicus.*, 25:319–334, 2008.

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

[22] A. Gibbard. Manipulation of voting schemes: A general result. *Econometrica.*, 41(4):587–601, 1973.

[23] T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoret. Comput. Sci.*, 38:293–306, 1985.

[24] E. Hemaspaandra, L. A. Hemaspaandra, and J. Rothe. Anyone but him: The complexity of precluding an alternative. *Artif. Intell.*, 171(5-6):255–285, 2007.

[25] E. Hemaspaandra, H. Spakowski, and J. Vogel. The complexity of Kemeny elections. *Theor. Comput. Sci.*, 349(3):382–391, 2005.

[26] E. Ianovski, L. Yu, E. Elkind, and M. C. Wilson. The complexity of safe manipulation under scoring rules. In *IJCAI*, pages 246–251, 2011.

[27] J. Katajainen and J. L. Träff. A meticulous analysis of mergesort programs. In *CIAC*, pages 217–228, 1997.

[28] A. P. Lin. *Solving Hard Problems in Election Systems*. PhD thesis, Rochester Institute of Technology, 2012.

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

[30] H. K. Mohajan. Majority judgment in an election with Borda majority count. *Int. J. Manag. Trans.*, pages 19–31, 2012.

[31] N. Narodytska and T. Walsh. The computational impact of partial votes on strategic voting. In *ECAI*, pages 657–662, 2014.

[32] J. Rothe, H. Spakowski, and J. Vogel. Exact complexity of the winner problem for Young elections. *Theory. Comput. Syst.*, 36(4):375–386, 2003.

[33] M. Satterthwaite. Strategy-proofness and Arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *J. Econ. Theory.*, 10(2):187–217, 1975.

[34] W. D. Smith. Descriptions of single-winner voting systems. http://m-schulze.9mail.de/votedesc.pdf, 2006.

[35] Y. Yang. Anyone but them: The complexity challenge for a resolute election controller. In *AAMAS*, 2017. To appear.

[36] Y. Yang and J. Guo. The control complexity of r-Approval: from the single-peaked case to the general case. In *AAMAS*, pages 621–628, 2014.

[37] Y. Yang and J. Guo. Controlling elections with bounded single-peaked width. In *AAMAS*, pages 629–636, 2014.

[38] Y. Yang and J. Guo. Possible winner problems on partial tournaments: A parameterized study. In *ADT*, pages 425–439, 2013.

[39] Y. Yang, Y. R. Shrestha, and J. Guo. How hard is bribery with distance restrictions? In *ECAI*, pages 363–371, 2016.

[40] M. A. Zahid and H. de Swart. The Borda majority count. *Inform. Sci.*, 295:429 – 440, 2015.