Auton Agent Multi-Agent Syst (2015) 29:1091–1124
DOI 10.1007/s10458-014-9277-x

Complexity of manipulation, bribery, and campaign management in Bucklin and fallback voting

Piotr Faliszewski · Yannick Reisch · Jörg Rothe · Lena Schend

Published online: 2 October 2014

Abstract A central theme in computational social choice is to study the extent to which voting systems computationally resist manipulative attacks seeking to influence the outcome of elections, such as manipulation (i.e., strategic voting), control, and bribery. Bucklin and fallback voting are among the voting systems with the broadest resistance (i.e., NP-hardness) to control attacks. However, only little is known about their behavior regarding manipulation and bribery attacks. We comprehensively investigate the computational resistance of Bucklin and fallback voting for many of the common manipulation and bribery scenarios; we also complement our discussion by considering several campaign-management problems for these two voting rules.

Keywords Computational social choice · Complexity theory · Voting theory · Manipulation · Bribery · Campaign management · Bucklin voting · Fallback voting

1 Introduction

A central theme in computational social choice (see, e.g., the bookchapter by Brandt et al. [11]) is to study manipulative attacks that seek to influence the outcome of elections, such

A preliminary version of this paper appeared as an extended abstract in the proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS-2014).

P. Faliszewski
AGH University, Krakow, Poland
e-mail: faliszew@agh.edu.pl

Y. Reisch · J. Rothe · L. Schend (✉)
Heinrich-Heine-Universität Düsseldorf, Düsseldorf, Germany
e-mail: schend@cs.uni-duesseldorf.de

Y. Reisch
e-mail: yannick.reisch@uni-duesseldorf.de

J. Rothe
e-mail: rothe@cs.uni-duesseldorf.de

as manipulation (i.e., strategic voting), control, and bribery. In *manipulation* (introduced by Bartholdi et al. [1,2] and, more generally, by Conitzer et al. [14]), voters try to do so by casting insincere votes. In *control* (introduced by Bartholdi et al. [3] and extended by Hemaspaandra et al. [36] to capture further control actions), an election chair tries to influence the election outcome by changing the structure of the election via adding/deleting/partitioning either candidates or voters. In *bribery* (introduced by Faliszewski et al. [27]), an external agent tries to influence the election outcome by bribing certain voters without exceeding some given budget. Some types of bribery-like actions have been studied under the term *campaign management* (see the work of Elkind et al. [20,21]), paying tribute to the fact that certain bribery scenarios have a more positive touch when looking at them through the lenses of a political campaign manager (in fact, every form of bribery can be seen through these lenses, but some are particularly well-suited for this interpretation).

Since the various types of manipulation, control, and bribery attacks are possible in principle for many (or even for all reasonable) voting systems,[1] it has been studied to what extent computational hardness can provide some kind of protection for specific voting systems. These lines of research have been surveyed by Faliszewski et al. [28], Faliszewski and Procaccia [32], and Rothe and Schend [45], as well as in the bookchapters by Conitzer and Walsh [15], Faliszewski and Rothe [33], Faliszewski et al. [30], and Baumeister et al. [4]. In particular, it is known from the work of Erdélyi et al. [22,23,25,26] that Bucklin and fallback voting are among the voting systems with the broadest resistance (in terms of NP-hardness[2]) to control attacks.[3] However, only little is known about the behavior of these two voting systems regarding manipulation and bribery attacks. The only related results we are aware of are due to Schlotter et al. [47] who have studied Bucklin and fallback voting with respect to campaign management, focusing on so-called shift bribery and support bribery.

Closing this wide gap, we comprehensively investigate the computational resistance of Bucklin and fallback voting to many of the common manipulation and bribery scenarios, and we also complement the results of Schlotter et al. [47] by studying two other campaign-management problems, namely swap bribery and extension bribery (to be defined in Sect. 5.1). In each such scenario, we will handle

(a) both the *weighted* and the *unweighted* case (where the voters in a weighted election may be of varying importance, whereas the voters in an unweighted election have unit weight each—"one person, one vote"),
(b) both the *constructive* case (aiming to make a given candidate win) and the *destructive* case (aiming to prevent a given candidate's victory),
(c) both the *nonunique-winner* and the *unique-winner* model (where the former means that it is enough to be one among possibly several winners, whereas the latter requires a candidate to be the one and only winner), and

[1] Recall, for example, the celebrated Gibbard–Satterthwaite theorem [35,46] that, essentially, says that only dictatorial voting rules are strategy-proof; see also its extension to irresolute voting rules that is due to Duggan and Schwartz [18]. The notion of strategic candidacy, studied by Dutta et al. [19], has a similar flavor from the point of view of candidate control. From the point of view of voter control, it is absolutely natural to expect that if we add sufficiently many votes supporting a given candidate, this candidate should win. The same applies to bribery: If we can bribe everyone, we should be able to ensure any candidate's victory.

[2] Resistance to manipulative actions is most often meant to be NP-hardness in the literature. Being a worst-case measure only, NP-hardness does have its limitations. There are also a number of other approaches that challenge such NP-hardness results, surveyed in [45]; for example, there are some experimental results on the control complexity of Bucklin and fallback voting [44].

[3] Other voting systems whose control complexity has been thoroughly studied include plurality, Condorcet, and approval voting [3,36], Llull and Copeland voting [29], a variant of approval voting known as SP-AV [24], range voting and normalized range voting [39], and Schulze voting [40,42] (see [33] for an overview).

(d) in the case of standard bribery problems, we will consider both the case where voters may have *varying prices* to be bribed and the case where they are bribable at *unit price* only.

This paper is organized as follows. In Sect. 2, we define the voting systems Bucklin and fallback and provide the needed background on complexity theory. We then present our results on manipulation in Sect. 3, on bribery in Sect. 4, and on campaign management (i.e., on swap bribery and extension bribery) in Sect. 5. Finally, Sects. 6 and 7 will give some discussion and our conclusions.

2 Preliminaries

2.1 Bucklin and fallback elections

An *election* is a pair (C, V), where $C = \{c_1, \ldots, c_m\}$ is a set of m candidates and $V = (v_1, \ldots, v_n)$ is a list of votes (or ballots) specifying the n voters' preferences over the candidates in C. How these preferences are represented depends on the voting system used. We allow voters to be weighted, i.e., a nonnegative integer weight w_i is associated with each vote v_i. For example, a vote v_i of a voter with weight $w_i = 3$ is counted as if three voters with unit weight would have cast the same ballot. An unweighted election is the special case of a weighted election where each voter has unit weight.

A voting system is a rule for how to determine the winner(s) of a given election. Here we focus on Bucklin and fallback voting only. Both systems use the notion of *(weighted) majority threshold in* V, which is defined by $maj(V) = \lfloor W/2 \rfloor + 1$, where $W = \sum_{i=1}^{n} w_i$ is the total weight of the votes in V. In *Bucklin voting (BV)*, votes are linear rankings of all candidates, denoted by, e.g., $c_2 > c_3 > c_1$, which means that this voter (strictly) prefers c_2 to c_3 and c_3 to c_1. We call the top position in a vote *level* 1, the next position *level* 2, and so on. Starting with the top position and proceeding level by level through the votes in V, we determine the smallest level ℓ such that some candidate(s) occur(s) in at least $maj(V)$ votes up to this level.[4] A bit more formally, for each candidate $c \in C$, let $score^{i}_{(C,V)}(c)$ denote the number of occurrences of c among the top i levels of the votes in V. The *Bucklin score of* c *in* (C, V) is the smallest level k such that $score^{k}_{(C,V)}(c) \geq maj(V)$. Among the candidates from C with smallest Bucklin score, say ℓ, those occurring most often up to level ℓ are the *Bucklin winners* (sometimes specifically called the *level* ℓ *Bucklin winners*).

Fallback voting is a hybrid voting system designed by Brams and Sanver [10] to combine Bucklin with approval voting. Let us first define approval voting, which was proposed by Brams and Fishburn [7] (see also, e.g., [4,8] for more background). In *approval voting*, votes in an election (C, V) are approval vectors from $\{0, 1\}^{\|C\|}$ indicating for each candidate $c \in C$ whether c is approved ("1") by this voter or not ("0"). Every candidate with the highest approval score is an *approval winner*. For each vote $v \in V$, let S_v denote the *approval strategy of* v, i.e., $S_v \subseteq C$ contains the candidates approved by v. In *fallback voting (FV)*, each voter first partitions the set of candidates into the approved ones and the disapproved ones and then provides a linear ranking of the approved candidates. For example, some voter might disapprove of c_1 and c_4, but approve of c_2 and c_3, preferring c_2 to c_3; this vote is denoted by $c_2 > c_3 \mid \{c_1, c_4\}$. To determine the winners in fallback voting, we first try to

[4] In *simplified Bucklin voting*, all these candidates win. However, we consider *Bucklin voting* in the unsimplified version where winners are determined by a slightly more involved procedure. Note that every Bucklin winner, as defined in the main text, also wins in simplified Bucklin voting, but not necessarily the other way round.

find the Bucklin winners when they exist. If so, all Bucklin winners are *fallback winners*. However, due to disapprovals it might happen that there is no Bucklin winner, and in that case all approval winners are *fallback winners*. A bit more formally, given a fallback election (C, V), let $A(c) = \{v \in V \mid c \in S_v\}$ denote the set of voters that approve of candidate $c \in C$, let $A^j(c)$ denote the set of voters that approve of candidate c up to the jth level, and define

$$score_{(C,V)}(c) = \sum_{v_i \in A(c)} w_i \quad \text{and} \quad score^j_{(C,V)}(c) = \sum_{v_i \in A^j(c)} w_i .$$

The *fallback score of c in* (C, V) is the smallest level k such that $score^k_{(C,V)}(c) \geq maj(V)$ (or ∞ if the candidate is never approved by a majority of voters). Among the candidates from C with smallest fallback score, say ℓ, those occurring most often up to level ℓ are the *(level ℓ) fallback winners*. Otherwise (i.e., if no candidate in C satisfies $score^k_{(C,V)}(c) \geq maj(V)$ for any $k \leq m$), all candidates c with maximum $score_{(C,V)}(c)$ are the *fallback winners*.

Bucklin elections are special fallback elections where all voters approve of all candidates. This means that NP-hardness results for control problems in Bucklin elections directly transfer to the same control problems in the more general fallback elections. However, this is not the case for manipulation and bribery because in these problems some votes may change and the two systems offer different ways of changing the votes. In those campaign-management scenarios that we will study for both systems, though, NP-hardness in fallback elections does immediately follow from NP-hardness in Bucklin elections.

2.2 Basics from complexity theory

We assume the reader is familiar with the basic notions from complexity theory such as the complexity classes P and NP, the polynomial-time many-one ($\leq^p_m$) and Turing ($\leq^p_T$) reducibility, and with hardness and completeness with respect to $\leq^p_m$. For more background on complexity theory, see, e.g., the textbooks [41,43]. In our proofs of NP-hardness, we use PARTITION and EXACT COVER BY THREE- SETS, two well-known NP-complete problems [34].

PARTITION	
Given	A set $A = \{1, \ldots, k\}$ and a list $(a_1, \ldots, a_k)$ of nonnegative integers with $\sum_{i=1}^{k} a_i = 2K$, where K is some positive integer.
Question	Is there a set $A' \subseteq A$ such that $\sum_{i \in A'} a_i = \sum_{i \notin A'} a_i = K$?

EXACT COVER BY THREE- SETS (X3C)	
Given	A set $B = \{b_1, b_2, \ldots, b_{3m}\}$, $m \geq 1$, and a collection $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$ of subsets, where for each S_i in $\mathcal{S}$ we have that $S_i \subseteq B$ and $\|S_i\| = 3$.
Question	Is there a subcollection $\mathcal{S}' \subseteq \mathcal{S}$ such that each element of B occurs in exactly one set in $\mathcal{S}'$?

When discussing the running times of our algorithms, we assume arithmetic operations to have unit costs. This is to model the fact that even if the voters are weighted, it is hard to

imagine elections where the weights would be so large as to require arithmetic beyond what current computers handle as unit operations. If one is uncomfortable with this approach, one should multiply the running times of our algorithms for the weighted cases by the logarithm of the sum of the weights of all the voters.

3 Manipulation in Bucklin and fallback voting

3.1 Definitions and overview of results

Conitzer et al. [14] introduced the following decision problem to model manipulation by a coalition of weighted voters. For a given voting system $\mathscr{E}$, define:

$\mathscr{E}$-Constructive Coalitional Weighted Manipulation ($\mathscr{E}$-CCWM)	
Given	A set C of candidates, a list V of nonmanipulative votes over C each having a nonnegative integer weight, where W_V is the list of these weights, a list W_S of the weights of k manipulators in S (whose votes over C are still unspecified) with $V \cap S = \emptyset$, and a designated candidate $c \in C$.
Question	Can the votes in S be set such that c is an $\mathscr{E}$ winner of $(C, V \cup S)$?

The unweighted case $\mathscr{E}$-CCUM is the special case of $\mathscr{E}$-CCWM where all voters and manipulators have unit weight. By changing the question to "… such that c is not a winner in $(C, V \cup S)$?," we obtain the destructive variants, $\mathscr{E}$-DCWM and $\mathscr{E}$-DCUM. Bartholdi et al. [1,2] first studied these problems with only one manipulator, but here we focus on manipulations exerted by coalitions of manipulators.

As we ask in the constructive problems $\mathscr{E}$-CCUM and $\mathscr{E}$-CCWM above whether the designated candidate c can be made *a* winner (or, in the destructive cases, $\mathscr{E}$-DCWM and $\mathscr{E}$-DCUM, whether c can be prevented from being *a* winner) of the resulting election, we say that the problem is stated in the so-called *nonunique-winner model* (sometimes referred to as the *co-winner* or simply the *winner model*). By asking whether the designated candidate can be made a *unique* winner (or, again in the destructive cases, can be prevented from being a *unique* winner) of the resulting election, the so-called *unique-winner model* is described. For the latter, we denote the corresponding decision problems by $\mathscr{E}$-UCCWM, $\mathscr{E}$-UCCUM, $\mathscr{E}$-UDCWM, and $\mathscr{E}$-UDCUM.[5] With that notation, the following proposition follows immediately from the definitions.

Proposition 1 *1. $\mathscr{E}$-CCUM $\leq_m^p$ $\mathscr{E}$-CCWM and $\mathscr{E}$-uCCUM $\leq_m^p$ $\mathscr{E}$-uCCWM.*
2. $\mathscr{E}$-DCUM $\leq_m^p$ $\mathscr{E}$-DCWM and $\mathscr{E}$-uDCUM $\leq_m^p$ $\mathscr{E}$-uDCWM.
3. $\mathscr{E}$-uDCUM $\leq_T^p$ $\mathscr{E}$-CCUM.
4. $\mathscr{E}$-uDCWM $\leq_T^p$ $\mathscr{E}$-CCWM.

Table 1 gives an overview of our results for manipulation in Bucklin and fallback voting. Note that all our results hold in both the unique-winner and the nonunique-winner model. In the following sections, we will provide the detailed proofs for the nonunique-winner model and will explicitly state how the proofs can be adapted to work in the unique-winner model.

[5] We introduce this notation for Proposition 1 only; throughout the rest of the paper we will slightly abuse notation by always using the abbreviations $\mathscr{E}$-CCWM, $\mathscr{E}$-CCUM, $\mathscr{E}$-DCWM, and $\mathscr{E}$-DCUM whenever the winner model will be clear from the context.

Table 1 Overview of results for manipulation in Bucklin and fallback voting

Problem	Bucklin voting		Fallback voting	
	Complexity	Reference	Complexity	Reference
$\mathscr{E}$-CCUM	P	Corollary 1	P	Proposition 3
$\mathscr{E}$-DCUM	P	Corollary 1	P	Proposition 3
$\mathscr{E}$-CCWM	NP-complete	Theorem 1	P	Proposition 2
$\mathscr{E}$-DCWM	P	Theorem 2	P	Proposition 2

3.2 Results for weighted manipulation

In this section we analyze the complexity of weighted manipulation in Bucklin and fallback voting.

In fallback elections, manipulators that try to make a certain candidate a winner by changing their votes can follow a simple strategy: They can limit their approval strategy to only this candidate and thus preclude all other candidates from gaining points from their votes. It is easy to see that if this attempt is not successful, no other way of constructing the manipulators' votes can make their designated candidate win (in both winner models). Since the best strategy for the manipulators is to cast identical votes, we immediately have that fallback-CCWM is in P for both winner models, which with Proposition 1 implies that the destructive variant in the unique-winner model is in P, as well. For the destructive case in the nonunique-winner model, a likewise simple strategy can be followed: The manipulators determine a candidate that is closest to the designated one (that is, they find a candidate who gets most points until the designated candidate's winning level) and approve of this candidate only. We state this observation in the following proposition.

Proposition 2 *Fallback-*CCWM *and fallback-*DCWM *are in P, each in both winner models.*

In weighted Bucklin elections, on the other hand, a coalition of manipulators trying to make a certain candidate win is faced with a harder challenge, as the following result shows.

Theorem 1 *For elections with at least three candidates, Bucklin-*CCWM *is NP-complete in both winner models.*

Proof It is easy to see that Bucklin-CCWM is in NP in both winner models and for all numbers of candidates.

We show NP-hardness of this problem by a reduction from PARTITION. Let an instance of PARTITION be given by $A = \{1, \ldots, k\}$ and $(a_1, \ldots, a_k)$ with $\sum_{i=1}^{k} a_i = 2K$. We will separate the proof into the case of an odd number of candidates, $m \geq 3$, and an even number of candidates, $m \geq 4$, and we will start with the former:

In our reduction, we will use the candidate set $C = \{c_1, c_2, \ldots, c_{m-1}\} \cup \{p\}$, where $m \geq 3$ is an odd number (the desired number of candidates in the constructed election). To simplify the description of the votes, we will use the following interval-like notation:

$$C[i, j] = \begin{cases} c_i > c_{i+1} > \cdots > c_j & \text{if } i < j, \\ c_i > c_{i+1} > \cdots > c_{m-1} > c_1 > \cdots > c_j & \text{otherwise.} \end{cases}$$

For example, by writing $C[1, 4] > p > \cdots$ we mean a preference order described by $c_1 > c_2 > c_3 > c_4 > p > \cdots$ (i.e., we rank candidates c_1, c_2, c_3, and c_4 first, then p, and then all the remaining candidates in some arbitrary-but-easy-to-compute order). Similarly,

Table 2 Voter list V in the proof of Theorem 1 for an odd number $m \geq 3$ of candidates

Preference	Weight
$C\left[1, \frac{m-1}{2}\right] > p > \cdots$	$\frac{m-1}{2}K$
$C\left[\frac{m-1}{2}+1, m-1\right] > p > \cdots$	$\frac{m-1}{2}K$
$C[1, m-1] > p$	K
$C[2, 1] > p$	K
$\vdots$	$\vdots$
$C[m-1, m-2] > p$	K

Table 3 Level-i scores, $i \in \left\{\frac{m-1}{2}, \frac{m-1}{2}+1\right\}$, of the candidates in C for odd $m \geq 3$

	$c \in C - \{p\}$	p
(a) Original election (C, V)		
$score^{\frac{m-1}{2}}$	$(m-1)K$	0
$score^{\frac{m-1}{2}+1}$	mK	$(m-1)K$
(b) Manipulated election $(C, V \cup S)$		
$score^{\frac{m-1}{2}}$	$\leq mK$	$2K$
$score^{\frac{m-1}{2}+1}$	$mK + K$	$mK + K$

$p > C[m-2, 2] > \cdots$ would mean a preference order of the form $p > c_{m-2} > c_{m-1} > c_1 > c_2 > \cdots$.

We construct a Bucklin election (C, V), where the candidate set C is as already specified, and where the voter list is as given in Table 2. Note that the overall weight of the voters in V is $2(m-1)K$.

Let there be k manipulators in S with weights $a_1, a_2, \ldots, a_k$. While in the original election we have a majority threshold of $(m-1)K+1$ points, the majority threshold is reached with $mK+1$ points in the election with the manipulators.

Since p receives no points at all in (C, V) before level $\frac{m-1}{2}+1$ and has fewer points than any other candidate on this level (see Table 3a), p is not a Bucklin winner of the original election (C, V).

We claim that $(A, (a_1, a_2, \ldots, a_k)) \in$ PARTITION if and only if p can be made a Bucklin winner in $(C, V \cup S)$.

From left to right: Assume there is a subset $A' \subseteq A$ such that $\sum_{i \in A'} a_i = K$. We can set the preferences of the manipulators as follows:

- $p > C\left[1, \frac{m-1}{2}\right] > \cdots$ for all manipulators with weight a_i for $i \in A'$, and
- $p > C\left[\frac{m-1}{2}+1, m-1\right] > \cdots$ for the remaining manipulators.

Then we have the level i scores, $i \in \left\{\frac{m-1}{2}, \frac{m-1}{2}+1\right\}$, that are shown in Table 3b for the manipulated election $(C, V \cup S)$, and we see that p is a level $\frac{m-1}{2}+1$ Bucklin winner in $(C, V \cup S)$.

From right to left: Assume that p can be made a Bucklin winner by setting the preferences of the k manipulators in S accordingly, and that without loss of generality, the manipulators each position p on top. Since the smallest level on which p receives any points from the

voters in V is $\frac{m-1}{2}+1$, p cannot win on any smaller level. Together with the fact that

$$score^{\frac{m-1}{2}}_{(C,V\cup S)}(p) = 2K \quad \text{and} \quad score^{\frac{m-1}{2}+1}_{(C,V\cup S)}(p) = mK + K,$$

p has to be a level $\frac{m-1}{2}+1$ Bucklin winner with the above score of $mK + K$.

Recalling Table 3a, we know that the other candidates, $c \in C - \{p\}$, have already mK points up to level $\frac{m-1}{2}+1$, even without the points from the manipulators. Since we have $m-1$ candidates that have to fill $\frac{m-1}{2}$ positions in the manipulators' votes, namely positions 2 through $\frac{m-1}{2}+1$, the overall sum of the points that all candidates $c \in C - \{p\}$ will gain up to level $\frac{m-1}{2}+1$ from the manipulators will be $(m-1)K$. Together with the restriction that none of the candidates is allowed to gain more than K points on the levels 1 to $\frac{m-1}{2}+1$ to ensure that p is still a Bucklin winner, we have that every candidate $c \in C - \{p\}$ is allowed to receive exactly K points from the manipulators. This implies that choosing any candidate in $C - \{p\}$, say c_1, and defining $A' \subseteq A$ by

$$A' = \left\{ i \;\middle|\; \text{manipulator with weight } a_i \text{ has } c_1 \text{ in one of the positions 2 through } \frac{m-1}{2} \right\},$$

it holds that $\sum_{i\in A'} a_i = K$.

Thus, $(A, (a_1, a_2, \ldots, a_k)) \in$ PARTITION.

For solving the unique-winner case, simply add two voters that have the following preferences and weights:

- $C\left[1, \frac{m-1}{2}\right] > p > \cdots$ with weight 1, and
- $C\left[\frac{m-1}{2}+1, m-1\right] > p > \cdots$ with weight 1.

The rest of the argument can be adapted in a straightforward manner.

For the case of an even number $m \geq 4$ of candidates, construct the following Bucklin election (C, V). The candidate set is $C = \{c_1, c_2, \ldots, c_{m-1}\} \cup \{p\}$. For $i, j \in \{1, 2, \ldots, m-1\}$, in addition to the notation $C[i, j]$ introduced above, for $i > j$, we write $C_p[i, j]$ to denote the preference "$c_i > c_{i+1} > \cdots > c_{m-1} > p > c_1 > \cdots > c_j$." (That is, the meaning of $C_p[i, j]$ is the same as the meaning of $C[i, j]$, except that we insert p after candidate c_{m-1}.) The voter list V, divided into three groups of voters, is defined by the preferences and weights shown in Table 4. Unspecified positions (denoted by "...") in the preferences) can be filled arbitrarily with the candidates not explicitly occurring in this preference. Furthermore, we have k manipulators in S with weights $a_1, a_2, \ldots, a_k$.

Let us now analyze the voter list V. First, we note that there are $\frac{m}{2}$ weight-K voters in Group 1, $\frac{m}{2}-1$ weight-K voters in Group 2, and three weight-K voters in Group 3. Thus, the total weight of the voters (not counting the manipulators) is $mK + 2K$. If we include the manipulators, the total weight goes up to $mK + 4K$. In effect, after including the manipulators, the majority theshold is $\frac{m}{2}K + 2K + 1$ (or $\frac{m}{2}K + K + 1$ not counting the manipulators).

Let us now compute the level-$\frac{m}{2}$ scores of the candidates in election (C, V) (these scores are also given in Table 5a; c_1 is the unique level $\frac{m}{2}$ Bucklin winner in (C, V)). We will consider particular groups of candidates:

- Candidate p:

  (1) Each voter in Group 1 ranks some $\frac{m}{2}$ candidates from $C - \{p\}$ first, so neither of the voters in this groups contributes toward the score of p at level $\frac{m}{2}$.

Table 4 Voter list V in the proof of Theorem 1 for an even number $m \geq 4$ of candidates

Voter group	Preference	Weight
Group 1	$C\left[1, \frac{m}{2}\right] > p > \cdots$	K
	$C\left[2, \frac{m}{2}+1\right] > p > \cdots$	K
	$C\left[3, \frac{m}{2}+2\right] > p > \cdots$	K
	$\vdots$	$\vdots$
	$C\left[\frac{m}{2}, m-1\right] > p > \cdots$	K
Group 2	$C_p[\frac{m}{2}+1, 1] > \cdots$	K
	$C_p[\frac{m}{2}+2, 2] > \cdots$	K
	$C_p[\frac{m}{2}+3, 3] > \cdots$	K
	$\vdots$	$\vdots$
	$C_p\left[m-1, \frac{m}{2}-1\right] > \cdots$	K
Group 3	$p > C\left[1, \frac{m}{2}\right] > \cdots$	K
	$c_1 > C\left[2, \frac{m}{2}\right] > p > \cdots$	K
	$c_1 > C\left[\frac{m}{2}+1, m-1\right] > p > \cdots$	K

(2) Each voter in Group 2 ranks p among the top $\frac{m}{2}$ candidates, so p's level-$\frac{m}{2}$ score from these voters is $\left(\frac{m}{2}-1\right)K$. (3) Only the first voter in Group 3 ranks p among the top $\frac{m}{2}$ positions, so p gets K level-$\frac{m}{2}$ points from Group 3.

– Candidate c_1:

(1) c_1 gets K level-$\frac{m}{2}$ points from the first voter in Group 1.
(2) c_1 gets K level-$\frac{m}{2}$ points from $\frac{m}{2}-2$ voters in Group 2 (all of them except for the first one) for a total of $\left(\frac{m}{2}-2\right)K$ level-$\frac{m}{2}$ points.
(3) c_1 gets K level-$\frac{m}{2}$ points from each of the voters in Group 3.

– Each candidate c_i, for $i \in \left\{2, \ldots, \frac{m}{2}-1\right\}$:

(1) c_i gets K level-$\frac{m}{2}$ points from each of the first i voters in Group 1, for a total of iK level-$\frac{m}{2}$ points.
(2) c_i gets K level-$\frac{m}{2}$ points from all but the first i voters in Group 2, for a total of $\left(\frac{m}{2}-1-i\right)K$ level-$\frac{m}{2}$ points.
(3) c_i gets K level-$\frac{m}{2}$ points from the first two voters in Group 3, for a total of $2K$ level-$\frac{m}{2}$ points.

– Each candidate $c_{\frac{m}{2}+i}$, for $i \in \left\{0, \ldots, \frac{m}{2}-1\right\}$:

(1) $c_{\frac{m}{2}+i}$ gets K level-$\frac{m}{2}$ points from each but the first i voters in Group 1, for a total of $(\frac{m}{2}-i)K$ level-$\frac{m}{2}$ points.
(2) $c_{\frac{m}{2}+i}$ gets K level-$\frac{m}{2}$ points from the first i voters in Group 2, for the total of iK level-$\frac{m}{2}$ points.
(3) $c_{\frac{m}{2}+i}$ gets K level-$\frac{m}{2}$ points from exactly one voter in Group 3 (for $i=0$ these points come from the second voter,[6] and for the other values of i, these points come from the third voter).

[6] Note that the first voter in Group 3 ranks $c_{\frac{m}{2}}$ on position $\frac{m}{2}+1$.

Table 5 Level-i scores, $i \in \left\{\frac{m}{2}, \frac{m}{2}+1\right\}$, of the candidates in C for even $m \geq 4$

	c_1	$c \in C - \{p, c_1\}$	p
(a) Original election (C, V)			
$score^{\frac{m}{2}}$	$\frac{m}{2}K + 2K$	$\frac{m}{2}K + K$	$\frac{m}{2}K$
(b) Manipulated election $(C, V \cup S)$			
$score^{\frac{m}{2}}$	$\frac{m}{2}K + 2K$	$\frac{m}{2}K + 2K$	$\frac{m}{2}K + 2K$
$score^{\frac{m}{2}+1}$	$< mK + 4K$	$< mK + 4K$	$mK + 4K$

We claim that $(A, (a_1, a_2, \ldots, a_k)) \in$ PARTITION if and only if p can be made a Bucklin winner in $(C, V \cup S)$.

From left to right: Assume there is a subset $A' \subseteq A$ such that $\sum_{i \in A'} a_i = K$. The preferences of the manipulators can then be set in the following way:

- $p > C\left[2, \frac{m}{2}\right] > \cdots$ for all manipulators with weight a_i for $i \in A'$, and
- $p > C\left[\frac{m}{2}+1, m-1\right] > \cdots$ for the remaining manipulators.

Then we have the level i scores, $i \in \left\{\frac{m}{2}, \frac{m}{2}+1\right\}$, that are shown in Table 5b. Hence, p is the unique winner (at level $\frac{m}{2}+1$) and, thus, a Bucklin winner in $(C, V \cup S)$.

From right to left: Assume that p can be made a Bucklin winner by setting the manipulators' votes accordingly. Without loss of generality, we assume that all manipulators position p on top. So we have that candidate p can reach exactly

- $\frac{m}{2}K + 2K$ points on level $\frac{m}{2}$ and
- $mK + 4K$ points on level $\frac{m}{2}+1$.

For p to be a Bucklin winner, she has to be a level $\frac{m}{2}+1$ Bucklin winner with maximum score. Thus, for all candidates $c \in C - \{p\}$, it has to hold that

$$score^{\frac{m}{2}}_{(C,V\cup S)}(c) \leq \frac{m}{2}K + 2K$$

to ensure that none of these candidates reaches a strict majority on a smaller level than p does. This directly implies that c_1 cannot be in the top $\frac{m}{2}$ positions of any manipulator. For the remaining candidates $c \in C - \{c_1, p\}$, it holds that

$$score^{\frac{m}{2}}_{(C,V)}(c) = \frac{m}{2}K + K,$$

so each of them is only allowed to gain at most K points on the first $\frac{m}{2}$ levels. But due to the restriction of c_1's position, the 2nd through $\frac{m}{2}$th positions in all manipulators' votes have to be filled with candidates from $C - \{c_1, p\}$. It follows that the sum of their scores will increase by $(m-2)K$ points, while $score^{\frac{m}{2}}_{(C,S)}(c) \leq K$ still has to hold. This is possible only if $score^{\frac{m}{2}}_{(C,S)}(c) = K$ for all $c \in C - \{c_1, p\}$. Thus, choosing any candidate in $C - \{c_1, p\}$, say c_2, there has to be a set $A' \subseteq A$ with

$$A' = \left\{ i \;\middle|\; \text{manipulator with weight } a_i \text{ has } c_2 \text{ in one of the positions 2 through } \frac{m}{2} \right\},$$

such that $\sum_{i \in A'} a_i = K$. Thus, $(A, (a_1, a_2, \ldots, a_k)) \in$ PARTITION.

Since p is the unique winner in the manipulated election in the proof "from left to right," and since the proof "from right to left" only involves arguments about the level on which other candidates than p reach the majority threshold, this construction solves the unique-winner case without further adaptions. □

Algorithm 1: Algorithm for Bucklin-DCWM

input : C set of candidates
V list of voters
W_V weights of the voters
W_S weights of the manipulators
p designated candidate
output: "YES" if $(C, V, W_V, W_S, p) \in$ Bucklin-DCWM
"NO" if $(C, V, W_V, W_S, p) \notin$ Bucklin-DCWM

```
1  if ∑_{w∈W_S} w > ∑_{w∈W_V} w then
2  |  return "YES";
3  foreach c ∈ C − {p} do
4  |  put p in the last position in the manipulators' votes;
5  |  put c in the first position in the manipulators' votes;
6  |  fill the remaining positions in the manipulators' votes arbitrarily;
7  |  let S be the list of the manipulators' votes;
8  |  if (p is not a Bucklin winner of (C, V ∪ S) with weights W_V ∪ W_S) then
9  |  |  return "YES";
10 |
11 return "NO";
```

We now turn to the destructive variant of coalitional weighted manipulation and give a deterministic polynomial-time algorithm for this problem in Bucklin voting (Algorithm 1). Intuitively, Algorithm 1 proceeds as follows: It tries all the candidates $c \in C - \{p\}$ and for each of them checks if having each of the manipulators cast an identical vote of the form $c > \cdots > p$ (where candidates in $C - \{p, c\}$ are ranked arbitrarily) ensures that p does not win. If it succeeds for even one c, it accepts. Otherwise, it rejects. (In other words, the problem of destructive coalitional weighted manipulation under Bucklin disjunctively truth-table reduces to testing if it is possible to choose the manipulators' votes so that a given candidate c obtains a better result in the election than p.) Before formally proving the runtime and correctness of this algorithm, we state the following useful lemma, which is easily seen to hold.

Lemma 1 *Let (C, V) be a weighted Bucklin election with total weight W and let $c, p \in C$. Then the following holds.*

1. *Assume that c is not a (unique) Bucklin winner in (C, V) and that the votes in V are changed such that the position of c is made worse in some votes, all else being equal.*[7] *Then c is still not a (unique) Bucklin winner.*
2. *Assume that c is a (unique) Bucklin winner of the election and that the votes in V are changed such that the position of c is improved in some votes, all else being equal. Then c remains a (unique) Bucklin winner.*
3. *Assume that c is a (unique) Bucklin winner of the election and that p is not a (unique) Bucklin winner. If in some votes the positions of candidates are swapped without changing the positions of c and p, all else being equal, then p is still not a (unique) Bucklin winner.*

We now analyze Algorithm 1 for Bucklin-DCWM.

[7] By "all else being equal" we tacitly mean that all other candidates remain in the same position in each vote, except those candidates that improve their position by one due to shifting c toward the bottom. An analogous comment applies to the cases where c's position is improved in the second statement of this lemma and where other candidates are swapped in the third statement of this lemma.

Theorem 2 *In both winner models, Bucklin*-DCWM *can be decided in time* $\mathcal{O}(m^2(n + \|W_S\|))$*, where* W_S *is the list of the manipulators' weights.*

Proof We begin with analyzing the runtime of Algorithm 1. Obviously, the algorithm always terminates and the input size is in $\mathcal{O}(\underbrace{m}_{\|C\|} + \underbrace{nm}_{\|V\|} + \underbrace{n}_{\|W_V\|} + \|W_S\| + \underbrace{1}_{\|\{p\}\|}) = \mathcal{O}(nm + \|W_S\|)$.

The most costly part of the algorithm is the for-loop. To construct the manipulators' votes, $\mathcal{O}(\|W_S\|m)$ steps are needed. The winner-determination procedure for Bucklin can be implemented with a runtime of $\mathcal{O}(nm)$, so the if-statement in line 8 can be computed in time $\mathcal{O}(m(n + \|W_S\|))$. Thus, the whole for-loop runs in time $\mathcal{O}(m^2(n + \|W_S\|))$.

To prove the correctness of the algorithm, we show that it gives the output "YES" if and only if $(C, V, W_V, W_S, p) \in$ Bucklin-DCWM. (Note that by changing the condition of the if-statement in line 8 to "(p is not a unique Bucklin winner of $(C, V \cup S)$ with weights $W_V \cup W_S$)," the algorithm solves Bucklin-DCWM in the unique-winner model which can be shown with an analogous argumentation as below.)

From left to right: If the algorithm outputs "YES" in line 2, then we have $\sum_{w \in W_S} w > \sum_{w \in W_V} w$, i.e., the sum of the manipulators' weights is greater than the sum of the weights of the nonmanipulative voters. In this case, any of the candidates $c \neq p$ can be made a unique level 1 Bucklin winner in $(C, V \cup S)$ by putting c in the first position of all the manipulators' votes and filling the remaining positions arbitrarily. Hence, $(C, V, W_V, W_S, p) \in$ Bucklin-DCWM. If the algorithm outputs "YES" in line 9, the manipulators' votes have been constructed such that p is not a Bucklin winner in $(C, V \cup S)$. Thus, we have that (C, V, W_V, W_S, p) is a yes-instance of Bucklin-DCWM.

From right to left: Assume that $(C, V, W_V, W_S, p) \in$ Bucklin-DCWM. If $\sum_{w \in W_S} w > \sum_{w \in W_V} w$, then the algorithm correctly outputs "YES." Otherwise, the following holds: Since the given instance is a yes-instance of Bucklin-DCWM, the votes of the manipulators in S can be set such that p is not a Bucklin winner of the election $(C, V \cup S)$. We know from Lemma 1 that successively swapping p with her neighbor until p is in the last position in all votes in S does not change the fact that p is not a Bucklin winner in $(C, V \cup S')$ (where S' are the new manipulative votes with p in the last position). Assume that $c \in C - \{p\}$ is a Bucklin winner in $(C, V \cup S)$. Then swap her position successively with her neighbor in the votes in S' until c is in the first position of all manipulative votes. Let S'' denote the accordingly changed list of manipulative votes. Again, from Lemma 1 we know that c still wins in $(C, V \cup S'')$. Let S''' be the list of manipulative votes that the algorithm constructs. We can transform S'' into S''' by swapping the corresponding candidates $c', c'' \in C - \{c, p\}$ accordingly. Since the positions of c and p remain unchanged, we have with Lemma 1 that p is still not a Bucklin winner in $(C, V \cup S''')$. Thus, the algorithm outputs "YES" in line 9.□

3.3 Results for unweighted manipulation

The unweighted manipulation cases in fallback elections can all be solved in deterministic polynomial time. This follows, together with Proposition 1, directly from the results presented for the weighted cases in Propostition 2. We state this in the following proposition.

Proposition 3 *Fallback*-CCUM *and fallback*-DCUM *are in P, each in both winner models.*

In Bucklin elections, however, the argumentation is more involved, since the manipulators do not have the possibility to preclude any candidate from gaining points from their votes. So the manipulators' votes have to be carefully constructed to ensure that no other candidate than the designated candidate gains too many points on the relevant levels.

Nevertheless, we can show that Bucklin-CCUM is in P by adapting an algorithm for *simplified-Bucklin*-CCUM (recall Footnote 4 for the definition of simplified Bucklin voting) that is due to Xia et al. [51]. We will do so by giving a high-level description of the algorithm and the relevant information for understanding the pseudocode shown in Algorithm 2. The correctness of the algorithm and the run-time analysis will then be shown in the proof of Lemma 2 and Theorem 3.

Algorithm 2: Algorithm for Bucklin-CCUM

input : C set of candidates
V list of voters
k number of manipulators
p designated candidate
output: "YES" if $(C, V, k, p) \in$ Bucklin-CCUM
"NO" if $(C, V, k, p) \notin$ Bucklin-CCUM

if $k > \|V\|$ **then**
  return "YES";
let $max\_scr^{\ell_{\min}}$, $max\_scr^{\ell_{\min}-1}$, $num^{\ell_{\min}}$, $num^{\ell_{\min}-1}$ be arrays of length m;
$maj = \lfloor \frac{\|V\|+k}{2} \rfloor + 1$;
$\ell_{\min} = \min\{i \mid score^{i}_{(C,V)}(p) + k \geq maj\}$;
foreach $c \in C - \{p\}$ **do**
  if $\min\{i \mid score^{i}_{(C,V)}(c) \geq maj\} < \ell_{\min}$ OR $score^{\ell_{\min}}_{(C,V)}(c) > score^{\ell_{\min}}_{(C,V)}(p) + k$ **then**
    return "NO";
  $max\_scr^{\ell_{\min}}[c] = score^{\ell_{\min}}_{(C,V)}(p) + k - score^{\ell_{\min}}_{(C,V)}(c)$;
  $max\_scr^{\ell_{\min}-1}[c] = maj - score^{\ell_{\min}-1}_{(C,V)}(c) - 1$;
  $num^{\ell_{\min}}[c] = \min\{max\_scr^{\ell_{\min}}[c], k\}$;
  $num^{\ell_{\min}-1}[c] = \min\{max\_scr^{\ell_{\min}-1}[c], max\_scr^{\ell_{\min}}[c], k\}$;
if $\sum_{c\in C-\{p\}} \min\{max\_scr^{\ell_{\min}-1}[c], max\_scr^{\ell_{\min}}[c], k\} < (\ell_{\min} - 2)k$ OR $\sum_{c\in C-\{p\}} \min\{max\_scr^{\ell_{\min}}[c], k\} < (\ell_{\min} - 1)k$ **then**
  return "NO";

return "YES";

The given input consists of a Bucklin election (C, V) with the set of candidates C and the list of voters V with specified preferences. Candidate $p \in C$ is the candidate we want to make a winner of the resulting election by determining the yet unspecified preferences of k manipulators.

To decide whether the given election can be manipulated successfully, that is, whether p can indeed be made a winner by setting the preferences of the k manipulators, the following variables and arrays of length m will be introduced.

maj: Denotes the strict majority threshold in the final election counting both the number of regular voters and the k manipulators.

$\ell_{\min}$: Denotes the smallest level on which candidate p reaches the majority threshold maj in the manipulated election, assuming that all the manipulators position p on top. This means that if p is to win, p has to win at level $\ell_{\min}$, having $score^{\ell_{\min}}_{(C,V)}(p) + k$ points.

$max\_scr^{\ell_{\min}}$: This array indicates how many further points each candidate c can gain without having strictly more points than p on level $\ell_{\min}$.

$max\_scr^{\ell_{\min}-1}$: This array indicates how many further points each candidate c may gain without reaching or exceeding the majority threshold maj on one of the levels 1 through $\ell_{\min} - 1$.

$num^{\ell_{\min}-1}$: This array indicates the number of manipulators that may have candidate c in the first $\ell_{\min} - 1$ positions of their votes without preventing p from winning, that is, $num^{\ell_{\min}-1}[c] = \min\{max\_scr^{\ell_{\min}}[c], max\_scr^{\ell_{\min}-1}[c], k\}$.

$num^{\ell_{\min}}$: This array indicates the number of manipulators that can place c among their top $\ell_{\min}$ positions without preventing p from winning, that is, $num^{\ell_{\min}}[c] = \min\{max\_scr^{\ell_{\min}}[c], k\}$.

We have that for all $c \in C - \{p\}$, $max\_scr^{\ell_{\min}}$ and $max\_scr^{\ell_{\min}-1}$ contain positive numbers and that $num^{\ell_{\min}}[c] \geq num^{\ell_{\min}-1}[c]$.

The algorithm proceeds as follows. In a first step (in line 1) it is tested whether there are more manipulators than nonmanipulative voters which would lead to a trivial yes-instance. If this test fails, the algorithm proceeds and tests whether the given instance is a trivial no-instance in the sense that there is at least one other candidate that cannot be dethroned by p with the given number k of manipulators (in line 7; we will go into further detail in the proof of Lemma 2). If no such candidate is found, the algorithm computes the neccessary arrays described above and proceeds to line 13. In this final step, assuming that p is in the first position in every manipulator's preference, the algorithm checks whether the remaining positions in the preferences can be filled while still ensuring that no candidate $c \in C - \{p\}$ beats p.

The following lemma gives the detailed proof of correctness of Algorithm 2.

Lemma 2 *Considering the notation C, V, k, p, $max\_scr^{\ell_{\min}}$, $max\_scr^{\ell_{\min}-1}$, $num^{\ell_{\min}}$, $num^{\ell_{\min}-1}$, $\ell_{\min}$, and maj as in Algorithm 2, it holds that:*

1. *If $k > \|V\|$ then $(C, V, k, p) \in$ Bucklin-CCUM.*
2. *If there is a candidate $c \in C - \{p\}$ with*
   (a) $\min\{i \mid score^{i}_{(C,V)}(c) \geq maj\} < \ell_{\min}$ *or*
   (b) $score^{\ell_{\min}}_{(C,V)}(c) > score^{\ell_{\min}}_{(C,V)}(p) + k$, *then* $(C, V, k, p) \notin$ *Bucklin*-CCUM.
3. *If neither of the above two conditions is met, then $(C, V, k, p) \notin$ Bucklin-CCUM if and only if*
   (a) $\sum_{c \in C-\{p\}} \min\{max\_scr^{\ell_{\min}}[c], max\_scr^{\ell_{\min}-1}[c], k\} < (\ell_{\min} - 2)k$ *or*
   (b) $\sum_{c \in C-\{p\}} \min\{max\_scr^{\ell_{\min}}[c], k\} < (\ell_{\min} - 1)k$.

Proof 1. If the number of manipulators is greater than the number of nonmanipulative voters, a successful manipulation is always possible. The manipulators simply position p on top in their votes, so p reaches the majority threshold already on the first level. Then, $(C, V, k, p) \in$ Bucklin-CCUM trivially holds.

2. Let $c \in C - \{p\}$ be an arbitrary candidate.
   (a) It holds that $\min\left\{i \;\middle|\; score^{i}_{(C,V)}(c) \geq maj\right\} < \ell_{\min}$: That means that we have a candidate c that reaches maj votes on an earlier level than p, and c does so even without the manipulators' votes. Thus $(C, V, k, p) \notin$ Bucklin-CCUM.
   (b) It holds that $score^{\ell_{\min}}_{(C,V)}(c) > score^{\ell_{\min}}_{(C,V)}(p) + k$: This means that, on exactly the level where p would have to win the manipulated election, c gets at least one point more from the nonmanipulative voters than p gains in the election where the manipulators'

votes have already been added. That means that p cannot be made a Bucklin winner of the manipulated election and thus $(C, V, k, p) \notin$ Bucklin-CCUM.

3. We assume that neither of the above conditions holds.
   From right to left:

   (a) Suppose that $\sum_{c\in C-\{p\}} \min\{max\_scr^{\ell_{\min}}[c], max\_scr^{\ell_{\min}-1}[c], k\} < (\ell_{\min}-2)k$. In this case, it is not possible to fill the remaining $(\ell_{\min}-2)k$ positions (positions 2 through $\ell_{\min}-1$) in the manipulators' votes without having for at least one candidate $d \in C-\{p\}$ that

   $$\text{either } max\_scr^{\ell_{\min}-1}[d] - score^{\ell_{\min}-1}_{(C,S)}(d) < 0$$
   $$\text{or } max\_scr^{\ell_{\min}}[d] - score^{\ell_{\min}}_{(C,S)}(d) < 0$$

   holds. That is equivalent to

   $$\text{either } maj - score^{\ell_{\min}-1}_{(C,V)}(d) - 1 - score^{\ell_{\min}-1}_{(C,S)}(d) < 0$$
   $$\text{or } score^{\ell_{\min}}_{(C,V)}(p) + k - score^{\ell_{\min}}_{(C,V)}(d) - score^{\ell_{\min}}_{(C,S)}(d) < 0,$$

   which in turn is equivalent to

   $$\text{either } score^{\ell_{\min}-1}_{(C,V\cup S)}(d) = score^{\ell_{\min}-1}_{(C,V)}(d) + score^{\ell_{\min}-1}_{(C,S)}(d) > maj - 1$$
   $$\text{or } score^{\ell_{\min}}_{(C,V\cup S)}(d) = score^{\ell_{\min}}_{(C,V)}(d) + score^{\ell_{\min}}_{(C,S)}(d) > score^{\ell_{\min}}_{(C,V)}(p) + k.$$

   So we have that either d is a Bucklin winner in the manipulated election on a smaller level than $\ell_{\min}$, or it holds that on level $\ell_{\min}$ candidate d has at least one point more than p. Thus $(C, V, k, p) \notin$ Bucklin-CCUM.

   (b) Suppose that $\sum_{c\in C-\{p\}} \min\{max\_scr^{\ell_{\min}}[c], k\} < (\ell_{\min}-1)k$. In this case, it is not possible to fill the remaining $(\ell_{\min}-1)k$ positions (positions 2 through $\ell_{\min}$) in the manipulators' votes without having for at least one candidate $d \in C-\{p\}$ that:

   $$max\_scr^{\ell_{\min}}[d] - score^{\ell_{\min}}_{(C,S)}(d) < 0$$
   $$\Leftrightarrow score^{\ell_{\min}}_{(C,V)}(p) + k - score^{\ell_{\min}}_{(C,V)}(d) - 1 - score^{\ell_{\min}}_{(C,S)}(d) < 0$$
   $$\Leftrightarrow score^{\ell_{\min}}_{(C,V\cup S)}(d) = score^{\ell_{\min}}_{(C,V)}(d) + score^{\ell_{\min}}_{(C,S)}(d) > score^{\ell_{\min}}_{(C,V)}(p) + k.$$

   So we have that d has at least one point more than p on level $\ell_{\min}$. So $(C, V, k, p) \notin$ Bucklin-CCUM.

From left to right: We show the contrapositive. Assume that both

(a) $\sum_{c\in C-\{p\}} \min\{max\_scr^{\ell_{\min}}[c], max\_scr^{\ell_{\min}-1}[c], k\} \geq (\ell_{\min}-2)k$ and

(b) $\sum_{c\in C-\{p\}} \min\{max\_scr^{\ell_{\min}}[c], k\} \geq (\ell_{\min}-1)k$

hold. Then we can fill positions 2 through $\ell_{\min}$ of the manipulators' votes such that for each candidate $e \in C-\{p\}$, the following holds:

$$max\_scr^{\ell_{\min}-1}[e] - score^{\ell_{\min}-1}_{(C,S)}(e) \geq 0 \text{ and}$$
$$max\_scr^{\ell_{\min}}[e] - score^{\ell_{\min}}_{(C,S)}(e) \geq 0,$$

which is equivalent to

$$score^{\ell_{\min}-1}_{(C,V\cup S)}(e) = score^{\ell_{\min}-1}_{(C,V)}(e) + score^{\ell_{\min}-1}_{(C,S)}(e) \leq maj - 1 \text{ and}$$
$$score^{\ell_{\min}}_{(C,V\cup S)}(e) = score^{\ell_{\min}}_{(C,V)}(e) + score^{\ell_{\min}}_{(C,S)}(e) \leq score^{\ell_{\min}}_{(C,V)}(p) + k.$$

So we have that $(C, V, k, p) \in$ Bucklin-CCUM.
This completes the proof. □

Now we are ready to show that Algorithm 2 runs in polynomial time and correctly solves Bucklin-CCUM.

Theorem 3 *In both winner models, Bucklin-*CCUM *can be decided in time* $\mathcal{O}(m^2 + nm)$.

Proof It follows immediately from Lemma 2 that Algorithm 2 is correct. It is also clear that it always terminates. To compute the needed scores $score^i_{(C,V)}(c)$ for all candidates c and every level i, $\mathcal{O}(m^2 + nm)$ steps are needed. The for-loop in line 6 needs $\mathcal{O}(m)$ steps. So the algorithm has a runtime of $\mathcal{O}(m^2 + nm)$ in total.

Note that the algorithm can easily be adapted to solve Bucklin-CCUM also in the unique-winner model. □

The algorithm can easily be adapted to solve the unique-winner case by slightly modifying the definition of the array $max\_scr^{\ell_{\min}}$ (subtracting 1) and allowing equality in the second inequality in line 7. The argumentation for the runtime analysis and the correctness can then be adapted straightforwardly.

With Theorem 3 and Proposition 1, we have the following corollary.

Corollary 1 *Bucklin-*CCUM *and Bucklin-*DCUM *are in P, each in both winner models.*

4 Bribery in Bucklin and fallback voting

4.1 Definitions and overview of results

We begin with defining the standard bribery scenarios proposed by Faliszewski et al. [27] (see also [29]) that will be applied here to fallback and Bucklin elections. Let $\mathcal{E}$ be a given election system.

$\mathcal{E}$-Constructive Unweighted Bribery ($\mathcal{E}$-CUB)	
Given	An $\mathcal{E}$ election (C, V), a designated candidate p, and a nonnegative integer k.
Question	Is it possible to make p an $\mathcal{E}$ winner by changing the votes of at most k voters?

This basic bribery scenario can be extended by either considering voters with different weights, or allowing that each voter has a different price for changing her vote, or both. These three scenarios are formally defined by the following problems:

$\mathcal{E}$-Constructive Weighted Bribery ($\mathcal{E}$-CWB)	
Given	An $\mathcal{E}$ election (C, V) with each voter $v_i \in V$ having a nonnegative integer weight w_i, a designated candidate p, and a positive integer k.
Question	Is it possible to make p an $\mathcal{E}$ winner by changing the votes of at most k voters?

$\mathscr{E}$-Constructive Unweighted Priced Bribery ($\mathscr{E}$-CUB-$)	
Given	An $\mathscr{E}$ election (C, V) with each voter $v_i \in V$ having a nonnegative integer price π_i, $1 \leq i \leq n$, a designated candidate p, and a positive integer k.
Question	Is there a set $B \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in B} \pi_i \leq k$ and the voters v_i with $i \in B$ can be bribed so that p is an $\mathscr{E}$ winner of the resulting election?

$\mathscr{E}$-Constructive Weighted Priced Bribery ($\mathscr{E}$-CWB-$)	
Given	An $\mathscr{E}$ election (C, V) with each voter $v_i \in V$ having nonnegative integer weight w_i and price π_i, $1 \leq i \leq n$, a designated candidate p, and a positive integer k.
Question	Is there a set $B \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in B} \pi_i \leq k$ and the voters v_i with $i \in B$ can be bribed so that p is an $\mathscr{E}$ winner of the resulting election?

Table 6 Overview of results for bribery in Bucklin and fallback voting

Problem	Bucklin voting		Fallback voting	
	Complexity	Reference	Complexity	Reference
$\mathscr{E}$-CUB	NP-complete	Theorem 4	NP-complete	Theorem 5
$\mathscr{E}$-DUB	P	Corollary 4	P	Theorem 7
$\mathscr{E}$-CUB-$	NP-complete	Corollary 2	NP-complete	Corollary 3
$\mathscr{E}$-DUB-$	P	Theorem 6	P	Theorem 7
$\mathscr{E}$-CWB	NP-complete	Corollary 2	NP-complete	Corollary 3
$\mathscr{E}$-DWB	P	Theorem 6	P	Theorem 7
$\mathscr{E}$-CWB-$	NP-complete	Corollary 2	NP-complete	Corollary 3
$\mathscr{E}$-DWB-$	NP-complete	Theorem 8	NP-complete	Theorem 8

By changing the question in the above four problems to whether p can be prevented from being a winner of the election by bribing some of the voters, we obtain the destructive variants of these bribery scenarios, and we denote the corresponding problems by $\mathscr{E}$-DUB, $\mathscr{E}$-DWB, $\mathscr{E}$-DUB-$, and $\mathscr{E}$-DWB-$.

As for the manipulation scenarios, we state the problem in the nonunique-winner model but all our results shown in Table 6 hold in both models. The proofs in the following sections will be presented in detail for the nonunique-winner model while we will only briefly describe the needed adaptions that have to be made for the argumentation to also apply to the unique-winner model.

4.2 Results for bribery

We start with the constructive cases of the standard bribery scenarios.

Theorem 4 CUB *is NP-complete for Bucklin voting in both winner models.*

Proof The following proof applies to both winner models with exactly the same argumentation, no adaptions are needed. Membership of Bucklin-CUB in NP is obvious. We show NP-hardness by a reduction from X3C.

Let $(B, \mathscr{S})$ be an instance of X3C with $B = \{b_1, b_2, \ldots, b_{3m}\}$ and $\mathscr{S} = \{S_1, S_2, \ldots, S_n\}$. Without loss of generality, we may assume that $n \geq 2m$. We construct a Bucklin-CUB

Table 7 Construction of Bucklin election (C, V) in the proof of Theorem 4

Position 1		Position 2		Position 3		...	Position $3m+1$		Position $3m+2$	
#	Cand.	#	Cand.	#	Cand.	...	#	Cand.	#	Cand.
m	c	$n+1-\ell_1$	b_1	$n+1-\ell_2$	b_2	...	$n+1-\ell_{3m}$	b_{3m}	$n-m+1$	p
m	d	ℓ_1-1	$g \in G'_2$	ℓ_2-1	$g \in G'_3$	...	$\ell_{3m}-1$	$g \in G'_{3m+1}$	$m-1$	$g \in G'_{3m+2}$
$n-2m$	$g \in G'_1$					...				

instance $((C, V), p, k)$, where (C, V) is a Bucklin election with the candidates $C = B \cup \{c, d\} \cup G \cup \{p\}$, p is the designated candidate, and $k = m$.

The set G is a set of "padding candidates," which are used to ensure that certain candidates do not gain points up to a certain level. Padding candidates are positioned in the votes such that, up to a certain level, they themselves do not gain enough points to be relevant for the central argument of the proof. (Specifically, we will ensure that up to a given level, each padding candidate gets at most one point.) Thus, their scores are not listed in the tables giving the scores of the relevant candidates.

For every $b_j \in B$, define ℓ_j to be the number of sets $S_i \in \mathscr{S}$ candidate b_j is contained in. V consists of the following $2n$ voters (i.e., a strict majority is reached with $n+1$ votes):

- The first voter group consists of n voters. For each i, $1 \leq i \leq n$, we have one voter of the form

$$c > d > S_i > G_i > \{C - (\{c, d\} \cup S_i \cup G_i)\},$$

  where $G_i \subseteq G$ is a set of $3m-3$ padding candidates. When a set X of candidates is given in such a ranking, the order of the candidates from X can be fixed arbitrarily in this ranking.
- The second voter group consists of n voters as well. We will present the preferences level by level from the first to the $(3m+2)$nd position in Table 7, indicating the number (by #) of occurrences of each candidate in these positions. Thus, the first column can be read as follows: m of the n voters position c on the first place, m of the n voters have candidate d on the first position while the remaining $n-2m$ voters each position a different candidate from G on their top position. The second column indicates that $n+1-\ell_1$ of the n voters position candidate b_1 on the second place and the remaining ℓ_1-1 voters in this group each have a different candidate from G on the second position in their ballot. The remaining columns can be read analogously. Note that the notation for the padding candidates has been chosen to keep the table as readable as possible while still emphasizing that each candidate from G is only positioned once within the top $3m+2$ positions in this voter group and thus only gains at most 1 point up to level $3m+2$. The G'_r-sets are disjoint subsets of G each containing exactly as many candidates as voters are denoted by # in the respective column.

Table 8a shows the scores of the relevant candidates in (C, V) (namely, c, d, p, and each $b_j \in B$) for the relevant levels (namely, 1, 2, $3m$, $3m+1$, and $3m+2$). In particular, one can see that c is the unique level 1 Bucklin winner in (C, V).

We claim that $\mathscr{S}$ has an exact cover $\mathscr{S}'$ for B if and only if p can be made a Bucklin winner by changing at most m votes in V.

From left to right: Let $\mathscr{S}'$ be an exact cover for B and let $I \subseteq \{1, \ldots, n\}$ be the set of indices of the m elements in $\mathscr{S}'$. To make p a Bucklin winner, we only have to change votes in the first voter group: For each $i \in I$, change the corresponding vote

Table 8 Level-i scores for $i \in \{1, 2, 3m, 3m+1, 3m+2\}$ and the candidates in $C - G$

	$b_j \in B$	c	d	p
(a) Original election (C, V)				
$score^1$	0	$n+m$	m	0
$score^2$	$\leq n+1$	$n+m$	$m+n$	0
$score^{3m}$	$\leq n+1$	$n+m$	$m+n$	0
$score^{3m+1}$	$\leq n+1$	$n+m$	$m+n$	0
$score^{3m+2}$	$n+1$	$n+m$	$m+n$	$n-m+1$
(b) Modified election (C, V')				
$score^1$	0	n	m	m
$score^2$	$\leq n$	n	n	m
$score^{3m}$	$\leq n$	n	n	m
$score^{3m+1}$	$\leq n$	n	n	m
$score^{3m+2}$	$\leq n$	n	n	$n+1$

$$c > d > S_i > G_i > \{C - (\{c, d\} \cup S_i \cup G_i)\}$$

to

$$p > G_i > g'_1 > g'_2 > g'_3 > g'_4 > \{C - (\{g'_1, g'_2, g'_3, g'_4, p\} \cup G_i)\},$$

where each g'_j, $1 \leq j \leq 4$, is from G but not in G_i.

With these new votes, c and d both lose m points on the first two levels from the first voter group and p gains m points on the first level. Every candidate $b_j \in B$ loses exactly one point on one of the levels 3, 4, or 5. The scores in the resulting election (C, V') are shown in Table 8b. As one can see, p is the first candidate to reach a strict majority of $n+1$ votes (namely, on level $3m+2$) and is a level $3m+2$ Bucklin winner in the new election.

From right to left: Assume that p is a Bucklin winner of the election (C, V'), where V' is the new voter list containing the m changed votes. Since only m votes can be changed and p did not score any points prior to level $3m+2$ in the original election, p has to be a level $3m+2$ Bucklin winner in (C, V'). Candidates c and d originally reach the majority threshold already on, respectively, the first and the second level, so all votes that can be changed must place c and d on the first two positions. The only votes doing so are those in the first voter group. Finally, to prevent the candidates in B from reaching a strict majority on a level prior to level $3m+2$, each of the $3m$ candidates has to lose at least one point by changing at most m votes. This, again, can only be done by changing votes from the first voter group and, hence, there has to be an exact cover $\mathscr{S}'$ for B, corresponding to the voters from the first voter group that have to be changed. □

The next corollary follows immediately from Theorem 4.

Corollary 2 *In Bucklin elections,* CWB*,* CUB-$*, and* CWB-$ *are NP-complete, each in both winner models.*

Based on the corresponding proof for approval voting that is due to Faliszewski et al. [27], we can show NP-completeness for unweighted bribery in fallback elections as well.

Table 9 Construction of fallback election (C, V) in the proof of Theorem 5

#	For each ...	Number of votes	Ranking of candidates
1	$i \in \{1, \ldots, n\}$	1	$S_i \mid (B - S_i) \cup E \cup \{p\}$
2	$i \in \{1, \ldots, n\}$	1	$B_i \mid (B - B_i) \cup E \cup \{p\}$
3		$n - m - 1$	$p \mid B \cup E$
4	$\ell \in \{1, \ldots, n + m\}$	1	$e_\ell \mid B \cup (E - \{e_\ell\} \cup \{p\}$

Theorem 5 CUB *is NP-complete for fallback voting in both winner models.*

Proof Fallback-CUB obviously is in NP. To show NP-hardness, we give a reduction from X3C. Let $(B, \mathscr{S})$ be an instance of X3C with $B = \{b_1, b_2, \ldots, b_{3m}\}$ and $\mathscr{S} = \{S_1, S_2, \ldots, S_n\}$. (We assume that $n > m$; otherwise, it would be immediate to check if $(B, \mathscr{S})$ is a yes-instance of X3C or not.) We define the fallback election (C, V) with the candidate set $C = B \cup E \cup \{p\}$, where p is the designated candidate and E is a set of $n + m$ padding candidates. For every $j \in \{1, \ldots, 3m\}$, we again define ℓ_j as the number of subsets $S_i \in \mathscr{S}$ candidate $b_j \in B$ is contained in. Using this notation, we define the subsets $B_i = \{b_j \in B \mid i \leq n - \ell_j\}$ for $i \in \{1, \ldots, n\}$. V consists of the $4n - 1$ voters whose preferences are given in Table 9.

In this election, we have that $score(p) = n - m - 1$, $score(b_j) = n$ for all $j \in \{1, \ldots, 3m\}$, and $score(e_\ell) = 1$ for all $e_\ell \in E$ and $\ell \in \{1, \ldots, n + m\}$. Since no candidate reaches a strict majority (at least $2n$ points), all candidates $b_j \in B$ are fallback winners of this election.

We claim that $\mathscr{S}$ has an exact cover $\mathscr{S}'$ for B if and only if p can be made a fallback winner by bribing at most m voters.

From left to right: Suppose that $\mathscr{S}$ has an exact cover $\mathscr{S}'$ for B. We change the votes of those voters in the first voter group where $S_i \in \mathscr{S}'$ from $S_i \mid (B - S_i) \cup E \cup \{p\}$ to $p \mid B \cup E$. In the resulting election (C, V'), only the scores of the candidates in B and the score of p change: p gains m points, whereas each $b_j \in B$ loses exactly one point. Thus, with an overall score of $n - 1$, candidate p wins the election together with the candidates in B.

From right to left: Suppose that p can be made a fallback winner by changing at most m votes in V. That means that p can gain at most m points, so the maximum overall score that p can reach is $n - 1$. Since each $b_j \in B$ has an overall score of n, every candidate in B has to lose at least one point by changing at most m votes (otherwise, there would be at least one candidate in B who beats p). This is possible only if in m votes of the first voter group the candidates in S_i are removed from the approval strategy such that these m sets S_i form an exact cover for B.

For the unique-winner model, simply change the third voter group in V to contain $n - m$ voters. □

This result immediately implies NP-hardness for the remaining constructive bribery scenarios in fallback elections as well.

Corollary 3 *In fallback elections,* CWB, CUB-$, *and* CWB-$ *are NP-complete, each in both winner models.*

We now turn to the destructive cases. The following result is an adaption of a result due to Xia [49] who showed that DUB is in P for simplified Bucklin voting (again, recall Footnote 4).

Theorem 6 *In Bucklin elections,* DWB *and* DUB-$ *are in P, each in both winner models.*

Proof Both problems, Bucklin-DWB and Bucklin-DUB-$, can be solved by deterministic polynomial-time algorithms that use Algorithm 1, which was designed in Sect. 3 to solve the destructive coalitional weighted manipulation problem for Bucklin elections, Bucklin-DCWM. The main difference between a bribery and a manipulation instance is that in the latter only the preferences of the manipulators have to be found, whereas in the former both the votes that will be bribed and the new preferences for these voters have to be found. If we have the set of votes we want to change, we can use the algorithm for the manipulation problem to construct the preferences. Thus, for the runtime of the algorithm the determination of these voter sets is crucial, and we show that in Bucklin elections the number of voter sets whose modification might actually lead to a successful bribery is bounded by a polynomial in both the number of voters and the number of candidates.

Consider Algorithm 3 and a given input (C, V, W_V, p, k) to it. In particular, p is the designated candidate that we want to prevent from winning and assume that we have a yes-instance, i.e., our bribery action is successful. We denote by (C, V'') the election resulting from (C, V) where the k votes that can be changed have already been changed. Then there is a candidate $c \in C - \{p\}$ that reaches a strict majority on level i, and it holds that $score^i_{(C,V'')}(c) > score^i_{(C,V'')}(p)$, which means that p is not a Bucklin winner in (C, V''). To achieve that, for each $i < m$, there are only five types of preferences that might have been changed in V, and they can be grouped into the following subsets $T_{i,j} \subseteq V$, $1 \leq j \leq 5$:

- $T_{i,1}$: p is among the top $i-1$ positions and c is among the top i positions (when changing: p loses points, c does neither lose nor win points up to level i).
- $T_{i,2}$: p is among the top $i-1$ positions and c is not among the top i positions (when changing: p loses points, c wins points up to level i).
- $T_{i,3}$: p is on position i and c is among the top $i-1$ positions (when changing: p loses points, c does neither lose nor win points up to level i).
- $T_{i,4}$: p is on position i and c is not among the top $i-1$ positions (when changing: p loses points, c wins points up to level i).
- $T_{i,5}$: both p and c are not among the top i positions (when changing: p does neither lose nor win points, c wins points up to level i).

For a sublist of voters $V' \subseteq V$, denote their total weight by W'_V. Algorithm 3 for Bucklin-DWB works as follows.

It is easy to see that Algorithm 3 runs in deterministic polynomial time: the two outer for-loops iterate up to m times, whereas the inner loop tests up to k^5 variations of the vector $(a_1, a_2, \ldots, a_5)$. Since $k \leq n$, we have that the number of executions of Algorithm 1 is in $\mathcal{O}(m^2 n^5)$.

For the proof of correctness, we show that given a bribery instance (C, V, W_V, k, p), the output of Algorithm 3 is "YES" if and only if $(C, V, W_V, k, p) \in$ Bucklin-DWB.

From left to right: If the algorithm returns "YES" in line 10, then Algorithm 1 could find a successful destructive manipulation regarding p for k manipulators with total weight $W_{V'}$. So p is not a Bucklin winner in the election (C, V''), where V'' is the list of voters with k changed votes. That means that $(C, V, W_V, k, p) \in$ Bucklin-DWB.

From right to left: Assume that $(C, V, W_V, k, p) \in$ Bucklin-DWB. Thus, there exists a set of k voters V' with total weight $W_{V'}$ such that changing these votes prevents p from being a Bucklin winner in (C, V''), where V'' is the new voter list containing the k changed votes. We want to show that such a V'' can always be transformed to the list of votes V' that is changed in Algorithm 3. From our assumptions it follows that we have a candidate

Algorithm 3: Algorithm for Bucklin-DWB

input : C set of candidates
V list of voters
W_V list of weights of voters
k number of votes that may be changed
p designated candidate
output: "YES" if $(C, V, W_V, k, p) \in$ Bucklin-DWB
"NO" if $(C, V, W_V, k, p) \notin$ Bucklin-DWB

```
1  let A = {(a_1, a_2, ..., a_5) | a_i ∈ {0, 1, ..., k}}, V' = ∅;
2  foreach c ∈ C − {p} do
3      foreach i < m do
4          foreach (a_1, a_2, ..., a_5) ∈ A do
5              if Σ_{ℓ=1}^{5} a_ℓ ≤ k then
6                  foreach j ∈ {1, 2, ..., 5} do
7                      add the a_j heaviest votes in T_{i,j} to V';
8                  run Algorithm 1 on input (C, V − V', W_{V−V'}, W_{V'}, p);
9                  if Bucklin-DCWM(C, V − V', W_{V−V'}, W_{V'}, p)= "YES" then
10                     return "YES";
11
12
13 return "NO";
```

$c \in C - \{p\}$ and a level $i < m$ such that c is a level i Bucklin winner that prevents p from being a Bucklin winner (in other words, in every Bucklin election there is *some* winner, so if p is not a winner then some other candidate c is).

Assume that there are voters in V'' whose preferences were not in one of the $T_{i,j}$ before the changes were made, i.e., votes were changed that not necessarily needed to be changed to prevent p from being a Bucklin winner. Undo these changes and change the same number of votes in the lists $T_{i,j}$ that were not changed before. We then have that all changed votes are in one of the $T_{i,j}$.

Since Bucklin is monotonic, we can always replace votes with higher weight by votes with lower weight (in one $T_{i,j}$) without risking that p would win just because of this exchange. Thus, we know that we can transform any given list of bribed votes to a list that the algorithm would construct and this would still prevent p from winning. This implies that if there is a list of k voters that can be successfully bribed to prevent p from being a Bucklin winner, the algorithm will find it.

For the Bucklin-DUB-$ problem the same algorithm can be used. The only difference is that all weights have to be set to one, the cheapest instead of the heaviest votes are added to V' in line 7 (i.e., in line 7 we add the votes with the lowest price instead of the ones with the greatest weight), and it has to be tested whether the sum of the prices of the chosen votes does not exceed the budget.

For the unique-winner case, run the algorithm solving the unique-winner variant of Bucklin-DCWM in line 8. □

From Theorem 6 we have the following corollary.

Corollary 4 *In Bucklin elections,* DUB *is in P in both winner models.*

This algorithm can be easily adapted for fallback elections. Due to the fact that in fallback elections the voters do not have to rank all candidates, it is possible that a candidate wins

only on level m. Thus, we can decide DWB for fallback elections as well by making the following changes in Algorithm 3:

- Change "$i < m$" in line 3 to "$i \leq m$,"
- use the fallback analogue of Algorithm 1 in line 8, and
- change "Bucklin-DCWM" in line 9 to "fallback-DCWM,"

Theorem 7 *In fallback elections,* DWB*,* DUB*, and* DUB-$ *are in P, each in both winner models.*

It remains to show the complexity of the destructive variant of priced bribery in weighted Bucklin and fallback elections.

Theorem 8 *Both Bucklin-*DWB-$ *and fallback-*DWB-$ *are NP-complete, each in both winner models.*

Proof That Bucklin-DWB-$ and fallback-DWB-$ are in NP in both winner models is again easy to see. We show NP-hardness by a reduction from PARTITION. (The same reduction works for both problems.) Let $(A, (a_1, \ldots, a_k))$ with $A = \{1, \ldots, k\}$ and $\sum_{i=1}^{k} a_i = 2K$ be an instance of PARTITION. We construct the following Bucklin (fallback) election (C, V) with $C = \{c, p\}$ and $k + 1$ votes in V: For each $i \in \{1, \ldots, k\}$, we have one voter v_i with weight $w_i = a_i$, price $\pi_i = a_i$, and preference $p > c$, and we have one voter v_{k+1} with weight $w_{k+1} = 1$, price $\pi_{k+1} = K + 1$, and preference $c > p$ (for fallback, all voters approve of both candidates).

The total weight of the voters in (C, V) is $2K + 1$, so $maj(V) = K + 1$. Let K be the budget that may not be exceeded and let p be the designated candidate. Obviously, p is the unique level 1 Bucklin (fallback) winner in (C, V).

We claim that $(A, (a_1, \ldots, a_k)) \in$ PARTITION if and only if p can be prevented from being a Bucklin (fallback) winner by changing votes in V without exceeding the budget K.

From left to right: Let $(A, (a_1, \ldots, a_k)) \in$ PARTITION with $A' \subseteq A$ such that $\sum_{i \in A'} a_i = K$. Change the votes of those voters with weight $w_i = a_i$ for $i \in A'$ from $p > c$ to $c > p$. With these changes we have that on the first level, p has K points and c has $K + 1$ points, so c is the new level 1 Bucklin (fallback) winner and p has been prevented from winning.

From right to left: Assume that p is not a Bucklin (fallback) winner in the bribed election. Since there are only two levels, c has to win on the first level to prevent p from winning. Changing the vote of voter v_{k+1} would provide no gain (and would be too expensive), so only the votes of $v_1, \ldots, v_k$ may be changed. For each of the voters, the price equals the weight, so voters with a total weight of K can be changed. Candidate c has one point on the first level in the original election, so it is only possible to make c a unique level 1 Bucklin winner by fully exhausting the budget and changing the votes with a total weight of K from $p > c$ to $c > p$ (or, for the case of fallback, to approve of c only, which gives the same effect). Thus, there is a subset $A' \subseteq A$ such that $\sum_{i \in A'} a_i = K$, so $(A, (a_1, \ldots, a_k)) \in$ PARTITION.

For the unique-winner model, simply omit voter v_{k+1} in the voter list. □

5 Campaign management

In the discussion so far, we have focused on bribery and manipulation as means of attacking Bucklin and fallback elections. However, it is also quite natural to consider bribery scenarios through the lenses of running a political campaign. After all, in a successful campaign, the candidates spend their effort (measured in terms of time, in terms of financial cost of

organizing promotional activities, and even in terms of the difficulty of convincing particular voters) to change the minds of the voters. Thus a campaign preceding an election can be seen as spending some resources for voters' support. Formally, this idea is very close to bribery (indeed, this view of campaign management was first presented in a paper whose focus was on a bribery problem [21]). In the next section we will describe the two campaign-management problems that we focus on in this paper, in the following two we will provide our results, and in the last one we will briefly discuss other campaign-management problems from the literature.

5.1 Definitions and overview of results

We start by discussing one of the most general campaign-management problems, namely the SWAP BRIBERY problem introduced by Elkind et al. [21]. This problem models a situation where a campaign manager, who is interested in the victory of a given candidate p, can organize meetings with specific voters (the unweighted variant of the problem) or with groups of like-minded voters (the weighted variant) and convince them to change their preference orders. However, the difficulty (or, as we will say from now on, the cost) of changing the voters' preference orders depends both on the voter and on the extent of the change (for example, it might be expensive to swap a voter's most preferred candidate with this voter's least preferred one, but it might be very cheap to swap the voter's two least preferred candidates). Formally, Elkind et al. [21] define *swap-bribery price functions* that for each voter and for each pair of candidates give the cost of swapping these two candidates in the voter's preference order (provided the candidates are adjacent in this order).

Definition 1 (Elkind et al. [21]) A *swap-bribery price function* for voter v_i is a function $\pi_i : C \times C \to \mathbb{N}$ that specifies for each ordered pair (c_r, c_s) of candidates the price for changing v_i's preference order from $\cdots > c_r > c_s > \cdots$ to $\cdots > c_s > c_r > \cdots$. Only candidates that are adjacent in a vote can be swapped.

In the $\mathscr{E}$-CONSTRUCTIVE SWAP BRIBERY problem, where $\mathscr{E} \in \{\text{BV}, \text{FV}\}$, we ask if there exists a sequence of swaps of adjacent candidates that lead to a given candidate being an $\mathscr{E}$ winner (note that the swaps are performed in sequence; even if some candidates are not adjacent at first, they may become adjacent in the course of performing the swaps and, then, can be swapped themselves).

$\mathscr{E}$-CONSTRUCTIVE UNWEIGHTED SWAP BRIBERY ($\mathscr{E}$-CUSB)	
Given	An $\mathscr{E}$ election (C, V), where $V = (v_1, \ldots, v_n)$, a designated candidate p, a list $(\pi_1, \ldots, \pi_n)$ of swap bribery price functions, and a nonnegative integer k.
Question	Can p be made an $\mathscr{E}$ winner of an election resulting from the input election by conducting a sequence of swaps of adjacent candidates in the voters' ballots such that the total cost of the swaps does not exceed the budget k?

We define the weighted variant of the problem, $\mathscr{E}$-CWSB, in the standard way (as far as we can tell, the weighted variant of the problem has not been studied before). However, it will soon become clear why the weighted variant is not particularly interesting and so we omit stating explicitly the easy modification of the definition. We also define the destructive variants of the swap bribery problems ($\mathscr{E}$-DUSB and $\mathscr{E}$-DWSB) in the usual way, by changing the question to ask whether p can be prevented from being an $\mathscr{E}$ winner.

Swap bribery is a very difficult problem—it is NP-complete for almost all natural voting rules (and, in particular, in the next section we will see a very strong hardness result for the Bucklin and fallback rules). However, in Sect. 5.4 we briefly mention its natural special case which is in P for both Bucklin and fallback voting.

The definition of swap bribery is very natural for voting rules—such as Bucklin—where each voter ranks all the candidates. However, we need to extend it for the case of fallback voting where the ballots consist of the approved part (with the candidates being ranked) and of the disapproved part (with the candidates not being ranked). In our approach, we define swap bribery under fallback to allow the swaps within the approved parts of the votes only. Naturally, one could also define costs for including given disapproved candidates in the approved part and, indeed, Elkind et al. [21] did so for SP-AV (SP-AV is another variant of the approval system).[8] However, following Schlotter et al. [47] (and Baumeister et al. [5]), we believe that it is more informative to study the complexity of modifying the rankings within the approved parts and the complexity of modifying the sets of approved candidates separately.

Specifcally, in addition to SWAP BRIBERY, we consider EXTENSION BRIBERY introduced by Baumeister et al. [5]. The idea of extension bribery is to capture very noninvasive campaign actions, where we try to convince some voters to include the designated candidate at the end of the ranking of approved candidates.

Definition 2 (Baumeister et al. [5]) The *extension bribery price function* $\delta_i : \mathbb{N} \to \mathbb{N}$ of a voter v_i defines the price for extending the approved part of v_i's vote with a given number of so-far-disapproved candidates (these new candidates are ranked below the previously-approved candidates, but among themselves are ranked as the briber requests).

EXTENSION BRIBERY is defined in the following way.

FV- CONSTRUCTIVE UNWEIGHTED EXTENSION BRIBERY (FV- CUEB)	
Given	A fallback election (C, V), where $V = (v_1, \ldots, v_n)$, a designated candidate p, a list $(\delta_1, \ldots, \delta_n)$ of extension bribery price functions, and a nonnegative integer k.
Question	Can p be made a fallback winner by extending the approved parts of the voters' ballots without exceeding the budget k?

Again, the weighted variant (FV- CWEB) is defined in the natural way and so are the destructive variants (FV- DUEB and FV- DWEB).

Table 10 summarizes the results of this section. Note that all these results hold in both winner models and that we give the detailed proofs for the nonunique-winner models only, while briefly providing the needed changes for the adaption to the unique-winner model. (The dashes "–" in Table 10 indicate that extension bribery is not applicable to voting rules such as Bucklin where each voter ranks all the candidates.)

[8] Like fallback voting, SP-AV is a hybrid variant of approval voting. It has been introduced by Brams and Sanver [9] and slightly modified by Erdélyi et al. [24] to cope with certain control actions (see also the chapter by Baumeister et al. [4] for a thorough discussion of this voting system).

Table 10 Overview of results for swap bribery and extension bribery in Bucklin and fallback voting

Problem	Bucklin voting		Fallback voting	
	Complexity	Reference	Complexity	Reference
$\mathscr{E}$-CUSB	NP-complete	Theorem 10	NP-complete	Corollary 5
$\mathscr{E}$-DUSB	NP-complete	Corollary 5	NP-complete	Corollary 5
$\mathscr{E}$-CWSB	NP-complete	Theorem 9	NP-complete	Theorem 9
$\mathscr{E}$-DWSB	NP-complete	Theorem 9	NP-complete	Theorem 9
FV- CUEB	–		P	Theorem 12
FV- DUEB	–		P	Theorem 12
FV- CWEB	–		NP-complete	Theorem 11
FV- DWEB	–		NP-complete	Theorem 11

The dashes "–" indicate that extension bribery is not applicable to Bucklin voting

5.2 Results for swap bribery

We start by quickly observing that (constructive and destructive) WEIGHTED SWAP BRIBERY is NP-complete for both the Bucklin and fallback rules. Note that we will use *BV* as a shorthand for Bucklin voting.

Theorem 9 BV- CWSB, BV- DWSB, FV- CWSB, *and* FV- DWSB *are NP-complete, each in both winner models, even for elections with only two candidates.*

Proof The proof for Bucklin is a direct consequence of the fact that CWB-$ is NP-complete for plurality, even for just two candidates [27] (the result holds both for the uniqe-winner case and for the nonunique-winner case). For two candidates, the Bucklin rule is identical to the plurality rule. Further, for two candidates CWB-$ is, in essence, identical to CWSB (the only possible bribery is to swap the only two candidates), and the nonunique-winner variant of CWB-$ is, in essence, identical to DWSB.

For fallback, membership of the problems in NP is clear, and NP-hardness follows by the same arguments as for Bucklin, by considering the setting where every voter approves of all candidates. □

For the unweighted case, NP-completeness of BV- CUSB follows immediately from the fact that the possible winner problem for Bucklin is NP-complete (see the papers of Konczak and Lang [37], for the definition of the possible winner problem, and of Xia and Conitzer [50], for the result regarding Bucklin) and the fact that, for a given voting rule, the possible winner problem reduces to the swap bribery problem [21]. However, on the one hand, NP-hardness of the possible winner problem was established for the simplified variant of Bucklin's rule only (recall once more Footnote 4), and on the other hand, we can show that BV- CUSB is NP-complete even for elections with just two voters.

Theorem 10 BV- CUSB *is NP-complete in both winner models, even for elections with only two voters.*

Proof It is easy to see that BV- CUSB is in NP. We show NP-hardness by a reduction from the following problem (which we will refer to as SINGLE- VOTE SWAP BRIBERY): Given a vote v (expressed as a preference order over some candidate set C), a swap-bribery price

function π for v, a designated candidate $p \in C$, and two nonnegative integers ℓ and k, is there a sequence of swaps of adjacent candidates, of total cost at most k, that ensure that p is ranked among the top ℓ positions in v? Elkind et al. [21] studied this problem as a variant of the swap bribery problem for ℓ-approval elections, where ℓ is part of the input and the election consists of a single vote; they established NP-completeness of the problem in their Theorem 6.

Let $I = (C, v, \pi, p, \ell, k)$ be an instance of SINGLE- VOTE SWAP BRIBERY, where $\|C\| = m$. We form a Bucklin election $E = (A, V)$ as follows. Let C' be a collection of $m-1$ dummy candidates with $C \cap C' = \emptyset$. We set $A = C \cup C' \cup \{d\}$. We partition C' into two sets, C'_1 and C'_2, such that $\|C'_1\| = \ell - 1$ and $\|C'_2\| = \|C'\| - (\ell - 1) = m - \ell$. (We pick any easily computable partition.) We let V be a collection of two voters, v_1 and v_2, with price functions π_1 and π_2:

1. v_1 has preference order $d > v > C'$ (i.e., v_1 ranks d on the top position, then all the candidates from C in the same order as v, and then all the candidates from C', in some arbitrary-but-easy-to-compute order). For each two candidates $x, y \in A$, if both x and y are in C then we set $\pi_1(x, y) = \pi(x, y)$, and otherwise we set $\pi_1(x, y) = k + 1$.
2. v_2 has preference order $p > C'_1 > d > C'_2 > C - \{p\}$ (that is, v_2 ranks p first, then the $\ell - 1$ candidates from C'_1 followed by d, followed by the remaining candidates from C', which are then followed by the candidates from $C - \{p\}$). For each two candidates $x, y \in A$, we set $\pi_2(x, y) = k + 1$.

Note that in our election $maj(V) = 2$. Further, the only two candidates that are ranked among the top $m + 1$ positions of both voters are p and d. Candidate d has Bucklin score $\ell + 1$ and, thus, we have the following situation:

1. If p is ranked among the top ℓ positions in v_1, then p is the unique Bucklin winner of the election.
2. If p is ranked in the $(\ell + 1)$st position by v_1, then both p and d are Bucklin winners.
3. If p is ranked in a position worse than the $(\ell + 1)$st position by v_1, then d is the unique Bucklin winner.

We claim that p can become a Bucklin winner of election E through a swap bribery of cost at most k if and only if I is a yes-instance of SINGLE- VOTE SWAP BRIBERY.

From right to left: Assume that I is a yes-instance of SINGLE- VOTE SWAP BRIBERY. This means that there is a sequence of swaps within v after which p is ranked among the top ℓ positions in v. Applying the same swaps to v_1 would cost the same and would put p among the top $\ell + 1$ positions in v_1, making p a Bucklin winner.

From left to right: Assume that there is a cost-at-most-k sequence of swaps within V that make p a Bucklin winner. Since any swap that is not in the v part of v_1 costs $k + 1$, we have that d's Bucklin score is still $\ell + 1$, and, thus, after the swaps, p's Bucklin score is in $\{2, \ldots, \ell + 1\}$. Executing the same swaps within v shows that I is a yes-instance of SINGLE- VOTE SWAP BRIBERY. For the unique-winner model, simply move d one position lower in v_2. □

To establish that BV- DUSB in the nonunique-winner model also is NP-complete for the case of two voters, it suffices to use the unique-winner construction from the proof of Theorem 10, but with the goal to prevent candidate d from being a Bucklin winner (the reader can see that p is the only candidate who can threaten d without exceeding the given budget). For the unique-winner destructive case, it suffices to use the BV- CUSB nonunique-winner constructive construction, but with the goal to prevent candidate d from being a unique

Bucklin winner. Finally, as noted at the end of Sect. 2.1, NP-hardness of FV- CUSB and FV- DUSB follows directly from the NP-hardness of BV- CUSB and BV- DUSB in both winner models. We summarize these observations in the following corollary.

Corollary 5 BV- DUSB, FV- CUSB, *and* FV- DUSB *are NP-complete, each in both winner models, even for elections with only two voters.*

5.3 Results for extension bribery

Let us now move on to the study of extension bribery, which here applies to fallback voting only. The following observation will simplify our discussion.

Observation 1 *In (constructive) extension bribery problems for the fallback rule it is never profitable to extend any vote in any other way than by asking the voter to include the designated candidate on the last unranked position.*

Thus we will often specify the extension bribery price functions by simply giving the cost of extending the vote by just one candidate (we will refer to this number as *extension cost* of the vote).

Not surprisingly, the weighted variants of extension bribery are NP-complete.

Theorem 11 *For elections with at least three candidates, both* FV- CWEB *and* FV- DWEB *are NP-complete, each in both winner models.*

Proof Obviously, FV- CWEB is in NP for any number of candidates. To show NP-hardness, we use a reduction from PARTITION. Note that our reduction, which has three candidates, can be modified so that an election with any number $m \geq 3$ of candidates will be constructed: Simply add the needed number of candidates to C and let all voters disapprove of the newly added candidates. Let $(A, (a_1, \ldots, a_k))$ with $A = \{1, \ldots, k\}$ and $\sum_{i=1}^{k} a_i = 2K$ be an instance of PARTITION.

We define the fallback election (C, V) with the candidate set $C = \{b, c, p\}$, the designated candidate is p, and we let V consist of $k + 2$ voters:

1. There is one voter v_0 with the ballot $p \mid \{b, c\}$, with weight K, and extension cost $K + 1$.
2. For each i, $1 \leq i \leq k$, there is a voter v_i who casts the ballot $c \mid \{b, p\}$, has weight $w_i = a_i$, and extension cost a_i.
3. There is one voter v_{k+1} with the ballot $b \mid \{c, p\}$, with weight K, and extension cost $K + 1$.

The total sum of the voters' weights in this election is $4K$, so $maj(V) = 2K + 1$. The weighted scores of the candidates in (C, V) are shown in Table 11a. As no candidate reaches the majority threshold, candidate c wins by approval score and is the unique fallback winner in (C, V).

We claim that there is a set $A' \subseteq A$ such that $\sum_{i \in A'} a_i = \sum_{i \notin A'} a_i = K$ if and only if p can be made a fallback winner by extension-bribing some of the voters without exceeding the budget K.

From left to right: We assume that there is a set $A' \subseteq A$ such that $\sum_{i \in A'} a_i = \sum_{i \notin A'} a_i = K$. We can change the votes from the voters v_i where $i \in A'$ from $c \mid \{b, p\}$ to $c > p \mid \{b\}$. Each of these changes costs a_i, so the overall sum of the costs is K. The candidates have the weighted scores in the resulting election (C, V') that are shown in Table 11b. So p can be made a fallback winner by extension-bribing voters in V without exceeding the budget K.

Table 11 Scores in the election constructed in the proof of Theorem 11

	b	c	p
(a) Scores in (C, V)			
score	K	$2K$	K
(b) Scores in (C, V')			
$score^1$	K	$2K$	K
$score^2$	K	$2K$	$2K$

From right to left: We assume that p is a fallback winner in election (C, V'), where V' is the changed voter list and the costs for the changes are at most K. Since the cost limit is K the only changes that can be made, and that are profitable for p, are adding p to the approval strategies of some of the voters $v_1, \ldots, v_k$. The weighted score of candidate c cannot be decreased, so p has to gain K points to tie with candidate c. Hence, there has to be a set $A' \subseteq A$ such that $\sum_{i \in A'} a_i = \sum_{i \notin A'} a_i = K$ and p has to be added to the approval strategies of the voters v_i where $i \in A'$.

For the unique-winner case of FV- CWEB, only the weight of v_0 has to be changed to $K + 1$ in the above election.

To show the result for the destructive case, for the nonunique-winner model it suffices to use the same construction as for the constructive unique-winner case, with the goal to prevent c from winning (it can be accomplished either by p or by b). Similarly, for the unique-winner destructive case, we use the same construction as for the nonunique-winner constructive case, with the goal to prevent c from being a unique winner (again, either p or b can be used for this purpose). □

On the other hand, the unweighted variant of the problem is in P. This is a nice complement to the hardness results of Schlotter et al. [47] regarding support bribery. The main difference regarding support bribery and extension bribery is that under the former we assume the voters to rank all the candidates but declare as approved only some of their top candidates, whereas in the latter (and, in general, in our model) we assume the voters to rank only the approved candidates and completely disregard the disapproved candidates.

Theorem 12 FV- CUEB *and* FV- DUEB *are in P, each in both winner models.*

Proof Let us consider FV- CUEB first. We claim that Algorithm 4 solves the problem in polynomial time. The algorithm considers each level s in which p could possibly become a fallback winner and tries the cheapest bribery that might achieve this. The algorithm clearly runs in polynomial time and its correctness follows from Observation 1.

It is clear how to adapt Algorithm 4 to the case of unique winners. Then, to solve the destructive unique-winner variant of the problem, it suffices to check if any candidate other than p can be made a fallback winner within the budget. For the destructive case in the nonunique-winner model, simply check if either (a) candidate p that we want to prevent from being a winner already is not a winner, or (b) there is some other candidate that can become a unique fallback winner through extension bribery (use the above algorithm for the unique-winner case).[9] □

[9] We note that this approach to solving the destructive cases is not general, but it is easy to see that it works for fallback.

Algorithm 4: Algorithm for fallback-CUEB

input : C set of candidates
V list of voters
$\Delta = (\delta_1, \ldots, \delta_n)$ list of extension bribery price functions
k budget
p designated candidate
output: "YES" if $(C, V, \Delta, k, p) \in$ fallback-CUEB
"NO" if $(C, V, \Delta, k, p) \notin$ fallback-CUEB

```
1 foreach s ∈ {1, ..., ||C||} do
2     let (v'_1, ..., v'_r) be a sublist of V containing voters that approve of at most s − 1 candidates and do
      not approve of p, sorted by extension costs in ascending order;
3     foreach t ∈ {0, ..., r} do
4         if changing v'_1, ... v'_t to approve p makes p a fallback winner then
5             if the sum of extension costs of v'_1, ..., v'_t is less than or equal to k then
6                 return "YES";
7
8
9 return "NO";
```

5.4 Other campaign-management problems

There is a number of other problems that model various ways in which a political campaign could be run. For example, motivated by the apparent hardness of swap bribery, Elkind et al. [21] defined its much-simplified variant, shift-bribery, where every swap has to involve the designated candidate p (that is, only the designated candidate can be "shifted" forward in selected votes). The complexity of shift bribery was studied for a number of voting rules [12,17,20,21], including Bucklin and fallback voting [47]. Interestingly, even though we gave strong hardness results for swap bribery under Bucklin and fallback, Schlotter et al. [47] have shown that shift bribery for these rules is in P.

Schlotter et al. [47] also introduced support bribery, a problem that models such actions as increasing or decreasing the number of candidates that a given voter approves of. The problem is very similar in spirit to extension bribery, but there is also a crucial difference: In extension bribery we can extend each vote in any arbitrary way, whereas in support bribery each voter has a complete preference order over the whole set of candidates and we can only affect the number of candidates that he or she approves of. Further, in extension bribery we can only increase the number of candidates approved by a given voter, but in support bribery both increasing and decreasing the number of approved candidates is possible. Schlotter et al. [47] show that this problem is NP-complete for fallback.[10]

6 Discussion and future research

We believe that the complexity of manipulation, bribery, and campaign management is particularly interesting for the case of Bucklin and fallback voting. These rules are, in an intuitive sense, natural generalizations of the k-approval family of rules, for which many of our problems are (relatively) easy, and it is interesting to see how this generalization affects our

[10] They also show that the problem is hard in the sense of parametrized complexity for two natural parameters describing the extent of change to the number of approved candidates. Interestingly, they show the problem to be fixed-parameter tractable if the number of approved candidates can either only increase or only decrease.

problems. Further, for many of the more complex rules, such as Borda and Copeland, manipulation, bribery and campaign-management problems tend to be hard. In our study, we are interested if the complexity of these problems for Bucklin and fallback is more similar to that for k-approval rules or to that for rules such as Borda and Copeland. We have presented specific results in the preceding sections and here we would like to give a high-level, intuitive overview.

We compare Bucklin and fallback voting to k-approval, Borda, and Copeland. Under k-approval, each candidate receives a point for each vote in which this candidate is ranked among the top k positions (we assume that k is a fixed constant; for example, we may consider 2-approval, 3-approval, and so on). Under the Borda rule, each voter v gives each candidate c as many points as there are candidates that v ranks below c. Under Copeland, for each two candidates c and d we check if more voters prefer c to d or the other way round. In the former case c receives a point and in the latter case d does (for the sake of specificity, in case of a tie we assume that neither of them receives a point, i.e., we consider the system that Faliszewski et al. [29] call Copeland^0). For each of these rules, the candidates that get the most points are the winners.

First, let us consider the family of manipulation problems. In this case, Bucklin and fallback voting behave similarly to the k-approval family of rules (see the thesis of Lin [38] for an overview of known results regarding the complexity of manipulating k-approval elections), and unlike Borda [6,14,16] and Copeland [14,31]. For example, for k-approval the unweighted constructive coalitional manipulation problem is in P and for Borda and Copeland it is NP-complete. Fallback voting is even easier to manipulate than most rules, including those in the k-approval family, because for fallback voting constructive coalitional manipulation is easy even for the case of weighted voters. This is so because in fallback voting the voters have an easy way of passing maximum support to their most preferred candidate without passing any support to anyone else. On the other hand, destructive coalitional manipulation problems tend to be easy for most of the voting rules and so there is no significant difference between Bucklin, fallback voting, and the other rules.

The complexity of bribery for Bucklin and fallback is similar to its complexity for the k-approval family of rules (for large enough k, i.e., for $k \geq 4$; we point the reader to the thesis of Lin [38] for an overview of results regarding constructive bribery under k-approval and to the work of Xia [49] for results regarding the destructive cases) and for Borda [13,27], but differs from that for Copeland [29]. Informally put, for Bucklin, fallback, k-approval (for large enough k), and for Borda constructive bribery is NP-complete and destructive bribery is in P (with the exception of the variant with both weights and prices, which is NP-complete). On the other hand, all variants of bribery are NP-complete for Copeland.

Finally, let us consider the complexity of campaign-management problems, focusing on swap bribery and extension bribery. Swap bribery is NP-complete for essentially all our rules [21] (i.e., for k-approval, $k \geq 2$, Borda, Copeland, Bucklin, and fallback voting). So Bucklin and fallback voting do not stand out in any particular way. The situation with extension bribery is, however, significantly different. The problem is in P for fallback voting, but is NP-complete for variants of Borda and Copeland adjusted to work in the setting where voters rank some of their top candidates only [5] (however, we should mention that there is a variant of Borda for which extension bribery is in P and that the weighted variant of extension bribery is NP-complete for fallback voting). Additionally, it is known that shift bribery for Bucklin and fallback voting are in P [47] (analogously to the case of the k-approval family of rules [21]), whereas shift bribery is NP-complete for both Borda and Copeland.

All in all, we conclude that, typically, the complexity of manipulation, bribery, and campaign-management problems for Bucklin and fallback voting is similar to that for

simpler rules, such as k-approval. However, it is also the case that the proofs for Bucklin and fallback voting are often more involved than those for k-approval and, certainly, not all results translate.

It is natural to ask what is the value of worst-case analysis of manipulation, bribery, and campaign management for voting rules in general, and for Bucklin and fallback voting specifically. There are several answers. On the one hand, NP-hardness results can be seen as some forms of (rather weak) safety guarantees against the respective forms of manipulation. From a more practical point of view, however, such results should rather be interpreted as justifications for designing and using heuristic approaches when solving the respective problems. Further, reductions that prove NP-hardness typically show those aspects of a particular problem and a particular voting rule that are most difficult to analyze algorithmically. Identifying such features of a problem can help in developing heuristics and approximation algorithms. From a yet different perspective, results such as those presented in this paper contribute to understanding the nature of the considered voting rules (as we have argued in this section, for the case of Bucklin and fallback voting the comparison to k-approval rules is particularly interesting).

As far as future work goes, we believe that the most interesting and promising direction would be to study heuristic and approximation algorithms for our problems. Such attacks were already taken in the literature regarding other problems and/or voting rules [16,44,48]. We believe that one should try to develop appropriate algorithms not only for those problems for which we have NP-hardness results, but even for those where polynomial-time algorithms exist. Specifically, it would be very interesting to develop distributed heuristics that require very limited communication from the agents involved in manipulating election results.

7 Conclusions

We have given an in-depth study of the complexity of manipulation, bribery, and campaign management for Bucklin and fallback voting. Our results complement those regarding control, campaign management, and possible/necessary winner problems and, in effect, we now have an almost complete picture of the (worst-case) complexity of Bucklin and fallback voting for all the standard election problems (there is only a single result missing for one of the control problems). Having this complete picture is particularly useful for Bucklin and fallback voting: For control problems they are among the hardest voting rules, but for bribery and manipulation—together with k-approval voting—they often lay on the edge of (in)tractability. It is thus interesting to see their complexity for each of the problems separately.

Acknowledgments We thank Edith Hemaspaandra and Lane A. Hemaspaandra for interesting discussions on these results after a RIT/UR Theory Canal talk. This work was supported in part by DFG Grant RO 1202/15-1, by a DAAD Grant for a PPP Project in the PROCOPE Program, by NCN Grants 2012/06/M/ST1/00358 and 2011/03/B/ST6/01393, and by AGH University Grant 11.11.230.124.

References

1. Bartholdi III, J., & Orlin, J. (1991). Single transferable vote resists strategic voting. *Social Choice and Welfare*, *8*(4), 341–354.
2. Bartholdi III, J., Tovey, C., & Trick, M. (1989). The computational difficulty of manipulating an election. *Social Choice and Welfare*, *6*(3), 227–241.
3. Bartholdi III, J., Tovey, C., & Trick, M. (1992). How hard is it to control an election? *Mathematical and Computer Modelling*, *16*(8/9), 27–40.

4. Baumeister, D., Erdélyi, G., Hemaspaandra, E., Hemaspaandra, L., & Rothe, J. (2010). Computational aspects of approval voting. In J. Laslier & R. Sanver (Eds.), *Handbook on approval voting, chap. 10* (pp. 199–251). Berlin: Springer.
5. Baumeister, D., Faliszewski, P., Lang, J., & Rothe, J. (2102). Campaigns for lazy voters: Truncated ballots. *Proceedings of the 11th International Joint Conference on Autonomous Agents and Multiagent Systems, IFAAMAS* (pp. 577–584).
6. Betzler, N., Niedermeier, R., & Woeginger, G. (2011). Unweighted coalitional manipulation under the Borda rule is NP-hard. *Proceedings of the 22nd International Joint Conference on Artificial Intelligence* (pp. 55–60).
7. Brams, S., & Fishburn, P. (1978). Approval voting. *American Political Science Review*, *72*(3), 831–847.
8. Brams, S., & Fishburn, P. (1983). *Approval voting*. Boston: Birkhäuser.
9. Brams, S., & Sanver, R. (2006). Critical strategies under approval voting: Who gets ruled in and ruled out. *Electoral Studies*, *25*(2), 287–305.
10. Brams, S., & Sanver, R. (2009). Voting systems that combine approval and preference. In S. Brams, W. Gehrlein, & F. Roberts (Eds.), *The mathematics of preference, choice, and order: Essays in honor of Peter C. Fishburn* (pp. 215–237). Berlin: Springer.
11. Brandt, F., Conitzer, V., & Endriss, U. (2013). Computational social choice. In G. Weiß (Ed.), *Multiagent systems* (2nd ed., pp. 213–283). Cambridge, MA: MIT Press.
12. Bredereck, R., Chen, J., Faliszewski, P., Nichterlein, A., & Niedermeier, R. (2014). Prices matter for the parameterized complexity of shift bribery. *Proceedings of the 28th AAAI Conference on Artificial Intelligence* (pp. 1398–1404). AAAI Press.
13. Brelsford, E., Faliszewski, P., Hemaspaandra, E., Schnoor, H., & Schnoor, I. (2008). *Approximability of manipulating elections. In: Proceedings of the 23rd AAAI Conference on Artificial Intelligence* (pp. 44–49). AAAI Press.
14. Conitzer, V., Sandholm, T., & Lang, J. (2007). When are elections with few candidates hard to manipulate? *Journal of the ACM*, *54*(3), 14.
15. Conitzer, V., & Walsh, T. (2014). Barriers to manipulation. In F. Brandt, V. Conitzer, U. Endriss, J. Lang, & A. Procaccia (Eds.), *Handbook of computational social choice*. Cambridge: Cambridge University Press.
16. Davies, J., Katsirelos, G., Narodytska, N., & Walsh, T. (2011). Complexity of and algorithms for Borda manipulation. *Proceedings of the 25th AAAI Conference on Artificial Intelligence* (pp. 657–662).
17. Dorn, B., & Schlotter, I. (2012). Multivariate complexity analysis of swap bribery. *Algorithmica*, *64*(1), 126–151.
18. Duggan, J., & Schwartz, T. (2000). Strategic manipulability without resoluteness or shared beliefs: Gibbard–Satterthwaite generalized. *Social Choice and Welfare*, *17*(1), 85–93.
19. Dutta, B., Jackson, M., & Le Breton, M. (2001). Strategic candidacy and voting procedures. *Econometrica*, *69*(4), 1013–1037.
20. Elkind, E., & Faliszewski, P. (2010). Approximation algorithms for campaign management. *Proceedings of the 6th International Workshop on Internet and Network Economics* (pp. 473–482). Springer, Lecture Notes in Computer Science #6484.
21. Elkind, E., Faliszewski, P., & Slinko, A. (2009). Swap bribery. *Proceedings of the 2nd International Symposium on Algorithmic Game Theory* (pp. 299–310). Springer, Lecture Notes in Computer Science #5814
22. Erdélyi, G., & Fellows, M. (2010). Parameterized control complexity in Bucklin voting and in fallback voting. In V. Conitzer & J. Rothe (Eds.), *Proceedings of the 3rd International Workshop on Computational Social Choice* (pp. 163–174). Universität Düsseldorf.
23. Erdélyi, G., Fellows, M., Rothe, J., & Schend, L. (2012). Control complexity in Bucklin and fallback voting. Technical Report arXiv:1103.2230 [cs.CC], Computing Research Repository, arXiv:org/corr/ (2012). March, 2011. Revised August, 2012. An extended version merging the CATS-2010, COMSOC-2010, AAMAS-2011, and SEA-2012 papers [26,22,25,44] has been accepted for publication in Journal of Computer and System Sciences.
24. Erdélyi, G., Nowak, M., & Rothe, J. (2009). Sincere-strategy preference-based approval voting fully resists constructive control and broadly resists destructive control. *Mathematical Logic Quarterly*, *55*(4), 425–443.
25. Erdélyi, G., Piras, L., & Rothe, J. (2011). The complexity of voter partition in Bucklin and fallback voting: Solving three open problems. *Proceedings of the 10th International Joint Conference on Autonomous Agents and Multiagent Systems. IFAAMAS* (pp. 837–844).
26. Erdélyi, G., & Rothe, J. (2010). Control complexity in fallback voting. *Proceedings of Computing: the 16th Australasian Theory Symposium* (pp. 39–48). Australian Computer Society Conferences in Research and Practice in Information Technology Series, vol. 32, no. 8.

27. Faliszewski, P., Hemaspaandra, E., & Hemaspaandra, L. (2009). How hard is bribery in elections? *Journal of Artificial Intelligence Research*, *35*, 485–532.
28. Faliszewski, P., Hemaspaandra, E., & Hemaspaandra, L. (2010). Using complexity to protect elections. *Communications of the ACM*, *53*(11), 74–82.
29. Faliszewski, P., Hemaspaandra, E., Hemaspaandra, L., & Rothe, J. (2009). Llull and Copeland voting computationally resist bribery and constructive control. *Journal of Artificial Intelligence Research*, *35*, 275–341.
30. Faliszewski, P., Hemaspaandra, E., Hemaspaandra, L., & Rothe, J. (2009). A richer understanding of the complexity of election systems. In S. Ravi & S. Shukla (Eds.), *Fundamental problems in computing: Essays in honor of professor Daniel J. Rosenkrantz, chap. 14* (pp. 375–406). Dordrecht: Springer.
31. Faliszewski, P., Hemaspaandra, E., & Schnoor, H. (2010). Manipulation of Copeland elections. *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems* (pp. 367–374). International Foundation for Autonomous Agents and Multiagent Systems.
32. Faliszewski, P., & Procaccia, A. (2010). AI's war on manipulation: Are we winning? *AI Magazine*, *31*(4), 53–64.
33. Faliszewski, P., & Rothe, J. (2014). Control and bribery. In F. Brandt, V. Conitzer, U. Endriss, J. Lang, & A. Procaccia (Eds.), *Handbook of computational social choice*. Cambridge: Cambridge University Press.
34. Garey, M., & Johnson, D. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. New York: W. H. Freeman and Company.
35. Gibbard, A. (1973). Manipulation of voting schemes. *Econometrica*, *41*(4), 587–601.
36. Hemaspaandra, E., Hemaspaandra, L., & Rothe, J. (2007). Anyone but him: The complexity of precluding an alternative. *Artificial Intelligence*, *171*(5–6), 255–285.
37. Konczak, K., & Lang, J. (2005). Voting procedures with incomplete preferences. *Proceedings of the Multidisciplinary IJCAI-05 Workshop on Advances in Preference Handling* (pp. 124–129).
38. Lin, A. (2012). Solving hard problems in election systems. Ph.D. thesis, Rochester Institute of Technology, Rochester, NY.
39. Menton, C. (2013). Normalized range voting broadly resists control. *Theory of Computing Systems*, *53*(4), 507–531.
40. Menton, C., & Singh, P. (2013). Control complexity of Schulze voting. *Proceedings of the 23rd International Joint Conference on Artificial Intelligence* (pp. 286–292). AAAI Press/IJCAI.
41. Papadimitriou, C. (1995). *Computational complexity* (2nd ed.). Reading, MA: Addison-Wesley.
42. Parkes, D., & Xia, L. (2012). A complexity-of-strategic-behavior comparison between Schulze's rule and ranked pairs. *Proceedings of the 26th AAAI Conference on Artificial Intelligence* (pp. 1429–1435). AAAI Press.
43. Rothe, J. (2005). *Complexity theory and cryptology. An introduction to cryptocomplexity. EATCS texts in theoretical computer science*. Berlin: Springer.
44. Rothe, J., & Schend, L. (2012). Control complexity in Bucklin, fallback, and plurality voting: An experimental approach. *Proceedings of the 11th International Symposium on Experimental Algorithms* (pp. 356–368). Springer, Lecture Notes in Computer Science #7276.
45. Rothe, J., & Schend, L. (2013). Challenges to complexity shields that are supposed to protect elections against manipulation and control: A survey. *Annals of Mathematics and Artificial Intelligence*, *68*(1–3), 161–193.
46. Satterthwaite, M. (1975). Strategy-proofness and Arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory*, *10*(2), 187–217.
47. Schlotter, I., Faliszewski, P., & Elkind, E. (2011). Campaign management under approval-driven voting rules. *Proceedings of the 25th AAAI Conference on Artificial Intelligence* (pp. 726–731).
48. Walsh, T. (2011). Where are the hard manipulation problems? *Journal of Artificial Intelligence Research*, *42*, 1–29.
49. Xia, L. (2012). Computing the margin of victory for various voting rules. *Proceedings of the 13th ACM Conference on Electronic Commerce* (pp. 982–999). ACM Press.
50. Xia, L., & Conitzer, V. (2011). Determining possible and necessary winners given partial orders. *Journal of Artificial Intelligence Research*, *41*, 25–67.
51. Xia, L., Zuckerman, M., Procaccia, A., Conitzer, V., & Rosenschein, J. (2009). Complexity of unweighted coalitional manipulation under some common voting rules. *Proceedings of the 21st International Joint Conference on Artificial Intelligence* (pp. 348–353). IJCAI.