Autonomous Agents and Multi-Agent Systems (2021) 35:38
https://doi.org/10.1007/s10458-021-09507-9

On coalitional manipulation for multiwinner elections: shortlisting

Robert Bredereck[2] · Andrzej Kaczmarczyk[1] · Rolf Niedermeier[1]

Accepted: 12 May 2021 / Published online: 8 July 2021

Abstract
Shortlisting of candidates—selecting a group of "best" candidates—is a special case of multiwinner elections. We provide the first in-depth study of the computational complexity of strategic voting for shortlisting based on the perhaps most basic voting rule in this scenario, ℓ-Bloc (every voter approves ℓ candidates). In particular, we investigate the influence of several different group evaluation functions (e.g., egalitarian versus utilitarian) and tie-breaking mechanisms modeling pessimistic and optimistic manipulators. Among other things, we conclude that in an egalitarian setting strategic voting may indeed be computationally intractable regardless of the tie-breaking rule. Altogether, we provide a fairly comprehensive picture of the computational complexity landscape of this scenario.

Keywords Computational social choice · Utility aggregation · Strategic voting · Parameterized computational complexity · Tie-breaking · SNTV · Bloc

1 Introduction

Assume that a university wants to select the two favorite pieces in classical style to be played during the next graduation ceremony. The students were asked to submit their favorite pieces. Then a jury consisting of seven members (three juniors and four seniors)

A preliminary version of this article appeared in the *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI '17)* [12]. In this full version we included all proofs and algorithms (together with our ILP formulations). Furthermore, we formalized the concept of simulation among tie-breaking rules.

✉ Robert Bredereck
robert.bredereck@hu-berlin.de

Andrzej Kaczmarczyk
a.kaczmarczyk@tu-berlin.de

Rolf Niedermeier
rolf.niedermeier@tu-berlin.de

[1] Faculty IV, Algorithmics and Computational Complexity, Technische Universität Berlin, Ernst-Reuter-Platz 7, 10587 Berlin, Germany

[2] Institut für Informatik, Algorithm Engineering, Humboldt-Universität zu Berlin, Rudower Chausse 25, 12489 Berlin, Germany

from the university staff selects from the six most frequently submitted pieces as follows: Each jury member approves two pieces and the two winners are those obtaining most of the approvals. The six options provided by the students are "Beethoven: Piano Concerto No. 5 (b_1)", "Beethoven: Symphony No. 6 (b_2)", "Mozart: Clarinet Concerto (m_1)", "Mozart: Jeunehomme Piano Concerto (m_2)", "Uematsu: Final Fantasy (o_1)", and "Badelt: Pirates of the Caribbean (o_2)." The three junior jury members are excited about recent audio-visual presentation arts (both interactive and passive) and approve o_1 and o_2. Two of the senior jury members are Mozart enthusiasts, and the other two senior jury members are Beethoven enthusiasts. Hence, when voting truthfully, two of them would approve the two Mozart pieces and the other two would approve the two Beethoven pieces. The winners of the selection process would be o_1 and o_2, both receiving three approvals whereas every other piece receives only two approvals.

The senior jury members meet every Friday evening and discuss important academic issues which include the graduation ceremony music selection processes, why "movie background noise" recently counts as classical music,[1] and the influence of video games on the ability of making important decisions. During such a meeting they agreed that a graduation ceremony should always be accompanied by pieces of traditional, first-class composers. Thus, finally all four senior jury members decide to approve b_1 and m_1 so these two pieces are played during the graduation ceremony.

Already this toy example above (which will be the basis of our running example throughout the paper) illustrates important aspects of strategic voting in multiwinner elections. In case of coalitional manipulation for single-winner elections (where a coalition of voters casts untruthful votes in order to influence the outcome of an election; a topic which has been intensively studied in the literature [9, 16]) one can always assume that a coalition of manipulators agrees on trying to make a distinguished alternative win the election. In case of *multiwinner elections*, however, already determining concrete possible goals of a coalition seems to be a non-trivial task: There may be exponentially many different outcomes which can be reached through strategic votes of the coalition members and each member could have its individual evaluation of these outcomes.

Multiwinner voting rules come up very naturally whenever one has to select from a large set of candidates a smaller set of "the best" candidates. Surprisingly, although at least as practically relevant as single-winner voting rules, the multiwinner literature is much less developed than the single-winner literature. In recent years (see a survey of Faliszewski et al. [26]), however, research into multiwinner voting rules, their properties, and algorithmic complexity grew significantly [1–5, 7, 10, 23, 25, 27, 33, 38, 42, 44, 45]. When selecting a group of winning candidates, various criteria can be interesting; namely, proportional representation, diversity, or excellence (see Elkind et al. [23]). We focus on the last scenario, where the goal is to select the best (say highest-scoring) group of candidates.

Aiming at excellence comes very naturally in the context of shortlisting, where the objective is a *short list* of candidates selected from an initial, much larger list of candidates. For instance, a human resource department wanting to fill a vacancy would select, from all job candidates, a short list of prospective applicants who should be further assessed to find the best fitting applicant. This example neatly illustrates the universal purpose of shortlisting, that is, saving effort at the same time increasing the quality of evaluating suitable candidates. Indeed, human resource departments will either waste a lot of time and

[1] http://www.classicfm.com/radio/hall-of-fame/.

effort interviewing every applicant in detail or they will significantly decrease the quality of interviewing to speed up the process unless they apply shortlisting beforehand.

A standard way of candidate selection in the context of shortlisting is to use scoring-based voting rules. We focus on the two most natural ones: SNTV (single non-transferable vote—each voter gives one point to one candidate) and ℓ-Bloc (each voter gives one point to each of ℓ different candidates, so SNTV is the same as 1-Bloc).[2] Obviously, for such voting rules it is trivial to determine the score of each individual candidate.

The main goal of our work is to model and understand coalitional manipulation in a computational sense—that is, to introduce a formal description of how a group of manipulators can influence the election outcome by casting strategic votes and whether it is possible to find an effective strategy for the manipulators to change the outcome in some desired way. We find studying coalitional manipulability from the computational complexity point of view relevant for two main reasons. First, in natural way we complement well-known work on manipulation for single-winner rules initiated by Bartholdi III et al. [8], coalitional manipulation for single-winner rules initiated by Conitzer et al. [17], and (non-coalitional) manipulation for multiwinner rules initiated by Meir et al. [38]. Second, we provide efficient algorithms that allow for experimental study of coalitional manipulation that might be interesting both for verifying how likely is or what is an impact of coalitional manipulation in practice (analogously to studies for the single-winner case [15, 19, 24, 36, 47]) and for interdisciplinary study on human's behavior when manipulating (like the one recently conducted for multiwinner elections by Scheuerman et al. [43]).

In coalitional manipulation scenarios, given full knowledge about other voters' preferences, one has a set of manipulative voters who want to influence the election outcome in a favorable way by casting their votes strategically. To come up with a useful framework for coalitional manipulation for multiwinner elections, we first have to identify the exact mathematical model and questions to be asked. A couple of straightforward extensions of coalitional manipulation for single-winner elections or (non-coalitional) manipulation for multiwinner elections do not fit. Extending the single-winner variant directly, one would probably assume that the coalition agrees on making a distinguished candidate part of the winners or that the coalition agrees on making a distinguished candidate group part of the winners. The former is unrealistic because in multiwinner settings one typically cares about more than just one candidate—especially in shortlisting it is natural that one wants rather some group of "similarly good" candidates to be winning instead of only one representative of such a group. The latter, that is, agreeing on a distinguished candidate group to be part of the winners is also problematic since there may be exponentially many "equally good" candidate groups for the coalition.[3] Notably, this was not a problem in the single-winner case; there, one can test for a successful manipulation towards each possible candidate avoiding an exponential increase of the running time (compared to the running time of such a test for a single candidate).

We address the aforementioned issue of modeling coalitional manipulation for multiwinner election by extending a single-manipulator model for multiwinner rules of Meir

[2] Although, in general, ℓ-Bloc does not satisfy *committee monotonicity* which is considered as a necessary condition for shortlisting [26], this rule seems quite frequent in practice—for example The Board of Research Excellence in Poland was elected using a variant of ℓ-Bloc [39].

[3] Indeed, assume a coalition has $x = 20$ favorite candidates that the coalition members equally prefer to be winning but the voting rule will select at most $k = 10$ of them. This means, when manipulating the election the coalition may support only one out of $\binom{x}{k} = 184756$ possible candidate groups.

et al. [38]. In their work, the manipulator specifies the utility of each candidate and the utility for a candidate group is obtained by adding up the utilities of each group member. We build on their idea and let each manipulator report the utility of each candidate. However, aggregating utilities for a coalition of manipulators (in other words, computing a collective utility of manipulators) becomes conceptually nontrivial—especially for a coalition of manipulators who have diverse utilities for single candidates but still have strong incentives to work together (e.g., as we have seen in our introductory example).

In fact, in our paper we only consider coalitions that are fixed, that is, irrespectively of how different the opinions of manipulators are, none of them leaves the coalition. For some situations and applications, this assumption might look too restrictive and unrealistic. In many situations, however, we believe there are good reasons for making this assumption. First, changing already existing coalitions in the real world usually requires a significant overhead (e.g., formal agreements and negotiations that cost both money and time) which makes such a change rather a last, not a first, resort. This holds true especially if a coalition is aimed at long-term benefits as, for example, strategic cooperations among firms or governments. Second, there are real-world cases where coalitions are forced, for example, in hierarchical administrative divisions (and their local governments) in countries. Third, computing a best possible manipulation for a given coalition is an important step in deciding whether it might be useful to actually form such a coalition in possible future. To wrap up, instead of focusing on coalitions dynamics (which is important work but not part of this paper), we rather concentrate on an analysis of a strength of, intuitively speaking, potential, fixed, or forced coalitions.

Our Contributions. We devise a formal description of coalitional manipulation in multiwinner elections arriving at a new, nontrivial model capturing two types of manipulators' attitudes and a few natural ways of utility aggregation. To this end, in our model, we distinguish between optimistic and pessimistic manipulators and we formalize aggregation of utilities in a utilitarian and an egalitarian way.

Using our model, we analyze the computational complexity of finding a successful manipulation for a coalition of voters, assuming elections under rules from the family of ℓ-Bloc voting rules. We show that, even for these fairly simple rules, the computational complexity of coalitional manipulation is diverse. In particular, we observe that finding a manipulation maximizing the utility of a worst-off manipulator (egalitarian aggregation) is NP-hard (regardless of the manipulators' attitude). This result stands in sharp contrast to the polynomial-time algorithms that we give for finding a manipulation maximizing the sum of manipulators' utilities (utilitarian aggregation). Additionally, we show how to circumvent the cumbersome NP-hardness for the egalitarian aggregation providing an (FPT) algorithm that is efficient for scenarios with few manipulators and few different values of utility that manipulators assign to agents. We survey all our computational complexity results in Table 1 (Sect. 6).

Related Work. To the best of our knowledge, there is no previous work on *coalitional* manipulation in the context of multiwinner elections. We refer to recent textbooks for an overview of the huge literature on single-winner (coalitional) manipulation [9, 16]. Most relevant to our work, Lin [35] showed that coalitional manipulation in single-winner elections under ℓ-Approval is solvable in linear time by a greedy algorithm. Meir et al. [38] introduced (non-coalitional) manipulation for multiwinner elections. While pinpointing manipulation for several voting rules as NP-hard, they showed that manipulation remains polynomial-time solvable for Bloc (which can be interpreted as a multiwinner equivalent of 1-Approval). Obraztsova et al. [42] extended the latter result for different tie-breaking strategies and identified further tractable special cases of multiwinner scoring rules but conjectured manipulation to be hard in general for (other) scoring rules. Summarizing, Bloc is simple but comparably well-studied, and hence we selected it as a showcase for our study of the presumably computationally harder *coalitional* manipulation.

As mentioned above, our model assumes an existing fixed coalition of manipulators. There is a significant amount of research on coalition dynamics and coalition forming in context of cooperative games. We refer to recent text books for an overview on this rich research area [22, 32]. Notably, our model is relevant for situations after some dynamic coalition forming process lead to a fixed coalition as well as during a coalition forming process when a group of agents wants to compute their potential utility from working together. Our aggregation models may be used for transferable utilities (utilitarian aggregation) as well as non-transferable utilities (egalitarian aggregation).

Another closely related model is the NP-hard multiwinner voting rule Minimax Approval Voting [13] which selects a group of candidates that maximizes the minimum satisfaction over all voters. This rule almost resembles the tie-breaking issue of our model for the egalitarian case and 0/1 utility values only. Each voter in a Minimax Approval Voting election can, however, approve an arbitrary number of candidates as opposed to the ℓ-Bloc rule that we consider, where each voter approves exactly ℓ candidates.

Organization. Section 2 introduces basic notation and formal concepts. In Sect. 3, we develop our model for coalitional manipulation in multiwinner elections. Its variants respect different ways of evaluating candidate groups (utilitarian vs. egalitarian) and two kinds of manipulators behavior (optimistic vs. pessimistic). In Sect. 4, we present algorithms and complexity results for computing the output of several tie-breaking rules that allow to model optimistic and pessimistic manipulators. In Sect. 5, we formally define the coalitional manipulation problem and explore its computational complexity using ℓ-Bloc as a showcase. We refer to our conclusion and Table 1 (Sect. 6) for a detailed overview of our findings.

2 Preliminaries

For a positive integer n, let $[n] := \{1, 2, \dots, n\}$. We use the toolbox of parameterized complexity [18, 21, 29, 40] to analyze the computational complexity of our problems in a fine-grained way. To this end, we always identify a *parameter* ρ that is typically a positive integer. We call a problem parameterized by ρ *fixed-parameter tractable* (in FPT) if it is solvable in $f(\rho) \cdot |I|^{O(1)}$ time, where $|I|$ is the size of a given instance encoding, ρ is the value of the parameter, and f is an arbitrary computable (typically super-polynomial) function. To preclude fixed-parameter tractability, we use an established complexity hierarchy of classes of parameterized problems, FPT $\subseteq$ W[1] $\subseteq$ W[2] $\subseteq \cdots \subseteq$ XP. It is widely believed that all inclusions are proper. The notions of hardness for parameterized classes are defined through parameterized reductions similar to classical polynomial-time many-one reductions—in this work, it suffices to ensure that the value of the parameter in the problem we reduce to depends only on the value of the parameter of the problem we reduce from. Occasionally, we use a *combined parameter* $\rho' + \rho''$ which is a more explicit way of expressing a parameter $\rho = \rho' + \rho''$.

An *election* (C, V) consists of a set C of m candidates and a multiset V of n votes. Votes are linear orders over C—for example, for $C = \{c_1, c_2, c_3\}$ we write $c_1 \succ_v c_2 \succ_v c_3$ to express that candidate c_1 is the most preferred and candidate c_3 is the least preferred according to vote v. We write $\succ$ if the corresponding vote is clear from the context.

A *multiwinner voting rule*[4] is a function that, given an election (C, V) and an integer $k \in [|C|]$, outputs a family of co-winning size-k subsets of C representing the co-winning *k-excellence-groups*. We decided to use the term k-excellence-group, abbreviated to *k-egroup*, following Debord [20].[5] Thus we emphasize our focus on shortlisting and bring our terminology closer to shortlisting (real-life) applications where the word "committee" traditionally rather refers to voters and not to candidates. For the sake of brevity, we use *egroup* if the size of an excellence-group is either not relevant or clear from the context.

We consider *scoring rules*—multiwinner voting rules that assign points to candidates based on their positions in the votes. By score (c), we denote the total number of points that candidate c obtains, and we use score $_{V'}(c)$ when restricting the election to a subset $V' \subset V$ of voters. A (multiwinner) scoring rule selects a family $\mathscr{X}$ of co-winning k-egroups with the maximum total sum of scores. It holds that $X \in \mathscr{X}$ if and only if for every candidate c in X and every candidate c' outside of X' it is true that score $(c) \geq$ score (c'). We focus on the family of ℓ-Bloc multiwinner voting rules, that is a family of scoring rules that assign, for each vote, one point to each of the top $\ell < |C|$ candidates.[6]

Example 1 Referring back to our introductory example, we have a set $C = \{b_1, b_2, m_1, m_2, o_1, o_2\}$ of candidates and a set $V = \{v^1_\mathrm{y}, v^2_\mathrm{y}, v^3_\mathrm{y}, v^1_\mathrm{b}, v^2_\mathrm{b}, v^1_\mathrm{m}, v^2_\mathrm{m}\}$ of voters. The voters v^1_y , v^2_y , and v^3_y represent the three junior jury members, whereas v^1_b , v^2_b and v^1_m , v^2_m represent, respectively, the Beethoven and Mozart enthusiasts among the senior jury members. In the example, we described a way of manipulating the election by the senior jury members which leads to selecting two traditional classical music pieces. There are several ways to illustrate this manipulation using our model. Below we present one of the possible sets of casted votes that represents the manipulated election:

$$\begin{aligned} v^1_\mathrm{y}, v^2_\mathrm{y}, v^3_\mathrm{y}: &\quad o_1 \succ o_2 \succ b_1 \succ b_2 \succ m_1 \succ m_2, \\ v^1_\mathrm{b}, v^2_\mathrm{b}, v^1_\mathrm{m}, v^2_\mathrm{m}: &\quad b_1 \succ m_1 \succ b_2 \succ m_2 \succ o_1 \succ o_2. \end{aligned}$$

Following the introductory example, we are choosing an egroup of size $k = 2$. Using the Bloc multiwinner voting rules (which coincides with our introductory example), the winning 2-egroup consist of candidates b_1 and m_1 . However, under the SNTV voting rule the situation would change, and the winners would be o_1 and b_1 . SNTV and Bloc alike output a single winning egroup in this example, and thus tie-breaking is ineffective.

To select a single k-egroup from the set of co-winning k-egroups one has to consider tie-breaking rules. A *multiwinner tie-breaking rule* is a mapping that, given an election and a family of co-winning k-egroups, outputs a single k-egroup. Among them, there is a set of natural rules that is of particular interest in order to model the behavior of manipulative

[4] One may argue that we should rather use the name multiwinner voting correspondence (instead of multiwinner voting rule) because the function returns a set of tied committees instead of a single committee (see, for example, the discussion on ties by Obraztsova et al. [42]; a textbook chapter on elections [9]; or a work by Barberà et al. [6], where voters instead of ordering the candidates are assumed to order all subsets of candidates of a given size). However, the phrasing we used is now well-established [11, 26] and without doubt most frequently used in the literature.

[5] We modified his term "elite" as we feel that it might carry negative connotations.

[6] The case where ℓ coincides with the size k of the egroup is typically referred to as Bloc; 1-Bloc corresponds to SNTV [38]. The case where $1 < \ell < k$ is also referred to as Limited Vote (or Limited Voting).

voters. Indeed, in addition to lexicographic and randomized tie-breaking, both *pessimistic* and *optimistic* tie-breaking rules have already been used to model the manipulator's behavior in case of a single manipulator [38, 42]. To model optimistic and pessimistic manipulators in a meaningful manner,[7] we use the model introduced by Obraztsova et al. [42] in which a manipulative voter v is described not only by the preference order $\succ_v$ of the candidates but also by a utility function $u : C \rightarrow \mathbb{N}$. To cover this in the tie-breaking process, *coalition-specific* tie-breaking rules get—in addition to the original election, the manipulators' votes, and the co-winning excellence-groups—the manipulators' utility functions in the input. The formal implementations of these rules and their properties are discussed in Sect. 3.2.

3 Model for coalitional manipulation

In this section, we formally define and explain our model and the respective variants, which we also motivate with short real-world examples. To this end, we discuss how we evaluate a k-egroup in terms of utility for a coalition of manipulators and introduce tie-breaking rules that model optimistic or pessimistic viewpoints of the manipulators.

3.1 Evaluating k-egroups

As already discussed in the introduction, one should not extend the model of coalitional manipulation for single-winner elections to multiwinner elections in the simplest way (e.g., by assuming that the manipulators agree on some distinguished candidate or on some distinguished egroup). Instead, we follow Meir et al. [38] and assume that we are given a utility function over the candidates for each manipulator and a utility level which, if achieved, indicates a successful manipulation.

Considering a collection of such utility functions there are several ways, each coming with distinct features, of computing the utility of an egroup. In the paper, we study the following three variants: *utilitarian*, *egalitarian*, and *candidate-wise egalitarian*.

In the *utilitarian variant* (considered by Meir et al. [38]) the utility of an egroup is the sum of utility values assigned by each manipulator to every candidate in the egroup. This is perhaps the most intuitive way of evaluating the utility of an egroup. Although it does not provide any guarantee on a single manipulator's utility after a manipulation (it might even happen that a single manipulator is significantly worse off compared to voting sincerely; see Example 2) the utilitarian variant is justified if the manipulators are able to "internally" compensate such loses, for example, by paying money to each other. For a real-world example, imagine an international company with branches (voters) scattered around the world considering actions to be taken (candidates) in order to reduce its carbon footprint. Seeking the most efficient solution, the company surveys the branches for pointing

[7] We cannot simply use ordinal preferences: Obraztsova et al. [42] observed that already in case of a single manipulator one cannot simply set the fixed lexicographic order of the manipulators' preferences (resp. the reverse of it) over candidates to model optimistic (resp. pessimistic) tie-breaking. For example, it is a strong restriction to assume that a manipulator would always prefer its first choice together with its fourth choice towards its second choice together with its third choice. It might be that only its first choice is really acceptable (in which case the assumption is reasonable) or that the first three choices are comparably good but the fourth choice is absolutely unacceptable (in which case the assumption is wrong).

the actions that reduce the footprint the most (utilities). A coalition of strategic voters can now be formed by all branches within a country that subsidizes companies that reduce the thier emission. Moreover, it is understandable that the office branches, whose footprint is rather small and thus so is the possible reduction, will rather support the actions that help the factory branches to reduce the footprint. To compensate, the coalition might decide to distribute more money from the country's benefit to the office branches.

Example 2 Consider the election $E = (C, V)$ where $C = \{b_1, b_2, m_1, m_2, o_1, o_2\}$ is a set of candidates and $V = \{v_1, v_2, v_3\}$ is the following multiset of three votes:

$$\begin{aligned} v_1, v_2: &\quad o_1 \succ o_2 \succ m_1 \succ m_2 \succ b_1 \succ b_2, \\ v_3: &\quad m_2 \succ m_1 \succ b_2 \succ b_1 \succ o_1 \succ o_2. \end{aligned}$$

Additionally, consider two manipulators, u_1 and u_2, that report utilities to the candidates as depicted in the table below.

$u(\cdot)$	b_1	b_2	m_1	m_2	o_1	o_2
u_1	10	5	4	0	0	0
u_2	1	2	5	7	0	0

Let us analyze the winning 2-egroup under the SNTV voting rule. Observe that if the manipulators vote sincerely, then together they give one point to b_1 and one to m_2 (one point from each manipulator). Combining the manipulators' votes with the non-manipulative ones, the winning 2-egroup consists of candidates o_1 and m_2 that both have score two; no other candidate has greater or equal score, so tie-breaking is unnecessary. The value of such a group is equal to seven according to the utilitarian evaluation variant. Manipulator u_2's utility is seven. However, both manipulators can do better by giving their points to candidate b_1. Then, the winners are candidates o_1 and b_1, giving the total utility of 11 (according to the utilitarian variant). Observe that in spite of growth of the total utility, the utility value gained by u_2, which is one, is lower than in the case of sincere voting.

The *egalitarian variant* comes in handy, for the scenarios where it is essential to guarantee a certain level of utility for every manipulator. Specifically, the utility of an egroup is the utility of a manipulator whose sum of utilities of candidates from the egroup is the smallest; thus, the egalitarian variant aims at maximizing this number. For a real-world example, imagine a parliament (voters) deciding about possible steps to reduce particulate matter (candidates). Seeking a way to reduce particulate matter below a certain threshold, a coalition of representatives from districts currently not meeting the threshold decides to vote strategically. Naturally, particulate matter reduction (utilities) are differently affected by different steps in the respective districts. The goal of the coalition is that even the worst district is below the threshold, which corresponds to egalitarian aggregation.

The *candidate-wise egalitarian variant* models again scenarios where, as in the utilitarian variant, the overall utilities from members of an egroup are summed up. The utility of the respective candidates, however, is aggregated in a pessimistic way, that is, assuming the lowest utility assigned by any member of the coalition is taken into the sum. For a real-world example, imagine a parliament (voters) deciding on different actions (candidates) to support the economy after a crisis. Representatives of the same party naturally work together as a coalition of strategic voters. Each representative has a different prediction from their own group of experts on the effectiveness of the actions (utilities), which at the end will sum up. To be on the safe side, the coalition decides to take the most pessimistic evaluation any expert group makes for a respective candidate into account for their decision, which corresponds to our candidate-wise egalitarian aggregation variant.

We formalize the described variants of k-egroup evaluation (for r manipulators) with Definition 1.

Definition 1 Given a set of candidates C, a k-egroup $S \subseteq C$, and a family of manipulator utility functions $U = \{u_1, u_2, \dots, u_r\}$ where $u_i : C \to \mathbb{N}$, we consider the following functions:

- $\mathrm{util}_U(S) := \sum_{u \in U} \sum_{c \in S} u(c)$,
- $\mathrm{egal}_U(S) := \min_{u \in U} \sum_{c \in S} u(c)$,
- $\mathrm{candegal}_U(S) := \sum_{c \in S} \min_{u \in U} u(c)$.

Intuitively, these functions determine the utility of a k-egroup S according to, respectively, the utilitarian, the candidate-wise egalitarian, and the egalitarian variant of evaluating S by a group of r manipulators (identifying manipulators with their utility functions). We omit subscript U when U is clear from the context. To illustrate Definition 1 we apply it in Example 3.

Example 3 Consider our example set of candidates $C = \{b_1, b_2, m_1, m_2, o_1, o_2\}$ and two manipulators u_1, u_2 whose utility functions over the candidates are depicted in the table below.

$u(\cdot)$	b_1	b_2	m_1	m_2	o_1	o_2
u_1	10	5	4	0	0	0
u_2	1	2	5	7	0	0

Then, evaluating the utility of 2-egroup $S = \{b_1, m_1\}$ applying the three different evaluation variants gives:

- $\mathrm{util}\,(S) = (10 + 4) + (1 + 5) = 20$,
- $\mathrm{egal}\,(S) = \min\{(10 + 4);(1 + 5)\} = 6$,
- $\mathrm{candegal}\,(S) = \min\{10, 1\} + \min\{4, 5\} = 5$.

Analyzing Example 3, we observe that we can compute the utilitarian value of egroup S by summing up the overall utilities that each candidate in S contributes to all manipulators; for instance candidate b_1 always contributes the utility of $11 = 10 + 1$ to the manipulators, independently of other candidates in the egroup. Following this observation, instead of coping with a collection of utility function, we can "contract" all manipulator's functions to a single function. The new function assigns each candidate a utility value equal to the sum of utilities that the contracted functions assign to this candidate. Analogously, we can deal with the candidate-wise egalitarian variant by taking the minimum utility associated to each candidate as the utility of this candidate in a new function. Thus, in both variants, we can consider a single utility function instead of a family of functions. The conclusion from the above discussion is summarized in the following observation.

Observation 1 Without loss of generality, one can assume that there is a single utility function over the candidates under the utilitarian or candidate-wise egalitarian evaluation.

Proof Consider a multiset of manipulator utility functions $U = \{u_1, u_2, \dots, u_r\}$ and a k-egroup S. For the utilitarian variant, create a new utility function u' that assigns to each candidate the sum of utilities given to this candidate by all manipulators; that is, $u'(c) := \sum_{i\in[r]} u_i(c)$ for all $c \in C$. Since for each candidate function u' returns the sum of utilities given to a candidate by all functions from family U, it holds that $\mathrm{util}_U(S) = \sum_{i\in[r]} \sum_{c\in S} u_i(c) = \sum_{c\in S} \sum_{i\in[r]} u_i(c) = \sum_{c\in S} u_i'(c)$.

We follow a similar strategy proving Observation 1 for the candidate-wise egalitarian evaluation. We introduce a function u' defined as $u'(c) := \min_{u\in U} u(c)$ for each candidate $c \in C$. Naturally, $\mathrm{candegal}_U(S) = \sum_{c\in S} \min_{u\in U} u(c) = \sum_{c\in S} u'(c)$. ☐

3.2 Breaking ties

In this work, we consider three different ways of breaking ties which are all established already in the literature: lexicographic tie-breaking, optimistic tie-breaking and pessimistic tie-breaking. Lexicographic tie-breaking is standard in the great majority of works about single-winner or multiwinner voting. Even though the usual motivation for it is to obtain simpler models, still there are situations in which lexicographic tie-breaking is what one can encounter in reality (e.g., if ties are broken by age). Nevertheless, our main motivation to consider lexicographic tie-breaking is to have a simple baseline, also allowing us easily compare our findings with other works. The two other tie-breaking mechanisms we consider are optimistic and pessimistic tie-breaking. They are used to model the two most natural types of manipulators: Optimistic manipulators would try to manipulate whenever there is a chance that ties are broken so that they are better off compared to truthful voting. Pessimistic manipulators will only manipulate when they are better off compared to truthful voting no matter how ties will be broken.

Before formally defining our tie-breaking rules, we briefly discuss some necessary notation and central concepts. Consider an election (C, V), a size k for the egroup to be chosen, and a scoring-based multiwinner voting rule $\mathscr{R}$. We can partition the set of candidates C into three sets C^+, P, and C^- as follows: The set C^+ contains the *confirmed candidates*, that is, candidates that are in all co-winning k-egroups. The set P contains the *pending candidates*, that is, candidates that are only in some co-winning k-egroups. The set C^- contains the *rejected candidates*, that is, candidates that are in no co-winning k-egroup. Observe that $|C^+| \leq k$, $|C^+ \cup P| \geq k$, and that every candidate from $P \cup C^-$ receives fewer points than every candidate from C^+. Additionally, all candidates in P receive the same number of points.

We define the following families of tie-breaking rules which are considered in this work. In order to define optimistic and pessimistic rules, we assume that in addition to C^+, P, and k, we are given a family of utility functions which are used to evaluate the k-egroups as discussed in Sect. 3.1. We call such a quadruple a *tie-breaking perspective*.

Lexicographic. $\mathscr{F}_{\text{lex}}$. A tie-breaking rule $\mathscr{F}$ belongs to $\mathscr{F}_{\text{lex}}$ if and only if ties are broken lexicographically with respect to some predefined order $>_{\mathscr{F}}$ of the candidates from C. That is, $\mathscr{F}$ selects all candidates from C^+ and the top $k - |C^+|$ candidates from P with respect to $>_{\mathscr{F}}$.

Optimistic. $\mathscr{F}^{\text{eval}}_{\text{opt}}$, eval $\in \{\text{util}, \text{egal}, \text{candegal}\}$. A tie-breaking rule belongs to $\mathscr{F}^{\text{eval}}_{\text{opt}}$ if and only if it always selects some k-egroup S such that $C^+ \subseteq S \subseteq (C^+ \cup P)$ and there is no other k-egroup S' with $C^+ \subseteq S' \subseteq (C^+ \cup P)$ and $\text{eval}(S') > \text{eval}(S)$.

Pessimistic. $\mathscr{F}^{\text{eval}}_{\text{pess}}$, eval $\in \{\text{util}, \text{egal}, \text{candegal}\}$. A tie-breaking rule belongs to $\mathscr{F}^{\text{eval}}_{\text{pess}}$ if and only if it always selects some k-egroup S such that $C^+ \subseteq S \subseteq (C^+ \cup P)$ and there is no other k-egroup S' with $C^+ \subseteq S' \subseteq (C^+ \cup P)$ and $\text{eval}(S') < \text{eval}(S)$.

We remark that the definitions above come in two, substantially different variants. For each lexicographic tie-breaking rule, there is always exactly one egroup that will be selected by the rule for a particular set of pending set candidates. However, it is not the case for the families of pessimistic and optimistic families of rules. In fact, there might be many possible egroups whose value, computed in terms of a respective evaluation variant, is exactly the same. Such a feature seems to contradict the idea of a tie-breaking rule that should not, by itself, introduce ties again. However, we argue that choosing arbitrary equally-valued ("tied") egroup is a proper way to circumvent this problem. Indeed, according to a particular evaluation all egroups with the same value are indistinguishable from each other.

3.3 Limits of lexicographic tie-breaking

From the above discussion, we conclude that lexicographic tie-breaking is straightforward in the case of scoring-based multiwinner voting rules. Basically any subset of the desired cardinality from the set of pending candidates can be chosen; in particular, the best pending candidates with respect to the given order can be chosen. Although this sounds naturally at a glance, for many other prominent multiwinner voting rules, such as Chamberlin-Courant [14] or STV [46], not every subset of the desired cardinality from the set of pending candidates can be chosen.

It remains to be clarified whether one can find a reasonable order of the pending candidates in order to model optimistic or pessimistic tie-breaking rules in a simple way. We show that this is possible for every $\mathscr{F}^{\text{eval}}_{\text{bhav}}$, with every combination of eval $\in \{\text{util}, \text{candegal}\}$ and bhav $\in \{\text{opt}, \text{pess}\}$, using the fact that in these cases we can safely assume that there is a single utility function (see Observation 1). On the contrary, there is a counterexample for eval $=$ egal and bhav $\in \{\text{opt}, \text{pess}\}$. On the way to prove these claims we need to formally define what it means that one family of tie-breaking rules can be used to simulate another family of tie-breaking rules. To this end, we introduce the concept of equivalence between tie-breaking perspectives.

Definition 2 Let $\{\boxed{\mathbb{C}}, \boxed{\mathbb{P}}, \boxed{\mathbb{K}}, \boxed{\mathbb{U}}\}$ be a set of attribute tags (treated exactly as usual characters), C be a fixed set of candidates, $X = (C^+, P, k, U)$ and $\hat{X} = (\hat{C}^+, \hat{P}, \hat{k}, \hat{U})$ be tie-breaking perspectives over C, where $C^+ \subset C$ and $P \subseteq C$. Then, X and X' are:

- $\boxed{\mathbb{C}}$-*equivalent* if and only if $C^+ = \hat{C}^+$,
- $\boxed{\mathbb{P}}$-*equivalent* if and only if $P = \hat{P}$,
- $\boxed{\mathbb{K}}$-*equivalent* if and only if $k = \hat{k}$, and
- $\boxed{\mathbb{U}}$-*equivalent* if and only if $U = \hat{U}$.

Additionally, for a subset $\mathscr{P}$ of symbols $\{\boxed{\mathbb{C}}, \boxed{\mathbb{P}}, \boxed{\mathbb{K}}, \boxed{\mathbb{U}}\}$, we say that X and X' are $\mathscr{P}$-equivalent if and only if, for each symbol $\mathbb{S} \in \mathscr{P}$, they are $\mathbb{S}$-equivalent.

One can easily verify that the equivalence notions from Definition 2 indeed meet the requirements of equivalence relations; thus, we can speak of equivalence classes of tie-breaking perspectives (with respect to a given equivalence notion).

Definition 3 For a nonempty set $\{\boxed{\mathbb{C}}, \boxed{\mathbb{P}}, \boxed{\mathbb{K}}, \boxed{\mathbb{U}}\}$ of attribute tags, and for two tie-breaking families $\mathscr{F}$ and $\mathscr{F}'$, we say that $\mathscr{F}$ can *$\mathscr{P}$-simulate* $\mathscr{F}'$ if, for every nonempty set of candidates C and for every tie-breaking perspective X over C, there exists a rule $F \in \mathscr{F}$ such that for each tie-breaking perspective in the equivalence class of X according to $\mathscr{P}$-equivalence there exists a rule $F' \in \mathscr{F}'$ such that F and F' yield the same output for this perspective. We call rule F an *$\mathscr{P}$-simulator*.

At first glance, Definition 3 might seem overcomplicated. However, it is tailored to grasp different degrees of simulation possibilities. On the one hand, one can always find a lexicographic order and use it for breaking ties if all of the following are known: confirmed candidates, pending candidates, utility functions, and the size of an egroup; formally, the family of lexicographic tie-breaking rules $\{\boxed{\mathbb{C}}, \boxed{\mathbb{P}}, \boxed{\mathbb{K}}, \boxed{\mathbb{U}}\}$-simulates every other family. Thus, one needs some flexibility in the definition of simulation for it to be non-trivial. On the other hand, it is somewhat clear that without fixing the utility functions, one cannot simulate optimistic or pessimistic tie-breaking rules. In other words, we have the following observation.

Observation 2 The family of lexicographic tie-breaking rules does not $\{\boxed{\mathbb{C}}, \boxed{\mathbb{P}}, \boxed{\mathbb{K}}\}$-simulate $\mathscr{F}^{\text{eval}}_{\text{bhav}}$.

Proof Suppose $k = 1$, $C^+ = \emptyset$, and $P = \{b_1, b_2\}$; that is, we are going to select either b_1 or b_2 who are tied. Let us fix a family $U = \{u_1\}$ of utility functions such that $u_1(b_1) = 1$ and $u_1(b_2) = 0$. For the family U of utility functions clearly $\mathscr{F}^{\text{eval}}_{\text{opt}}$ selects candidate b_1 . Now, consider a family $U' = \{u'_1\}$ of utility functions where u'_1 assigns utility one to candidate b_2 and zero otherwise. For this family, $\mathscr{F}^{\text{eval}}_{\text{opt}}$ selects candidate b_2. This means that we cannot find a $\{\boxed{\mathbb{C}}, \boxed{\mathbb{P}}, \boxed{\mathbb{K}}\}$-simulator F from family $\mathscr{F}_{\text{lex}}$ of tie-breaking rules because in the first case F would have to choose b_1 and in the second case b_2 would have to be chosen. This is impossible using a single preference order over $\{b_1, b_2\}$. Similar families of functions (obtained by exchanging each one with zero and vice versa) yield a proof for $\mathscr{F}^{\text{eval}}_{\text{pess}}$ as well. □

Next, we show that for some cases it is sufficient to fix just the utility functions in order to simulate optimistic or pessimistic tie-breaking rules (see Proposition 1). For other cases, however, one *has* to fix all of the following: confirmed candidates, pending candidates, utility functions, and the size of an egroup (see Proposition 2).

Proposition 1 *For every* eval $\in \{$ util , candegal $\}$ *and* bhav $\in \{$opt, pess$\}$, *the family* $\mathscr{F}_{\text{lex}}$ *can* $\{\boxed{\mathbb{U}}\}$*-simulate* $\mathscr{F}^{\text{eval}}_{\text{bhav}}$; *additionally, for m candidates and r utility functions, a* $\{\boxed{\mathbb{U}}\}$*-simulator $F \in \mathscr{F}_{\text{lex}}$ can be found in $O(m \cdot (r + \log m))$ time.*

Proof Recall from Observation 1 that if eval $\in \{$ util , candegal $\}$, then there exists a single utility function u' that is equivalent to the given family of utility functions (with respect to the evaluation of egroup utilities). Hence, we compute such a function u' in $O(m \cdot r)$ time precisely following its definition as in the proof of Observation 1. We say an order $>_{\mathscr{F}}$ of the candidates is *consistent* with some utility function u if $c >_{\mathscr{F}} c'$ implies $u(c) \geq u(c')$ for optimistic tie-breaking and $c >_{\mathscr{F}} c'$ implies $u(c) \leq u(c')$ for pessimistic tie-breaking. Any lexicographic tie-breaking rule defined by an order $>_{\mathscr{F}}$ that is consistent with the utility function u' simulates $\mathscr{F}^{\text{eval}}_{\text{bhav}}$. We compute a consistent order by sorting the candidates according to u' in $O(m \cdot \log m)$ time. □

Proposition 1 describes a strong feature of optimistic utilitarian and candidate-wise egalitarian tie-breaking and their pessimistic variants. Intuitively, the proposition says that for these tie-breaking mechanisms one can compute a respective linear order of candidates. Then one can forget all the details of the initial tie-breaking mechanism and use the order to determine winners. The order can be computed even without knowing the details of an election. Unfortunately, the simulation of pessimistic and optimistic *egalitarian* tie-breaking turns out to be more complicated.

Proposition 2 *For each nonempty set $\mathscr{P} \subseteq \{\boxed{\mathbb{C}}, \boxed{\mathbb{P}}, \boxed{\mathbb{K}}, \boxed{\mathbb{U}}\}$ of size at most three, the lexicographic tie-breaking family of rules does not $\mathscr{P}$-simulate $\mathscr{F}^{\text{egal}}_{\text{bhav}}$ assuming* bhav $\in \{\text{opt}, \text{pess}\}$.

Proof From Observation 2 we already know that the family of lexicographic tie-breaking rules cannot $\{\boxed{\mathbb{C}}, \boxed{\mathbb{P}}, \boxed{\mathbb{K}}\}$-simulate the family of egalitarian pessimistic tie-breaking rules or the family of egalitarian optimistic tie-breaking rules.

Next, we build one counterexample for each of the remaining size-three subsets of $\{\boxed{\mathbb{C}}, \boxed{\mathbb{P}}, \boxed{\mathbb{K}}, \boxed{\mathbb{U}}\}$ to show our theorem. To this end, let us fix a set of candidates $C = \{b_1, b_2, m_1, m_2, o_1, o_2\}$ (compatible with our running example) and a family $U = \{u_1, u_2\}$ of utility functions as depicted in the table below.

$u(\cdot)$	b_1	b_2	m_1	m_2	o_1	o_2
u_1	10	5	4	0	0	0
u_2	1	2	5	7	0	0

First, we prove that the family $\mathscr{F}_{\text{lex}}$ cannot $\{\boxed{\mathbb{C}}, \boxed{\mathbb{P}}, \boxed{\mathbb{U}}\}$-simulate $\mathscr{F}^{\text{egal}}_{\text{bhav}}$ for any bhav $\in \{\text{opt}, \text{pess}\}$. Let us fix $C^+ = \emptyset$, $P = C \setminus \{o_1, o_2\}$. We consider the optimistic variant of egalitarian tie-breaking for $k = 1$, so we are searching for a 1-egroup. Looking at the values of U, we see that candidate m_1 gives the best possible egalitarian evaluation value which is four. This means that a $\{\boxed{\mathbb{C}}, \boxed{\mathbb{P}}, \boxed{\mathbb{U}}\}$-simulator $F \in \mathscr{F}_{\text{lex}}$ has to use an order where m_1 precedes both b_1 and m_2. However, it turns out that if we set $k = 2$, then the best 2-egroup consists exactly of candidates b_1 and m_2. This leads to a contradiction because now candidates b_1 and m_2 should precede m_1 in F's lexicographic order. Consequently, family $\mathscr{F}_{\text{lex}}$ does not $\{\boxed{\mathbb{C}}, \boxed{\mathbb{P}}, \boxed{\mathbb{U}}\}$-simulate $\mathscr{F}^{\text{egal}}_{\text{opt}}$. Using the same values of utility functions and the same sequence of the values of k we get a proof for the pessimistic variant of egalitarian evaluation.

Second, we prove that the family $\mathscr{F}_{\text{lex}}$ cannot $\{\boxed{\mathbb{P}}, \boxed{\mathbb{K}}, \boxed{\mathbb{U}}\}$-simulate $\mathscr{F}^{\text{egal}}_{\text{bhav}}$ for bhav $\in \{\text{opt}, \text{pess}\}$. This time, we fix $P = C \setminus \{o_1, o_2\}$, $k = 2$. We construct the first case by setting $C^+ = \{o_1\}$. Using the fact that in both functions candidate o_1 has utility zero, we choose exactly the same candidate as in the proof of $\{\boxed{\mathbb{C}}, \boxed{\mathbb{P}}, \boxed{\mathbb{U}}\}$-simulation for the case $k = 1$; that is, for the optimistic variant, the winning 2-egroup is m_1 and o_1. Consequently,

m_1 precedes b_1 and m_2 in the potential $\{P, k, U\}$-simulator's lexicographic order. Towards a contradiction, we set $C^+ = \emptyset$. The situation is exactly the same as in the proof of the $\{\boxed{\mathbb{C}}, \boxed{\mathbb{P}}, \boxed{\mathbb{U}}\}$-simulation case. Now, the winning 2-egroup consists of b_1 and m_2 which ends the proof for the optimistic case. By almost the same argument, the result holds for the pessimistic variant.

Finally, we prove that the family $\mathscr{F}_{\text{lex}}$ cannot $\{\boxed{\mathbb{C}}, \boxed{\mathbb{K}}, \boxed{\mathbb{U}}\}$-simulate $\mathscr{F}^{\text{egal}}_{\text{bhav}}$ for bhav $\in \{\text{opt}, \text{pess}\}$. We fix $C^+ = \emptyset$, $k = 2$. For the first case we pick $P = \{b_2, m_1, m_2\}$. The best egalitarian evaluation happens for the 2-egroup consisting of b_2 and m_1. This imposes that, in the potential $\{\boxed{\mathbb{C}}, \boxed{\mathbb{K}}, \boxed{\mathbb{U}}\}$-simulator's order, b_2 and m_1 precede the remaining candidates (in particular, m_1 precedes m_2). However, for $P = C$ the best 2-egroup changes to that consisting of b_1 and m_2 which gives a contradiction (m_2 precedes m_1). As in the previous cases, the same argument provides a proof for the pessimistic variant. $\square$

Proposition 2 implies that pessimistic and optimistic egalitarian tie-breaking cannot be simulated without having full knowledge about an election. In terms of computational complexity, however, finding winners for pessimistic egalitarian tie-breaking remains tractable whereas the same task for optimistic egalitarian tie-breaking is intractable. We devote the next section to show this dichotomy as well as to establish computational hardness of computing winners for the other introduced tie-breaking rules.

4 Complexity of tie-breaking

It is natural to ask whether the tie-breaking rules proposed in Sect. 3.2 are practical in terms of their computational complexity. If not, then there is little hope for effective and efficient coalitional manipulation because tie-breaking might be an inevitable subtask to be solved by the manipulators. Indeed, manipulators might not be "powerful" enough to secure victory of their desired egroup completely avoiding tie-breaking.

Clearly, we can perform every lexicographic tie-breaking rule that is defined through some predefined order of the candidates in linear time. Hence, we focus on the rules that model optimistic or pessimistic manipulators. To this end, we analyze the following computational problem.

$\mathscr{F}^{\text{eval}}_{\text{bhav}}$-Tie-Breaking ($\mathscr{F}^{\text{eval}}_{\text{bhav}}$-TB)

eval $\in$ { util , egal , candegal }, bhav $\in$ { opt , pess }

Input: A set of candidates C partitioned into a set P of pending candidates and a set C^+ of confirmed candidates, the size k of the excellence-group such that $|C^+| < k < |C|$, a family of manipulator utility functions $U = \{u_1, u_2, \ldots, u_r\}$ where $u_i : C \to \mathbb{N}$, and a non-negative, integral evaluation threshold q.

Question: Is there a size-k set $S \subseteq C$ such that S is selected according to $\mathscr{F}^{\text{eval}}_{\text{bhav}}$, $C^+ \subseteq S$, and eval $(S) \geq q$?

Naturally, we may assume that the number of candidates and the number of utility functions are polynomially upper-bounded in the size of the input. However, both the evaluation threshold and the utility function values are encoded in binary.

Note that an analogous problem has not been considered for single-winner elections. The reason behind this is that, for single-winner elections, optimistic and pessimistic tie-breaking rules can be easily simulated by lexicographic tie-breaking rules. To obtain them, it is sufficient to order the candidates with respect to their value to manipulators, computed separately for every candidate. However, one cannot simply apply this approach for egroups, because there might be exponentially many different egroups to consider. Even if this exponential blow-up were acceptable, it would still be unclear how to derive an order of candidates from the computed values of egroups. Yet, using a different technique, we can simulate tie-breaking in multiwinner elections with a lexicographic tie-breaking rule for several variants of evaluation.

4.1 Utilitarian and candidate-wise egalitarian: tie-breaking is easy

As a warm-up, we observe that tie-breaking can be applied and performed efficiently if the k-egroups are evaluated according to the utilitarian or candidate-wise egalitarian variant. The corresponding result follows almost directly from Proposition 1.

Corollary 1 *Let m denote the number of candidates and r denote the number of manipulators. Then one can solve $\mathscr{F}^{\text{eval}}_{\text{bhav}}$ -Tie-Breaking in $O(m \cdot (r + \log m))$ time for* eval $\in$ { util , candegal }, bhav $\in$ {opt, pess}.

Proof The algorithm works in two steps. First, compute a lexicographic tie-breaking rule $\mathscr{F}_{\text{lex}}$ that simulates $\mathscr{F}^{\text{eval}}_{\text{bhav}}$ in $O(m \cdot (r + \log m))$ time as described in Proposition 1. Second, apply tie-breaking rule $\mathscr{F}_{\text{lex}}$, and evaluate the resulting k-egroup in $O(k \cdot r)$ time. The running time of applying a lexicographic tie-breaking rule is linear with respect to the input length (see Sect. 3.3). □

4.2 Egalitarian: being optimistic is hard

In this subsection, we consider the optimistic and pessimistic tie-breaking rules when applied for searching a k-egroup evaluated according to the egalitarian variant. First, we show that applying and evaluating egalitarian tie-breaking is computationally easy for pessimistic manipulators but computationally intractable for optimistic manipulators even if the size of the egroup is small. Being pessimistic, the main idea is to "guess" the manipulator that is least satisfied and select the candidates appropriately. We show the computational worst-case hardness of the optimistic case via a reduction from the W[2]-complete Set Cover problem parameterized by solution size [21].

Theorem 1 *Let m denote the number of candidates, r denote the number of manipulators, q denote the evaluation threshold, and k denote the size of an egroup. Then one can solve $\mathscr{F}^{\text{egal}}_{\text{pess}}$ -Tie-Breaking in $O(r \cdot m \log m)$ time, but $\mathscr{F}^{\text{egal}}_{\text{opt}}$ -Tie-Breaking is* NP*-hard and* W[2] *-hard when parameterized by k even if $q = 1$ and every manipulator only gives either utility one or zero to each candidate.*

Proof For the pessimistic case, it is sufficient to "guess" the least satisfied manipulator x by iterating through r possibilities. Then, select $k - |C^+|$ pending candidates with the smallest total utility for this manipulator in $O(m \log m)$ time. Finally, comparing the k-egroup with the worst minimum satisfaction over all manipulators to the given lower bound q on satisfaction level solves the problem.

We prove the hardness for the optimistic case reducing from the W[2]-hard Set Cover problem which, given a collection $\mathscr{S} = \{S_1, S_2, \dots, S_m\}$ of subsets of universe $X = \{x_1, x_2, \dots, x_n\}$ and an integer h,[8] asks whether there exists a family $\mathscr{S}' \subseteq \mathscr{S}$ of size at most h such that $\bigcup_{S \in \mathscr{S}'} S = X$. Let us fix an instance $I = (X, \mathscr{S}, h)$ of Set Cover. To construct an $\mathscr{F}^{\text{egal}}_{\text{opt}}$ -Tie-Breaking instance, we introduce pending candidates $P = \{c_1, c_2, \dots, c_m\}$ representing subsets in $\mathscr{S}$ and manipulators $u_1, u_2, \dots, u_n$ representing elements of the universe. Note that there are no confirmed and rejected candidates. Each manipulator u_i gives utility one to candidate c_j if set S_j contains element x_i and zero otherwise. We set the excellence-group size $k := h$ and the threshold q to be 1.

Observe that if there is a size-k subset $P' \subseteq P$ such that $\min_{i \in [n]} \sum_{c' \in P'} u_i(c') \geq 1$, then there exists a family $\mathscr{S}' = \{S_j : c_j \in P'\}$—consisting of the sets represented by candidates in P'—such that each element of the universe belongs to the set $\bigcup_{S \in \mathscr{S}'} S$. For the reverse direction, assume there is a family $\mathscr{S}'$ such that each element of the universe belongs to the set $\bigcup_{S \in \mathscr{S}'} S$. Then, consider the size-k subset $P' = \{c_j : S_j \in \mathscr{S}'\}$. Assume towards a contradiction that there is an $i \in [n]$ with $\sum_{c' \in P'} u_i(c') < 1$. Then, by construction of the utility functions, it must hold that $x_i \notin \bigcup_{S \in \mathscr{S}'} S$—a contradiction. Thus, $\min_{i \in [n]} \sum_{c' \in P'} u_i(c') \geq 1$.

Since Set Cover is NP-hard and W[2]-hard with respect to parameter h, we obtain that our problem is also NP-hard and W[2]-hard when parameterized by the size k of an excellence-group. □

Inspecting the W[2]-hardness proof of Theorem 1, we learn that a small egroup size (alone) does not make $\mathscr{F}^{\text{egal}}_{\text{opt}}$ -Tie-Breaking computationally tractable even for very simple utility functions. Next, using a parameterized reduction from the W[1]-complete Multicolored Clique problem [28], we show that there is still no hope for fixed-parameter tractability (under standard assumptions) even for the combined parameter "number of manipulators and egroup size"; intuitively, this parameter covers situations where few manipulators are going to influence an election for a small egroup.

Theorem 2 *Let k denote the size of an egroup and r denote the number of manipulators. Then, parameterized by $r + k$, $\mathscr{F}^{\text{egal}}_{\text{opt}}$ -Tie-Breaking is* W[1]-*hard.*

Proof We describe a parameterized reduction from the W[1]-hard Multicolored Clique problem [28]. In this problem, given an undirected graph $G = (V, E)$, a non-negative integer h, and a vertex coloring $\phi : V \to \{1, 2, \dots, h\}$, we ask whether graph G admits a colorful h-clique, that is, a size-h vertex subset $Q \subseteq V$ such that the vertices in Q are pairwise adjacent and have pairwise distinct colors. Without loss of generality, we assume that the number of vertices of each color is the same; to be referred to as y in the following. Let (G, ϕ), $G = (V, E)$, be a Multicolored Clique instance. Let $V(i) = \{v^i_1, v^i_2, \dots, v^i_y\}$ denote the set of vertices of color $i \in [h]$, and let $E(i,j) = \{e^{i,j}_1, e^{i,j}_2, \dots, e^{i,j}_{|E(i,j)|}\}$, defined for $i, j \in [h]$, $i < j$, denote the set of edges that connect a vertex of color i to a vertex of color j. We

[8] Note that we indeed reuse the variable names m and n as there will be a one-to-one correspondence between subsets and candidates as well as between elements and voters so that the usual meaning of m and n is preserved.

disregard possible edges between vertices of the same color as they cannot be part of any multicolored clique anyway.

Candidates. We create one confirmed candidate c^* and $|V| + |E|$ pending candidates. More precisely: for each $\ell \in [y]$, we create one *vertex candidate* a^i_ℓ for each vertex $v^i_\ell \in V(i)$, $i \in [h]$ and for each $i,j \in [h]$ such that $i < j$ we create one *edge candidate* $b^{i,j}_t$ for each edge $e^{i,j}_t \in E(i,j)$, $t \in [|E(i,j)|]$. We set the size k of the egroup to $h + \binom{h}{2} + 1$ and set the evaluation threshold $q := y + 1$. Next, we describe the manipulators and explain the high-level idea of the construction.

Manipulators and main idea. Our construction will ensure that there is a k-egroup X with $c^* \in X$ and $\mathrm{egal}(X) \geq q$ if and only if X contains h vertex candidates and $\binom{h}{2}$ edge candidates that encode a colorful h-clique. To this end, we introduce the following manipulators.

1. For each color $i \in [h]$, there is a *color manipulator* μ_i ensuring that the k-egroup contains a vertex candidate $a^i_{z_i}$ corresponding to a vertex of color i. Herein, variable z_i denotes the id of the vertex candidate (resp. vertex) that is *selected* for color i.
2. For each $i,j \in [h]$ such that $i < j$, there is one *color pair manipulator* $\mu_{i,j}$ ensuring that the k-egroup contains an edge candidate $b^{i,j}_{z_{i,j}}$ corresponding to an edge connecting vertices of colors i and j. Herein, variable $z_{i,j}$ denotes the id of the edge candidate (resp. edge) that is *selected* for color pair $\{i,j\}$, $i < j$.
3. For each $i,j \in [h]$ such that $i \neq j$, there are two *verification manipulators* $\nu_{i,j}$, $\nu'_{i,j}$ ensuring that vertex $v^i_{z_i}$ is incident to edge $e^{i,j}_{z_{i,j}}$ if $i < j$ or incident to edge $e^{j,i}_{z_{j,i}}$ otherwise.

It is easy to verify that if there exists a k-egroup in agreement with the description in the previous three points, then this k-egroup must encode a colorful h-clique.

Utility functions. Let us now describe how we can guarantee the intended roles of the manipulators introduced in points 1 to 3 above using utility functions.

1. Color manipulator μ_i, $i \in [h]$, has utility y for the confirmed candidate c^*, utility one for each candidate corresponding to a vertex of color i, and utility zero for the remaining candidates.
2. Color pair manipulator $\mu_{i,j}$, $i,j \in [h]$, $i < j$, has utility y for the confirmed candidate c^*, utility one for each candidate corresponding to an edge connecting a vertices of colors i and j, and utility zero for the remaining candidates.
3. Verification manipulator $\nu_{i,j}$, $i,j \in [h]$, $i \neq j$, has utility ℓ for candidate a^i_ℓ, $\ell \in [y]$, utility $q - \ell$ for each candidate corresponding to an edge that connects vertex v^i_ℓ to a vertex of color j, and utility zero for the remaining candidates.
4. Verification manipulator $\nu'_{i,j}$, $i,j \in [h]$, $i \neq j$, has utility $q - \ell$ for candidate a^i_ℓ, $\ell \in [y]$, utility ℓ for each candidate corresponding to an edge that connects vertex v^i_ℓ, to a vertex of color j, and utility zero for the remaining candidates.

Correctness. We argue that the graph G admits a colorful clique of size h if and only if there is a k-egroup X with $c^* \in X$ and $\mathrm{egal}(X) \geq q$.

Suppose that there exists a colorful clique H of size h. Create the k-egroup X as follows. Start with $\{c^*\}$ and add every vertex candidate that corresponds to some vertex of H and every edge candidate that corresponds to some edge connecting vertices of H. Each color manipulator and color pair manipulator receives total utility $y+1$, because H contains, by definition, one vertex of each color and one edge connecting two vertices for each color pair. It is easy to verify that the verification manipulator $v_{i,j}$ must receive utility ℓ from a vertex candidate and utility $q-\ell$ from an edge candidate and that the verification manipulator $v'_{i,j}$ must receive utility $q-\ell$ from a vertex candidate and utility ℓ from an edge candidate. Thus, $\text{egal}\,(X)=q=y+1$.

Suppose that there exists a k-egroup $X \subseteq C$ such that $\text{egal}\,(X) \geq q$. Since each color manipulator cannot achieve utility $y+1$ unless c^* belongs to the winning k-egroup, it follows that $c^* \in X$. Because each color manipulator μ_i receives total utility at least $y+1$, X must contain some vertex candidate $a^i_{z_i}$ corresponding to a vertex of color i for some $z_i \in [y]$. We say that X *selects* vertex $v^i_{z_i}$. Since each color pair manipulator $\mu_{i,j}$ receives total utility at least $y+1$, X must contain some edge candidate $b^{i,j}_{z_{i,j}}$ corresponding to an edge connecting a vertex of color i and a vertex of color j for some $z_{i,j}$. We say that X *selects* edge $e^{i,j}_{z_{i,j}}$. We implicitly assumed that each color manipulator and color pair manipulator contributes exactly one selected candidate to X. This assumption is true because there are exactly $k-1$ such manipulators and each needs to select at least one candidate; hence, X is exactly of the desired size. In order to show that the corresponding vertices and edges encode a colorful h-clique, it remains to show that no selected edge is incident to a vertex that is not selected. Assume towards a contradiction that X selects an edge $e^{i,j}_{z_{i,j}}$ and some vertex $v^i_{z_i} \notin e^{i,j}_{z_{i,j}}$. However, either verification manipulator $v_{i,j}$ or verification manipulator $v'_{i,j}$ receives the total utility at most $q-1$; a contradiction. □

Finally, devising an ILP formulation, we show that $\mathcal{F}^{\text{egal}}_{\text{opt}}$ -Tie-Breaking becomes fixed-parameter tractable when parameterized by the combined parameter "number of manipulators and number of different utility values." This parameter covers situations with few manipulators that have simple utility functions; in particular, when few voters have 0/1 utility functions. The subsequent Theorem 3 shows that neither few utility functions (Theorem 1) nor few manipulators (Theorem 2) make $\mathcal{F}^{\text{egal}}_{\text{opt}}$ -TB fixed-parameter tractable, but only combining these two parameters allows us to deal with the problem in FPT time.

Theorem 3 *Let u_{diff} denote the number of different utility values and r denote the number of manipulators. Then, parameterized by $r+u_{\text{diff}}$, $\mathcal{F}^{\text{egal}}_{\text{opt}}$ -Tie-Breaking is fixed-parameter tractable.*

Proof We define the type of any candidate c_i to be the size-r vector $t=(u_1(c_i),u_2(c_i),\dots,u_r(c_i))$. Let $\mathcal{T}=\{t_1,t_2,\dots,t_{|\mathcal{T}|}\}$ be the set of all possible types. Naturally, the size of $\mathcal{T}$ is upper-bounded by u^r_{diff}. We denote the set of candidates of type $t_i \in \mathcal{T}$ by T_i. Now, the ILP formulation of the problem using exactly $|\mathcal{T}|+1$ variables reads as follows. For each type $t_i \in \mathcal{T}$, we introduce an integer variable x_i indicating the number of candidates of type t_i in an optimal k-egroup. We use variable s to represent the minimal value of the total utility achieved by manipulators. We define the following ILP with the goal to maximize s (indicating the utility gained by the least satisfied manipulator) subject to:

$$\forall t_i \in \mathcal{T}\colon\ 0 \leq x_i \leq |T_i|, \tag{1}$$

$$\sum_{t_i \in \mathcal{T}} x_i = k, \tag{2}$$

$$\forall \ell \in [r] : \sum_{t_i \in \mathcal{T}} x_i \cdot t_i[\ell] \geq s. \tag{3}$$

Constraint set (1) ensures that the solution is achievable with given candidates. Constraint (2) guarantees a choice of an egroup of size k. The last set of constraints imposes that s holds at most the minimal value of the total utility gained by manipulators. By a famous result of Lenstra [34], this ILP formulation with the number of variables bounded by $u^r_{\text{diff}} + 1$ yields that $\mathcal{F}^{\text{egal}}_{\text{opt}}$-Tie-Breaking is fixed-parameter tractable when parameterized by the combined parameter $r + u_{\text{diff}}$. □

5 Complexity of coalitional manipulation

In the previous section, we have seen that breaking ties optimistically or pessimistically—an essential subtask to be solved by the manipulators in general—can become computationally challenging; in most cases, however, this problem turned out to be computationally easy. In this section, we move on to our full framework and analyze the computational difficulty of voting strategically for a coalition of manipulators. To this end, we formalize our central computational problem. Let $\mathcal{R}$ be a multiwinner voting rule and let $\mathcal{F}$ be a multiwinner tie-breaking rule.

$\mathcal{R}$-$\mathcal{F}$-eval-Coalitional Manipulation ($\mathcal{R}$-$\mathcal{F}$-eval-CM)

	eval $\in$ { util , egal , candegal }
Input: t	An election (C, V), an egroup size $k < \lvert C \rvert$, r manipulators represented by their utility functions $U = \{u_1, u_2, \dots, u_r\}$ such that, for all $i \in [r]$, $u_i : C \to \mathbb{N}$, and a non-negative, integral evaluation threshold q.
Question:	Is there a size-r multiset W of manipulative votes over C such that an k-egroup $S \subset C$ that wins the election $(C, V \cup W)$ under $\mathcal{R}$ and $\mathcal{F}$ yields eval $(S) \geq q$?

The $\mathcal{R}$-$\mathcal{F}$-eval-CM problem is defined very generally; namely, one can consider any multiwinner voting rule $\mathcal{R}$ (in particular, any single-winner voting rule is a multiwinner voting rule with $k = 1$). In our paper, however, we focus on ℓ-Bloc; hence, from now on, we narrow down our analysis of $\mathcal{R}$-$\mathcal{F}$-eval-CM to the ℓ-Bloc-$\mathcal{F}$-eval-CM problem.

In line with our intention to model optimistic and pessimistic attitudes of manipulators, we require that the evaluation of an optimistic/pessimistic tie-breaking rule $\mathcal{F}$ is the same as that of the manipulator's. Indeed, only when this is the case the tie-breaking rule reflects that the manipulator's expect a certain (pessimistic or optimistic) outcome of an election in case of a tie. More formally, for every eval $\in$ { util , egal , candegal }, we focus on variants of ℓ-Bloc-$\mathcal{F}$-eval-CM where $\mathcal{F} \in \{\mathcal{F}_{\text{lex}}, \mathcal{F}^{\text{eval}}_{\text{opt}}, \mathcal{F}^{\text{eval}}_{\text{pess}}\}$.[9] We always allow lexicographic

[9] The excluded problem variants might become relevant for situations where a tie-breaking is performed by a third party external to the manipulators. However, in such a case, the third party should have its own utility function over the candidates, which is beyond our model.

tie-breaking because it models cases where a tie-breaking rule is fixed, known to all voters and, more importantly, irrelevant of manipulator utility functions.

On the way to show our results, we also use a restricted version of ℓ-Bloc-$\mathcal{F}$-eval-Coalitional Manipulation that we call *ℓ-Bloc-$\mathcal{F}$*-eval-Coalitional Manipulation . In this variant, the input stays the same, but all manipulators cast exactly the same vote to achieve the objective.

To increase readability, we decided to represent manipulators by their utility functions. As a consequence, we frequently use, for example, u_1 referring to the manipulator itself, even if we do not care about the values of utility function u_1 at the moment of usage. In the paper, we also stick to the term "voters" meaning the set V of voters of an input election. We never call manipulators "voters"; however, we speak about the manipulative votes they cast.

As for the encoding of the input of $\mathcal{R}$-$\mathcal{F}$-eval-CM, we use a standard assumption; namely, that the number of candidates, the number of voters, and the number of manipulators are polynomially upper-bounded in the size of the input. Analogously to $\mathcal{F}^{\text{eval}}_{\text{bhav}}$ -Tie-Breaking, both the evaluation threshold and the utility function values are encoded in binary.

In the subsequent sections, we first focus on the (computationally simpler) utilitarian and candidate-wise egalitarian evaluation variants (Sect. 5.1) and then consider the egalitarian evaluation (Sect. 5.2).

5.1 Utilitarian and candidate-wise egalitarian: manipulation is tractable

We show that ℓ-Bloc-$\mathcal{F}$-eval-Coalitional Manipulation can be solved in polynomial time for any $\ell \in \mathbb{N}$, any $\text{eval} \in \{\text{util}, \text{candegal}\}$, and any tie-breaking rule $\mathcal{F} \in \{\mathcal{F}_{\text{lex}}, \mathcal{F}^{\text{eval}}_{\text{opt}}, \mathcal{F}^{\text{eval}}_{\text{pess}}\}$. Whereas in general, for $|I|$ being the input size, our algorithm requires $O(|I|^5)$ steps, for Bloc (i.e., $\ell = k$), we present a better, quadratic-time algorithm (with respect to n).

In several proofs in Sect. 5.1 we use the value of a candidate for manipulators (coalition) and say that a candidate is more valuable or less valuable than another candidate. Although we cannot directly measure the value of a candidate for the whole manipulators' coalition in general, thanks to Observation 1 we can assume a single utility function when discussing the utilitarian and candidate-wise egalitarian variants. Thus, assigning a single value to each candidate is justified.

We start with an algorithm solving the general case of ℓ-Bloc-$\mathcal{F}$-eval-Coalitional Manipulation, $\text{eval} \in \{\text{util}, \text{candegal}\}$, $\mathcal{F} \in \{\mathcal{F}_{\text{lex}}, \mathcal{F}^{\text{eval}}_{\text{opt}}, \mathcal{F}^{\text{eval}}_{\text{pess}}\}$. The basic idea is to "guess" the lowest final score of a member of a k-egroup and (assuming some lexicographic order over the candidates) the least preferred candidate of the k-egroup that obtains the lowest final score; there are at most polynomially many (with respect to the input size) pairs to be guessed. Then, the algorithm, in polynomial time (with respect to the input size), finds an optimal manipulation leading to a k-egroup represented by the guessed pair. At first glance it might seem that it is enough to use a greedy algorithm for finding an optimal manipulation for a guessed pair. However, observe that a fundamental task here is, given a number of points to distribute (by manipulators), to find a selection of most-preferred members of a k-egroup where each selected member requires a certain number of points to be selected. This task however is just another interpretation of (a variant of) the weakly NP-hard Knapsack problem [31]. Fortunately, the weights of the items can be upper-bounded by the number of manipulators times size of the egroup, so that this Knapsack variant can indeed by solved in polynomial time.

Theorem 4 *Let m denote the number of candidates, n the number of voters, k the size of a desired egroup, and r the number of manipulators. One can solve ℓ-Bloc-$\mathscr{F}$-eval-Coalitional Manipulation in $O(k^2m^2r(n+r))$ time for any* $\text{eval} \in \{\text{util}, \text{candegal}\}$ *and* $\mathscr{F} \in \{\mathscr{F}_{\text{lex}}, \mathscr{F}^{\text{eval}}_{\text{opt}}, \mathscr{F}^{\text{eval}}_{\text{pess}}\}$.

Proof We prove the theorem for the lexicographic tie-breaking rule $\mathscr{F}_{\text{lex}}$. This is sufficient since, using Proposition 1, one can generalize the result for the cases of utilitarian and candidate-wise egalitarian variants. The basic idea of our algorithm is to fix certain parameters of a solution and then to reduce the resulting subproblem to a variant of the Knapsack problem with polynomial-sized weights. The algorithm iterates through all possible value combinations of the following two parameters:

- the lowest final score $z < |V \cup W|$ of any member of the k-egroup and
- the candidate $\hat{c}$ with final score z such that c is the least preferred member of the k-egroup with respect to tie-breaking rule $\mathscr{F}_{\text{lex}}$.

For each combination of the parameters, the algorithm computes an optimal solution if it exists. In this case, an optimal solution is a manipulation leading to an egroup that maximizes the utility for the manipulation among all egroups described by the parameters z and $\hat{c}$. The algorithm outputs "yes" if, among the solutions computed for all combinations of the parameters, there exists a manipulation resulting in an egroup that has at least the utility requested by the instance's input. Otherwise, the algorithm outputs "no."

To show how to compute an optimal manipulation for some combination of the parameters, let us fix some z and $\hat{c}$. We denote by C^+ the set of candidates who get at least $z+1$ approvals from the non-manipulative votes or who are preferred to $\hat{c}$ according to $\mathscr{F}_{\text{lex}}$ and get exactly z approvals from the non-manipulative votes. Assuming that the combination of parameter values is correct, all candidates from $C^+ \cup \{\hat{c}\}$ must belong to the k-egroup. Let $k^+ := |C^+|$. For sanity, we check whether $k^+ < k$, that is, whether candidate $\hat{c}$ can belong to the k-egroup if the candidate obtains final score z. We discard the corresponding combination of solution parameter values if the check fails. Next, we ensure that $\hat{c}$ obtains the final score exactly z. If $\hat{c}$ receives less than $z-r$ or more than z approvals from non-manipulative votes, then we discard this combination of solution parameter values. Otherwise, let $\hat{s} := z - \text{score}_V(\hat{c})$ denote the number of additional approvals candidate $\hat{c}$ needs in order to get final score z. Let $k^* := k - k^+ - 1$ be the number of remaining (not yet fixed) members of the k-egroup. Let $s^* := r \cdot \ell - \hat{s}$ be the number of approvals to be distributed to candidates in $C \setminus \{\hat{c}\}$.

Now, the manipulators have to influence further k^* candidates to join the k-egroup (so far only consisting of $C^+ \cup \{\hat{c}\}$) and distribute exactly s^* approvals in total to candidates in $C \setminus \{\hat{c}\}$ but at most r approvals per candidate. To this end, let C^* denote the set of candidates which can possibly join the k-egroup. For each candidate $c \in C \setminus (C^+ \cup \{\hat{c}\})$ it holds that $c \in C^*$ if and only if

1. $z - r \leq \text{score}_V(c) \leq z - 1$ if c is preferred to $\hat{c}$ with respect to $\mathscr{F}_{\text{lex}}$, or
2. $z - r + 1 \leq \text{score}_V(c) \leq z$ if $\hat{c}$ is preferred to c with respect to $\mathscr{F}_{\text{lex}}$.

A straightforward idea is to select the k^* elements from C^* which have the highest values (that is, utility) for the coalition. However, there are two issues: First, s^* might be too small; that is, there are too few approvals to ensure that each of the k^* best-valued candidates gets the final score at least z (resp. at least $z+1$). Second, s^* might be too large; that is, there are too many approvals to be distributed so that there is no way to do this without causing unwanted candidates to get final score at least z (resp. at least $z+1$).

Fortunately, we can easily detect these cases and deal with them efficiently. In the former scenario we reduce the remaining problem to an instance of Exact k -item Knapsack—the problem in which, for a given set of items, their values and weights, and a knapsack capacity, we search for k items that maximize the overall value and do not exceed the knapsack capacity. In the latter case, we show that we can discard the corresponding combination of solution parameters.

First, if $s^* \le r \cdot k^*$, then one can certainly distribute all s^* approvals (e.g., to the k^* candidates that will finally join the k-egroup). Of course, it could still be the case that there are too few approvals available to push the desired candidates into the k-egroup in a greedy manner. To solve this problem, we build an Exact k^* -item Knapsack instance where each candidate $c^* \in C^*$ is mapped to an item. We set the weight of c^* to $z - \text{score}_V(c^*)$ if c^* is preferred to $\hat{c}$ with respect to $\mathscr{F}_{\text{lex}}$ and otherwise to $(z+1) - \text{score}_V(c^*)$. We set the value of each $c^* \in C^*$ to be equal to the utility that candidate c^* contributes to the manipulators. Now, an optimal solution (given the combinations of parameter values is correct) must select exactly k^* elements from C^* such that the total weight is at most s^*. This corresponds to Exact k -item Knapsack if we set our knapsack capacity to s^*. Furthermore, finding any such set with maximum total value leads to an optimal solution. Even if the final total weight s' of the chosen elements is smaller than s^*, we can transfer the Exact k -item Knapsack solution to the correct solution of our problem. The total weight corresponds to the number of approvals used. Thus, with the Exact k -item Knapsack solution we spend s' approvals and, because of the monotonicity of ℓ-Bloc together with the assumption that $s^* \le r \cdot k^*$, we use $s^* - s'$ approvals to approve the chosen candidates even more.

Second, if $s^* > r \cdot k^*$, then one can certainly ensure that each of the k^* most valued candidates from C^* achieves the final score at least z (resp. at least $z+1$). In many cases, it will not be a problem to distribute the remaining approvals; for example, one can safely spend up to r approvals for each candidate from $C \setminus C^*$, that is, to candidates that have no chance to get enough points to join the k-egroup or to candidates which are already fixed to be in the k-egroup. Furthermore, each candidate from C^* that is not among the k^* most valued candidates can be safely approved $z - \text{score}_V(c^*) - 1$ times (resp. $z - \text{score}_V(c^*)$ times) without reaching final score z (resp. $z+1$); we denote by s^+ the total number of approvals distributed in this way. So, if $s^* \le s^+ + r \cdot k^*$ (note that we also assume $s^* > r \cdot k^*$), then we can greedily push the k^* most valued candidates from C^* into the k-egroup (spending $r \cdot k^*$ approvals) and then safely distribute the remaining at most s^+ approvals to other candidates as discussed. If $s^* > s^+ + r \cdot k^*$, then there is no possibility of distributing approvals in a way that $\hat{c}$ is part of the k-egroup. Towards a contradiction assume that $\hat{c}$ is part of the k-egroup obtained after distributing $s^+ + r \cdot k^* + 1$ approvals. This means that we spend all possible s^+ approvals so that $\hat{c}$ is not beaten and $r \cdot k^*$ approvals to push k^* candidates to the winning k-egroup. Giving one more approval to some candidate c' from C^* that is not yet in the k-egroup, by definition of C^* and s^+, means that the score of c' is enough to push $\hat{c}$ out of the final k-egroup; a contradiction. Consequently, for the case of $s^* > s^+ + r \cdot k^*$, we discard the corresponding combination of solution parameters.

As for the running time, the first step is sorting the candidates according to their values in $O(m(r + \log(m)))$ time. Then let us consider the running time of two cases $s^* \le r \cdot k^*$

and $s^* > r \cdot k^*$ separately. In the former case, we solve Exact k-item Knapsack in $O(k^2mr)$ time by using dynamic programming based on analyzing all possible total weights of the selected items until the final value is reached [31, Chapter 9.7.3][10] (note that the maximum possible total weight is upper-bounded by kr). If $s^* > r \cdot k^*$, then each manipulator approves at most m candidates which gives running time $O(m \cdot r)$. Thus, we can conclude that the running time of the discussed cases is $O(k^2mr)$. Additionally, there are at most $n + r$ values of z and at most m choices of $\hat{c}$. Summarizing, we get the running time $O(k^2m^2r(n + r))$. □

Next, we show that Bloc-$\mathscr{F}$-eval-CM (i.e., the special case of ℓ-Bloc-$\mathscr{F}$-eval-CM where $\ell = k$) can be solved in quadratic time, that is, much faster than the general variant of the problem. On our way to present this results, we first give an algorithm for ℓ-Bloc-$\mathscr{F}$-eval -CM with consistent manipulators. Then, we argue that it also solves Bloc-$\mathscr{F}$-eval-CM. The algorithm "guesses" the minimum score among all members of the winning egroup and then (according to the tie-breaking method) selects the best candidates that can reach this score.

Proposition 3 *Let m denote the number of candidates, n denote the number of voters, and r denote the number of manipulators. Then one can solve* ℓ-Bloc-$\mathscr{F}$-eval-Coalitional Manipulation *with consistent manipulators in $O(m(m + r + n))$ time for any* eval $\in \{$ util , candegal $\}$ *and* $\mathscr{F} \in \{\mathscr{F}_{\text{lex}}, \mathscr{F}^{\text{eval}}_{\text{opt}}, \mathscr{F}^{\text{eval}}_{\text{pess}}\}$.

Proof Consider an instance of ℓ-Bloc-$\mathscr{F}_{\text{lex}}$-eval-CM with consistent manipulators with an election $E = (C, V)$ where C is a candidate set and V is a multiset of non-manipulative votes, r manipulators, an egroup size k, and a lexicographic order $>_{\mathscr{F}}$ used by $\mathscr{F}_{\text{lex}}$ to break ties. In essence, we introduce a constrained solution form called a *canonical solution* and argue that it is sufficient to aim for only this type of solutions. Then we provide an algorithm that efficiently seeks for an optimal canonical solution.

At the beginning, we observe that when manipulators vote consistently, then we can arrange the top ℓ candidates of a manipulative vote in any order. Hence, the solution to our problem is a size-ℓ subset (instead of an order) of candidates which we call a set of *supported candidates*; we call each member of this set a *supported candidate*. We now introduce a vital concept of the proof, the "strength" of the candidates.

Strength order of the candidates. Additionally, we introduce a new order $>_{\mathscr{S}}$ of the candidates. It sorts them descendingly with respect to the score they receive from voters and, as a second criterion, according to the position in $>_{\mathscr{F}}$. Intuitively, the easier it is for some candidate to be a part of a winning k-egroup, the higher is the candidate's position in $>_{\mathscr{S}}$. As a consequence, we state Claim 1.

Claim 1 *Let us fix an instance of* ℓ-Bloc-$\mathscr{F}_{\text{lex}}$-eval-*CM with consistent manipulators and a solution X which leads to a winning k-egroup S. For every supported (resp. unsupported) candidate c, the following holds:*

[10] Kellerer et al. [31] present dynamic programming based on all possible total *values* of items. However (what they also remark), these can be exchanged with all possible total *weights* of items leading to an algorithm with running time polynomial in the maximum weight of items.

1. *If c is part of the winning k-egroup, then every supported (resp. unsupported) predecessor of c, according to* $>_{\mathscr{S}}$, *belongs to S.*
2. *If c is not part of the winning k-egroup, then every supported (resp. unsupported) successor of c, according to* $>_{\mathscr{S}}$, *does not belong to S.*

Proof Fix an instance of ℓ-Bloc-$\mathscr{F}_{\text{lex}}$-eval-CM with consistent manipulators and a solution X resulting in winning k-egroup S. Let us consider the respective order $>_{\mathscr{S}}$ over the candidates in the instance.

We first show that Statement 1 regarding supported candidates holds. According to the statement, fix some supported candidate $c \in S$ and let p be a predecessor of c (according to $>_{\mathscr{S}}$). Towards a contradiction, let us assume that $p \notin S$. This implies that either (i) the score of p is smaller than the score of c or (ii) their scores are the same but $c >_{\mathscr{F}} p$. Let us focus on case (i). Both considered candidates are supported by all manipulators (note that manipulators vote consistently). Thus, as a consequence of $p >_{\mathscr{S}} c$, we have that the score of p is at least as high as the score of c; a contradiction. Next, consider case (ii), where p and c have the same scores. Consequently, the mutual order of c and p in $>_{\mathscr{S}}$ is the same as their order in $>_{\mathscr{F}}$ (in other words, the order of c and p in $>_{\mathscr{S}}$ does not depend on scores of c and p because those must be the same prior to any manipulation). Since $c >_{\mathscr{F}} p$, it follows that, by definition of $>_{\mathscr{S}}$, it must hold that $c >_{\mathscr{S}} p$; a contradiction again. Eventually, we obtain that p has to be part of S which completes the argument.

An analogous approach leads to proofs for the remaining three cases stated in the theorem. □

Claim 1 justifies thinking about $>_{\mathscr{S}}$ as a "strength order"; hence, in the proof we use the terms *stronger* and *weaker* candidate. Using Claim 1, we can fix some candidate c as the weakest in the winning k-egroup and then infer candidates that have to be and that cannot be part of this k-egroup. To formalize this idea, we introduce the concept of a *canonical solution*.

Canonical solutions. Assuming the case where $k \leq \ell$, we call a solution X leading to a winning k-egroup S canonical if all candidates of the winning egroup are supported; that is, $S \subseteq X$. In the opposite case, $k > \ell$, solution X is canonical if $X \subset S$ and X is a set of the ℓ weakest candidates in S. For the latter case, the formulation describes the solution which favors supporting weaker candidates first and ensures that no approval is given to a candidate outside the winning k-egroup.

Canonical solutions are achievable from every solution without changing the winning k-egroup. One cannot prevent a candidate from winning by supporting the candidate more because this only increases the candidate's score. Consequently, we can always transfer approvals to all candidates from the winning k-egroup. For the case $k > \ell$, we then have to rearrange the approvals in such a way that only the weakest members of the k-egroup are supported. However, such a rearrangement cannot change the outcome because, according to Claim 1, we can transfer an approval from some stronger candidate c to a weaker candidate c' keeping both of them in the winning k-egroup.

Dropped and kept candidates. By the assumption that $k < m$, for every solution (including canonical solutions) we can always find the strongest candidate who is not part of the winning egroup. We call this candidate the *dropped candidate*. Note that we use the strength order in the definition of the dropped candidate; this order does not take manipulative votes into account. Further applying the assumption that $\ell < m$, without loss of generality, we can assume that the dropped candidate is not a supported candidate. This holds true because if the dropped candidate is not in the winning k-egroup even if supported, then we can support any other candidate c (which must exist because $\ell < m$) without changing the winning k-egroup. Due to Claim 1, if c is not in the winning k-egroup, then, even after supporting, c (is by definition weaker than the dropped candidate) cannot become a member of k-egroup. Otherwise, supporting c clearly cannot prevent it from being a member of the winning k-egroup. Naturally, by definition of the dropped candidate, all candidates stronger than the dropped candidate are members of the winning k-egroup. We call these candidates *kept candidates*.

High-level description of the algorithm. The algorithm solving ℓ-Bloc-$\mathscr{F}_{\text{lex}}$-eval-CM with consistent manipulators iteratively looks for an optimal canonical solution for every possible (non-negative) number t of kept candidates (alternatively the algorithm checks all feasible possibilities of choosing the dropped candidate). Then, the algorithm compares all solutions and picks one that is resulting in an egroup liked the most by the manipulators. Observe that $k - \ell \leq t \leq k$. The upper bound k is the consequence of the fact that each kept candidate is (by definition) in the winning k-egroup. Since all candidates except for kept candidates have to be supported to be part of the winning egroup, we need at least $k - \ell$ kept candidates in order to be able to complete the k-egroup.

What remains to be done. Procedure 1 describes how to look for an optimal canonical solution for a fixed number t of kept candidates. First, partition the candidate set in the following way. By C^* we denote the kept candidates (which are the top t candidates according to $>_{\mathscr{S}}$). Consequently, the $(t + 1)$-st strongest candidate is the dropped candidate; say c^*. For every value of t, the corresponding dropped candidate, by definition, is not allowed to be part of the winning egroup. Let

$$D = \{c \in C \setminus (\mathrm{C}^* \cup \{c^*\}) \,|(\operatorname{score}_V(c) + r > \operatorname{score}_V(c^*)) \vee \\ (\operatorname{score}_V(c) + r = \operatorname{score}_V(c^*) \wedge c >_{\mathscr{F}} c^*)\}$$

be the set of *distinguished candidates*. Each distinguished candidate, if supported, is preferred over c^* to be selected into the winning k-egroup. Consequently, the distinguished candidates are all candidates who can potentially be part of the winning k-egroup. We remark that to fulfill our assumption that the dropped candidate is not belonging to a winning egroup, it is obligatory to support at least $k - t$ distinguished candidates. Note that $\mathrm{C}^* \cup \{c^*\} \cup D \neq C$ is possible. The remaining candidates cannot be part of the winning k-egroup under any circumstances assuming t kept candidates. Also, set D might consist of less than $k - t$ required candidates (which is the case when there are too few candidates that, after supported, would outperform c^*). If such a situation emerges, then we skip the respective value of t. Making use of the described division into c^*, D, and C^*, Procedure 1 incrementally builds the set X of supported candidates associated with an optimal solution until all possible approvals are used. Observe that since $k < |C|$ and $\ell < |C|$, it is guaranteed that for $t = k$ Procedure 1 will return a feasible solution for t; in fact, this solution will always result in a winning egroup consisting of all t kept candidates (irrespective of D).

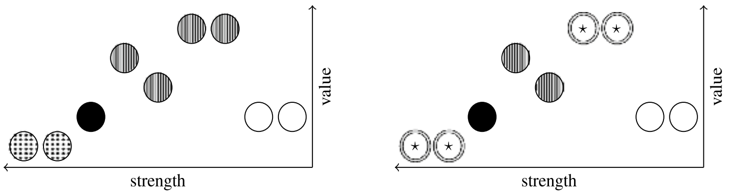
Procedure 1: A procedure of finding an optimal set of supported candidates.

Input: Election $E = (C, V)$; number ℓ of approvals in ℓ-Bloc rule; size k of the winning k-egroup; a partition of C into kept candidates C^* (such that $|\mathrm{C}^*| = t$ and $k - \ell \leq t \leq k$), a dropped candidate c^*, and distinguished candidates D (such that $|D| \geq k - t$).
Output: Optimal supported candidates set X

1 $X \longleftarrow \{k - t$ most valuable candidates from $D\}$
2 $X \longleftarrow X \cup \{\min\{t, \ell - |X|\}$ arbitrary candidates from $\mathrm{C}^*\}$
3 **if** $\ell \neq |X|$ **then**
4     $A \longleftarrow \{\ell - |X|$ weakest candidates from $C \setminus X\}$
5     $B \longleftarrow \{$top k strongest candidates from $X \cup A\}$
6     $p \longleftarrow |B \setminus X|$
7     $X \longleftarrow X \cup \{\ell - |X| - p$ weakest candidates from $C \setminus X\}$
8     $X \longleftarrow X \cup \{p$ most valuable candidates from $D \setminus X\}$
9 **end**
10 **return** X

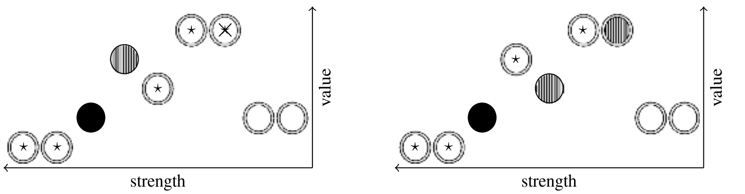
Detailed description of the algorithm. Before studying Procedure 1 in detail, consider Fig. 1 illustrating the procedure on example data. In line 1, the procedure builds set X of supported candidates using the $k - t$ best valued distinguished candidates. Since only the distinguished candidates might be a part of the winning k-egroup besides the kept candidates, there is no better outcome achievable. Then, in line 2, the remaining approvals, if they exist, are used to support kept candidates. This operation does not change the resulting k-egroup. Then Procedure 1 checks whether all ℓ approvals were used; that is, whether $\ell = |X|$. If not, then there are exactly $\ell - |X|$ remaining approvals to use. Note that at this stage set X contains k supported candidates which correspond to the best possible k-egroup; however, without spending all approvals. Let us call this k-egroup S. It is possible that there is no way to spend the remaining $\ell - |X|$ approvals without changing the winning k -egroup S. Then substitutions of candidates occur. The new candidates in the k-egroup can be only those that are distinguished and so far unsupported whereas the exchanged ones can be only so far supported distinguished candidates. This means that each substitution lowers the overall value of the winning k-egroup. So, the best what can be achieved is to find the minimal number of substitutions and then pick the most valuable remaining candidates from D to be substituted. The minimal number of substitutions can be found by analyzing how many candidates would be exchanged in the winning k-egroup if the weakest $\ell - |X|$ previously unsupported candidates were supported. The procedure makes such a simulation and computes the number p of necessary substitutions, in lines 4-6. Supporting the $\ell - |X| - p$ weakest unsupported candidates and then the p most valuable so far unsupported distinguished candidates gives the optimal k-egroup for t kept candidates (when all approvals are spent). Note that the number ℓ of approvals is strictly lower than the number of candidates, so one always avoids supporting c^*.

Running time. To analyze the running time of the algorithm, several steps need to be considered. At the beginning we have to compute values of candidates and then sort the candidates with respect to their value. This step runs in $O(rm + m \log m)$ time. Similarly, computing $>_{\mathscr{S}}$ takes $O(\ell n + m \log m)$ time. Moreover, Procedure 1 needs $O(m)$ steps to find an optimal canonical solution for some fixed number t of kept candidates. Finally, we have at most $\ell + 1$ possible values of t. Summing the times up, together with the fact that $\ell < m$, we obtain the running time $O(m(m + r + n))$.



(a) The division of the candidates into the kept candidates (dotted), the dropped candidate (filled), the distinguished candidates (vertical lines), and the others who cannot be part of the winning egroup.

(b) An illustration of lines 1-2 of Procedure 1. The double-edged candidates form set X. The starred candidates would form the winning k-egroup if the double-edged candidates were supported.



(c) Supporting the weakest possible candidates to use all approvals. The winning egroup changes. The winners are marked with stars while the candidate which is not any more in the winning egroup is crossed out. Such a simulation is done in lines 4-6. As a result, the minimum number of substitutions in the k-egroup is computed.

(d) An illustration of the solution of the considered case computed by Procedure 1 in lines 7 to 8. One candidate from the k-egroup presented in Figure 1b has to be substituted; naturally, it is optimal to pick the most valuable possible candidate as a replacement for the substituted one. Supported candidates are double-edged and the winning k-egroup is starred.

Fig. 1 An illustrative example of a run of Procedure 1 for $t = 2$, nine candidates, 7-Bloc, and 4-egroup. The horizontal position indicates the strength of a candidate—with the strength decresing from left to right—and the vertical position indicates the value of a candidate. Since the number r of manipulators determines only the set of distinguished candidates, we do not specify r explicitly. We indicate the set of distinguished candidates instead. Figure 1a–d step by step present the execution of Procedure 1 on the way to find an optimal 4-egroup.

Pessimistic and Optimistic Evaluation. Due to Proposition 1, the algorithm we presented can be applied also for pessimistic and optimistic evaluation because of the possibility of simulating these evaluations by a lexicographic order in time $O(m(r + \log(m)))$. □

For Bloc, we will show that manipulators can always vote identically to achieve an optimal k-egroup. In a nutshell, for every egroup the manipulators can only increase the scores of the k-egroup's members by voting exactly for them. This fact leads to the following theorem.

Theorem 5 *Let m denote the number of candidates, n denote the number of voters, and r denote the number of manipulators. One can solve* Bloc-$\mathscr{F}$-eval-Coalitional Manipulation *in $O(m(m + r + n))$ time for any* eval $\in$ { util , candegal } *and* $\mathscr{F} \in \{\mathscr{F}_{\text{lex}}, \mathscr{F}^{\text{eval}}_{\text{opt}}, \mathscr{F}^{\text{eval}}_{\text{pess}}\}$.

Proof We show that for Bloc-$\mathscr{F}$-eval-Coalitional Manipulation the manipulators have no incentive to deviate from one optimal profile (i.e., they vote in the same manner). Let us fix an optimal k-egroup S. If there exists a candidate $c \in S$ which is not approved by some manipulator u^*, then there exists also some candidate $c' \notin S$ which is approved by u^* (u^* approves at most $k - 1$ candidates from S). Observe that in the Bloc voting rule by shifting a candidate up in a preference order we only increase the candidate's score; as a result, we cannot prevent the candidate from winning by doing such a shift. Using this observation, we can exchange some candidate $c \in S$ with some candidate $c' \notin S$ in the preference order of u^* without preventing c from winning. We repeat exchanging candidates until all manipulators approve only candidates from S. Then we obtain an optimal vote by fixing a preference order over those candidates arbitrarily (there might be more than one optimal vote but all of them place only candidates from set S at the first k places). Concluding, we can use the algorithm from Proposition 3 which works in the given time. □

5.2 Egalitarian: manipulation is hard even for simple tie-breaking

In Sect. 4.2, we showed that already breaking ties might be computationally intractable. These intractability results only hold with respect to the egalitarian evaluation and optimistic manipulators. We now show that this intractability of $\mathscr{F}^{\text{egal}}_{\text{opt}}$ -Tie-Breaking extends to coalitional manipulation for any tie-breaking rule and egalitarian evaluation. This includes the pessimistic egalitarian case which we consider to be highly relevant as it naturally models searching for a "safe" voting strategy.

Proposition 4 *For any tie-breaking rule $\mathscr{F}$, there is a polynomial-time many-one reduction from $\mathscr{F}^{\text{egal}}_{\text{opt}}$ -Tie-Breaking to ℓ*-Bloc-$\mathscr{F}$-egal-*Coalitional Manipulation.*

Proof We reduce an instance of $\mathscr{F}^{\text{egal}}_{\text{opt}}$ -Tie-Breaking to ℓ-Bloc-$\mathscr{F}$-egal-Coalitional Manipulation; however, before we describe the actual reduction, we present a useful observation concerning $\mathscr{F}^{\text{egal}}_{\text{opt}}$ -Tie-Breaking in the next paragraph.

Let us fix an instance I of $\mathscr{F}^{\text{egal}}_{\text{opt}}$ -Tie-Breaking with a confirmed set C^+, a pending set P, a size k of an egroup, a threshold q, and a set of manipulators represented by a family U of utility functions. We construct a new equivalent instance I' of $\mathscr{F}^{\text{egal}}_{\text{opt}}$ -Tie-Breaking with a larger set of manipulator utility functions $U' \supseteq U$. The construction is a polynomial-time many-one reduction which proves that we can "pump" the number of manipulators arbitrarily for instance I. To add a manipulator, it is enough to set to q the utility that the manipulator gives to every candidate. Naturally, such a manipulator cannot have the total utility smaller than q, so the correct solution for I is also correct for I'. Contrarily, when there is no solution for I, it means that for every possible k-egroup S' there is some manipulator $\bar{u}$ such that $\text{egal}_{\bar{u}}(S') < q$. Consequently, one cannot find a solution for I' as well, because the set of possible k-egroups and their values of egalitarian utility do not change.

Now we can phrase our reduction from $\mathscr{F}^{\text{egal}}_{\text{opt}}$ -Tie-Breaking to ℓ-Bloc-$\mathscr{F}$-egal-Coalitional Manipulation. Let us fix an instance I of $\mathscr{F}^{\text{egal}}_{\text{opt}}$ -Tie-Breaking with a confirmed set C^+, a pending set P, a size k of an egroup, a threshold q, and a set U of r utility functions. Because of the observation about "pumping" instances of $\mathscr{F}^{\text{egal}}_{\text{opt}}$ -Tie-Breaking, we can assume, without loss of generality, that $\ell \cdot r \geq k - |C^+|$ holds. In the constructed instance of ℓ-Bloc-$\mathscr{F}$-egal-CM equivalent to I, we build an election that yields sets P and C^+ and aim at an egroup of size k. However, it is likely that we need to add a set of dummy candidates that we denote by D. It is important to ensure that the dummy candidates cannot be the winners of the constructed election. To do so, we keep the score of each dummy candidate to be at most 1, the score of each pending candidate to be $r + 2$, and the score of each confirmed candidate to be at least $2r + 3$. The construction starts from ensuring the scores of the confirmed candidates. Observe that in this step we add at most $(2r + 3) \cdot |C^+|$ voters (in case $\ell = 1$). If $\ell > |C^+|$, then we have to add some dummy candidates in this step. We can upper-bound the number of the added dummy candidates by $(2r+3)\cdot|C^{+}|(\ell-1)$ (this bound is not tight). Analogously, we add new voters such that each pending candidate has score exactly $r+2$. At this step we have the election where we are able to spend $\ell\cdot r\geq k-|C^{+}|$ approvals. We can select every possible subset of pending candidates to form the winning k-egroup by approving candidates in this subset exactly once. However, to be sure that we are able to distribute all approvals such that there is no tie, we ensure that the remaining $\ell\cdot r-(k-|C^{+}|)$ approvals can be distributed to some candidates without changing the outcome. To achieve this goal, we add exactly $\ell\cdot r-(k-|C^{+}|)$ dummy candidates with score 0. We set the evaluation threshold of the newly constructed instance to q.

By our construction, we are always able to approve enough pending candidates to form a k-egroup without considering ties, and we cannot make a dummy candidate a winner under any circumstances. Thus, if $\mathscr{F}^{\text{egal}}_{\text{opt}}$ -Tie-Breaking has a solution S, then we approve every candidate $c\in S$ such that c was in the pending set P before, and we obtain a solution to the reduced instance. In the opposite case, if there is no such a k-egroup whose egalitarian utility value is at least q, then the corresponding instance of ℓ-Bloc-$\mathscr{F}$-egal-Coalitional Manipulation also has no solution since the possible k-egroups are exactly the same. The reduction runs in polynomial time. $\square$

Observe that the reduction proving Proposition 4 does not change the egroup size k. Additionally, the increase of the number of manipulators in the resulting instances is polynomially upper-bounded in the egroup size k of input instances. This is due to the fact that even if we need to "pump" an initial instance to achieve $\ell\cdot r\geq k-|C^{+}|$, then we add at most $\left\lceil\frac{k-|C^{+}|}{\ell}\right\rceil\leq k$ manipulators. Thus, together with Theorem 1 and Theorem 2, Proposition 4 leads to the following theorem.

Theorem 6 *Let $\mathscr{F}$ be an arbitrary tie-breaking rule. Then, ℓ-Bloc-$\mathscr{F}$-egal-Coalitional Manipulation is* NP*-hard. Let r denote the number of manipulators, q denote the evaluation threshold and k denote the size of an egroup. Then, parameterized by $r+k$, ℓ-Bloc-$\mathscr{F}$-egal -CM is* W[1]*-hard. Parameterized by k, ℓ-Bloc-$\mathscr{F}$-egal-CM is* W[2]*-hard even if $q=1$ and every manipulator only gives either utility one or zero to each candidate.*

Combining exhaustive enumeration of values describing essential properties of solutions and an extension of the ILP from Theorem 3, we show that, for the combined parameter "number of manipulators and number of different utility values," fixed-parameter tractability of $\mathscr{F}^{\text{egal}}_{\text{opt}}$ -Tie-Breaking extends to coalitional manipulation for both optimistic and pessimistic egalitarian tie-breaking.

Theorem 7 *Let r denote the number of manipulators and u_{diff} denote the number of different utility values. Parameterized by $r+u_{\text{diff}}$, ℓ-Bloc-$\mathscr{F}$-egal-Coalitional Manipulation with $\mathscr{F}\in\{\mathscr{F}^{\text{egal}}_{\text{pess}},\mathscr{F}^{\text{egal}}_{\text{opt}}\}$ is fixed-parameter tractable.*

Proof In a nutshell, we divide ℓ-Bloc-$\mathscr{F}^{\text{egal}}_{\text{pess}}$-egal-CM and ℓ-Bloc-$\mathscr{F}^{\text{egal}}_{\text{opt}}$-egal-CM into subproblems solvable in FPT time with respect to the combined parameter "number of manipulators and number of different utility values." We show that solving polynomially many subproblems is enough.

The main idea. We split the proof into two parts. In the first part, we define subproblems and show how to find a solution assuming that the subproblems are solvable in FPT time with respect to the parameter. In the second part, we show that, indeed, the subproblems are fixed-parameter tractable using their ILP formulations. The inputs for ℓ-Bloc-$\mathscr{F}^{\text{egal}}_{\text{pess}}$-egal-CM and ℓ-Bloc-$\mathscr{F}^{\text{egal}}_{\text{opt}}$-egal-CM are the same, so let us consider an arbitrary input with an election $E = (C, V)$ where $|V| = n$, $|C| = m$, a size k of an excellence-group, and r manipulators represented by a set $U = \{u_1, u_2, \dots, u_r\}$ of their utility functions. Let u_{diff} be the number of different utility values.

An election resulting from a manipulation and a corresponding k-egroup emerging from the manipulation can be described by three non-negative integer parameters:

1. the lowest final score z of any member of the k-egroup;
2. the number p of *promoted candidates* from the k-egroup with a score higher than z which, at the same time, have score at most z without taking manipulative votes into consideration;
3. the number b of *border candidates* with score z.

Observe that if as a result of a manipulation the lowest final score of members in a final k-egroup is z, then the promoted candidates are part of the k-egroup regardless of the tie-breaking method used. For border candidates, however, it might be necessary to run the tie-breaking rule to determine the k-egroup. In other words, border candidates become pending candidates unless all of them are part of the k-egroup. By definition, no candidate scoring lower than the border candidates is a member of the k-egroup, which gives border candidates their name. From now on, we refer to the election situation characterized by parameters z, p, b as a *state* (resp. *input state*). Additionally, we call a set of manipulator votes a *manipulation*.

Part 1: High-level description of the algorithm. For now, we assume that there is a procedure $\mathscr{P}$ which runs in FPT time with respect to the combined parameter "number of manipulators and number of different utility values." Procedure $\mathscr{P}$ takes values z, p, b and an instance of the problem as an input, and it finds a manipulation which leads to a k-egroup maximizing the egalitarian utility under either egalitarian optimistic or egalitarian pessimistic tie-breaking with respect to the input state. If such a manipulation does not exist, then procedure $\mathscr{P}$ returns "no." The algorithm solving ℓ-Bloc-$\mathscr{F}^{\text{egal}}_{\text{pess}}$-egal-CM and ℓ-Bloc-$\mathscr{F}^{\text{egal}}_{\text{opt}}$-egal-CM runs $\mathscr{P}$ for all possible combinations of values z, p, and b. Eventually, it chooses the best manipulation returned by $\mathscr{P}$ or returns "no" if $\mathscr{P}$ always returned so. Since the value of z is at most $|V| + |W|$ and b together with p are both upper-bounded by the number of candidates, we run $\mathscr{P}$ at most $(n + r)m^2$ times. Because the input size grows polynomially with respect to the growth of the values r, m, and n, the overall algorithm runs in FPT time with respect to the combined parameter "number of manipulators and number of different utility values."

Part 2: Basics and preprocessing for the ILP To complete the proof, we describe procedure $\mathscr{P}$ used by the above algorithm. In short, the procedure builds and solves an ILP program that finds a manipulation leading to the state described by the input values. Before we describe the procedure in detail we start with some notation. Fix some values of z, b, p and some election $E = (C, V)$ that altogether form the input of $\mathscr{P}$. For each candidate $c \in C$, let a size-r vector $t = (u_1(c), u_2(c), \dots, u_r(c))$, referred to as a *type vector*, define the *type* of c. We denote the set of all possible type vectors by $\mathscr{T} = \{t_1, t_2, \dots, t_{|\mathscr{T}|}\}$. Observe that $|\mathscr{T}| \leq u^r_{\text{diff}}$. With each type vector t_i, $i \in [|\mathscr{T}|]$, we associate a set T_i consisting of all candidates of type t_i. We also distinguish the candidates with respect to their initial score compared to z. A candidate of type $t_i \in \mathscr{T}$, $i \in |\mathscr{T}|$, with score $z - j$, $j \in [r] \cup \{0\}$, belongs to group G^j_i. We denote all candidates with a score (excluding manipulative votes) higher than z by C^+, whereas by C^- we denote the candidates with a score (excluding manipulative votes) strictly lower than $z - r$. For each type $t_i \in \mathscr{T}$ of a candidate, we define function $\mathrm{obl}(t_i) := |C^+ \cap T_i|$, which gives the number of candidates of type t_i that are obligatory part of the winning k-egroup.

At the beginning, procedure $\mathscr{P}$ tests whether the input values z, b, and p represent a correct state. From the fact that there has to be at least one candidate with score z, we get the upper bound $k - |C^+| - 1$ for value p. To have enough candidates to complete the k-egroup, we need at least $k - |C^+| - p$ candidates with score z after the manipulation which gives $b \geq k - |C^+| - p$. Finally, the state is incorrect if the corresponding set C^+ contains k or more candidates. If the input values are incorrect, then $\mathscr{P}$ returns "no." Otherwise, $\mathscr{P}$ continues with building a corresponding ILP program. We give two separate ILP programs—one for the optimistic egalitarian tie-breaking and the other one for the pessimistic egalitarian tie-breaking. Both programs consist of two parts. The first part models all possible manipulations leading to the state described by values z, p, and b. The second one is responsible for selecting the best k-egroup assuming the particular tie-breaking and considering all possible manipulations according to the first part. Although the whole programs are different from each other, the first parts stay the same. Thus, we postpone distinguishing between the programs until we describe the second parts. For the sake of readability, we present the ILP programs step by step.

ILP: Common part. For each group G_i^j, $i \in [|\mathscr{T}|]$, $j \in [r] \cup \{0\}$, we introduce variables x_i^j and x_i^{j+} indicating the numbers of, respectively, border and promoted candidates from group G_i^j. Additionally, we introduce variables o and $\bar{o}$. The former represents the number of approvals used to get the obligatory numbers of border and promoted candidates. The latter indicates the number of approvals which are to be spent without changing the final k-egroup (thus, in some sense a complement of the obligatory approvals) resulting from the manipulation (e.g., approving candidates in C^+, who are part of the winning k-egroup anyway, cannot change the outcome). We begin our ILP program with ensuring that the values of x_i^j and x_i^{j+} are feasible:

$$\forall t_i \in \mathscr{T}, j \in [r] \cup \{0\} : x_i^{j+} + x_i^j \leq |G_i^j|, \tag{4}$$

$$\sum_{t_i \in \mathscr{T}, j \in [r] \cup \{0\}} x_i^{j+} = p, \tag{5}$$

$$\sum_{t_i \in \mathscr{T}, j \in [r] \cup \{0\}} x_i^j = b, \tag{6}$$

$$\forall t_i \in \mathscr{T}\colon x_i^{0+} + x_i^0 = |G_i^0|, \tag{7}$$

$$\forall t_i \in \mathscr{T}\colon x_i^{r+} = 0. \tag{8}$$

The constraints ensure that exactly p candidates are selected to be promoted (5), exactly b candidates are selected to be border ones (6), and that, for every group, the sum of border and promoted candidates is not greater than the cardinality of the group (4). The last two constaint sets ensure that candidates who have score z are either promoted or border candidates (7) and that candidates with initial score $z - r$ cannot be promoted (i.e., get a score higher than z) (8). Next, we add the constraints concerning the number of approvals we need to use to perform the manipulation described by all variables x_i^j and x_i^{j+}. We start with ensuring that the manipulation does not exceed the number of possible approvals. As mentioned earlier, we store the number of required approvals using variable o.

$$o = \sum_{t_i \in \mathcal{T},\, j \in [r] \cup \{0\}} \left(x_i^j \cdot j + x_i^{j+} \cdot (j+1) \right), \tag{9}$$

$$o \le \ell r. \tag{10}$$

Then, we model spending the $\bar{o}$ remaining votes (if any) to use all approvals.

$$\begin{aligned} \bar{o} \le r|C^- \cup C^+| &+ \sum_{t_i \in \mathcal{T}} \sum_{j \in [r]} \left(|G_i^j| - x_i^j - x_i^{j+} \right)(j-1) \\ &+ \sum_{t_i \in \mathcal{T}, j \in [r]} \left(x_i^{j+} \cdot (r-j-1) \right), \end{aligned} \tag{11}$$

$$\bar{o} + o = \ell r. \tag{12}$$

The upper bound on the number of votes one can spend without changing the outcome presented in constraint (11) consists of three summands. The first one indicates the number of approvals which can be spent for candidates whose initial score was either too high or too low to make a difference in the outcome of the election resulting from the manipulation. The second summand counts the approvals we can spend for potential promoted and border candidates that eventually are not part of the winning k-egroup; we can give them less approvals than are needed to make them border candidates. The last summand represents the number of additional approvals that we can spend on the promoted candidates to reach the maximum of r approvals per candidate. This completes the first part of the ILP program in which we modeled the possible variants of promoted and border candidates for the fixed state (z, b, p).

ILP extension for optimistic egalitarian tie-breaking. In the second part, we find the final k-egroup by completing it with the border candidates according to the particular tie-breaking mechanism. Let us first focus on the case of the optimistic egalitarian tie-breaking. We introduce constraints allowing us to maximize the total egalitarian utility value of the final egroup; namely, for each group G_i^j, $i \in [|\mathcal{T}|]$, $j \in [r] \cup \{0\}$, we add a non-negative, integral variable $x_i^{j\bullet}$ indicating the number of border candidates of the given group chosen to be in the final k-egroup. The following constraints ensure that we select exactly $k - |C^+| - p$ border candidates to complete the winning egroup and that, for each group G_i^j, we do not select more candidates than available.

$$\sum_{t_i \in \mathcal{T},\, j \in [r] \cup \{0\}} x_i^{j\bullet} = k - |C^+| - p, \tag{13}$$

$$\forall t_i \in \mathcal{T},\, j \in [r] \cup \{0\} : x_i^{j\bullet} \le x_i^j. \tag{14}$$

To complete the description of the ILP, we add the following final set of constraints defining the egalitarian utility s of the final k-excellence-group:

$$\forall q \in [r]: \sum_{t_i \in \mathcal{T},\, j \in [r] \cup \{0\}} t_i[q] \cdot (x_i^{j+} + x_i^{j\bullet}) + \sum_{t_i \in \mathcal{T}} t_i[q] \cdot \mathrm{obl}(t_i) \geq s. \tag{15}$$

We set the goal of the program to maximize s and thus our program simulates the egalitarian optimistic tie-breaking.

ILP extension for pessimistic egalitarian tie-breaking. To solve our subproblem for the case of pessimistic egalitarian tie-breaking, we need a different approach. We start with an additional notation. For each type of candidate $t_i \in \mathcal{T}$, let $b_i = \sum_{j \in [r] \cup \{0\}} x_i^j$ denote the number of border candidates of this type. For each type $t_i \in \mathcal{T}$ and manipulator u_q, $q \in [r]$, we introduce a new integer variable d_i^q. Its value corresponds to the number of border candidates of type t_i who are part of the worst possible winning k-egroup according to manipulator u_q's preferences; we call these candidates the *designated candidates* of type t_i of manipulator u_q. For each variable d_i^q, we define a binary variable $\overline{\text{used}}\,[d_i^q]$ which has value one if at least one candidate of type t_i is a designated candidate of manipulator u_q. Similarly, we define $\overline{\text{fullyused}}\,[d_i^q]$ to indicate that all candidates of type t_i are designated by manipulator u_q. To give a program which solves the case of pessimistic egalitarian tie-breaking, we copy the first part of the previous ILP program (constraints from (4) to (12)) and add new constraints. First of all, we ensure that each manipulator designates not more than the number of available border candidates from each type and that every manipulator designates exactly $k - p - |C^+|$ candidates.

$$\forall t_i \in \mathcal{T}, q \in [r]: \; 0 \leq d_i^q \leq b_i, \tag{16}$$

$$\forall q \in [r]: \sum_{t_i \in \mathcal{T}} d_i^q = k - p - |C^+|. \tag{17}$$

The following forces the semantics of the variables $\overline{\text{used}}$; that is, a variable $\overline{\text{used}}\,[d_i^q]$, $i \in [|\mathcal{T}|]$, $q \in [r]$, has value one if and only if variable d_i^q is at least one.

$$\forall t_i \in \mathcal{T}, q \in [r]: \; \overline{\text{used}}\,[d_i^q] \leq d_i^q, \tag{18}$$

$$\forall t_i \in \mathcal{T}, q \in [r]: \; \overline{\text{used}}\,[d_i^q] n \geq d_i^q. \tag{19}$$

Similarly, for the variables $\overline{\text{fullyused}}$, we ensure that $\overline{\text{fullyused}}\,[d_i^q]$, $i \in [|\mathcal{T}|]$, $q \in [r]$, is one if and only if manipulator u_q designates all available candidates of type t_i.

$$\forall t_i \in \mathcal{T}, q \in [r]: \; \overline{\text{fullyused}}\,[d_i^q] \geq 1 - (b_i - d_i^q), \tag{20}$$

$$\forall t_i \in \mathcal{T}, q \in [r]: \; b_i - d_i^q \leq n(1 - \overline{\text{fullyused}}\,[d_i^q]). \tag{21}$$

Since our task is to perform pessimistic tie-breaking, we have to ensure that the designated candidates for each manipulator are the candidates whom the manipulator gives the least utility. We impose this by forcing that the more valuable candidates (for a particular manipulator) are used only when all candidates of all less valuable types (for the manipulator) are used (i.e., they are fully used). To achieve this we make use of the $\overline{\text{used}}$ and $\overline{\text{fullyused}}$ variables in the following constraints.

$$\forall q \in [r] \cup \{0\} \forall t_i, t_{i'} \in \mathcal{T} \text{ with } t_i[q] > t_{i'}[q]: \; \overline{\text{used}}\,[d_i^q] \leq \overline{\text{fullyused}}\,[d_{i'}^q]. \tag{22}$$

Finally, we give the last set of constraints where s represents the pessimistic egalitarian k-egroup's utility which our ILP program wants to maximize:

$$\forall q \in [r] : \sum_{t_i \in \mathcal{T}} (d_i^q + \mathrm{obl}(t_i)) \cdot t_i[q] \geq s. \tag{23}$$

The ILP programs, for both tie-breaking variants, use $O(ru_{\mathrm{diff}}^r)$ variables. So, according to Lentra's result [34], we obtain fixed-parameter tractability with respect to the combined parameter $r + u_{\mathrm{diff}}$. Consequently, procedure $\mathcal{P}$ is in FPT with respect to the same parameter. □

Finally, by using ideas from Theorem 4, we proceed with Theorem 8—a counterpart of Theorem 7 but for lexicographic tie-breaking; namely, we obtain fixed-parameter tractability for egalitarian coalitional manipulation with lexicographic tie-breaking. As byproduct (following from Proposition 1), the fixed-parameter tractability from Theorem 8 also holds for the remaining tie-breaking variants: optimistic and pessimistic tie-breaking for both utilitarian and candidate-wise egalitarian variants.

Theorem 8 *Let r denote the number of manipulators and u_{diff} denote the number of different utility values. For every* $\mathrm{eval} \in \{\mathrm{util}, \mathrm{candegal}\}$, ℓ-Bloc-$\mathcal{F}$-egal-*Coalitional Manipulation with $\mathcal{F} \in \{\mathcal{F}_{\mathrm{lex}}, \mathcal{F}_{\mathrm{opt}}^{\mathrm{eval}}, \mathcal{F}_{\mathrm{pess}}^{\mathrm{eval}}\}$ parameterized by $r + u_{\mathrm{diff}}$ is fixed-parameter tractable.*

Proof The general proof idea is to show an algorithm which solves problem ℓ-Bloc-$\mathcal{F}_{\mathrm{lex}}$-egal-Coalitional Manipulation in the requested time. Proposition 1 implies that the result holds for all $\mathcal{F} \in \{\mathcal{F}_{\mathrm{lex}}, \mathcal{F}_{\mathrm{opt}}^{\mathrm{eval}}, \mathcal{F}_{\mathrm{pess}}^{\mathrm{eval}}\}$ for every $\mathrm{eval} \in \{\mathrm{util}, \mathrm{candegal}\}$.

To solve ℓ-Bloc-$\mathcal{F}_{\mathrm{lex}}$-egal-Coalitional Manipulation we create an ILP for all possible value combinations of the following parameters:

- the lowest final score $z < |V \cup W|$ of any member of the k-egroup and
- the candidate $\hat{c}$ which is the least preferred member of the k-egroup with final score z with respect to the tie-breaking rule $\mathcal{F}_{\mathrm{lex}}$.

Having z fixed, let C^+ denote the set of candidates which get at least $z + 1$ approvals from the non-manipulative votes or which are preferred to $\hat{c}$ with respect to $\mathcal{F}$ and get exactly z approvals from the non-manipulative votes. Assuming that the combination of parameter values is correct, all candidates from $C^+ \cup \{\hat{c}\}$ must belong to the k-egroup. We check whether $|C^+| < k$, that is, whether there is space for candidate $\hat{c}$ in the k-egroup. If the check fails, then we skip the corresponding combination of solution parameter values. Next, we ensure that $\hat{c}$ obtains final score exactly z. If $\hat{c}$ receives less than $z - r$ or more than z approvals from non-manipulative votes, then we discard this combination of solution parameter values. Otherwise, let $\hat{s} := z - \mathrm{score}_V(\hat{c})$ denote the number of additional approvals candidate $\hat{c}$ needs in order to get final score z.

We define the type of some candidate c_i to be the size-r vector $t_j = (u_1(c_i), u_2(c_i), \dots, u_r(c_i))$. We denote by $\mathcal{T} = \{t_1, t_2, \dots, t_{|\mathcal{T}|}\}$ the set of all possible types. Observe that $|\mathcal{T}| \leq u_{\mathrm{diff}}^r$. With each type vector t_i, $i \in [|\mathcal{T}|]$, we associate a set T_i consisting of all candidates of type t_i. Having $\hat{c}$ (and z) fixed, we distinguish candidates according to types further. For $j \in [r] \cup \{0\}$, all candidates with score $z - j$ (obtained from the non-manipulative votes) that are preferred (resp. not preferred) to candidate $\hat{c}$ according to $\mathscr{F}$, fall into group G_i^{j+} (resp. G_i^{j-}). We denote by C_r candidates that do not fall into any of such groups. For each type $t_i \in \mathscr{T}$ of a candidate, we define function $\mathrm{obl}(t_i) = |C^+ \cap T_i|$ which gives the number of candidates of type t_i who are obligatory part of the winning k-egroup. For each manipulator $q \in [r]$, by $g(q) := \sum_{t_i \in \mathscr{T}} \mathrm{obl}(t_i) \cdot t_i[q]$ we denote the *guaranteed utility* of q, that is, the utility achieved from the candidates in C^+.

We give the following ILP formulation of the problem using $2r|\mathscr{T}| + 2$ variables. For all groups G_i^{j+} and G_i^{j-}, $i \in [|\mathscr{T}|]$, $j \in [r] \cup \{0\}$, we introduce variables x_i^{j+} and x_i^{j-} respectively. The variables indicate, respectively, the number of candidates from groups G_i^{j+} and G_i^{j-} whom we push to the winning k-egroup. Also, we introduce two additional variables s and u. The former one represents the minimal value of the total utility achieved by manipulators. The latter one indicates the number of approvals which were spent without changing the outcome. To shorten the ILP we define

$$\mathrm{Mful}_z^{\hat{c}} := \sum_{t_i \in \mathscr{T}, j \in [r]} x_i^{j+} \cdot j + \sum_{t_i \in \mathscr{T}, j \in [r-1] \cup \{0\}} x_i^{j-} \cdot (j+1).$$

Intuitively, $\mathrm{Mful}_z^{\hat{c}}$ is the number of approvals used to add potential egroup members to the winning k-egroup. Also, we define

$$\mathrm{Fbid}_z^{\hat{c}} := \sum_{t_i \in \mathscr{T}, j \in [r] \cup \{0\}} \Big[(r-j+1)(|G_i^{j+}| - x_i^{j+}) + (r-j)(|G_i^{j-}| - x_i^{j-}) \Big].$$

$\mathrm{Fbid}_z^{\hat{c}}$ represents the number of approvals which cannot be used if one wants to avoid pushing candidates outside of the solution (given by values of the variables x) to the winning k-egroup; for example, if some candidate c needs j approvals to be part of the winning k-egroup, then we subtract $r-j+1$ approvals from the whole pool of r approvals for this candidate because we can use only $j-1$ approvals not to push c into the k-egroup. We define the following constraints to construct our program the goal of which is to maximize s:

$$\forall t_i \in \mathscr{T}, j \in [r] \cup \{0\}, \star \in \{+,-\} : x_i^{j\star} \leq |G_i^{j\star}|, \tag{24}$$

$$\forall t_i \in \mathscr{T}\colon x_i^{z-} = 0, \tag{25}$$

$$\forall t_i \in \mathscr{T}\colon x_i^{0+} = |G_i^{0+}|, \tag{26}$$

$$\mathrm{Mful}_z^{\hat{c}} \leq r \cdot \ell - \hat{s}, \tag{27}$$

$$(|C|-1)r - \mathrm{Mful}_z^{\hat{c}} - \mathrm{Fbid}_z^{\hat{c}} \geq u, \tag{28}$$

$$u + \mathrm{Mful}_z^{\hat{c}} = r \cdot \ell - \hat{s}, \tag{29}$$

$$\sum_{\substack{t_i \in \mathscr{T}, j \in [r] \cup \{0\}, \\ \star \in \{+,-\}}} x_i^{j\star} = k - |C^+| - 1, \tag{30}$$

$$\forall q \in [r]:\ g(q) + \sum_{\substack{t_i \in \mathscr{T},\ \star \in \{+,-\}, \\ j \in [r] \cup \{0\}}} x_i^{j\star} \cdot t_i[q] \quad \geq s. \tag{31}$$

Constraint (24) ensures that the candidates picked into a solution are available and can be part of the solution. Observe that candidates in G_i^{0+} have to be part of the solution and candidates in G_i^{z-} cannot be part of the solution. These two facts are ensured by Constraints (25) and (26). Constraint (27) forbids spending more approvals than possible to push some candidates to the k-egroup. The same role for "wasted" approvals plays Constraint (28). The upper bound of wasted approvals is counted in the following way: From the maximal number of possible approvals (we subtract one from the number of candidates because we give exactly $\hat{s}$ approvals to candidate $\hat{c}$), we first subtract the approvals already given to candidates in the k-egroup (i.e., $\mathrm{Mful}_z^{\hat{c}}$); next, we subtract all approvals that would push candidates outside of the solution given by the variables x to the k-egroup (i.e., $\mathrm{Fbid}_z^{\hat{c}}$). Constraint (29) ensures that, altogether, we spend exactly as many approvals as required, and Constraint (30) holds only when a proper number of candidates are pushed to be part of k-egroup. The last constraint forces maximization of the egalitarian utility of the winning k-egroup when s is maximized.

Using our technique we can obtain a solution by making $O(nm)$ ILPs with at most $2ru^r_{\mathrm{diff}} + 2$ variables. We return "yes" if there exists an ILP that achieves a k-egroup with the utility at least the given threshold and "no" otherwise. According to Lenstra's result [34], the constructed ILPs yield fixed-parameter tractability with respect to the combined parameter $r + u_{\mathrm{diff}}$. □

6 Conclusion

We developed a new model for and started the first systematic study of coalitional manipulation for multiwinner elections. Our analysis revealed that multiwinner coalitional manipulation requires models which are significantly more complex than those for single-winner coalitional manipulation or multiwinner non-coalitional manipulation. As described in the introduction, depending on the aggregation function our model may assume a given, fixed coalition of manipulators can compensate their (potential) utility loss after a manipulation in some way. Thus, in particular, our model does not analyze the dynamics of a coalition but rather it tries to assess its potential and possible influence. Finding the quality of possible manipulations for a given coalition is essential to answer more general questions about coalitions such as "what is the most profitable coalition for a given agent?". Being able to answer such questions, one can investigate the dynamics of coalitions formation. We discuss this future direction in more detail in the last paragraph of this section.

In our work, on the one hand, we generalized several tractability results for coalitional manipulation of ℓ-Approval by Conitzer et al. [17] and Lin [35] and for (non-coalitional) manipulation of Bloc by Meir et al. [38] and Obraztsova et al. [42] to tractability of coalitional manipulation of ℓ-Bloc in case of utilitarian or candidate-wise egalitarian evaluation of egroups. On the other hand, we showed that coalitional manipulation becomes intractable in case of egalitarian evaluation of egroups.

Let us discuss a few findings in more detail (Table 1 surveys all our results). We studied lexicographic, optimistic, and pessimistic tie-breaking and showed that, with the exception of egalitarian group evaluation, winning excellence-groups can be determined very efficiently. The intractability (NP-hardness, parameterized hardness in form of W[1]- and W[2]-hardness) for the egalitarian case, however, turns out to hold even for quite restricted scenarios. We also demonstrated that numerous tie-breaking rules can be "simulated" by (carefully chosen) lexicographic tie-breaking, again except for the egalitarian case. Interestingly, the hardness of egalitarian tie-breaking holds only for the optimistic case while for the pessimistic case it is efficiently solvable. Hardness for the egalitarian optimistic scenario, however, translates into hardness results for coalitional manipulation *regardless* of

Table 1 Computational complexity of tie-breaking and coalitional manipulation

$\mathscr{F}^{\text{eval}}_{\text{bhav}}$-Tie-Breaking, easy cases:		
Settings (evaluation, behavior)	Complexity	Reference
Utilitarian or cand.wise egalitarian, optimistic or pessimistic	$O(m \cdot (r + \log m))$	Cor. 1 †
Egalitarian, pessimistic	$O(r \cdot m \log m)$	Thm. 1
$\mathscr{F}^{\text{egal}}_{\text{opt}}$-Tie-Breaking (egalitarian, optimistic):		
Parameters, restrictions	Complexity	Reference
General	NP-hard	Thm. 1
k, 0/1 utilities and $q = 1$	W[2]-hard	Thm. 1
$r + k$	W[1]-hard	Thm. 2
$r + u_{\text{diff}}$	FPT	Thm. 3
ℓ-Bloc-$\mathscr{F}$-eval-Coalitional Manipulation		
Utilitarian/cand.wise egalitarian, optimistic/pessimistic:		
Restrictions	Complexity	Reference
General	$O(k^2 m^2 r(n + r))$	Thm. 4 ⋄
Consistent manipulators	$O(m(m + r + n))$	Prop. 3 ⋄
$\ell = k$	$O(m(m + r + n))$	Thm. 5 ⋄
ℓ-Bloc-$\mathscr{F}$-eval-Coalitional Manipulation		
Egalitarian, optimistic/pessimistic:		
Parameters, restrictions	Complexity	Reference
General	NP-hard	Thm. 6 ⋄
k, 0/1 utilities and $q = 1$	W[2]-hard	Thm. 6 ⋄
$r + k$	W[1]-hard	Thm. 6 ⋄
$r + u_{\text{diff}}$	FPT	Thm. 7 and Thm. 8 ⋄

Our results for ℓ-Bloc hold for any $\ell \geq 1$, and thus cover SNTV. The parameters are the size k of the egroup, the number r of manipulators, and the number u_{diff} of different utility values. Furthermore, m is the number of candidates and n is the number of voters. The result marked with † holds for all possible combinations of the respective evaluation and behavior variants. The results marked with ⋄ hold also for $\mathscr{F} = \mathscr{F}_{\text{lex}}$

the specific tie-breaking rule. On the contrary, coalitional manipulation becomes tractable for the other two evaluation strategies—"candidate-wise" egalitarian and utilitarian. Additionally, for few manipulators and few different utility values the manipulators assign to the candidates, manipulation becomes tractable also for the egalitarian optimistic scenario.

Our study provides a handful of practically efficient algorithms allowing for experimental studies of coalitional manipulability, as shown by Kalkbrenner [30], who implemented and tested our algorithms. Among many issues that such studies can address, there are a few remarkable ones like "is finding a successful manipulation hard in practice?", "how likely is a successful manipulation?", and "how much, in practice, can an election outcome be affected by a coalition?" (all questions were previously studied in the single-winner case). For example, Kalkbrenner [30] asked this kind of questions for the San Francisco Election Data from Preflib [37]. In her preliminary results, she showed that under certain election parameters already a group of significantly fewer than 0.1% of voters could have manipulated the elections successfully replacing at least one candidate. Even though this result sounds alarming, it is unsure how robust it is with respect to different election parameters. Thus, we find it interesting to apply our algorithms to empirically advance our understanding of manipulability of real-life elections, in particular San Francisco Election Data.

One may also want to consider further evaluation functions to model different variants of manipulators' behavior. One also technically natural variant is to consider $\min_{c \in S} \min_{u \in U} u(c)$, introducing a very pessimistic viewpoint of a coalition: "The worst candidate from the shortlist is finally chosen and the most-pessimistic expert from the coalition is right with its evaluation." This evaluation function would have in common with util and candegal that one can also assume without loss of generality that there is just one utility function (Observation 1). However, results do not directly translate because, for the evaluation, only the worst candidate in the k-excellence-group would matter. This property has a Chamberlin-Courant flavor, but with essentially only one voter, which suggests that one might expect computational tractability for our coalitional manipulation problems assuming the pessimistic evaluation in question.

In our study, we entirely focused on shortlisting as one of the simplest tasks for multiwinner elections to analyze our evaluation functions. It is interesting and non-trivial to develop models for multiwinner rules that aim for proportional representation or diversity. For shortlisting, extending our studies to non-approval-like scoring-based voting correspondences would be a natural next step. In this context, already seeing what happens if one extends the set of individual scores from being only 0 or 1 to more (but few) numbers is of interest. Moreover, we focused on deterministic tie-breaking mechanisms, ignoring randomized tie-breaking—another issue for future research.

An analysis of the manipulators' behavior, briefly mentioned at the beginning of this section, directing towards game theory seems promising as well. (Even more so since we identified polynomial-time algorithms for a few variants of coalitional manipulation.) One very interesting question about coalitions is, for example, whether a particular coalition is stable. Intuitively, the utility for every voter that is a part of the manipulating coalition should not be below the utility the voter receives when voting sincerely. This is of course only a necessary condition to ensure the stability of a coalition. A more sophisticated analysis of stability needs to consider game-theoretic aspects such as Nash or core stability [41].

Acknowledgements We thank the anonymous reviewers of *IJCAI '17* and *JAAMAS* for their constructive and valuable feedback. Robert Bredereck was from mid-September 2016 to mid-September 2017 on postdoctoral leave at the University of Oxford, supported by the DFG fellowship BR 5207/2. Work partially done while Robert Bredereck was with TU Berlin. Andrzej Kaczmarczyk was supported by the DFG project AFFA (BR 5207/1 and NI 369/15).

Funding Open Access funding enabled and organized by Projekt DEAL.

References

1. Aziz, H., Gaspers, S., Gudmundsson, J., Mackenzie, S., Mattei, N., & Walsh, T. (2015). Computational aspects of multi-winner approval voting. In *Proceedings of the 14th international conference on autonomous agents and multiagent systems, AAMAS '15* (pp. 107–115).
2. Aziz, H., Brill, M., Conitzer, V., Elkind, E., Freeman, R., & Walsh, T. (2017a). Justified representation in approval-based committee voting. *Social Choice and Welfare, 48*(2), 461–485.
3. Aziz, H., Elkind, E., Faliszewski, P., Lackner, M., & Skowron, P. (2017b). The Condorcet principle for multiwinner elections: from shortlisting to proportionality. In *Proceedings of the 26th international conference on artificial intelligence, IJCAI '17* (pp. 84–90).
4. Barberà, S., & Coelho, D. (2008). How to choose a non-controversial list with *k* names. *Social Choice and Welfare, 31*(1), 79–96.
5. Barberà, S., & Coelho, D. (2010). On the rule of *k* names. *Games and Economic Behavior, 70*(1), 44–61.
6. Barberà, S., Sonnenschein, H., & Zhou, L. (1991). Voting by committees. *Econometrica, 59*(3), 595–609.
7. Barrot, N., Gourvès, L., Lang, J., Monnot, J., & Ries, B. (2013). Possible winners in approval voting. In *Proceedings of the 3rd international conference on algorithmic decision theory, ADT '13* (pp. 57–70).
8. Bartholdi, J. J., III., Tovey, C. A., & Trick, M. A. (1989). The computational difficulty of manipulating an election. *Social Choice and Welfare, 6*(3), 227–241.
9. Baumeister, D., & Rothe, J. (2015). Preference aggregation by voting, chap 4. In J. Rothe (Ed.), *Economics and computation: An introduction to algorithmic game theory, computational social choice, and fair division* (pp. 197–325). Berlin: Springer.
10. Betzler, N., Slinko, A., & Uhlmann, J. (2013). On the computation of fully proportional representation. *Journal of Artificial Intelligence Research, 47,* 475–519.
11. Brandt, F., Conitzer, V., Endriss, U., Lang, J., & Procaccia, A. D. (Eds.). (2016). *Handbook of computational social choice*. Cambridge: Cambridge University Press.
12. Bredereck, R., Kaczmarczyk, A., & Niedermeier, R. (2017). On coalitional manipulation for multiwinner elections: Shortlisting. In *Proceedings of the 26th international joint conference on artificial intelligence, IJCAI '17* (pp. 887–893).
13. Caragiannis, I., Kalaitzis, D., & Markakis, E. (2010). Approximation algorithms and mechanism design for minimax approval voting. In *Proceedings of the 24th AAAI conference on artificial intelligence (AAAI '10)* (pp. 737–742). AAAI Press.
14. Chamberlin, J. R., & Courant, P. N. (1983). Representative deliberations and representative decisions: Proportional representation and the Borda rule. *American Political Science Review, 77*(3), 718–733.
15. Coleman, T., & Teague, V. (2007). On the complexity of manipulating elections. In *Proceedings of computing: The 13th Australasian theory symposium* (pp. 25–33).
16. Conitzer, V., & Walsh, T. (2016). Barriers to manipulation in voting, chap 6. In F. Brandt, V. Conitzer, U. Endriss, J. Lang, & A. D. Procaccia (Eds.), *Handbook of computational social choice* (pp. 126–145). Cambridge: Cambridge University Press.
17. Conitzer, V., Sandholm, T., & Lang, J. (2007). When are elections with few candidates hard to manipulate? *Journal of the ACM, 54*(3), 1–33.
18. Cygan, M., Fomin, F. V., Kowalik, L., Lokshtanov, D., Marx, D., Pilipczuk, M., et al. (2015). *Parameterized algorithms*. Berlin: Springer.
19. Davies, J., Katsirelos, G., Narodytska, N., Walsh, T., & Xia, L. (2014). Complexity of and algorithms for the manipulation of Borda, Nanson's and Baldwin's voting rules. *Artificial Intelligence, 217,* 20–42.

20. Debord, B. (1992). An axiomatic characterization of Borda's k-choice function. *Social Choice and Welfare, 9*(4), 337–343.
21. Downey, R. G., & Fellows, M. R. (2013). *Fundamentals of parameterized complexity*. Berlin: Springer.
22. Elkind, E., & Rothe, J. (2015). Cooperative game theory, chap 3. In J. Rothe (Ed.), *Economics and computation: an introduction to algorithmic game theory, computational social choice, and fair division* (pp. 135–193). Berlin: Springer.
23. Elkind, E., Faliszewski, P., Skowron, P., & Slinko, A. M. (2017). Properties of multiwinner voting rules. *Social Choice and Welfare, 48*(3), 599–632.
24. Erdélyi, G., Fellows, M. R., Rothe, J., & Schend, L. (2015). Control complexity in Bucklin and fallback voting: An experimental analysis. *Journal of Computer and System Sciences, 81*(4), 661–670.
25. Faliszewski, P., Skowron, P., Slinko, A. M., & Talmon, N. (2016). Multiwinner analogues of the plurality rule: Axiomatic and algorithmic perspectives. In *Proceedings of the 30th AAAI conference on artificial intelligence, AAAI '16* (pp. 482–488).
26. Faliszewski, P., Skowron, P., Slinko, A. M., & Talmon, N. (2017a). Multiwinner voting: A new challenge for social choice theory, chap 2. In U. Endriss (Ed.), *Trends in computational social choice* (pp. 27–47). New York: AI Access.
27. Faliszewski, P., Skowron, P., & Talmon, N. (2017b). Bribery as a measure of candidate success: Complexity results for approval-based multiwinner rules. In *Proceedings of the 16th international conference on autonomous agents and multiagent systems, AAMAS '17* (pp. 6–14).
28. Fellows, M. R., Hermelin, D., Rosamond, F., & Vialette, S. (2009). On the parameterized complexity of multiple-interval graph problems. *Theoretical Computer Science, 410*(1), 53–61.
29. Flum, J., & Grohe, M. (2006). *Parameterized complexity theory*. Berlin: Springer.
30. Kalkbrenner, L. (2019). Coalitional manipulation for multiwinner elections: Algorithms and experiments. Bachelor thesis. http://fpt.akt.tu-berlin.de/publications/theses/BA-lydia-kalkbrenner.pdf.
31. Kellerer, H., Pferschy, U., & Pisinger, D. (2004). *Knapsack problems*. Berlin: Springer.
32. Klaus, B., Manlove, D. F., Rossi, F., Aziz, H., Savani, R., Chalkiadakis, G., & Wooldridge, M. (2016). Coalitional formation. In F. Brandt, V. Conitzer, U. Endriss, J. Lang, & A. D. Procaccia (Eds.), *Handbook of computational social choice* (pp. 331–396). Cambridge: Cambridge University Press (chap 14–16, part 3).
33. Lackner, M., & Skowron, P. (2018). Approval-based multi-winner rules and strategic voting. In *Proceedings of the 27th international joint conference on artificial intelligence, IJCAI '18* (pp. 340–346).
34. Lenstra, H. W. (1983). Integer programming with a fixed number of variables. *Mathematics of Operations Research, 8*(4), 538–548.
35. Lin, A. (2011). The complexity of manipulating *k*-approval elections. In *Proceedings of the 3rd international conference on agents and artificial intelligence, ICAART '11* (pp. 212–218).
36. Lu, T., Tang, P., Procaccia, A. D., & Boutilier, C. (2012). Bayesian vote manipulation: Optimal strategies and impact on welfare. In *Proceedings of the 28th conference on uncertainty in artificial intelligence, UAI '12* (pp. 543–553).
37. Mattei, N., & Walsh, T. (2013). Preflib: A library for preferences. In *Proceedings of the 3nd international conference on algorithmic decision theory, ADT '13* (pp. 259–270).
38. Meir, R., Procaccia, A. D., Rosenschein, J. S., & Zohar, A. (2008). Complexity of strategic behavior in multi-winner elections. *Journal of Artificial Intelligence Research, 33*(1), 149–178.
39. Ministry of Science and Higher Education of the Republic of Poland. (2019). Informations on the election of The Board of Research Excellence (in Polish). http://www.bip.nauka.gov.pl/g2/oryginal/2019_03/c435c5061f0aab7158eba2716553f240.pdf. Accessed July 30, 2019
40. Niedermeier, R. (2006). *Invitation to fixed-parameter algorithms*. Oxford: Oxford University Press.
41. Nisan, N., Roughgarden, T., Tardos, É., & Vazirani, V. V. (2007). *Algorithmic game theory*. Cambridge: Cambridge University Press.
42. Obraztsova, S., Zick, Y., & Elkind, E. (2013). On manipulation in multi-winner elections based on scoring rules. In *Proceedings of the 12th international conference on autonomous agents and multiagent systems, AAMAS '13* (pp. 359–366).
43. Scheuerman, J., Harman, J. L., Mattei, N., & Venable, K. B. (2019). Heuristics in multi-winner approval voting. CoRR abs/1905.12104.
44. Skowron, P. (2015). What do we elect committees for? A voting committee model for multi-winner rules. In *Proceedings of the 24th international conference on artificial intelligence, IJCAI '15* (pp. 1141–1147).
45. Skowron, P., Faliszewski, P., & Slinko, A. M. (2015). Achieving fully proportional representation: Approximability results. *Artificial Intelligence, 222,* 67–103.
46. Tideman, N., & Richardson, D. (2000). Better voting methods through technology: The refinement-manageability trade-off in the single transferable vote. *Public Choice, 103*(1–2), 13–34.

47. Walsh, T. (2011). Where are the hard manipulation problems? *Journal of Artificial Intelligence Research, 44,* 1–29.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.