Participatory Budgeting Project Strength via Candidate Control

Piotr Faliszewski, Łukasz Janeczko, Dušan Knop, Jan Pokorný,
Šimon Schierreich, Mateusz Słuszniak, and Krzysztof Sornat

Abstract

We study the complexity of candidate control in participatory budgeting elections. The goal of constructive candidate control is to ensure that a given candidate wins by either adding or deleting candidates from the election (in the destructive setting, the goal is to prevent a given candidate from winning). We show that such control problems are NP-hard to solve for many participatory budgeting voting rules, including Phragmén and Equal-Shares, but there are natural cases with polynomial-time algorithms. We also argue that control by deleting candidates is a useful tool for assessing the performance (or, strength) of initially losing projects, and we support this view with experiments on real-life PB instances.

1 Introduction

Participatory budgeting is a recent democratic innovation where cities allow their inhabitants to decide about a certain fraction of their budgets [9, 16, 26]. Specifically, some of the community members propose possible projects to be implemented and, then, all the citizens get a chance to vote as to which of them should be funded. Most commonly, such elections use approval ballots, where people indicate which projects they would like to see implemented, and the GreedyAV rule, which selects the most approved projects (subject to not exceeding the budget). However, there also are more advanced rules, such as Phragmén [8, 21] or Equal-Shares [24, 25], which produce arguably more fair—or, to be precise, more proportional—decisions [14]. Yet, with more advanced rules come issues about understanding the results. Indeed, recently [6] have argued that proposers whose projects were rejected may find it quite difficult to understand the reasons for these outcomes. To alleviate this problem, they introduced a number of performance measures—mostly based on the bribery family of problems [13, 12]—that attempt to answer the following question: As a proposer of a project that was not funded, what could I have done differently to have it funded? For example, they ask if the project would have been funded if its cost were lower (see also the work of [3]), or if its proposer convinced more people to vote for it, or if the proposer motivated some voters to only approve his or her project. Similar bribery-style problems were also used to evaluate the robustness of election results [27, 7, 2, 4, 5], or the margin of victory for the winners [22, 28].

In this paper, we follow-up on these ideas, but using candidate control. The main difference is that instead of focusing on circumstances that depend on a project's proposer (indeed, the project's cost is his or her choice, and it is his or her choice what support campaign he or she runs), we focus on external ones, independent of his or her actions (such as some other projects being submitted or not[1]). We believe that looking at both types of reasons for a project's rejection gives a more complete view of its performance.

[1]We disregard the possibility that a proposer might try to discourage other people from proposing projects, albeit we acknowledge that this may happen, and this might even be quite benign: A group of activists focused on making their city more green may discuss among themselves which projects to submit and which to withhold.

	unit		unary		binary	
	Del	Add	Del	Add	Del	Add
GreedyAV	P‡	P‡	P‡	P‡	NP-c	NP-c
GreedyCost	P‡	P‡	P‡	P‡	NP-c	NP-c
Phragmén	NP-c	NP-c	NP-c	NP-c	NP-c	NP-c
Equal-Shares	NP-c	NP-c	NP-c	NP-c	NP-c	NP-c

Table 1: A basic overview of our complexity results. In the first column, we list the rules we are interested in. All the remaining columns contain the complexity classification of our problem in one of the three variants: by `unit`, we mean that all input projects are of the same cost, `unary` stands for cases where the costs are of size polynomial in $n + m$, and `binary` applies for the variant where costs need to be encoded in binary (and hence can be exponential in n and m). By Del (Add, respectively), we mean that the control operation is project deletion (addition). The complexity classification is the same for both constructive and destructive objectives. Results marked with ‡ hold even in the weighted version of the control where projects' weights are encoded in binary.

Candidate Control. The idea of the control-in-elections family of problems is that we are given a description of an election, a designated candidate, and we ask if it is possible to ensure that this candidate is a winner (in constructive control) or ceases to be a winner (in destructive control) by modifying the structure of the election [1, 12]. Specifically, researchers consider control by adding or deleting either candidates or voters (some papers—including the one that introduced election control [1]—also consider various forms of arranging run-off elections as a type of control). So far, election control has been mostly studied theoretically, with a focus on the complexity analysis [1, 18, 23, 20, 10, 29], but some empirical results exist as well [11]. We study candidate control in participatory budgeting, that is, we ask if it is possible to ensure funding of a given project (or, preclude its funding) by either adding new projects—from some apriori known set of projects—or by deleting them. Our results are theoretical and focus on the complexity of our problems, but we motivate them by project performance analysis and we also show some experimental results that provide examples of such analysis. As our performance analysis is based on control by deleting projects, we pay most attention to results regarding this variant of control, and we include the addition case for the sake of completeness and to be in sync with the preceding literature.

Performance Analysis. Let us now discuss how one could use control by deleting candidates to analyze the performance of projects in participatory budgeting (we will use the terms "projects" and "candidates" interchangeably, e.g., using "candidates" in the names of control problems). Consider a participatory budgeting election and some not-funded project p. One basic measure of its performance is the smallest number of other projects that have to be removed from the election for p to be funded. The lower this number, the closer was the project to winning: Indeed, perhaps some proposers only managed to submit their projects in the last minute and it was possible that they would have missed the deadline, or some projects were close to being removed from the election due to formal reasons, but the city officials were not strict in this regard. However, it is more likely that such issues would affect cheaper projects than the more expensive ones—which, likely, had more careful proposers—so instead of asking for a smallest set of projects to delete, we may ask for a set with the lowest total cost.

Another way of using control by deleting projects to assess a project's performance is to use a probabilistic approach, along the lines of the one taken by [4], [5], and [2] for bribery: We ask for the probability that project p is funded assuming that a random subset of projects (of a given cardinality) is removed. The higher it is, and the lower is the number of removed projects, the closer was project p to winning.

A different interpretation of the above measures is that instead of thinking that some projects "barely made it" to participate in the election, we learn how many projects performed better than p. The more projects we need to delete to have p funded (or, to have p funded with sufficiently high probability) the more projects can be seen as critically stronger than p.

Finally, we can use candidate control as a way of assessing rivalry between projects. For example, if project p has a much higher probability of being funded after deleting a random set of projects under the condition that some other project q was included in this set, then we can view q as a strong rival of p.

Contributions. We provide theoretical and experimental results. First, we consider the complexity of candidate control for four well-known voting rules, depending on how the costs of the projects are encoded (either in binary, or in unary, or as unit costs, which means that each project costs the same amount). We show the overview of our results in Table 1. We mention that all our NP-hardness results also imply #P-hardness for respective problems where we ask for a number of solutions (e.g., number of ways in which we can ensure a victory of a given project by deleting a given number of others). This is interesting because solving such problems is necessary for estimating the probability that a project wins if a given randomly-selected set of projects is deleted. On the experimental side, we provide an analysis of real-world participatory budgeting instances, showing what one can learn about them via candidate control. To this end, we provide several performance measures and prove their usefulness in an extensive analysis of real-world PB instances.

We refer the interested reader to the full version of our paper for all missing details and proofs [15].

2 Preliminaries

An instance of *participatory budgeting* (PB) is a triple $E = (P, V, B)$, where $P = \{p_1, \dots, p_m\}$ is a set of *projects*, $V = \{v_1, \dots, v_n\}$ is a set of voters, and $B \in \mathbb{N}$ is an available budget. Each voter $v \in V$ is associated with an *approval set* $A(v) \subseteq P$, which is the set of projects they approve. For each project $p \in P$, we know $\mathrm{cost}(p) \in \mathbb{N}_{\geq 1}, \mathrm{cost}(p) \leq B$, i.e., the price for which this project can be implemented. We extend this notation from a single project p to a set of projects $S \subseteq P$ and set $\mathrm{cost}(S) = \sum_{p \in S} \mathrm{cost}(p)$. We say that $S \subseteq P$ is a set of B-*feasible* projects if $\mathrm{cost}(S) \leq B$.

A PB rule is a function $f \colon E \to 2^P$ that, for a given PB instance, outputs a B-feasible subset of projects. Note that we assume the rules to be resolute, which can be easily ensured by incorporating some tie-breaking order into them. Let $W = f(E)$ for some rule f and some PB instance E. We say that projects from W are *selected* or, equivalently, *funded*. The projects not in W are called *losing*.

In this work, we consider four different participatory budgeting rules. Each of these rules starts with an empty set W and sequentially, in rounds, extends W with additional projects unless the budget B is exhausted or all the projects were processed. More formally, given an instance $E = (P, V, B)$ of PB, the rules we consider work as follows:

GreedyAV. We define the score of a project $p \in P$ as the number of voters approving p; formally $\mathrm{score}_{\mathrm{AV}}(p) = |\{v \in V \mid p \in A(v)\}|$. The GreedyAV rule then processes the projects in non-increasing order according to their scores (with ties resolved according to a given tie-breaking order). If the rule can afford the currently processed project p, i.e., $\mathrm{cost}(W) + \mathrm{cost}(p) \leq B$, then it includes p in W. Otherwise, the rule continues with the next project. The rule terminates when all projects were processed.

GreedyCost. This rule is very similar to the GreedyAV rule. The only difference is in the order in which the projects are processed. Specifically, the score of a project $p \in P$ under GreedyCost rule is

$\text{score}_{\text{AV/c}}(p) = \text{score}_{\text{AV}}(p)/\text{cost}(p)$. The process is then identical to the one for the GreedyAV rule.

Phragmén. This rule is conceptually different from the two above. Here, each voter starts with an empty virtual bank account, which is, in a continuous manner, increased by one unit of money per unit of time. Once there is a project $p \in P \setminus W$ such that the sum of balances of voters approving p is exactly $\text{cost}(p)$, the current round ends, and the rule performs several steps. First, it includes project p into the set of funded projects. Next, it sets the balance of all voters approving p to zero. Finally, the rule removes all projects $p' \in P \setminus W$ such that $\text{cost}(W) + \text{cost}(p') > B$. Then, the rule continues with the next round. The rule terminates when there are no remaining projects in P.

Equal-Shares. Our last rule is also based on the idea of virtual bank accounts. However, this time, the budget is proportionally spread among the voters, meaning that each voter starts with the initial balance of B/n, and this initial value is never increased. Again, the rule works in rounds. In each round, the rule funds a project such that its supporters have enough cumulative budget to fund this project, and each of them covers as small a fraction of its cost as possible. Formally, let b_i be the current balance of a voter v_i. We say that a project $p \in P \setminus W$ is q-affordable, where $q \in [0, 1]$, if

$$\sum\nolimits_{v_i \in A(p)} \min\left(b_i, q \cdot \text{cost}(p)\right) = \text{cost}(p).$$

In each round, the rule funds a project that is q-affordable for the smallest q over all projects and adjusts the balances of voters supporting p: Specifically, the balance b_i of each agent is decreased by $q_i \cdot \text{cost}(p)$, where $q_i = q$ if $b_i > q \cdot \text{cost}(p)$ and $b_i/\text{cost}(p)$ otherwise. The rule terminates when no affordable project exists.

Control Problems. We focus on the control by adding or deleting projects and follow the standard notation from single- and multi-winner voting [12]. Let f be a PB rule. In the f-Constructive Control by Deleting Candidates projects (f-CCDC, for short), we are given an instance $E = (P, V, B)$ of PB, an integer r, and a project $p \notin f(E)$, and our goal is to decide whether it is possible to delete at most r projects D such that $p \in f((P \setminus D, V, B))$. In the f-Destructive Control by Deleting Candidates (f-DCDC), the project p is initially funded, and the goal is to decide whether we can delete at most r projects so that project p becomes a loser.

In control by adding projects, there are two disjoint sets of projects: P is a set of standard projects, and Q is a set of spoiler projects. The rule does not initially consider the spoiler projects. The question here is whether we can find at most r spoiler projects such that once we add them to the instance, then project p is (in the case of f-CCAC) or is not (in the case of f-DCAC) funded by the rule f.

In our algorithmic results, we are sometimes interested in the weighted variants of the above-defined problems. Under this consideration, each project p is additionally associated with its weight $\omega(p) \in \mathbb{N}$, and the goal is to decide whether a set D of projects securing our goal exists such that $\sum_{p \in D} \omega(p) \leq r$. We indicate the weighted variant by adding a dollar sign in front of the operation type. For example, the weighted variant of f-CCDC is referred to as f-CC\$DC. Even though the weighted variant might seem unnatural at first glance, the motivation for it is two-fold. First, it is studied in literature on control in elections [12]. Second, and more importantly, if we set the weight of each project equal to its cost, we can use a hypothetical algorithm for the weighted variant to find a set of the lowest total costs that secures our goal, one of the proposed performance measures.

3 Complexity Results

We start with the complexity picture of both constructive and destructive control by deleting projects. This operation is arguably more natural for real-life instances and is also assumed in our experimental results. Before we present our results, let us illustrate the concept of control by deleting projects using a toy example.

Example 1. *Assume an instance with three projects c_1, c_2, and p. The project c_1 is approved by three voters, the project c_2 by two voters, and project p by a single voter. The costs of the projects are $\mathrm{cost}(c_1) = 1$, $\mathrm{cost}(c_2) = 2$, and $\mathrm{cost}(p) = 1$. The total budget is $B = 2$. Under the GreedyAV rule, project c_1 is considered first, and since its cost is less than the budget, this project is funded. Next, the rule considers project c_2, but this project costs more than the remaining budget. Lastly, project p is considered and eventually also funded. If we remove project c_1, then project c_2 gets funded and exhausts the budget, and therefore, project p cannot be funded. Hence, removing c_1 from the instance is a successful destructive control that prevents p from winning and a successful control that makes c_2 win.*

We start with the GreedyAV rule. In our first result, we show that both constructive and destructive control are computationally intractable, even if the instance is unweighted. Maybe surprisingly, this hardness result holds even if the instance contains only two agents (we give a reduction from the classic NP-complete [17] problem RX3C, see below).

Definition 1. *In the RX3C problem, we are given a universe $U = \{u_1, \ldots, u_{3N}\}$ and a family $\mathcal{S}$ of $3N$ size-3 subsets $S_1, \ldots, S_{3N} \subset U$ such that every element $u_i \in U$ appears in exactly 3 subsets of $\mathcal{S}$. We ask if there are N subsets in $\mathcal{S}$ whose union is U.*

Theorem 1. *Both GreedyAV-CCDC and GreedyAV-DCDC are NP-complete, even if $|V| = 2$.*

Proof Sketch. To show NP-hardness, we give a reduction from RX3C. We focus only on the constructive variant and consider the destructive one in the full version.

The idea of the proof is that the projects are in 1-to-1 correspondence with the sets, and using project costs, we encode which elements are covered by each set. We achieve this by having project costs as a $3N$ digit-length number in base 4. A project p_j corresponding to a subset $S_j = \{u_{i_1}, u_{i_2}, u_{i_3}\} \in \mathcal{S}$, $i_1 < i_2 < i_3$, then has cost of the form

$$\underset{3N}{0}\,\underset{}{0}\,\underset{}{0}\cdots\underset{i_3}{0}\,\underset{}{1}\,\underset{}{0}\,\underset{}{0}\cdots\underset{i_2}{0}\,\underset{}{1}\,\underset{}{0}\,\underset{}{0}\cdots\underset{i_1}{0}\,\underset{}{1}\,\underset{}{0}\,\underset{}{0}\cdots\underset{2}{0}\,\underset{1}{0},$$

where the i-th digit of this cost has value one if and only if the element u_i belongs to S_j. All other digits are always zero. Next, we set the budget and the cost of our distinguished project p so that it gets funded only if the deleted projects correspond to subsets that form an exact cover in $\mathcal{I}$.

Formally, given an instance $\mathcal{I} = (U, \mathcal{S})$, we construct an instance $\mathcal{J}$ of GreedyAV-CCDC as follows. For each set $S_j \in \mathcal{S}$, $S_j = \{u_{i_1}, u_{i_2}, u_{i_3}\}$, we create a *set-project* p_j with cost $1 \cdot 4^{i_1} + 1 \cdot 4^{i_2} + 1 \cdot 4^{i_3}$. Next, we add our *distinguished project* p and $N + 1$ *guard-projects* $g_1, \ldots, g_{N+1}$. The cost of the distinguished project p is $\sum_{i=1}^{3N} 1 \cdot 4^i$ and for every $i \in [N + 1]$, we set $\mathrm{cost}(g_i) = \mathrm{cost}(p) + 1$. That is, the guard projects are only one unit more expensive compared to our distinguished project. This ensures that the budget left after we delete some set-projects is exactly the cost of p.

The set of voters consists of just two voters, v_1 and v_2. The first voter, v_1, approves all projects except for p. The second voter, v_2, approves only the set-projects. Such a preference profile secures that, regardless of the tie-breaking order, the method first processes all set-projects, then all guard-projects, and only as the last possibility, the method processes the distinguished project p.

To complete the construction, we set $B = \sum_{i=1}^{3N} 3 \cdot 4^i$ and $r = N$. Observe that the number of guard-projects is one greater than the number of projects we are allowed to delete; hence, no solution may delete all guard-projects. The budget is selected so that if we do not delete any project, all the set-projects are funded, and the budget is exhausted after the last set-project is taken into consideration by the rule. On the other hand, if we remove N projects corresponding to subsets forming an exact cover in $\mathcal{I}$, the remaining budget after the rule processes all the set-projects will be exactly the cost of p.

First, let us show that p is indeed initially not funded. The $\mathrm{score}_{\mathrm{AV}}$ of every set-project p_j, $j \in [3N]$, is exactly two, the $\mathrm{score}_{\mathrm{AV}}$ of the guard-projects is exactly one, while the $\mathrm{score}_{\mathrm{AV}}$ of our distinguished

project p is zero. Therefore, all other projects are processed before project p. Moreover, their total cost is $\sum_{i=1}^{3N} 3 \cdot 4^i$ due to the definition of the costs and the fact that every element u_i appears in exactly three subsets. Therefore, when the distinguished project p is processed by the rule, the budget is $B - \sum_{i=1}^{3N} 3 \cdot 4^i = \sum_{i=1}^{3N} 3 \cdot 4^i - \sum_{i=1}^{3N} 3 \cdot 4^i = 0$. Hence, project p is clearly not affordable.

For left-to-right implication, let $\mathcal{I}$ be a YES-instance and let $C \subset \mathcal{S}$ be an exact cover of U. We delete every set-project p_j such that $S_j \in C$, and we claim that p is now funded, that is, the control is successful. Since C is an exact cover, we spend exactly $\sum_{i=1}^{3N} 2 \cdot 4^i$ on the set-projects. Consequently, after the last set-project is processed by the rule, the remaining budget is $B - \sum_{i=1}^{3N} 2 \cdot 4^i = \sum_{i=1}^{3N} 3 \cdot 4^i - \sum_{i=1}^{3N} 2 \cdot 4^i = \sum_{i=1}^{3N} (3-2) \cdot 4^i = \sum_{i=1}^{3N} 1 \cdot 4^i$. This is one unit of money less than the cost of any guard-project. Therefore, no guard-project is funded, and once the rule processes p, the remaining budget is still $\sum_{i=1}^{3N} 1 \cdot 4^i$ which is the cost of p, so p is funded. Consequently, $\mathcal{J}$ is also a YES-instance, and C is a solution. □

The hardness construction from Theorem 1 requires prices of exponential size. That is, our problems are, from the computational complexity perspective, weakly NP-hard. It is natural to ask whether Theorem 1 can be strengthened to polynomial-size prices or if a pseudopolynomial time algorithm exists for the problem. In the following result, we answer this question positively by giving an algorithm based on dynamic programming, which works even in the case of weighted control.

Theorem 2. *If the costs of the projects are encoded in unary, both GREEDYAV-CC$DC and GREEDYAV-DC$DC can be solved in polynomial time, even if the projects' weights are encoded in binary.*

Proof Sketch. Our algorithm is based on the dynamic programming approach. We first present an algorithm for constructive control and later show what needs to be changed to also solve the destructive variant of control by deleting projects.

We suppose that $p_1, \ldots, p_{m-1}, p$ is the order in which the rule processes the projects in the original instance; that is, p_1 is processed first and p_ℓ is processed just before the distinguished project p. Note that the assumption that p is processed last is not in contradiction with the fact that our algorithm works for any tie-breaking order, as we can remove all projects that the GREEDYAV rule processes after the distinguished project p. Moreover, we can remove all projects with $\text{cost}(p_j) > B$, as such projects cannot be afforded under any condition.

The central part of the algorithm is to compute a dynamic programming table $\text{DP}[j, \beta]$, where $j \in [m-1]$ is an index of the last processed project, and $\beta \in [B]$ is a desired remaining budget just before the rule processes a project p_{j+1}. We call the pair (j, β) a *signature*. For every signature, the dynamic programming table stores the weight of a minimum-weight *partial solution* $D_{j,\beta} \subseteq \{p_1, \ldots, p_j\}$ such that if the projects from $D_{j,\beta}$ are removed, the remaining budget just before the GREEDYAV rule processed project p_{j+1} is exactly β. If no such partial solution exists, we store some large value ∞ (in fact, it is enough to store any value greater than r).

The computation is then defined as follows. The basic step is when $j = 1$. Here, we just decide whether p_1 needs to be deleted or not, based on the required remaining budget β. Formally, we set the dynamic programming table as follows:

$$\text{DP}[1, \beta] = \begin{cases} 0 & \text{if } \beta = B - \text{cost}(p_1), \\ \omega(p_1) & \text{if } \beta = B, \text{ and} \\ \infty & \text{otherwise.} \end{cases}$$

For every $j \in [2, m-1]$, the computation of the algorithm is defined as follows.

$$\mathrm{DP}[j,\beta] = \begin{cases} \min\{\mathrm{DP}[j-1,\beta] + \omega(p_j), \mathrm{DP}[j-1,\beta+\mathrm{cost}(p_j)]\} \\ \qquad\qquad \text{if } \mathrm{DP}[j-1,\beta] + \omega(p_j) \leq r \text{ and } \beta + \mathrm{cost}(p_j) \leq B, \\ \mathrm{DP}[j-1,\beta] \\ \qquad\qquad \text{if } \beta + \mathrm{cost}(p_j) > B, \\ \mathrm{DP}[j-1,\beta+\mathrm{cost}(p_j)] \\ \qquad\qquad \text{if } \mathrm{DP}[j-1,\beta] + \omega(p_j) > r. \end{cases}$$

The first case corresponds to a situation where we need to decide whether to include p_j in a solution or not. In the second case, we cannot fund p_j anyway, so we need not delete it, and we are only interested in whether the same budget can be achieved just before p_j is processed. In the last case, we cannot delete p_j, as it would exceed the budget.

Once all the cells of the dynamic programming table DP are correctly computed, we can decide the instances. Specifically, we return Yes whenever there exists a cell $\mathrm{DP}[m-1,\beta]$, where $\beta \geq \mathrm{cost}(p)$, such that $\mathrm{DP}[m-1,\beta] \leq r$. The dynamic programming table has $\mathcal{O}(m \cdot B)$ cells, and each cell can be computed in time $\mathcal{O}(\log(r))$. The final check can be done in $\mathcal{O}(B)$ time; therefore, the overall running time of the algorithm is $\mathcal{O}((m \cdot B) \cdot \log(r) + B)$, which, assuming the unary-encoded budget, is clearly a polynomial-time algorithm, even if the projects' weights are encoded in binary. □

Theorem 2 implies that for real-life elections, performance measures based on project control can be computed efficiently. Indeed, we implemented this algorithm and we use it in our experimental analysis in Section 4.

The hardness construction provided in Theorem 1 and the algorithm from Theorem 2 also work for the GreedyCost rule. For the former result, one can observe that even under the GreedyCost rule, the property that the set-projects are processed before the guard-projects, and that the guard-projects are processed before the distinguished project p is preserved. This comes from the fact that set-projects are approved by exactly two voters, guard-projects by exactly one voter, and p by no voter. Moreover, no set-project is more expensive than any guard-project. The algorithms require that the relative ordering of the projects is not affected by deletions. This is also clearly preserved under GreedyCost.

Corollary 1. *Both GreedyCost-CCDC and GreedyCost-DCDC are NP-complete, even if $|V| = 2$. If the projects' costs are encoded in unary, even weighted control can be solved in polynomial time.*

Now, we turn our attention to the Phragmén rule. Here, the situation is significantly less positive. Specifically, in the following theorem, we show that for this rule, it is NP-hard to decide whether successful control is possible, even if the instance is unweighted and all projects are of the same cost.

Theorem 3. *Both Phragmén-CCDC and Phragmén-DCDC are NP-complete, even if the projects are of unit cost.*

The idea behind the construction is that we have one project for every set $S_i \in \mathcal{S}$ in the RX3C instance and many direct competitors of the distinguished project p. The set-projects have significantly higher support than p or its competitors, and, moreover, the competitors of p share their voters with the set-projects. Hence, all the set-projects are always funded before the first project of a different type may be funded. These set-projects exhaust most of the budget, and unless every voter approving a competitor of p approves a funded set-project, the remainder of the budget is spent on the competitor of p, which is, consequently, not funded.

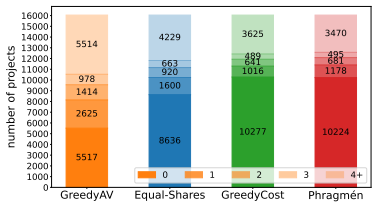


Figure 1: The distribution of projects according to their optimal control size. Each bar represents one rule and is partitioned into five parts whose sizes correspond to the number of projects that require $r \in \{0, 1, 2, 3, 4+\}$ deletions to get funded. The part with winning projects is always the darkest, and the part for projects with $r \geq 4$ is the lightest.

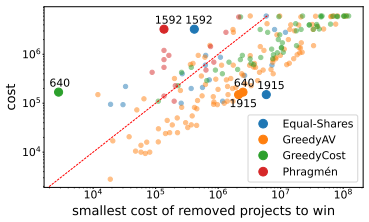


Figure 2: The ratio between the project's cost and the sum of costs of projects in the cheapest possible set of projects that makes a given project winning. The interesting projects are labeled with their name, and each project p is plotted for every rule f (unless $p \in f(E)$). Instance: Warsaw, Citywide, 2023.

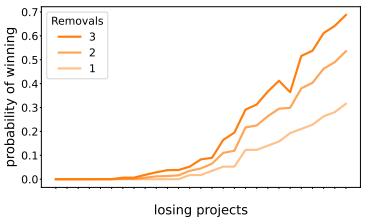


Figure 3: The probability of being funded for each initially losing project after removing $r \in \{1, 2, 3\}$ projects. The projects are ordered according to their winning probability for $r = 1$. The used rule is GreedyAV, and the depicted instance is Warsaw, Ursynow, 2019.

To finalize the complexity picture, we show that for the Equal-Shares rule, control by deleting projects is intractable under the same restriction as in the case of Phragmén. To prove this result, we exploit a reduction of [19], who showed that it is NP-hard to decide whether the Equal-Shares rule outputs the same outcome for every tie-breaking order.

Theorem 4. *Both Equal-Shares-CCDC and Equal-Shares-DCDC are NP-complete, even if the projects are of unit cost.*

For performance measures based on the probability that a project wins/loses if a randomly selected set of projects is deleted or added, it is essential to efficiently determine the number of solutions for an instance. However, it turns out that all our NP-hardness results, excluding Equal-Shares and the deletion operation, also imply #P-hardness for respective problems. Therefore, we do not expect the existence of a significantly faster algorithm for computing such measures than a simple enumeration of all possible solutions.

4 Experiments

We analyze the effect of project deletions on real-world data from Pabulib [14] and explore how different performance measures based on this operation can help with understanding and explanation of outcomes for proposers of losing projects and PB election organizers.

Data. In our experiments, we analyze 543 approval-based PB instances from Pabulib. We include every instance with approval ballots available as of October 2024 with at least one losing project. In total, our dataset contains 10531 losing projects for GreedyAV, 5771 for GreedyCost, 6004 for Phragmén, and 7412 for Equal-Shares. The largest instance consists of 160 projects and 90494 voters.

Experimental Setup. The experiments were run on computers with two AMD EPYC™ 7H12, 64-core, 2.6 GHz CPUs, and 256 GB DDR4 3200MT/s RAM. We use the same Python implementation for rules GreedyAV, GreedyCost, and Phragmén as in [6]. For Equal-Shares, we use our own implementation in C++ that significantly outperforms the available implementations (see **??** for details). For each rule, every instance, and every combination of $r \in \{1, 2, 3\}$ projects in this instance, we determined the winners after deleting these r projects. Evaluating the instances with our rules took us the following

number of core-hours: 38000 for Phragmén, 900 for both GreedyAV and GreedyCost, and 5000 for our C++ implementation of Equal-Shares. The overall running time is significantly skewed by instances with many projects, as we need to try all $\mathcal{O}(n^r)$ subsets (recall that significant speed-up for these rules is not possible due to Theorems 3 and 4). For computing the optimal control in the setting with either GreedyAV or GreedyCost, we use the dynamic programming approach from Theorem 2, which finishes for the whole dataset in less than 1 core-hour.

4.1 How Close is a Project to Being Funded?

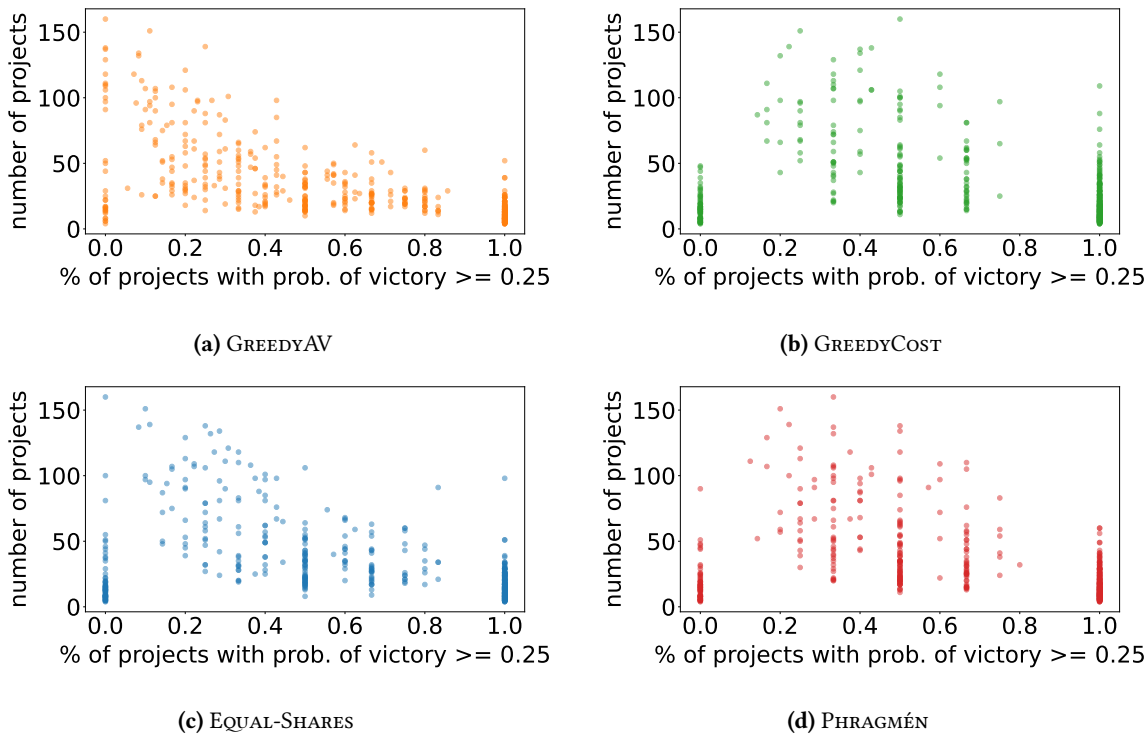
In general, PB rules do not provide any information about the performance of proposed projects—a project is either funded or not—and there are no direct ways of measuring how close a project was to being successful. Indeed, this is exactly what drove [6] to initiate the study of project performance measures. Here, we propose several further measures based on constructive control by deleting projects.

The first, very basic, approach we suggest is to count how many other projects need to be removed from the instance to make some initially losing project p funded. In Figure 1, we present an overview of the results for the whole dataset. For GreedyAV, more than 47% initially losing projects get funded after the removal of at most 3 projects; for the remaining rules, the value is smaller, but still significant – 37% for GreedyCost, 40% for Equal-Shares, and 43% for Phragmén, respectively. Another interesting piece of information we gain from Figure 1 is that for most of the projects for which it is enough to remove at most three projects, it is actually enough to remove only one other project. This is most evident in the case of GreedyAV.

Yet, saying that a project is close to winning simply because it can be funded after deleting some small number of carefully selected projects is overly simplistic. On the one hand, it is not too surprising that a project gets selected after deleting some other, expensive projects. On the other hand, we expect such expensive projects to be well-prepared and to not be removed for formal reasons, or due to missing some deadline. Consequently, instead of taking the number of projects that we need to delete to get some initially losing project p funded, we may rather seek the cheapest set of projects (of a given cardinality) whose deletion gets p funded. This measure indeed is much more fine-grained than our first one. To see this, we consider the results for Warsaw, Citywide, 2023 PB election, shown in Figure 2: There are certain projects where the removal of expensive projects is the only way to get them funded (e.g., project 1915), but there are also projects that get funded after removing rather cheap projects (see, e.g., project 1592). Note that this behavior is not very consistent among different PB rules, which is caused mostly by their underlying principles: whether the rule is more proportionality- or social-welfare-oriented (e.g., to get project 640 funded, we delete cheaper projects under GreedyCost, but more expensive ones under GreedyAV).

Both above measures have the downside that they focus on deleting exactly a particular subset of projects. However, even if we can get some initially losing project p funded after deleting projects whose cost is X, it is possible that p would be losing after deleting some other projects, whose cost is $2X$ (while this may seem unintuitive, it can happen due to various possible interactions among the projects and involved operation of our PB rules). Hence, in the remainder of this section we take a more stochastic approach and analyze the probability that a project gets funded if we remove a random subset of projects of some predefined size. We start with an overview of the whole dataset and later we analyze one specific instance, to show what information can be gained from measures based on the probability of winning.

In Figure 4, we have a data point for every instance and plot the percentage of initially losing projects that have at least 25% probability of being funded after the removal of three random projects. First, we see that for all the voting rules, there are instances where successful control is either unlikely for large fractions of projects or, on the contrary, where control is likely to be successful for nearly all projects. This affects more frequently instances of smaller size. Also, there is a visible difference



(a) GreedyAV

(b) GreedyCost

(c) Equal-Shares

(d) Phragmén

Figure 4: Each point of this plot represents a single instance. On the x axis, we have the percentage of losing projects with a probability of at least 0.25 for getting funded after the removal of 3 random projects. On the y axis, we have m, i.e., the number of projects.

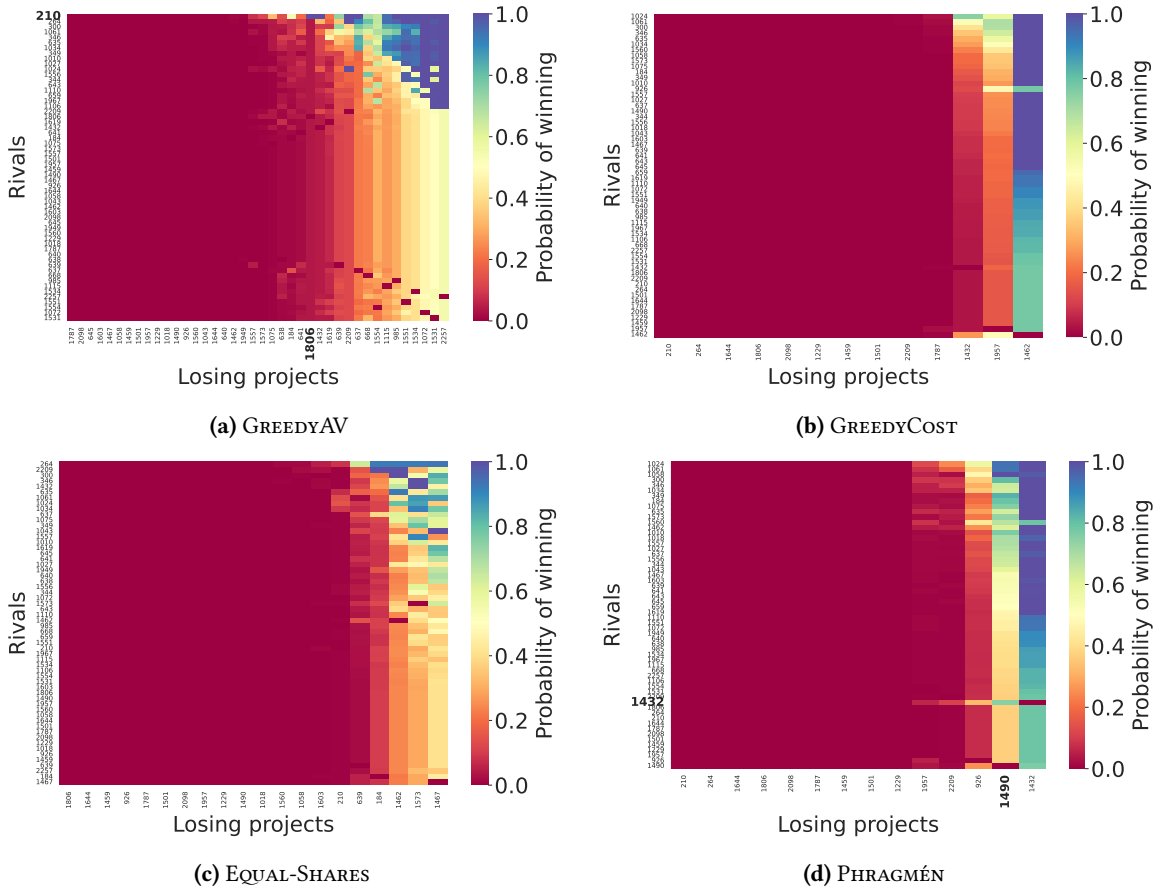
between GreedyAV and the remaining rules: unlike in GreedyAV, for these rules, the instances are more 'clustered' around certain percentage values, while for GreedyAV, the instances are more spread.

Now, we focus on a specific instance. In Figure 3, we plot for each losing project of the instance Warsaw, Ursynow, 2019 (under GreedyAV), the probability that this project becomes a winner if r projects are removed for different values of r. From this plot, we can distinguish between projects that are 'clear losers', meaning that their probability of winning is very close to zero regardless of the size of the removal set, and projects that are much closer to winning. This demonstrates how useful such a measure is of a project's strength is.

4.2 Who Are My Biggest Rivals?

The performance measures introduced so far allow us to compare projects from a 'global perspective', meaning that we can see how a losing project performed relative to other losing projects. However, for project proposers, it is very important to know why their project was not funded and what can be done to improve their project's performance in the future. One possibility is to identify a given project's rivals—projects whose removal significantly increases its chances of victory. Proposers of losing projects can then analyze such rivals, learn what they did differently, and improve their projects.

We propose the following measure of rivalry. We set the r-rivalry between a losing project p and some other project q equal to the probability that p is funded after we remove q and r other random projects. In Figure 5, we present the results for $r = 2$, instance Warsaw, Ursynow, 2019, and different PB rules. It is not surprising that the strongest rivals are usually the projects that were initially funded. However, this is not always the case. One such example is project 1490 under the Phragmén rule (Figure 5d), for which the (initially losing) project 1432 is a much stronger rival than most of the initially winning projects. It can also be the case that for some projects, their funding relies solely on the performance



(a) GreedyAV **(b)** GreedyCost

(c) Equal-Shares **(d)** Phragmén

Figure 5: Rivals for initially losing projects. For each losing project p (plotted on the x-axis) and every other (not necessarily losing) project q (plotted on the y-axis), we display the probability (red represents a value close to 0, blue represents a value close to 1) that p is funded when q and $r = 2$ other random projects are removed. Instance: Warsaw, Ursynow, 2019.

of a few other projects. A very good example of this behavior is project 1806 under the GreedyAV rule (Figure 5a): unless we remove project 210, there is almost no chance that project 1806 will ever be funded. This measure also nicely complements the measures from the previous subsection, as, based on plots similar to Figure 5, we can visually distinguish projects that are hopeless losers, which are somewhere in the middle, and which projects almost got in.

5 Discussion

In our experiments, we demonstrated the usefulness of project performance analysis. One can use our measures to compare different projects and provide election organizers with information on which projects were very close to being funded. In practice, cities often try to discuss popular losing projects and fund them from an increased or completely separate budget. Moreover, our rivalry measures help project proposers identify which other projects prevented their success. If such strong rivals share some similarities with the losing project, the proposer can learn from them and improve their project for the next round of participatory budgeting.

Ethical Statement

Even though our paper's goal is to improve the explainability and transparency of participatory budgeting outcomes, voting control is traditionally understood as malicious and highly undesired behavior. As such, one might object that our work could increase awareness of possible manipulation in PB elections. We want to stress that manipulations based on our performance measures are implausible: Our measures can be computed only after the elections have ended and we have complete (and anonymized) information about the whole instance. That is, the potential knowledge based on our measures can be used, if at all, to manipulate the next installation of PB elections. However, the new instance will most likely be different since some projects have already been funded, new projects will be proposed, and, in particular, voters' preferences may change over time.

Acknowledgments

We thank the anonymous reviewers for their helpful comments. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 101002854). ŁJ's research presented in this paper is supported in part from the funds assigned by Polish Ministry of Science and Technology to AGH University. This work was co-funded by the European Union under the project Robotics and advanced industrial production (reg. no. CZ.02.01.01/00/22_008/0004590). This work was supported by the MEYS of the Czech Republic through the e-INFRA CZ (ID: 90254), project OPEN-30-16. JP and ŠS acknowledge the additional support of the Grant Agency of the CTU in Prague, grant No. SGS23/205/OHK3/3T/18.

References

[1] John J. Bartholdi III, Craig A. Tovey, and Michael A. Trick. How hard is it to control an election? *Mathematical and Computer Modelling*, 16(8-9):27–40, 1992. doi: 10.1016/0895-7177(92)90085-Y.

[2] Dorothea Baumeister and Tobias Hogrebe. On the complexity of predicting election outcomes and estimating their robustness. In *Proceedings of the 18th European Conference on Multi-Agent Systems, EUMAS '21*, volume 12802 of *LNCS*, pages 228–244. Springer, 2021. doi: 10.1007/978-3-030-82254-5_14.

[3] Dorothea Baumeister, Linus Boes, and Johanna Hillebrand. Complexity of manipulative interference in participatory budgeting. In *Proceedings of the 7th International Conference on Algorithmic Decision Theory, ADT '21*, volume 13023 of *LNCS*, pages 424–439. Springer, 2021. doi: 10.1007/978-3-030-87756-9_27.

[4] Niclas Boehmer, Robert Bredereck, Piotr Faliszewski, and Rolf Niedermeier. Winner robustness via swap- and shift-bribery: Parameterized counting complexity and experiments. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence, IJCAI '21*, pages 52–58. ijcai.org, 2021. doi: 10.24963/ijcai.2021/8.

[5] Niclas Boehmer, Piotr Faliszewski, Łukasz Janeczko, and Andrzej Kaczmarczyk. Robustness of participatory budgeting outcomes: Complexity and experiments. In *Proceedings of the 16th International Symposium on Algorithmic Game Theory, SAGT '23*, volume 14238 of *LNCS*, pages 161–178. Springer, 2023. doi: 10.1007/978-3-031-43254-5_10.

[6] Niclas Boehmer, Piotr Faliszewski, Łukasz Janeczko, Dominik Peters, Grzegorz Pierczyński, Šimon Schierreich, Piotr Skowron, and Stanisław Szufa. Evaluation of project performance in participatory

budgeting. In *Proceedings of the 33rd International Joint Conference on Artificial Intelligence, IJCAI '24*, pages 2678–2686. ijcai.org, 2024. doi: 10.24963/ijcai.2024/296.

[7] Robert Bredereck, Piotr Faliszewski, Andrzej Kaczmarczyk, Rolf Niedermeier, Piotr Skowron, and Nimrod Talmon. Robustness among multiwinner voting rules. *Artificial Intelligence*, 290, 2021. doi: 10.1016/j.artint.2020.103403.

[8] Markus Brill, Rupert Freeman, Svante Janson, and Martin Lackner. Phragmén's voting methods and justified representation. *Mathematical Programming*, 203(1):47–76, 2024. doi: 10.1007/S10107-023-01926-8.

[9] Yves Cabannes. Participatory budgeting: A significant contribution to participatory democracy. *Environment and Urbanization*, 16(1):27–46, April 2004. ISSN 0956-2478. doi: 10.1177/095624780401600104.

[10] Gábor Erdélyi, Michael R. Fellows, Jörg Rothe, and Lena Schend. Control complexity in Bucklin and fallback voting: A theoretical analysis. *Journal of Computer and System Sciences*, 81(4):632–660, 2015. doi: 10.1016/j.jcss.2014.11.002.

[11] Gábor Erdélyi, Michael R. Fellows, Jörg Rothe, and Lena Schend. Control complexity in Bucklin and fallback voting: An experimental analysis. *Journal of Computer and System Sciences*, 81(4):661–670, 2015. doi: 10.1016/j.jcss.2014.11.003.

[12] Piotr Faliszewski and Jörg Rothe. Control and bribery in voting. In *Handbook of Computational Social Choice*, pages 146–168. Cambridge University Press, 2016. doi: 10.1017/CBO9781107446984.008.

[13] Piotr Faliszewski, Edith Hemaspaandra, and Lane A. Hemaspaandra. How hard is bribery in elections? *Journal of Artificial Intelligence Research*, 35:485–532, 2009. doi: 10.1613/jair.2676.

[14] Piotr Faliszewski, Jaroslaw Flis, Dominik Peters, Grzegorz Pierczyński, Piotr Skowron, Dariusz Stolicki, Stanisław Szufa, and Nimrod Talmon. Participatory budgeting: Data, tools and analysis. In *Proceedings of the 32nd International Joint Conference on Artificial Intelligence, IJCAI '23*, pages 2667–2674. ijcai.org, 2023. doi: 10.24963/ijcai.2023/297.

[15] Piotr Faliszewski, Łukasz Janeczko, Dušan Knop, Jan Pokorný, Šimon Schierreich, Mateusz Słuszniak, and Krzysztof Sornat. Participatory budgeting project strength via candidate control. In *Proceedings of the 34th International Joint Conference on Artificial Intelligence, IJCAI '25*. ijcai.org, 2025.

[16] Ashish Goel, Anilesh K. Krishnaswamy, Sukolsak Sakshuwong, and Tanja Aitamurto. Knapsack voting for participatory budgeting. *ACM Transactions on Economics and Computation*, 7(2):8:1–8:27, 2019. doi: 10.1145/3340230.

[17] Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985. doi: 10.1016/0304-3975(85)90224-5.

[18] Edith Hemaspaandra, Lane A. Hemaspaandra, and Jörg Rothe. Anyone but him: The complexity of precluding an alternative. *Artificial Intelligence*, 171(5-6):255–285, 2007. doi: 10.1016/j.artint.2007.01.005.

[19] Łukasz Janeczko and Piotr Faliszewski. Ties in multiwinner approval voting. In *Proceedings of the 32nd International Joint Conference on Artificial Intelligence, IJCAI '23*, pages 2765–2773. ijcai.org, 2023. doi: 10.24963/ijcai.2023/308.

[20] Hong Liu, Haodi Feng, Daming Zhu, and Junfeng Luan. Parameterized computational complexity of control problems in voting systems. *Theoretical Computer Science*, 410(27-29):2746–2753, 2009. doi: 10.1016/j.tcs.2009.04.004.

[21] Maaike Los, Zoé Christoff, and Davide Grossi. Proportional budget allocations: Towards a systematization. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence, IJCAI '22*, pages 398–404. ijcai.org, 2022. doi: 10.24963/ijcai.2022/57.

[22] Thomas R. Magrino, Ronald L. Rivest, Emily Shen, and David Wagner. Computing the margin of victory in IRV elections. Presented at 2011 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections, August 2011.

[23] Reshef Meir, Ariel D. Procaccia, Jeffrey S. Rosenschein, and Aviv Zohar. Complexity of strategic

behavior in multi-winner elections. *Journal of Artificial Intelligence Research*, 33:149–178, 2008. doi: 10.1613/jair.2566.

[24] Dominik Peters and Piotr Skowron. Proportionality and the limits of welfarism. In *Proceedings of the 21st ACM Conference on Economics and Computation, EC '20*, pages 793–794. ACM, 2020. doi: 10.1145/3391403.3399465.

[25] Dominik Peters, Grzegorz Pierczyński, and Piotr Skowron. Proportional participatory budgeting with additive utilities. In *Proceedings of the 35th Annual Conference on Neural Information Processing Systems, NeurIPS '21*, pages 12726–12737. Curran Associates, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/69f8ea31de0c00502b2ae571fbab1f95-Abstract.html.

[26] Simon Rey and Jan Maly. The (computational) social choice take on indivisible participatory budgeting. Technical Report arXiv.2303.00621, arXiv, 2023.

[27] Dmitry Shiryaev, Lan Yu, and Edith Elkind. On elections with robust winners. In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems, AAMAS '13*, pages 415–422. IFAAMAS, 2013. URL http://dl.acm.org/citation.cfm?id=2484987.

[28] Lirong Xia. Computing the margin of victory for various voting rules. In *Proceedings of the 13th ACM Conference on Electronic Commerce, EC '12*, pages 982–999. ACM, 2012. doi: 10.1145/2229012.2229086.

[29] Yongjie Yang. Complexity of manipulating and controlling approval-based multiwinner voting. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI '19*, pages 637–643. ijcai.org, 2019. doi: 10.24963/ijcai.2019/90.

Piotr Faliszewski
AGH University of Kraków
Kraków, Poland
Email: faliszew@agh.edu.pl

Łukasz Janeczko
AGH University of Kraków
Kraków, Poland
Email: ljaneczk@agh.edu.pl

Dušan Knop
Czech Technical University in Prague
Prague, Czechia
Email: dusan.knop@fit.cvut.cz

Jan Pokorný
Czech Technical University in Prague
Prague, Czechia
Email: jan.pokorny@fit.cvut.cz

Šimon Schierreich
Czech Technical University in Prague
Prague, Czechia
Email: schiesim@fit.cvut.cz

Mateusz Słuszniak
AGH University of Kraków
Kraków, Poland
Email: msluszniak1@gmail.com

Krzysztof Sornat
AGH University of Kraków
Kraków, Poland
Email: sornat@agh.edu.pl