Algorithms for the Coalitional Manipulation Problem

Michael Zuckerman* Ariel D. Procaccia† Jeffrey S. Rosenschein‡

Abstract

We investigate the problem of coalitional manipulation in elections, which is known to be hard in a variety of voting rules. We put forward efficient algorithms for the problem in Scoring rules, Maximin and Plurality with runoff, and analyze their windows of error. Specifically, given an instance on which an algorithm fails, we bound the additional power the manipulators need in order to succeed. We finally discuss the implications of our results with respect to the popular approach of employing computational hardness to preclude manipulation.

*School of Computer Science and Engineering, The Hebrew University of Jerusalem, Jerusalem 91904, Israel, email: michez@cs.huji.ac.il

†School of Computer Science and Engineering, The Hebrew University of Jerusalem, Jerusalem 91904, Israel, email: arielpro@cs.huji.ac.il

‡School of Computer Science and Engineering, The Hebrew University of Jerusalem, Jerusalem 91904, Israel, email: jeff@cs.huji.ac.il

1 Introduction

Social choice theory is an extremely well-studied subfield of economics. In recent years, interest in the computational aspects of social choice, and in particular in the computational aspects of voting, has sharply increased. Among other things, this trend is motivated by numerous applications of voting techniques and paradigms to problems in artificial intelligence [15, 12, 10].

In an election, a set of voters submit their (linear) preferences (i.e., rankings) over a set of candidates. The winner of the election is designated by a *voting rule*, which is basically a mapping from the space of possible *preference profiles* into candidates. A thorn in the side of social choice theory is formulated in the famous Gibbard-Satterthwaite Theorem [13, 22]. This theorem states that for any voting rule that is not a *dictatorship*, there are elections in which at least one of the voters would benefit by lying. A dictatorship is a voting rule where one of the voters—the dictator—single-handedly decides the outcome of the election.

Since the 1970s, when this impossibility result was established, an enormous amount of effort has been invested in discovering ways to circumvent it. Two prominent and well-established ways are allowing payments [23, 4, 14], or restricting the voters' preferences [18].

In this paper, we wish to discuss a third path—the "path less taken", if you will—which has been explored by computer scientists. The Gibbard-Satterthwaite Theorem implies that in theory, voters are able to *manipulate* elections, i.e., bend them to their advantage by lying. But in practice, deciding which lie to employ may prove to be a hard computational problem; after all, there are a superpolynomial number of possibilities of ranking the candidates.

Indeed, Bartholdi *et al.* [2] put forward a voting rule where manipulation is $\mathcal{NP}$-hard. In another important paper, Bartholdi and Orlin [1] greatly strengthened the approach by proving that the important Single Transferable Vote (STV) rule is hard to manipulate.

This line of research enjoys new life in recent years thanks to the influential work of Conitzer, Lang and Sandholm [7].[1] The foregoing paper studied the complexity of coalitional manipulation. In this setting, there is a coalition of potentially untruthful voters, attempting to coordinate their ballots so as to get their favorite candidate elected. The authors further assume that the votes are weighted: some voters have more power than others. Conitzer *et al.* show that in a variety of prominent voting rules, coalitional manipulation is $\mathcal{NP}$-hard, *even if there are only a constant number of candidates* (for more details, see Section 2). This work has been extended in numerous directions, by different authors [16, 21, 5, 8]; Elkind and Lipmaa [9], for example, strengthened the abovementioned results about coalitional manipulation by employing cryptographic techniques.

In short, computational complexity is by now a well-established method of circumventing the Gibbard-Satterthwaite Theorem. Unfortunately, a shortcoming of the results we mentioned above is that they are worst-case hardness results, and thus provide a poor obstacle against potential manipulators. Recent work regarding the average-case complexity of manipulation have argued that manipulation with respect to many worst-case-hard-to-manipulate voting rules is easy in practice [6]. In particular, Procaccia and Rosenschein [20, 19] have established some theoretical results regarding the tractability of the coalitional manipulation problem in practice. The matter was further discussed by Erdelyi *et al.* [11]. In spite of this, the question of the tractability of the manipulation problem, and in particular of the coalitional manipulation problem, in practical settings is still wide-open.

Our Approach and Results. We wish to convince that, indeed, the coalitional manipulation problem can be efficiently solved in practice, but our approach differs from all previous work. We

[1] Historical note: although we cite the JACM 2007 paper, this work originated in a AAAI 2002 paper.

present efficient heuristic algorithms for the problem which provide strong theoretical guarantees. Indeed, we characterize small windows of instances on which our algorithms may fail; the algorithms are proven to succeed on all other instances.

Specifically, we prove the following results regarding three of the most prominent voting rules (in which coalitional manipulation is known to be $\mathcal{NP}$-hard even for a constant number of candidates):

Theorem.

1. In the Borda rule, if it is possible to find a manipulation for an instance with certain weights, Algorithm 2 will succeed when given an extra manipulator with maximal weight.

2. In the Plurality with Runoff rule, if it is possible to find a manipulation for an instance with certain weights, Algorithm 3 will succeed when given an extra manipulator with maximal weight.

3. In the Maximin rule, if it is possible to find a manipulation for an instance with certain weights, Algorithm 1 will succeed when given two copies of the set of manipulators.

Structure of the Paper. In Section 2, we describe the major voting rules and formulate the coalitional manipulation problem. In Section 3, we present and analyze our algorithms in three subsections: Borda, Plurality with Runoff, and Maximin. Finally, we discuss our results in Section 4.

2 Voting Rules and Manipulation Problems

An election consists of a set $C = \{c_1, \ldots, c_m\}$ of candidates and a set $S = \{v_1, \ldots, v_{|S|}\}$ of voters. The voters provide a total order on the candidates. To put it differently, each voter submits a ranking of the candidates. The voting setting also includes a *voting rule*, which is a function from the set of all possible combinations of votes to C.

We shall discuss the following voting rules (whenever the voting rule is based on scores, the candidate with the highest score wins):

- *Scoring rules.* Let $\vec{\alpha} = \langle \alpha_1, \ldots, \alpha_m \rangle$ be a vector of non-negative integers such that $\alpha_1 \geq \alpha_2 \geq \ldots \geq \alpha_m$. For each voter, a candidate receives α_1 points if it is ranked first by the voter, α_2 if it is ranked second, etc. The *score* $s_{\vec{\alpha}}$ of a candidate is the total number of points the candidate receives. The scoring rules which we will consider are: *Borda*, where $\vec{\alpha} = \langle m-1, m-2, \ldots, 0 \rangle$, and *Veto*, where $\vec{\alpha} = \langle 1, 1, \ldots, 1, 0 \rangle$.

- *Maximin.* For any two distinct candidates x and y, let $N(x, y)$ be the number of voters who prefer x to y. The *maximin score* of x is $S(x) = \min_{y \neq x} N(x, y)$.

- *Copeland.* For any two distinct candidates x and y, let $C(x, y) = +1$ if $N(x, y) > N(y, x)$ (in this case we say that x beats y in their pairwise election), $C(x, y) = 0$ if $N(x, y) = N(y, x)$, and $C(x, y) = -1$ if $N(x, y) < N(y, x)$. The *Copeland score* of candidate x is $S(x) = \sum_{y \neq x} C(x, y)$.

- *Plurality with runoff.* In this rule, a first round eliminates all candidates except the two with the highest plurality scores. The second round determines the winner between these two by their pairwise election.

In some settings the voters are *weighted*. A *weight function* is a mapping $w : S \to \mathbb{N}$. When voters are weighted, the above rules are applied by considering a voter of weight l to be l different voters.

Definition 2.1.

1. In the Constructive Coalitional Weighted Manipulation (CCWM) problem, we are given a set C of candidates, with a distinguished candidate $p \in C$, a set of weighted voters S that already cast their votes (these are the truthful voters), and weights for a set of voters T that still have not cast their votes (the manipulators). We are asked whether there is a way to cast the votes in T such that p wins the election.

2. Constructive Coalitional Unweighted Manipulation (CCUM) problem is a special case of CCWM problem where all the weights equal 1.

Remark 2.2. We implicitly assume in both questions that the manipulators have full knowledge about the other votes. Unless explicitly stated otherwise, we also assume that ties are broken adversarially to the manipulators, so if p ties with another candidate, p loses.

Theorem 2.3 ([7])**.** *The CCWM problem in all the abovementioned voting rules is $\mathcal{NP}$-complete, even when the number of candidates is constant.*

Throughout this paper we will use the convention that $|C| = m$, $|S| = N$ and $|T| = n$. Whenever the voting rule is based on scores, we will denote by $S_{S,j}(c)$ the score of candidate c from the voters in S and the first j voters of T (fixing some order on the voters of T). Whenever it is clear from the context that S is fixed, we will use simply $S_j(c)$ for the same. Also, for $G \subseteq C$, $0 \le j \le n$ we will write $S_j(G) = \{S_j(g) \mid g \in G\}$. For two lists A, B (ordered multisets, possibly with multiplicities), we denote by $A + B$ the list which is obtained after B is appended to A.

3 Results

We begin our contribution by presenting a general greedy algorithm for the coalitional manipulation problem. Some of our main results concern this algorithm or its restriction to scoring rules.

The greedy algorithm is given as Algorithm 1. It works as follows: the manipulators, according to descending weights, each rank p first and rank the other candidates in a way that minimizes their maximum score. This algorithm is a generalization of the one that appeared in [3].

Remark 3.1. We refer to an iteration of the main for loop in line 4 of the algorithm as a *stage* of the algorithm.

We will use the fact that for many voting rules, if there exists a manipulation for a coalition of manipulators with weights list W, then there exists a manipulation for a coalition of manipulators with weights list W' where $W' \supseteq W$. Normally, if the coalition is too small then there is no manipulation, and this is indeed what the algorithm will report. On the other hand, if the coalition is large enough, then the greedy algorithm will find the manipulation. So there remains a window of error, where for some coalitions there could exist a manipulation, but the algorithm may not find it. We are interested in bounding the size of this window. We first formulate the monotonicity property described above.

Definition 3.2. In the context of the CCWM problem, a voting rule is said to be *monotone in weights* if it satisfies the following property: whenever there is a manipulation making p win for manipulator set T with weights list W, there is also a manipulation making p win for manipulator set T' with weights list W', where $T' \supseteq T$, $W' \supseteq W$.

Algorithm 1 Decides CCWM

procedure GREEDY(C, p, X_S, W) ▷ X_S is the set of preferences of voters in S, W is the list of weights for voters in T, $|W| = |T| = n$
  sort(W) ▷ Sort the weights in descending order
  $X \leftarrow \phi$ ▷ Will contain the preferences of T
  for $j = 1, \ldots, n$ **do** ▷ Iterate over voters by descending weights
    $P_j \leftarrow (p)$ ▷ Put p at the first place of the j-th preference list
    for $t = 2, \ldots, m$ **do** ▷ Iterate over places of j-th preference list
      ▷ Evaluate the score of each candidate if j would put it at the next available place
      Pick $c \in \text{argmin}_{c \in C \setminus P_j}\{\text{Score of } c \text{ from } X_S \cup X \cup \{P_j + \{c\}\}\}$
      $P_j = P_j + \{c\}$ ▷ Add c to j's preference list
    end for
    $X \leftarrow X \cup \{P_j\}$
  end for
  $X_T \leftarrow X$
  if $\text{argmax}_{c \in C}\{\text{Score of } c \text{ based on } X_S \cup X_T\} = \{p\}$ **then**
    return true ▷ p wins
  else
    return false
  end if
end procedure

Monotonicity in weights is a prerequisite for the type of analysis we wish to present. However, not all the basic voting rules have this property. In particular, the prominent Copeland rule does not possess this property. See Appendix B for an example.

3.1 Borda

In this subsection, we analyze the performance of Algorithm 1 with respect to the important Borda voting rule. Note that, in the context of scoring rules, Algorithm 1 reduces to Algorithm 2, which appears in Appendix G. This algorithm first appeared in Procaccia and Rosenschein [20]. In this specific instantiation of Algorithm 1, we do not require sorting the manipulators' weights, as this does not play a part in our analysis.

Lemma 3.3. *Scoring rules are monotone in weights.*

The proofs of all monotonicity Lemmas are deferred to Appendix A. We are now ready to present our theorem regarding the Borda rule.

Theorem 3.4. *In the Borda voting rule, let C be a set of candidates with $p \in C$ a preferred candidate, S a set of voters who already cast their votes. Let W be the weights list for the set T. Then:*

1. *If there is no ballot making p win the election, then Algorithm 2 will return* false.
2. *If there exists a ballot making p win the election, then for the same instance with weights list $W + \{w'_1, \ldots, w'_k\}$, where $k \geq 1, \sum_{i=1}^{k} w'_i \geq \max(W)$, Algorithm 2 will return* true.

A key notion for the proof of the theorem is the definition of the set G_W. Let W be list of weights; we define G_W as follows. Run the algorithm $n+1$ stages with the weights $W + \{w\}$, where

w is an arbitrary weight. Let $G_W^0 = \text{argmax}_{g\in C\setminus\{p\}}\{S_0(g)\}$, and, by induction, for $s = 1, 2, \dots$: $G_W^s = G_W^{s-1} \cup \{g \mid g \text{ was ranked after some } g' \in G_W^{s-1} \text{ at some stage } l,\ 1 \le l \le n+1\}$. Finally, let $G_W = \cup_{0\le s} G_W^s$. The definition is independent of the weight w, as this weight is used only at stage $n+1$, so it does not impact the preferences of the voters, and thus it does not impact G_W. From the definition, $G_W^0 \subseteq G_W^1 \subseteq \dots \subseteq C \setminus \{p\}$. Furthermore, as $|C \setminus \{p\}| = m-1$, it follows that there exists $0 \le s' \le m-2$ s.t. $G_W^{s'} = G_W^{s'+1}$, and thus $G_W = G_W^{s'} = G_W^{m-2}$.

Lemma 3.5. *Given W, the candidates in G_W were ranked at each stage l, $1 \le l \le n+1$ at $|G_W|$ last places, i.e., they were always granted the points $|G_W| - 1, \dots, 0$.*

Proof. If, by way of contradiction, there exists $c \in C \setminus G_W$ that was ranked at some stage at one of the last $|G_W|$ places, then there is $g \in G_W$ that was ranked before c at that stage. Let $s \ge 0$ such that $g \in G_W^s$. By definition, $c \in G_W^{s+1} \Rightarrow c \in G_W$, a contradiction. □

Lemma 3.6. *For all $c \in C \setminus G_W$, it holds that $S_n(c) \le \min_{g\in G_W}\{S_n(g)\}$.*

Proof. Suppose for contradiction that there are $c \in C \setminus G_W$ and $g \in G_W$, s.t. $S_n(c) > S_n(g)$. Then at stage $n+1$, c would have been ranked after g. Let $s \ge 0$ s.t. $g \in G_W^s$. Then $c \in G_W^{s+1} \Rightarrow c \in G_W$, a contradiction. □

Lemma 3.7. *Given W, $|W| = n$, let G_W be as before. Denote by $q(W)$ the average score of candidates in G_W after n stages: $q(W) = \frac{1}{|G_W|}\sum_{g\in G_W} S_n(g)$. Then:*

1. *If $S_n(p) \le q(W)$ then there is no manipulation that makes p win the election, and the algorithm will return* false.
2. *If $S_n(p) > \max_{g\in G_W}\{S_n(g)\}$, then there is a manipulation that makes p win, and the algorithm will find it.*

Proof. We first prove part 1. Denote $W = \{w_1, \dots, w_n\}$. We have the set G_W, and we suppose that $S_n(p) \le q(W)$. Let us consider any ballot X_T of votes in T, and let $S'_n(c)$ be the scores of the candidates $c \in C$ implied by this ballot (including also all the votes in S). As in Algorithm 2, p was placed at the top of the preference of each voter in T; it follows that

$$S_n(p) = S_0(p) + \sum_{j=1}^{n} w_j(m-1) \ge S'_n(p) \tag{1}$$

On the other hand, since by Lemma 3.5, in Algorithm 2 the candidates of G_W were ranked by all the voters in T at the last $|G_W|$ places, it follows that

$$q(W) = \frac{1}{|G_W|}\Big(\sum_{g\in G_W} S_0(g) + \sum_{j=1}^{n} w_j \sum_{i=0}^{|G_W|-1} i\Big) \le \frac{1}{|G_W|}\sum_{g\in G_W} S'_n(g) =: q'(X_T) \tag{2}$$

Combining together (1) and (2) we get that $S'_n(p) \le q'(X_T)$. There is at least one $g \in G_W$ such that $S'_n(g) \ge q'(X_T)$ (since $q'(X_T)$ is the average of the scores), hence $S'_n(p) \le S'_n(g)$, and so p will not win when X_T is applied.

Also note that Algorithm 2 returns *true* only if it constructs a (valid) ballot that makes p win, and so for the case $S_n(p) \le q(W)$ the algorithm will return *false*.

We now prove part 2 of the lemma. If $S_n(p) > \max_{g\in G_W}\{S_n(g)\}$, then by Lemma 3.6 for all $c \in C \setminus \{p\}$, $S_n(p) > S_n(c)$, and so the algorithm will find the manipulation. □

Lemma 3.8. *Let $G_W, q(W)$ be as before. Then for $w \geq 1$, $q(W+\{w\}) - q(W) \leq w\frac{m-2}{2}$.*

Proof. First, $G_W \subseteq G_{W+\{w\}}$, because for all $s \geq 0, G_W^s \subseteq G_{W+\{w\}}^s$. Now, for all $g \in G_{W+\{w\}} \setminus G_W$, g was not ranked in the first $n+1$ stages after any candidate in G_W, and so for all $g' \in G_W$, $S_n(g) \leq S_n(g')$, and hence

$$\frac{1}{|G_{W+\{w\}}|} \sum_{g \in G_{W+\{w\}}} S_n(g) \leq \frac{1}{|G_W|} \sum_{g' \in G_W} S_n(g') = q(W)$$

Now we can proceed:

$$\begin{aligned} q(W+\{w\}) &= \frac{1}{|G_{W+\{w\}}|} \sum_{g \in G_{W+\{w\}}} S_{n+1}(g) \\ &= \frac{1}{|G_{W+\{w\}}|} \sum_{g \in G_{W+\{w\}}} S_n(g) + \frac{w}{|G_{W+\{w\}}|} \sum_{i=0}^{|G_{W+\{w\}}|-1} i \\ &\leq q(W) + \frac{w}{m-1} \sum_{i=0}^{m-2} i \\ &= q(W) + w\frac{m-2}{2} \end{aligned}$$

And so, $q(W+\{w\}) - q(W) \leq w\frac{m-2}{2}$. ∎

We will prove in Lemmas 3.10 – 3.13 that for any weights list W, $|W| = n$: $\max_{g \in G_W}\{S_n(g)\} - q(W) \leq \max(W)\frac{m-2}{2}$. First we need to show that the scores of candidates in G_W are not scattered too much. We will need the following definition:

Definition 3.9. For an integer $w \geq 0$, a finite non-empty set of integers A is called *w-dense* if when we sort the set in nonincreasing order $b_1 \geq b_2 \geq \ldots \geq b_k$ (such that $\{b_1, \ldots, b_k\} = A$), it holds that for all $1 \leq j \leq k-1$, $b_{j+1} \geq b_j - w$.

Lemma 3.10. *Let W be a list of weights, $|W| = n$. Let $G_W = \cup_{0 \leq s} G_W^s$, as before. Then for all $s \geq 1$ and $g \in G_W^s \setminus G_W^{s-1}$ there exist $g' \in G_W^{s-1}$ and $X \subseteq C \setminus \{p\}$ (maybe $X = \phi$), s.t. $\{S_j(g), S_j(g')\} \cup S_j(X)$ is a $w_{\max}$-dense where $w_{\max} = \max(W)$ and $0 \leq j \leq n$.*

Proof. Let $s \geq 1$ and $g \in G_W^s \setminus G_W^{s-1}$. By definition there exist $g' \in G_W^{s-1}$ and $1 \leq j \leq n+1$ minimal s.t. g was ranked after g' at stage j. We distinguish between two cases:

Case 1: $j > 1$. In this case g was ranked before g' at stage $j-1$. So we have:

$$S_{j-1}(g) \geq S_{j-1}(g') \tag{3}$$

$$S_{j-2}(g) \leq S_{j-2}(g') \tag{4}$$

Denote $\alpha_d(h) := m -$ (place of $h \in C$ at the preference list of voter d). Further, denote by w_d the weight of voter d (so at stage d, h gets $w_d\alpha_d(h)$ points). g was ranked before g' at stage $j-1$, and hence $\alpha_{j-1}(g) > \alpha_{j-1}(g')$. Denote $l = \alpha_{j-1}(g) - \alpha_{j-1}(g')$. Let $g' = g_0, g_1, \ldots, g_l = g$ be the candidates that got at stage $j-1$ the points $w_{j-1}\alpha_{j-1}(g'), w_{j-1}(\alpha_{j-1}(g')+1), \ldots, w_{j-1}(\alpha_{j-1}(g')+l)$, respectively. Our purpose is to show that $\{S_{j-1}(g_0), \ldots, S_{j-1}(g_l)\}$ is w_{j-1}-dense, and therefore $w_{\max}$-dense. By definition of the algorithm,

$$S_{j-2}(g_0) \geq S_{j-2}(g_1) \geq \ldots \geq S_{j-2}(g_l) \tag{5}$$

Denote $u_t = S_{j-2}(g_t) + w_{j-1}\alpha_{j-1}(g')$ for $0 \leq t \leq l$. Then

$$\text{For all } 0 \leq t \leq l,\ S_{j-1}(g_t) = u_t + w_{j-1}t \tag{6}$$

So we need to show that $\{u_t + w_{j-1}t \mid 0 \leq t \leq l\}$ is w_{j-1}-dense. It is enough to show that:

(a) For all $0 \leq t \leq l$, if $u_t + w_{j-1}t < u_0$, then there exists $t < t' \leq l$ s.t. $u_t + w_{j-1}t < u_{t'} + w_{j-1}t' \leq u_t + w_{j-1}(t+1)$, and

(b) For all $0 \leq t \leq l$, if $u_t + w_{j-1}t > u_0$, then there exists $0 \leq t' < t$ s.t. $u_t + w_{j-1}(t-1) \leq u_{t'} + w_{j-1}t' < u_t + w_{j-1}t$.

Proof of (a): From (5) we get

$$u_0 \geq \ldots \geq u_l \tag{7}$$

Also from (3) and (6) we have $u_0 \leq u_l + w_{j-1}l$. Let $0 \leq t \leq l-1$ s.t. $u_t + w_{j-1}t < u_0$. Let us consider the sequence $u_t + w_{j-1}t, u_{t+1} + w_{j-1}(t+1), \ldots, u_l + w_{j-1}l$. Since $u_t + w_{j-1}t < u_0 \leq u_l + w_{j-1}l$, it follows that there is a minimal index t', $t < t' \leq l$ s.t. $u_t + w_{j-1}t < u_{t'} + w_{j-1}t'$. Then $u_{t'-1} + w_{j-1}(t'-1) \leq u_t + w_{j-1}t$, and thus

$$u_{t'-1} + w_{j-1}t' \leq u_t + w_{j-1}(t+1) \tag{8}$$

From (7) $u_{t'} \leq u_{t'-1}$, and then

$$u_{t'} + w_{j-1}t' \leq u_{t'-1} + w_{j-1}t' \tag{9}$$

Combining (8) and (9) together, we get $u_{t'} + w_{j-1}t' \leq u_t + w_{j-1}(t+1)$. This concludes the proof of (a). The proof of (b) is similar.

Case 2: $j = 1$. In this case $s \geq 2$, because otherwise, if $s = 1$, then $g' \in G_W^0$; therefore $S_0(g) \geq S_0(g') = \max_{h \in C\setminus\{p\}}\{S_0(h)\} \Rightarrow g \in G_W^0$, a contradiction. $g' \notin G_W^{s-2}$, because otherwise, by definition $g \in G_W^{s-1}$. Therefore there exists $g'' \in G_W^{s-2}$ s.t. g' was ranked after g'' at some stage j', i.e., $S_{j'-1}(g') \geq S_{j'-1}(g'')$. g has never been ranked after g'' (because otherwise $g \in G_W^{s-1}$), and it follows that $S_{j'-1}(g) \leq S_{j'-1}(g'')$, and we have $S_{j'-1}(g) \leq S_{j'-1}(g')$. Let j_0 be minimal s.t. $S_{j_0}(g) \leq S_{j_0}(g')$. As at stage 1 g was ranked after g', it holds that $S_0(g) \geq S_0(g')$. If $j_0 = 0$ then $S_0(g) = S_0(g')$, and hence $\{S_0(g), S_0(g')\}$ is 0-dense, and therefore $w_{\max}$-dense. Otherwise it holds that $S_{j_0-1}(g) > S_{j_0-1}(g')$, and as in Case 1, we can prove that $\{S_{j_0}(g), S_{j_0}(g')\} \cup S_{j_0}(X)$ is $w_{\max}$-dense for some $X \subseteq C \setminus \{p\}$. □

Lemma 3.11. *Let W be a list of weights, $|W| = n$, $w_{\max} = \max(W)$. Let $H \subseteq C\setminus\{p\}$ s.t. $S_j(H)$ is $w_{\max}$-dense for some $0 \leq j \leq n$. Then there exists $H', H \subseteq H' \subseteq C\setminus\{p\}$ s.t. $S_n(H')$ is $w_{\max}$-dense.*

Proof. We have $H \subseteq C \setminus \{p\}$ and $0 \leq j \leq n$, s.t. $S_j(H)$ is $w_{\max}$-dense. Denote $H_j := H$. Define inductively for $t = j, j+1, \ldots, n-1$: $H_{t+1} = \{g \in C \setminus \{p\} \mid \min_{h \in H_t}\{S_t(h)\} \leq S_t(g) \leq \max_{h \in H_t}\{S_t(h)\}\}$. Of course, for all t, $H_t \subseteq H_{t+1}$. It is easy to see that if for some $j \leq t \leq n-1$, $S_t(H_t)$ is $w_{\max}$-dense, then $S_{t+1}(H_{t+1})$ is also $w_{\max}$-dense. So we get by induction that $S_n(H_n)$ is $w_{\max}$-dense, and $H \subseteq H_n \subseteq C \setminus \{p\}$. □

Lemma 3.12. *Let W be a list of weights, $|W| = n$, $w_{\max} = \max(W)$. Let G_W be as before. Then the set $S_n(G_W)$ is $w_{\max}$-dense.*

Proof. Let $g = g_0 \in G_W$. If $g \notin G_W^0$, then $g \in G_W^s \setminus G_W^{s-1}$ for some $s \geq 1$. By Lemma 3.10 there exist $g_1 \in G_W^{s-1}$ and $X_1 \subseteq C \setminus \{p\}$ s.t. $\{S_j(g_0), S_j(g_1)\} \cup S_j(X_1)$ is $w_{\max}$-dense for some $0 \leq j \leq n$. By Lemma 3.11 there exists X_1', $X_1 \subseteq X_1' \subseteq C \setminus \{p\}$, s.t. $\{S_n(g_0), S_n(g_1)\} \cup S_n(X_1')$ is $w_{\max}$-dense. Denote $Z_1 := \{g_0, g_1\} \cup X_1'$. Similarly, if $g_1 \notin G_W^0$, then there exist $g_2 \in G_W^{s-2}$ and X_2' s.t. $\{S_n(g_1), S_n(g_2)\} \cup S_n(X_2')$ is $w_{\max}$-dense. Denote $Z_2 := \{g_1, g_2\} \cup X_2'$, etc. Thus, we can build a sequence of sets $Z_1, \ldots, Z_{s+1}$, s.t. for all $1 \leq t \leq s+1$, $S_n(Z_t)$ is $w_{\max}$-dense, $g = g_0 \in Z_1$ and for each $1 \leq t \leq s$ there exists $g_t \in G_W^{s-t}$ s.t. $g_t \in Z_t \cap Z_{t+1}$, and in particular, $g_s \in G_W^0$.

It is easy to see that for two w-dense sets A, A', if $A \cap A' \neq \phi$ then $A \cup A'$ is also w-dense, and hence we get $Z_g := \bigcup_{t=1}^{s+1} Z_t$ is $w_{\max}$-dense. Note that $S_0(G_W^0)$ is $w_{\max}$-dense, and hence there exists $\hat{Z}, G_W^0 \subseteq \hat{Z} \subseteq C \setminus \{p\}$ s.t. $S_n(\hat{Z})$ is $w_{\max}$-dense. From this we conclude that $\{S_n(g) \mid g \in \hat{Z} \cup \bigcup_{g \in G_W} Z_g\}$ is $w_{\max}$-dense. By Lemma 3.6, for all $h \in \hat{Z} \cup \bigcup_{g \in G_W} Z_g$, if $h \notin G_W$, then $S_n(h) \leq \min_{g \in G_W}\{S_n(g)\}$, and hence $S_n(G_W)$ is a also $w_{\max}$-dense. ◻

Lemma 3.13. *Let W be a list of weights, $|W| = n$, $w_{\max} = \max(W)$. Let G_W be as before, and denote $q(W) = \frac{1}{|G_W|}\sum_{g \in G_W} S_n(g)$, as before. Then $\max_{g \in G_W}\{S_n(g)\} - q(W) \leq w_{\max}\frac{m-2}{2}$.*

Proof. Sort the members of G_W by their scores after the n-th stage, i.e., $G_W = \{g_1, \ldots, g_{|G_W|}\}$ s.t. for all $1 \leq t \leq |G_W|-1$, $S_n(g_t) \geq S_n(g_{t+1})$. Denote for $1 \leq t \leq |G_W|$, $u_t = S_n(g_1) - w_{\max}(t-1)$, and let $U = \{u_1, \ldots, u_{|G_W|}\}$. $|U| = |G_W|$, $\max U = S_n(g_1) = \max_{g \in G_W}\{S_n(g)\}$. By Lemma 3.12, it is easy to see that for all $1 \leq t \leq |G_W|$, $S_n(g_t) \geq u_t$. Consequently, $q(W) \geq \frac{1}{|G_W|}\sum_{t=1}^{|G_W|} u_t$, hence $\max_{g \in G_W}\{S_n(g)\} - q(W) = u_1 - q(W) \leq u_1 - \frac{1}{|G_W|}\sum_{t=1}^{|G_W|} u_t = w_{\max}\frac{|G_W|-1}{2} \leq w_{\max}\frac{m-2}{2}$. ◻

Now we are ready to prove Theorem 3.4.

Proof of Theorem 3.4. Regarding part 1, Algorithm 2 returns *true* only if it constructs a (valid) ballot that makes p win, and thus if there is no ballot making p win, Algorithm 2 will return *false*.

We now prove part 2 of the theorem. Suppose that there exists a ballot making p win for weights list W, $|W| = n$. Let $W' := W + \{w_1', \ldots, w_k'\}$ for $k \geq 1, \sum_{i=1}^{k} w_i' \geq \max(W)$. By Lemma 3.7, $S_n(p) > q(W)$. From Lemma 3.8 we get by induction that

$$q(W') \leq q(W) + \sum_{i=1}^{k} w_i' \cdot \frac{m-2}{2} \tag{10}$$

By Lemma 3.13 and (10) we get:

$$\begin{aligned}
\max_{g \in G_{W'}}\{S_{n+k}(g)\} &\leq q(W') + \max(W') \cdot \frac{m-2}{2} \\
&\leq q(W') + \sum_{i=1}^{k} w_i' \cdot \frac{m-2}{2} \\
&\leq q(W) + \sum_{i=1}^{k} w_i' \cdot \frac{m-2}{2} + \sum_{i=1}^{k} w_i' \cdot \frac{m-2}{2} \\
&= q(W) + \sum_{i=1}^{k} w_i' \cdot (m-2) \\
&< S_n(p) + \sum_{i=1}^{k} w_i' \cdot (m-1) \\
&= S_{n+k}(p)
\end{aligned}$$

and hence, by Lemma 3.7 the algorithm will find a ballot making p win for set T' with weights W', and will return *true*. This completes the proof of Theorem 3.4. ☐

For an example where there is a manipulation for weights list W, but the algorithm will find a manipulation only for weights list $W + \{w'\}$, see Appendix C.

3.2 Maximin

In this subsection, we show that Algorithm 1 also does well with respect to the Maximin rule.

Lemma 3.14. *Maximin is monotone in weights.*

Theorem 3.15. *In the Maximin rule, let C be the set of candidates with $p \in C$ the preferred candidate, and S the set of voters who already cast their votes. Let W be the weights list for the set T. Then:*

1. *If there is no ballot making p win the election, then Algorithm 1 will return* false*.*
2. *If there is a ballot making p win the election, then for the same instance with weights list W' s.t. $W' \supseteq W + W$ (i.e., W' contains two copies of W), Algorithm 1 will return* true*.*

The proof of Theorem 3.15 is deferred to Appendix D.

3.3 Plurality with Runoff

In this subsection we present a heuristic algorithm for the CCWM problem in Plurality with runoff. The algorithm receives as a parameter a size of window $0 \leq u \leq \max(W)$ where it can give a wrong answer. Its running time depends on the size of its input and on u (see below). We begin by noting:

Lemma 3.16. *Plurality with runoff is monotone in weights.*

We will now give an informal description of the algorithm. We go over all the candidates other than p. To each candidate g we try to assign the voters with minimal total weight, such that if these voters place g first, g continues to the second round; the rest of the voters rank p first. If we succeeded in this way to make g and p survive the first round, and in the second round p beats g, then we found a valid ballot for making p win the election. If no candidate g was found in this way, then we report that there is no ballot.

A formal description of this algorithm, Algorithm 3, is given in Appendix G. The following additional notations are required. Denote by $\beta_X(g)$ the plurality score of g from voter set X (i.e., the sum of weights of the voters in X that put g at the top of their preferences). We also use $N_X(g, g') = \sum_{v \in U} w_v$, where U is the set of all the voters in X that prefer g to g', and w_v is the weight of voter v. Finally, for $g, g' \in C$ we denote $g \gg g'$ if a tie between g and g' is broken in favor of g.

Remark 3.17. In Algorithm 3 we do not rely on the assumption that for all $g \neq p$, $g \gg p$.

In the next theorem we prove the correctness of Algorithm 3, and analyze its time complexity. We will see that for getting an exact answer ($u = 0$), we will need running time which is polynomial in $\max(W)$ and the rest of the input. As the weights in W are specified in binary representation, this requires exponential time. However when the size of the error window increases, the complexity decreases, so for $u = \Omega(\frac{\max(W)}{\log(\max(W))})$ the complexity of the algorithm is polynomial in its input.

Theorem 3.18. *In the Plurality with runoff rule, let C be the set of candidates with $p \in C$ the preferred candidate, S the set of voters who already cast their votes. Let W be the weights list for the set T, and let $u \geq 0$ be the error window. Then:*

1. *If there is no ballot making p win the election, then Algorithm 3 will return* false*.*

2. *If there is a ballot making p win the election, then for the same problem with voter set $T' = T + \{v_{n+1}, \ldots, v_{n+l}\}$ with weights list $W' = W + \{w_{n+1}, \ldots, w_{n+l}\}$, where $l \geq 1, \sum_{j=1}^{l} w_{n+j} \geq u$, Algorithm 3 will return* true*.*

3. *On input C, p, X_S, W_S, W, u, where $|C| = m, |S| = N, |W| = n$, u is an integer, s.t. $0 \leq u \leq \max(W)$, the running time of Algorithm 3 is polynomial in $m, N, \log(\max(W_S)), n$ and $\frac{\max(W)}{u+1}$.*

The proof of the theorem is given in Appendix E.

4 Discussion

We would first like to discuss the application of our results to unweighted coalitional manipulation (the CCUM problem). It is known that this problem is tractable when the number of candidates is constant [7]. However, to the best of our knowledge there are no results regarding the complexity of the problem when the the number of candidates is not constant. We conjecture that CCUM in Borda and Maximin is $\mathcal{NP}$-complete.

In the context of unweighted manipulation, one can consider the following optimization problem (call it CCUO, where "O" stands for Optimization): given the votes of the truthful voters, find the minimum number of manipulators needed in order to make p win. Then, our theorems almost directly imply the following corollary:

Corollary 4.1.

1. *Algorithm 2 approximates CCUO in Borda up to an additive factor of 1.*

2. *Algorithm 1 is a 2-approximation algorithm for CCUO in Maximin.*

On the other hand, we have the following results:

Corollary 4.2. *Algorithm 3 efficiently solves the CCUM problem in Plurality with runoff.*

Theorem 4.3. *Algorithm 2 efficiently solves the CCUM problem in Veto.*

Corollary 4.2 is a special case of Theorem 3.18. The proof of Theorem 4.3 is given in Appendix F. We deduce that CCUM, and thus CCUO, in the Plurality with runoff and Veto rules are in $\mathcal{P}$.

We now turn to a brief and informal discussion regarding the implications of our results with respect to solving the weighted coalitional manipulation problem, CCWM, in practice. Our theorems imply that our algorithms err on only very specific configurations of the voters' weights. Intuitively, this means that if one looks at some (reasonable) distribution over the instances of the CCUM problem (such that weights are randomly selected), the probability of hitting the window of error is extremely small.

Procaccia and Rosenschein [20] show that Algorithm 2 has a small chance of mistake with respect to a *junta distribution*—which is arguably especially hard—over the instances of CCWM in scoring rules. This result uses a very loose analysis regarding the algorithm's window of error. Our result regarding Borda is far stronger, since the window of error is much more accurately characterized. However, since the result in Procaccia and Rosenschein deals with scoring rules in general, neither result subsumes the other.

5 Acknowledgments

The first author would like to thank Noam Nisan for a generous grant that supported this research.

References

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

[2] J. Bartholdi, C. A. Tovey, and M. A. Trick. The computational difficulty of manipulating an election. *Social Choice and Welfare*, 6:227–241, 1989.

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

[4] E. H. Clarke. Multipart pricing of public goods. *Public Choice*, 11:17–33, 1971.

[5] V. Conitzer and T. Sandholm. Universal voting protocol tweaks to make manipulation hard. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 781–788, 2003.

[6] V. Conitzer and T. Sandholm. Nonexistence of voting rules that are usually hard to manipulate. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence*, pages 627–634, 2006.

[7] Vincent Conitzer, Tuomas Sandholm, and Jerome Lang. When are elections with few candidates hard to manipulate? *Journal of the ACM*, 54(3):1–33, 2007.

[8] E. Elkind and H. Lipmaa. Hybrid voting protocols and hardness of manipulation. In *16th Annual International Symposium on Algorithms and Computation*, Lecture Notes in Computer Science, pages 206–215. Springer-Verlag, 2005.

[9] E. Elkind and H. Lipmaa. Small coalitions cannot manipulate voting. In *International Conference on Financial Cryptography*, Lecture Notes in Computer Science. Springer-Verlag, 2005.

[10] E. Ephrati and J. S. Rosenschein. A heuristic technique for multiagent planning. *Annals of Mathematics and Artificial Intelligence*, 20:13–67, 1997.

[11] G. Erdelyi, L. A. Hemaspaandra, J. Rothe, and H. Spakowski. On approximating optimal weighted lobbying, and frequency of correctness versus average-case polynomial time. Technical report, arXiv:cs/0703097v1, 2007.

[12] S. Ghosh, M. Mundhe, K. Hernandez, and S. Sen. Voting for movies: the anatomy of a recommender system. In *Proceedings of the Third Annual Conference on Autonomous Agents*, pages 434–435, 1999.

[13] A. Gibbard. Manipulation of voting schemes. *Econometrica*, 41:587–602, 1973.

[14] T. Groves. Incentives in teams. *Econometrica*, 41:617–631, 1973.

[15] T. Haynes, S. Sen, N. Arora, and R. Nadella. An automated meeting scheduling system that utilizes user preferences. In *Proceedings of the First International Conference on Autonomous Agents*, pages 308–315, 1997.

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

[17] D. S. Hochbaum. *Approximation Algorithms for NP-Hard Problems.* PWS Publishing Company, 1997.

[18] H. Moulin. On strategy-proofness and single peakedness. *Public Choice*, 35:437–455, 1980.

[19] A. D. Procaccia and J. S. Rosenschein. Average-case tractability of manipulation in voting via the fraction of manipulators. In *Proceedings of The Sixth International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 718–720, 2007.

[20] A. D. Procaccia and J. S. Rosenschein. Junta distributions and the average-case complexity of manipulating elections. *Journal of Artificial Intelligence Research*, 28:157–181, 2007.

[21] A. D. Procaccia, J. S. Rosenschein, and A. Zohar. Multi-winner elections: Complexity of manipulation, control and winner-determination. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, pages 1476–1481, 2007.

[22] M. Satterthwaite. Strategy-proofness and Arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory*, 10:187–217, 1975.

[23] W. Vickrey. Counter speculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16(1):8–37, 1961.

A Monotonicity Proofs

Proof of Lemma 3.3 (Scoring Rules). Let C be the candidates, $p \in C$ is the preferred candidate, S is the set of truthful voters, and W are the weights for the manipulators T. Denote $|C| = m$, $|S| = N$, $|W| = |T| = n$. It is enough to show that if there is a manipulation for the set T, then for the same instance with manipulator voters $T' = T + \{v\}$ with weights list $W' = W + \{w\}$, where $w \geq 1$ is any integer, there is also a manipulation, and the rest will follow by induction. Let $\vec{\alpha} = \langle \alpha_1, \dots, \alpha_m \rangle$ be the scores vector of the rule. Let X_S be the preference orders of voters in S, and X_T the preference orders of voters in T that make p win. Fix some order on voters in T. By definition, for all $c \in C \setminus \{p\}$, $S_n(c) < S_n(p)$. Let the additional voter of T' vote with p at the first place, and some arbitrary order on the other candidates. Then for all $c \in C \setminus \{p\}$, $S_{n+1}(p) = S_n(p) + w\alpha_1 > S_n(c) + w\alpha_1 \geq S_{n+1}(c)$, and so we got the ballot of votes of T' to make p win. □

Proof of Lemma 3.14 (Maximin). Let X_S be the preference orders of the voters in S, and X_T the preference orders of voters in T that make p win. We need to show that there are preference orders for $T' = T + \{v\}$ with weights list $W' = W + \{w\}$ where $w \geq 1$ is any integer, that make p win. Fix some order on voters in T. By definition, for all $c \in C \setminus \{p\}$, $S_n(c) < S_n(p)$. Let the additional voter of T' vote with p at the first place, and some arbitrary order on the other candidates. Then for all $c \in C \setminus \{p\}$, $S_{n+1}(p) = S_n(p) + w > S_n(c) + w \geq S_{n+1}(c)$, and so we got the ballot of votes of T' to make p win. □

Proof of Lemma 3.16 (Plurality with Runoff). Let C be the candidates, $p \in C$ is the preferred candidate, S is the set of truthful voters, and W are the weights for manipulators of T. Suppose that there is a ballot of votes of T that makes p win the election. We need to show that there is a ballot making p win for the set $W' = W + \{w\}$, where $w \geq 1$. Let g be the candidate that proceeds with p to the second round in the winning ballot for W. Let the additional voter vote $p \succ \ldots$. Then the plurality score of p and g will not decrease, while the plurality score of any other candidate will remain the same, and so p and g will proceed to the next round in the new ballot as well. In the second round p will beat g in the new ballot, since the total weight of the voters who prefer p to g increased, while the total weight of voters who prefer g to p remained the same. Thus, p will win the election in the new ballot. □

B Copeland is Not Monotone in Weights

When discussing Scoring rules, Maximin, and Plurality with runoff, we are motivated to look for approximate solutions to the CCWM problem by the fact that these voting rules are monotone in weights (see Appendix A). However, with respect to Copeland this is not the case. The next example illustrates this fact. Consider the next setting: $C = \{p, 1, 2, 3\}$, $N = |S| = 6$. All the weights equal 1. The votes of the voters in S are shown in the next table:

Voter in S	Vote
1	$p \succ 1 \succ 2 \succ 3$
2	$p \succ 2 \succ 1 \succ 3$
3	$3 \succ p \succ 1 \succ 2$
4	$3 \succ p \succ 2 \succ 1$
5	$1 \succ 2 \succ 3 \succ p$
6	$2 \succ 1 \succ 3 \succ p$

The pairwise results are given in the next table. In the cell corresponding to the row of candidate g and the column of candidate g', we write "$a : b$" to indicate that g is preferred to g' by a voters, and g' is preferred to g by b voters (i.e., $a = N_0(g, g')$, $b = N_0(g', g)$):

	p	1	2	3
p		4:2	4:2	2:4
1	2:4		3:3	4:2
2	2:4	3:3		4:2
3	4:2	2:4	2:4	

From the above table we calculate that $S_0(p) = 1$, $S_0(1) = S_0(2) = 0$, $S_0(3) = -1$, so p wins the election in this setting. However, if we add another voter (with weight 1), then no matter what his vote will be, p will not win the election: if the additional voter puts 1 above 2, then 1 will win, and otherwise 2 will win.

Remark B.1. It is easy to see, however, that whenever there is a manipulation for the coalition with weights W, then there is also a manipulation for coalition with weights $W + \{w\} + \{w\}$, where $w \geq 1$ is any integer: the first additional voter makes an arbitrary vote, and the second additional voter reverses the first's ranking.

C An Example for Algorithm 2 and Borda

We present an example of a case where there is a manipulation for weights list W, but Algorithm 2 will find a manipulation only for weights list $W + \{w'\}$. In our example $W = \{1, 1, 1, 1\}$, $w' = 1$, so we are actually talking about the special case of unweighted coalitions. Consider the set $C = \{p, 1, 2, 3, 4, 5, 6\}, m = |C| = 7,\ N = |S| = 5$. 3 voters in S voted $6 \succ 5 \succ 4 \succ 3 \succ 2 \succ p \succ 1$, and the other 2 voters in S voted $2 \succ 3 \succ 4 \succ 5 \succ 6 \succ p \succ 1$. When applying Algorithm 2 to this input, the voters in T will award the candidates with the following scores (we denote by $\alpha_j(c)$ the points that voter j gives to candidate c):[2]

Candidate $c \in C$	p	1	2	3	4	5	6
$S_0(c)$	5	0	18	19	20	21	22
$\alpha_1(c)$	6	5	4	3	2	1	0
$\alpha_2(c)$	6	5	0	1	2	3	4
$\alpha_3(c)$	6	5	4	3	2	1	0
$\alpha_4(c)$	6	5	0	1	2	3	4
$\alpha_5(c)$	6	5	4	3	2	1	0

So the cumulative scores will be as follows:

Candidate $c \in C$	p	1	2	3	4	5	6
$S_0(c)$	5	0	18	19	20	21	22
$S_1(c)$	11	5	22	22	22	22	22
$S_2(c)$	17	10	22	23	24	25	26
$S_3(c)$	23	15	26	26	26	26	26
$S_4(c)$	29	20	26	27	28	29	30
$S_5(c)$	35	25	30	30	30	30	30

Note that after 4 stages, the algorithm still did not find a manipulation: $S_4(p) = 29 < 30 = S_4(6)$. However, if we change the votes of the third and fourth voters of T, then we find an appropriate ballot:

Candidate $c \in C$	p	1	2	3	4	5	6
$S_0(c)$	5	0	18	19	20	21	22
$\alpha_1(c)$	6	5	4	3	2	1	0
$\alpha_2(c)$	6	5	0	1	2	3	4
$\alpha'_3(c)$	6	5	3	4	0	1	2
$\alpha'_4(c)$	6	5	3	1	4	2	0

Now the cumulative scores are:

Candidate $c \in C$	p	1	2	3	4	5	6
$S_0(c)$	5	0	18	19	20	21	22
$S_1(c)$	11	5	22	22	22	22	22
$S_2(c)$	17	10	22	23	24	25	26
$S'_3(c)$	23	15	25	27	24	26	28
$S'_4(c)$	29	20	28	28	28	28	28

Evidently, for any $c \in C \setminus \{p\},\ S'_4(p) = 29 > S'_4(c)$.

[2]We assumed here that when two candidates have the same scores up until a certain stage, the current voter will award fewer points to the candidate with lower index, but any tie-breaking rule will give the same results.

D Proof of Theorem 3.15

Let us introduce some more notation. For candidates $g, g' \in C$ and $0 \leq j \leq n$ we denote by $N_j(g, g')$ the total weight of the voters after j stages (including the voters in S) that prefer g over g'. So $S_j(g) = \min_{g' \in C \setminus \{g\}} N_j(g, g')$. We also denote for $g \in C$, $0 \leq j \leq n$: $\mathrm{MIN}_j(g) = \{h \in C \setminus \{g\} \mid N_j(g, h) = S_j(g)\}$. Fixing the set C, $p \in C$, and an order on the weights list W, we denote by $f(j)$ the maximal score of p's opponents distributed by Algorithm 1 after j stages: $f(j) = \max_{g \in C \setminus \{p\}} S_j(g)$.

In Algorithm 1, p is always placed at the top of each preference, and so with each voter its score grows by the weight of this voter. In our next lemma we will put forward an upper bound on the growth rate of the scores of p's opponents.

Lemma D.1. *Consider Algorithm 1 applied to the Maximin rule. Denote by w_j the weight of the j-th voter processed by the algorithm. Then for all $0 \leq j \leq n-2$, $f(j+2) \leq f(j) + \max\{w_{j+1}, w_{j+2}\}$.*

Proof. Let $0 \leq j \leq n-2$. Let $g \neq p$ be any candidate. By definition $S_j(g) \leq f(j)$. We would like to show that $S_{j+2}(g) \leq f(j) + \max\{w_{j+1}, w_{j+2}\}$.

If $S_{j+1}(g) \leq f(j)$, then $S_{j+2}(g) \leq S_{j+1}(g) + w_{j+2} \leq f(j) + \max\{w_{j+1}, w_{j+2}\}$, and we are done. So let us assume now that $S_{j+1}(g) > f(j)$. Define a directed graph $G = (V, E)$, where $V = \{g\} \cup \{x \in C \setminus \{p\} \mid x \text{ was ranked after } g \text{ at stage } j+1\}$, and $(x, y) \in E$ iff $y \in \mathrm{MIN}_j(x)$. There is at least one outgoing edge from g in E, since otherwise there was $g' \in \mathrm{MIN}_j(g)$ that voter $j+1$ put before g, and then $S_{j+1}(g) = S_j(g) \leq f(j)$, a contradiction. In addition, for all $x \in V \setminus \{g\}$, there is at least one outgoing edge from x in E, because otherwise there was $x' \in \mathrm{MIN}_j(x)$ that was ranked before g at stage $j+1$, and then Algorithm 1 would have ranked x before g (after x') since then $S_{j+1}(x) = S_j(x) \leq f(j) < S_{j+1}(g)$, a contradiction.

For $x \in V$, denote by $V(x)$ all the vertices y in V s.t. there exists a directed path from x to y. Denote by $G(x)$ the sub-graph of G induced by $V(x)$. It is easy to see that $G(g)$ contains at least one circle. Let U be one such circle. Let $g' \in U$ be the vertex that was picked first among the vertices of U at stage $j+1$. Let g'' be the vertex before g' in the circle: $(g'', g') \in U$. Since g'' was ranked after g' at stage $j+1$, it follows that $S_{j+1}(g'') = S_j(g'') \leq f(j)$.

Suppose by way of contradiction that $S_{j+2}(g) > f(j) + \max\{w_{j+1}, w_{j+2}\}$. g was ranked by $j+2$ at place t^*. Then g'' was ranked by $j+2$ at place $< t^*$, since otherwise when we would have reached the place t^*, we would pick g'' (with score $S_{j+2}(g'') \leq f(j) + w_{j+2} < S_{j+2}(g)$) instead of g, a contradiction.

Denote by X_1 all the vertices in $V(g)$ that have an outgoing edge to g'' in $G(g)$. For all $x \in X_1$, $g'' \in \mathrm{MIN}_j(x)$, i.e., $S_j(x) = N_j(x, g'')$. All $x \in X_1$ were ranked by $j+2$ before g, since otherwise, if there was $x \in X_1$, s.t. until the place t^* it still was not added to the preference list, then when evaluating its score on place t^*, we would get: $S_{j+2}(x) \leq N_{j+2}(x, g'') = N_{j+1}(x, g'') \leq N_j(x, g'') + w_{j+1} = S_j(x) + w_{j+1} < S_{j+2}(g)$, and so we would put x instead of g. Denote by X_2 all the vertices in $V(g)$ that have an outgoing edge to some vertex $x \in X_1$. In the same manner we can show that all the vertices in X_2 were ranked at stage $j+2$ before g. We continue this way backwards on the path in $G(g)$ from g'' to g, and we get that one vertex before g on that path, say g_0, was ranked by $j+2$ before g. And thus,

$$\begin{aligned} S_{j+2}(g) &\leq N_{j+2}(g, g_0) = N_{j+1}(g, g_0) \leq N_j(g, g_0) + w_{j+1} \\ &= S_j(g) + w_{j+1} \leq f(j) + \max\{w_{j+1}, w_{j+2}\} \\ &< S_{j+2}(g), \end{aligned}$$

a contradiction. ∎

We are now ready to prove the theorem.

Proof of Theorem 3.15. We prove part 1. Algorithm 1 returns *true* only if it constructs a (valid) ballot that makes p win, and thus if there is no ballot making p win, Algorithm 1 will return *false*.

We now prove part 2. Suppose that there exists a ballot Z_T making p win for weights list $W = \{w_1, \ldots, w_n\}$. Let $S'_j(g)$ be the scores implied by Z_T. Then:

$$f(0) < S'_n(p) \leq S_0(p) + \sum_{i=1}^{n} w_i \tag{11}$$

Let $W' = W + W + X$, where X is some list of weights (possibly empty). We need to show that $S_{|W'|}(p) > f(|W'|)$. In Algorithm 1, after sorting the weights of W', the equal weights of two copies of W will be adjacent, i.e., the order of weights in W' will be of the form:

$$x_1, \ldots, x_{q_1}, w_1, w_1, x_{q_1+1}, \ldots, x_{q_2}, w_2, w_2, \ldots, w_n, w_n, x_{q_n+1}, \ldots, x_{|X|}$$

By Lemma D.1, one can prove by induction that:

$$f(|W'|) \leq f(0) + \sum_{i=1}^{|X|} x_i + \sum_{i=1}^{n} w_i \tag{12}$$

And so by (11) and (12) we have:

$$S_{|W'|}(p) = S_0(p) + \sum_{i=1}^{|X|} x_i + 2\sum_{i=1}^{n} w_i > f(0) + \sum_{i=1}^{|X|} x_i + \sum_{i=1}^{n} w_i \geq f(|W'|)$$

□

E Proof of Theorem 3.18

We start with part 1. Note that

$$\begin{aligned} \overline{x} = (\overline{x}_1, \ldots, \overline{x}_n) \text{ satisfies } &\sum_{j=1}^{n} w_j \overline{x}_j \leq \sum_{j=1}^{n} w_j - \lambda_g \\ \iff x = \vec{1} - \overline{x} \text{ satisfies } &\sum_{j=1}^{n} w_j x_j \geq \lambda_g \end{aligned} \tag{13}$$

and thus when voters corresponding to weights returned by the function SUBSET-OF-WEIGHTS-APPROXIMATE(), vote $g \succ \ldots$, they ensure that g proceeds to the second round. It is easy to see that whenever Algorithm 3 returns *true*, it actually finds a (valid) ballot making p win the election, and so if there is no such ballot, then the algorithm will return *false*.

We now move on to part 2. Denote by A_W the instance of the problem with weights list W. Suppose that there exists ballot X_T of votes in T s.t. combined with preferences X_S of voters of S, it makes p win the election in A_W. We will denote by $\beta'_Y(g)$ the plurality score of g from voter set Y under the preferences $X_S \cup X_T$. Also, we denote $N'_Y(g, g') = \sum_{v \in U_{X_S \cup X_T}} w_v$, where $U_{X_S \cup X_T}$ is the set of all the voters in Y that prefer g to g' under $X_S \cup X_T$. Let $0 \leq u \leq \max(W)$, $W' = W + \{w_{n+1}, \ldots, w_{n+l}\}$, where $\sum_{j=1}^{l} w_{n+j} \geq u$. We need to show that Algorithm 3 will return *true* on the input W', u.

There is a candidate $g \neq p$ that passes together with p to the second round when applying the preferences X_T together with X_S on A_W, and thus for each $g' \notin \{p, g\}$ and $c \in \{p, g\}$, if $c \gg g'$, then $\beta'_S(c) + \beta'_T(c) \geq \beta'_S(g') + \beta'_T(g')$, and if $g' \gg c$, then $\beta'_S(c) + \beta'_T(c) > \beta'_S(g') + \beta'_T(g')$. Also,

$$\beta'_T(p) + \beta'_T(g) \leq \sum_{j=1}^{n} w_j \tag{14}$$

Now consider Algorithm 3 applied to $A_{W'}$. If it does not reach g in the main loop, then it will exit earlier returning "true", meaning that it will find a desired ballot making p win. Otherwise, it will reach the candidate g. λ_g is the minimal sum of weights that ensures that g will continue to the second round, and hence

$$\lambda_g \leq \beta'_T(g) \leq \sum_{j=1}^{n} w_j \leq \sum_{j=1}^{n+l} w_j \tag{15}$$

We will reach the function SUBSET-OF-WEIGHTS-APPROXIMATE(), and enter it with arguments W', λ_g and u. By (13), the vector $x = (x_1, \ldots, x_{n+l})$ returned by SUBSET-OF-WEIGHTS-APPROXIMATE() satisfies $\sum_{j=1}^{n+l} w_j x_j \geq \lambda_g$, and so g will continue to the next round. Now we show that p will also continue to the next round. Denote by H the maximization problem

$$\begin{aligned} &\max \sum_{j=1}^{n+l} w_j \overline{x}_j \\ &\text{s.t.} \sum_{j=1}^{n+l} w_j \overline{x}_j \leq \sum_{j=1}^{n+l} w_j - \lambda_g \\ &\overline{x}_j \in \{0, 1\}, \text{ for } j = 1, \ldots, n+l \end{aligned} \tag{16}$$

Let $J^* = \{j \mid \overline{x}_j = 1, \overline{x} = (\overline{x}_1, \ldots, \overline{x}_{n+l})$ is the optimal solution to $H\}$. Denote $P^* = \sum_{j \in J^*} w_j$. Let $H(k)$ the scaled version of the above maximization problem:

$$\begin{aligned} &\max \sum_{j=1}^{n+l} \left\lfloor \frac{w_j}{k} \right\rfloor \overline{x}_j \\ &\text{s.t.} \sum_{j=1}^{n+l} w_j \overline{x}_j \leq \sum_{j=1}^{n+l} w_j - \lambda_g \\ &\overline{x}_j \in \{0, 1\}, \text{ for } j = 1, \ldots, n+l \end{aligned} \tag{17}$$

Let $J(k) = \{j \mid \overline{x}_j = 1, \overline{x} = (\overline{x}_1, \ldots, \overline{x}_{n+l})$ is the optimal solution to $H(k)\}$. Let $P(k) = \sum_{j \in J(k)} w_j$. Now, $\overline{x} = (\overline{x}_1, \ldots, \overline{x}_{n+l})$ which we obtained in SUBSET-OF-WEIGHTS-APPROXIMATE() satisfies:

$$\begin{aligned} \sum_{j=1}^{n+l} w_j \overline{x}_j &= \sum_{j \in J(k)} w_j \geq \sum_{j \in J(k)} k \left\lfloor \frac{w_j}{k} \right\rfloor \geq \sum_{j \in J^*} k \left\lfloor \frac{w_j}{k} \right\rfloor \geq \sum_{j \in J^*} (w_j - (k-1)) \\ &= \sum_{j \in J^*} w_j - (k-1)|J^*| = P^* - (k-1)|J^*| \end{aligned} \tag{18}$$

Hence, the vector $x = \vec{1} - \overline{x}$ returned by the function, satisfies:

$$\begin{aligned}\sum_{j=1}^{n+l} w_j x_j &\leq \sum_{j=1}^{n+l} w_j - P^* + (k-1)|J^*| \\ &\leq \sum_{j=1}^{n+l} w_j - P^* + \left\lfloor \frac{u}{2(n+l)} \right\rfloor (n+l) \\ &\leq \sum_{j=1}^{n+l} w_j - P^* + \left\lfloor \frac{u}{2} \right\rfloor\end{aligned} \tag{19}$$

By definition of P^*, we get:

$$\begin{aligned}\sum_{j=1}^{n+l} w_j - P^* &= \min\{\sum_{j=1}^{n+l} w_j x_j \mid \sum_{j=1}^{n+l} w_j x_j \geq \lambda_g, x_j \in \{0,1\}, 1 \leq j \leq n+l\} \\ &\leq \min\{\sum_{j=1}^{n} w_j x_j \mid \sum_{j=1}^{n} w_j x_j \geq \lambda_g, x_j \in \{0,1\}, 1 \leq j \leq n\} \\ &\leq \beta'_T(g)\end{aligned} \tag{20}$$

Combining (19) and (20), we get that for vector x returned by the function SUBSET-OF-WEIGHTS-APPROXIMATE():

$$\beta_{T'}(g) = \sum_{j=1}^{n+l} w_j x_j \leq \beta'_T(g) + \left\lfloor \frac{u}{2} \right\rfloor \tag{21}$$

In the algorithm, all the voters j s.t. $x_j = 0$ will vote $p \succ \ldots$, and so we will have

$$\begin{aligned}\beta_{T'}(p) &= \sum_{j=1}^{n+l} w_j - \sum_{j=1}^{n+l} w_j x_j \geq \sum_{j=1}^{n} w_j + u - (\beta'_T(g) + \left\lfloor \frac{u}{2} \right\rfloor) \\ &= \sum_{j=1}^{n} w_j - \beta'_T(g) + \left\lceil \frac{u}{2} \right\rceil \geq \beta'_T(p) + \left\lceil \frac{u}{2} \right\rceil\end{aligned} \tag{22}$$

Therefore, p will continue to the next round.

We now prove that p beats g in the next round. If $g \gg p$, then in the winning ballot X_T, $N'_S(p,g) + N'_T(p,g) > N'_S(g,p) + N'_T(g,p)$, otherwise $N'_S(p,g) + N'_T(p,g) \geq N'_S(g,p) + N'_T(g,p)$. From (21) we get:

$$N_{T'}(g,p) = \beta_{T'}(g) \leq \beta'_T(g) + \left\lfloor \frac{u}{2} \right\rfloor \leq N'_T(g,p) + \left\lfloor \frac{u}{2} \right\rfloor \tag{23}$$

Thus, from (23):

$$\begin{aligned}N_{T'}(p,g) &= \sum_{j=1}^{n+l} w_j - N_{T'}(g,p) \geq \sum_{j=1}^{n} w_j + u - (N'_T(g,p) + \left\lfloor \frac{u}{2} \right\rfloor) \\ &= N'_T(p,g) + \left\lceil \frac{u}{2} \right\rceil\end{aligned} \tag{24}$$

So, for $g \gg p$ we get

$$\begin{aligned}N'_S(p,g) + N_{T'}(p,g) &\geq N'_S(p,g) + N'_T(p,g) + \left\lceil \frac{u}{2} \right\rceil \\ &> N'_S(g,p) + N'_T(g,p) + \left\lfloor \frac{u}{2} \right\rfloor \\ &\geq N'_S(g,p) + N_{T'}(g,p)\end{aligned} \tag{25}$$

In the same way, for $p \gg g$ we get

$$N'_S(p,g) + N_{T'}(p,g) \geq N'_S(g,p) + N_{T'}(g,p) \tag{26}$$

Therefore, by the algorithm p wins the second round of the election, and hence the entire election; the algorithm will return *true*.

Next, we prove part 3. Using the notation of the previous item, let $P(k)$ be the maximum sum of weights from $W = \{w_1, \dots, w_n\}$, solving the scaled maximization problem $H(k)$.[3] There is a well-known dynamic programming algorithm solving the knapsack problem $H(k)$ in $O(nP(k))$ [17, chapter 9]. Furthermore, $P(k) \leq \sum_{j=1}^{n} \lfloor \frac{w_j}{k} \rfloor \leq n \lfloor \frac{\max(W)}{k} \rfloor \leq n\frac{\max(W)}{k}$. $k = \lfloor \frac{u}{2n} \rfloor + 1 \geq \frac{u+1}{2n}$, and so we have:

$$P(k) \leq n\frac{\max(W)}{k} \leq n\frac{\max(W)}{\frac{u+1}{2n}} = 2n^2\frac{\max(W)}{u+1} \tag{27}$$

Thus we can solve $H(k)$ in $O(nP(k)) = O(n^3 \cdot \frac{\max(W)}{u+1})$. It is easy to see that all the other steps of Algorithm 3 are polynomial in its inputs; hence, the proof is completed. □

F Proof of Theorem 4.3

We prove this theorem via the Lemmas F.1 – F.4. First, we define the set $X_n = \{x \in C \setminus \{p\} \mid x$ was ranked last at stage j for $1 \leq j \leq n\}$. And define $Y_n = \{y \in C \setminus \{p\} \mid S_n(y) \geq \min(S_n(X_n))\}$. From the definition, $X_n \subseteq Y_n$. Also by definition:

$$\forall g \notin Y_n \cup \{p\},\ S_n(g) < \min(S_n(Y_n)) \tag{28}$$

We denote by $\alpha_j(x)$ the number of points x was awarded in stage j.

Lemma F.1. *For all $y_1, y_2 \in Y_n$, $|S_n(y_1) - S_n(y_2)| \leq 1$.*

Proof. Let $x^* \in X_n$ s.t. $S_n(x^*) = \min(S_n(X_n))$. Let $y \in Y_n$. By definition, $S_n(x^*) \leq S_n(y)$. We would like to show that $S_n(y) \leq S_n(x^*) + 1$. Suppose for contradiction that $S_n(y) - S_n(x^*) \geq 2$. Let $1 \leq j \leq n$ maximal s.t. $\alpha_j(x^*) = 0$. Then:

$$\begin{aligned} S_j(y) - S_j(x^*) &= \Big[S_n(y) - \sum_{k=j+1}^{n} \alpha_k(y)\Big] - \Big[S_n(x^*) - \sum_{k=j+1}^{n} \alpha_k(x^*)\Big] \\ &\geq \big[S_n(y) - (n-j)\big] - \big[S_n(x^*) - (n-j)\big] \\ &= S_n(y) - S_n(x^*) \geq 2 \end{aligned}$$

Therefore $S_{j-1}(y) - S_{j-1}(x^*) \geq 1$, and so $S_{j-1}(y) > S_{j-1}(x^*)$, a contradiction to $\alpha_j(x^*) = 0$. We showed that for all $y \in Y_n$, $S_n(x^*) \leq S_n(y) \leq S_n(x^*) + 1$, and hence for all $y_1, y_2 \in Y_n$, $|S_n(y_1) - S_n(y_2)| \leq 1$. □

Lemma F.2. *Define $q(n) := \frac{1}{|Y_n|}\sum_{y \in Y_n} S_n(y)$. Let Z_T be any preferences list of voters in T, and $S'_n(g)$ be the scores of $g \in C$ which are implied by Z_T (including votes in S). Then $q'(n) := \frac{1}{|Y_n|}\sum_{y \in Y_n} S'_n(y) \geq q(n)$.*

[3] We slightly abuse notation here, as we defined the optimization problems for weights set $W' = \{w_1, \dots, w_{n+l}\}$, but the definition for the set W is analogous.

Proof. The above fact is true since in the algorithm, at every stage j, there is some $x \in X_n \subseteq Y_n$ such that $\alpha_j(x) = 0$, and so for every j, the sum $\sum_{y \in Y_n} \alpha_j(y) = |Y_n| - 1$ is minimal. Formally, let us denote by $\alpha'_j(g)$ the number of points candidate g gets from voter j in Z_T. Then:

$$\begin{aligned} q(n) &= \frac{1}{|Y_n|}\big(\sum_{y \in Y_n} S_0(y) + n(|Y_n| - 1)\big) \\ &\leq \frac{1}{|Y_n|}\big(\sum_{y \in Y_n} S_0(y) + \sum_{j=1}^{n} \sum_{y \in Y_n} \alpha'_j(y)\big) \\ &= \frac{1}{|Y_n|} \sum_{y \in Y_n} S'_n(y) = q'(n) \end{aligned}$$

□

Lemma F.3. *If $S_n(p) > \max(S_n(Y_n))$ then Algorithm 2 will find the manipulation that makes p win.*

Proof. By Equation (28), for all $g \in C \setminus \{p\}$, $S_n(g) \leq \max(S_n(Y_n))$, and so if $S_n(p) > \max(S_n(Y_n))$, then for all $g \in C \setminus \{p\}$, $S_n(p) > S_n(g)$, and so the algorithm will find the manipulation. □

Lemma F.4. *If $S_n(p) \leq \max(S_n(Y_n))$ then there exists no manipulation.*

Proof. Let Z_T be any set of preferences of voters in T, and let $S'_n(g)$, $q'(n)$ and $\alpha'_j(g)$ as in Lemma F.2. As for all j, $\alpha_j(p) = 1 \geq \alpha'_j(p)$, it follows that $S_n(p) \geq S'_n(p)$. There is at least one $g_0 \in Y_n$ s.t. $S'_n(g_0) \geq \lceil q'(n) \rceil$. By Lemma F.2, $\lceil q'(n) \rceil \geq \lceil q(n) \rceil$. By Lemma F.1, $\lceil q(n) \rceil = \max(S_n(Y_n))$. Combining the foregoing steps, we obtain:

$$S'_n(g_0) \geq \lceil q'(n) \rceil \geq \lceil q(n) \rceil = \max(S_n(Y_n)) \geq S_n(p) \geq S'_n(p)$$

We conclude that p does not win under Z_T, and hence there is no ballot of votes in T that makes p win the election. □

G Algorithms

Algorithm 2 Decides CCWM in Scoring rules

procedure SCORING-RULES-GREEDY$(C, p, S_0(C), W)$ ▷ $S_0(C)$ is the list of scores of candidates distributed by voters in S, W is the list of weights for voters in T, $|W| = |T| = n$
    for $j = 1, \ldots, n$ **do** ▷ Go over voters in T
        $S_j(p) = S_{j-1}(p) + w_j \alpha_1$ ▷ Put p at the first place of the j-th preference list
        Let $t_1, t_2, \ldots, t_{m-1}$ s.t. $\forall l,\ S_{j-1}(c_{t_{l-1}}) \leq S_{j-1}(c_{t_l})$
        j votes $p \succ c_{t_1} \succ \ldots \succ c_{t_{m-1}}$
        for $l = 1, \ldots, m-1$ **do** ▷ Update the scores
            $S_j(c_{t_l}) = S_{j-1}(c_{t_l}) + w_j \alpha_{l+1}$
        end for
    end for
    if $\text{argmax}_{c \in C}\{S_n(c)\} = \{p\}$ **then** ▷ p wins
        return true
    else
        return false
    end if
end procedure

Algorithm 3 Decides CCWM in Plurality with runoff with desired accuracy

procedure PLURALITY-WITH-RUNOFF(C, p, X_S, W_S, W, u) ▷ X_S is the set of preferences of voters in S, W_S are the weights of voters in S, $W = \{w_1, \ldots, w_n\}$ are the weights of voters in T, u is the size of error window
    for g in $C \setminus \{p\}$ **do** ▷ Go over candidates in $C \setminus \{p\}$
        if there exists $g' \in \text{argmax}_{g' \in C\setminus\{p\}} \beta_S(g')$, $g' \neq g$ s.t. $g' \gg g$ **then**
            $\lambda_g \leftarrow \max_{g' \in C\setminus\{p\}} \beta_S(g') - \beta_S(g) + 1$
        else
            $\lambda_g \leftarrow \max_{g' \in C\setminus\{p\}} \beta_S(g') - \beta_S(g)$
        end if
        if $\lambda_g > \sum_{i=1}^{n} w_i$ **then** ▷ If we cannot make g pass to the next round
            continue ▷ Go to the next candidate in the main loop
        end if
        $x \leftarrow$ SUBSET-OF-WEIGHTS-APPROXIMATE(W, λ_g, u)
            ▷ $x \in \{0,1\}^n$ minimizes $\{\sum_{j=1}^{n} w_j x_j \mid \sum_{j=1}^{n} w_j x_j \geq \lambda_g, \forall j, x_j \in \{0,1\}\}$
        All the voters j s.t. $x_j = 1$ vote $g \succ \ldots$ ▷ Order of candidates except g is arbitrary
        All the voters j s.t. $x_j = 0$ vote $p \succ \ldots$
        if $\exists g' \in C \setminus \{p, g\}$ s.t. $\big(\beta_S(g') > \beta_S(p) + \beta_T(p)\big)$
            or $\big(\beta_S(g') = \beta_S(p) + \beta_T(p)$ and $g' \gg p\big)$ **then**
            continue ▷ p does not pass to next round
        end if
        if $\big(N_{S\cup T}(p, g) > N_{S\cup T}(g, p)\big)$ or $\big(N_{S\cup T}(p, g) = N_{S\cup T}(g, p)$ and $p \gg g\big)$ **then**
            return true ▷ p beats g in the second round
        else
            continue
        end if
    end for
    return false ▷ No appropriate g was found
end procedure

procedure SUBSET-OF-WEIGHTS-APPROXIMATE(W, λ_g, u) ▷ $W = \{w_1, \ldots, w_n\}$ are the weights of voters in T, λ_g is the minimum total sum of desired weights, u is the size of error window
    Check that $0 \leq u \leq \max(W)$
    $k \leftarrow \left\lfloor \frac{u}{2n} \right\rfloor + 1$
    Solve by dynamic prog.: $\max\{\sum_{j=1}^{n} \left\lfloor \frac{w_j}{k} \right\rfloor \overline{x}_j \mid \sum_{j=1}^{n} w_j \overline{x}_j \leq \sum_{j=1}^{n} w_j - \lambda_g, \forall j,\ \overline{x}_j \in \{0,1\}\}$
    Let $\overline{x} \in \{0,1\}^n$ be the vector that maximizes the above sum
    return $\vec{1} - \overline{x}$ ▷ $\vec{1}$ is the vector of n 1-s
end procedure