Computational Aspects of Distortion

Soroush Ebadian
University of Toronto
soroush@cs.toronto.edu

Aris Filos-Ratsikas
University of Edinburgh
Aris.Filos-Ratsikas@ed.ac.uk

Mohamad Latifian
University of Toronto
latifian@cs.toronto.edu

Nisarg Shah
University of Toronto
nisarg@cs.toronto.edu

Abstract

The distortion framework in social choice theory allows quantifying the efficiency of (randomized) selection of an alternative based on the preferences of a set of agents. We make two fundamental contributions to this framework.

First, we develop a linear-programming-based algorithm for computing the optimal randomized decision on a given instance, which is simpler and faster than the state-of-the-art solutions. For practitioners who may prefer to deploy a classical decision-making rule over the aforementioned optimal rule, we develop an algorithm based on non-convex quadratic programming for computing the *exact* distortion of any (and the best) randomized positional scoring rule. For a small number of alternatives, we find that the exact distortion bounds are significantly better than the asymptotic bounds established in prior literature and lead to different recommendations on which rules to use.

These results rely on a novel characterization of the instances yielding the worst distortion, which may be of independent interest.

1 Introduction

The area of computational social choice aims to find good ways of aggregating the preferences of a set of people, or *voters*, into a desirable collective decision. The notion of distortion in social choice was introduced by Procaccia and Rosenschein [27] in 2006 to measure the inefficiency of voting rules that only have access to limited information about the preferences of the agents. Over the past two decades, this literature has given rise to a plethora of interesting results about the distortion of rules in several different domains of interest, including single-winner and committee voting [2, 6, 11, 12, 13, 14, 24, 25], participatory budgeting [10, 21], and matching [3, 4, 5, 15, 19].

One of the most fundamental settings that have been studied under this framework is that of implicit utilitarian voting, in which the preferences of the agents are expressed by normalised but otherwise unrestricted cardinal values, and the voting rules only have access to the ordinal preference rankings induced by those values. For this setting, some of the very first works in the area already identified the limitations of voting rules, even those that employ randomization in their decisions, by providing a general lower bound of $\Omega(\sqrt{m})$ on the best achievable distortion [13], where m is the number of alternatives. Far more recently, Ebadian et al. [16] designed a voting rule that always achieves a distortion of $O(\sqrt{m})$, thus settling the asymptotic efficiency of general voting rules. Even more recently, Ebadian et al. [17] studied the distortion of a class of very natural randomized voting rules, called randomized positional scoring rules (RPSRs), which select an outcome proportionally to its score, according to some predefined scoring vector. Rules in this class display several attractive properties (such as anonymity, neutrality, and strategyproofness [9, 20]) and in [17] are deemed *explainable*, as they use randomization in a rather straightforward manner. The best achievable asymptotic distortion by RPSRs has been shown to be $\Theta(\sqrt{m \log m})$ [11, 13], and the asymptotic distortion bounds for several interesting RPSRs are known [17].

From the preceding paragraphs, it is evident that the *worst-case, asymptotic* distortion of voting rules for implicit utilitarian voting is pretty well understood. Still, many of the preference vectors that we encounter in applications exhibit a certain structure, which does not necessarily allow for the pathological examples that establish the worst-case lower bounds to arise. This motivates the following first question:

1. Given a vector of agents' preferences, how can we compute the best possible distortion on this vector? How can we find a voting rule that achieves it?

Additionally, even in a worst-case regime, in many cases we should not be satisfied with merely asymptotic bounds, as the hidden constants, even when small, can have a significant impact on the calculated distortion bounds. This latter fact is particularly pronounced in instances with a small number of possible outcomes, which are often encountered in practice. Indeed, elections for political leaders often exhibit a small number of alternatives, or selecting a candidate for an academic position usually boils down to a choice between a few shortlisted options. Especially for those cases, we would like to be able to compute the *exact* distortion bound of a given voting rule, captured in the following question:

2. Given a voting rule, how can we find its exact *worst-case distortion?*

1.1 Our Contributions

In this paper, we provide algorithms that answer Questions 1 and 2 above, which are based on continuous optimization techniques. In particular, we provide the following contributions for the distortion of voting rules in the *unit-sum normalized* utilitarian setting.

For Question 1

We design a polynomial-time algorithm which, for any vector of agents' (ordinal) preferences, i.e., for any *preference profile* with n agents and m alternatives, constructs a linear program with $O(nm)$ variables and $O(nm)$ constraints which computes the best possible distortion, and the voting rule that achieves this distortion. Our algorithm is conceptually and computationally simpler than the best known solution due to [13], which also employs a linear program with $O(nm^2)$ variables and constraints. For the first part of Question 1 in isolation, i.e., merely identifying the best possible distortion of any voting rule given a preference profile (but not the rule itself), we provide a much faster algorithm, which runs in time $O(nm \log(nm))$. We complement our theoretical results with experiments which demonstrate the far superior running time of our algorithms on both real and synthetic data.

For Question 2

We formulate the problem of finding the *exact* distortion of a given RPSR as a quadratic program. This quadratic program is non-convex, and thus our algorithm does not run in polynomial time. Still, it is fast enough to comfortably compute the exact distortion of a given RPSR for up to 5 candidates, which captures a plethora of social choice scenarios of interest. Our algorithm is the first to solve the exact distortion computation problem, but in the next section, we do highlight its computational advantages against previous techniques that could conceivably be applied to our domain. Using this quadratic program, we compute the exact distortion of several well-known RPSRs, as well as the best possible distortion achieved by any rule in this class. For the latter part, we employ an iterative technique that searches through the (continuous) domain of RSPRs, aided by distortion lower bounds for rules in this class obtained as solutions to linear programs, making use of classic characterization results from the literature [9, 20].

All of our results are enabled by a novel characterization of the structure of the worst-case instances for any (ordinal) voting rule, by means of what we refer to as *dichotomous utilities*, which could be of independent interest.

1.2 Related Work and Discussion

We discuss how our results for Questions 1 and 2 improve upon the state of the art results in the literature. For more works on the general topic of distortion in computational social choice theory, we defer the reader to the survey of Anshelevich et al. [7].

For finding the preference profile-optimal rule and its distortion in Question 1, Boutilier et al. [13] provided a polynomial-time algorithm based on linear-programming. At a high level, their linear program finds the smallest possible distortion value subject to a set of constructed inequalities that ensure that this distortion is feasible for some voting rule. In turn, these inequalities are obtained from the dual of a different linear program that checks the feasibility of a given distortion; the use of duality here is crucial, to avoid their resulting optimization program having quadratic constraints. Boutilier et al. [13], as also evidenced by the phrasing of Theorem 3.4. in their work, primarily presented their algorithm as a proof of polynomial-time solvability of the problem. That said, merely establishing polynomial-time solvability can be achieved by a simpler linear program, one with infinitely many constraints, coupled with an appropriate separation oracle so that it can be solved by the ellipsoid method; see Section 3 for more details. Still, even the algorithm of [13] turns out to not be fast enough for several applications, e.g., for computing the preference profile-optimal rule on a large set of inputs, to be used as a performance benchmark against other voting rules in experiments. Our algorithms are faster in theory and in practice, making them much more appealing for such applications. The main technical contribution that allows us to devise these faster algorithms is our novel structural restriction of dichotomous utilities as the worst-case preference profiles.

Finally, we remark that one can interpret our algorithm as a general social choice rule with a best-possible distortion of $O(\sqrt{m})$. Compared to the rule of Ebadian et al. [16], ours seems conceptually simpler, as it is based on linear programs. In contrast, the rule in [16] uses the concept of *stable lotteries*, which are computed via applying a multiplicative weight updates algorithm to approach the value of a certain zero-sum game, see [23] for more details.

Moving on to the exact distortion of RPSRs in Question 2, the literature had not really provided any methods for this task prior to our work. The most related approach is due to Filos-Ratsikas and Miltersen [18], who studied (among other voting rules), the exact distortion of RPSRs for three candidates, but crucially, for a different normalization of the utilities called *unit-range*.[1] Given our structural characterization of the worst-case utility profiles, their approach is in principle applicable for unit-sum as well, but it results in non-convex quadratic programs with at least 18 variables and constraints, even for $m = 3$. Our program, which is different from the one of [18], has much fewer constraints and can thus easily handle instances with $m = 5$ alternatives; we provide more details about the comparison with [18] in Appendix F.

To find the best RPSR and its distortion, we repeatedly apply our quadratic program on a set of candidate RPSRs. To guide our search, we follow another idea of [18], and obtain those candidates as solutions to a sequence of zero-sum games, which can easily be solved via linear programming. The zero-sum game formulation is enabled by well-known characterizations of RPSRs due to Gibbard [20] and Barbera [9] (see also [18]), and can also be seen as an application of Yao's minimax principle [29].

2 Preliminaries

Let $[t] := \{1, 2, \dots, t\}$ for $t \in \mathbb{N}$. For a set S, let $\Delta(S)$ be the set of probability distributions over S.

Utilitarian voting.

Let N be a set of n agents and A be a set of m alternatives. We assume that each agent $i \in N$ has a utility function $u_i : A \to \mathbb{R}_{\geqslant 0}$ over the alternatives. Following the literature, we adopt the unit-sum assumption: $\sum_{a \in A} u_i(a) = 1$ for each $i \in N$ [8]. These utility function collectively form the utility profile $\vec{u}$. With slight abuse of notation, we use $u_i(p) := \mathbb{E}_{a \sim p}\, u_i(a)$ to denote the (expected) utility of agent i under distribution $p \in \Delta(A)$. Given a utility profile $\vec{u}$, the social welfare of an alternative $a \in A$ is $\mathsf{sw}(a, \vec{u}) := \sum_{i \in N} u_i(a)$ and that of a distribution $p \in \Delta(A)$ is $\mathsf{sw}(p, \vec{u}) := \sum_{i \in N} u_i(p)$. We say that an alternative a^* is optimal if it has the maximum social welfare, i.e. $a^*(\vec{u}) \in \arg\max_{a \in A} \mathsf{sw}(a, \vec{u})$. We might drop $\vec{u}$ when it is clear from the context.

Elicitation and aggregation.

Based on their underlying utilities, agents submit their votes in form of a ranking over the alternatives. Let $\sigma_i : [m] \to A$ denote the ranking of agent $i \in N$ over the alternatives. We use $a \succ_i b$ to show that agent i prefers alternative a to alternative b, and $\mathsf{rank}_i(a) := \sigma_i^{-1}(a)$ to show the rank of alternative a in agent i's ranking; thus $a \succ_i b \Leftrightarrow \mathsf{rank}_i(a) < \mathsf{rank}_i(b)$. We say that σ_i is consistent with utility function u_i if, for any pair of alternatives $a, b \in A$, $a \succ_i b$ implies $u_i(a) \geqslant u_i(b)$; let $\mathcal{C}(\sigma_i)$ denote the set of utility functions consistent with σ_i. These rankings collectively form a preference profile $\vec{\sigma}$. We say that utility profile $\vec{u}$ is consistent with $\vec{\sigma}$ if $u_i \in \mathcal{C}(\sigma_i)$ for each agent $i \in N$; let $\mathcal{C}(\vec{\sigma})$ denote the set of utility profiles consistent with $\vec{\sigma}$.

[1]In unit-range, the values of each agent for the alternatives lie in $[0, 1]$ with the maximum value being 1 and the minimum value being 0. In unit-sum, the values of each agent for the alternatives sum to 1. The unit-sum normalization is the most widely-used in the related literature, see [7, 8].

Voting rules.

A (randomized) voting rule f takes a preference profile $\vec{\sigma}$ as input and outputs a distribution $f(\vec{\sigma}) \in \Delta(A)$ over the alternatives. If $f(\vec{\sigma})$ always has singleton support, we say f is *deterministic* and, with slight abuse of notation, use $f(\vec{\sigma})$ to denote the alternative in the support.

Distortion.

The distortion of a distribution $p \in \Delta(A)$ on a utility profile $\vec{u}$ is its social welfare approximation, $\mathsf{dist}(p, \vec{u}) := \frac{\max_{a \in A} \mathsf{sw}(a,\vec{\sigma})}{\mathsf{sw}(p,\vec{\sigma})}$. Its distortion on a preference profile $\vec{\sigma}$ is its worst-case distortion on any utility profile $\vec{u} \in \mathcal{C}(\vec{\sigma})$: $\mathsf{dist}(p, \vec{\sigma}) := \sup_{\vec{u} \in \mathcal{C}(\vec{\sigma})} \mathsf{dist}(p, \vec{u})$; note that this supremum is attained at some $\vec{u} \in \mathcal{C}(\vec{\sigma})$ due to a continuous function being optimized over a compact domain and, thus, can be replaced by a maximum. The distortion of a voting rule f is the worst-case distortion of its output over all preference profiles: $\mathsf{dist}(f) = \sup_{\vec{\sigma}} \mathsf{dist}(f(\vec{\sigma}), \vec{\sigma})$, where the worst case is taken over all preference profiles with m alternatives (and any number of agents). The *instance-optimal rule* f^* is the rule that, on each preference profile $\vec{\sigma}$, outputs the distribution with the smallest distortion: $f^*(\vec{\sigma}) \in \arg\min_{p \in \Delta(A)} \mathsf{dist}(p, \vec{\sigma})$.

3 Computing the Optimal Distribution

In this section, we focus on computing the instance-optimal rule f^*, i.e., computing the optimal distribution in $\arg\min_{p \in \Delta(A)} \mathsf{dist}(p, \vec{\sigma})$ given a preference profile $\vec{\sigma}$. Boutilier et al. [13] prove that this can be accomplished in polynomial time. Specifically, they first write a *nonlinear program* for the problem, which involves multiplying the probability of selecting an alternative with the utility of an agent for the alternative, both variables of the program. Then, by considering its dual program and combining it with the primal, they design a complicated *linear program* (LP) whose optimal solution yields the optimal distribution and its corresponding distortion. Their full LP is provided in Appendix A. This LP has $O(nm^2)$ variables and $O(nm^2)$ constraints, establishing that the optimal distribution (and its corresponding distortion) can be computed in polynomial time.

While Boutilier et al. [13] do not point it out, polynomial-time computability is actually much easier to establish. Given a preference profile $\vec{\sigma}$, one can write the following straightforward LP.

$$\begin{aligned}
\max \quad & \beta \\
\text{s.t.} \quad & \sum_{a \in A} p_a \cdot \mathsf{sw}(a, \vec{u}) \geqslant \beta \cdot \max_{a \in A} \mathsf{sw}(a, \vec{u}), && \forall \vec{u} \in \mathcal{C}(\vec{\sigma}) && (1) \\
& \sum_{a \in A} p_a = 1 && && (2) \\
& p_a \geqslant 0, && \forall a \in A. && (3)
\end{aligned}$$

Variable p_a denotes the probability of selecting alternative a and constraint (1) requires that $\mathsf{dist}(p, \vec{u}) \leqslant 1/\beta$ under every utility profile $\vec{u} \in \mathcal{C}(\vec{\sigma})$. Thus, $1/\beta$ becomes an upper bound on the distortion of p and maximizing β yields the optimal distribution p along with its distortion $1/\beta$. While this LP has *uncountably many* constraints, it admits a straightforward polynomial-time separation oracle. Given values of β and $(p_a)_{a \in A}$, checking whether constraints (2) or (3) are violated is trivial, and finding a violated constraint in (1) (if one exists) amounts to solving the following m linear programs, one for each $a^* \in A$:

$$\begin{aligned}
\min \quad & \sum_{a\in A} p_a \cdot \left(\sum_{i\in N} u_{i,a}\right) - \beta \cdot \left(\sum_{i\in N} u_{i,a^*}\right) & \\
\text{s.t.} \quad & u_{i,a} \geqslant u_{i,b}, & \forall i \in N, a, b \in A : a \succ_i b \\
& \sum_{a\in A} u_{i,a} = 1, & \forall i \in N \\
& u_{i,a} \geqslant 0, & \forall i \in N, a \in A.
\end{aligned}$$

If the optimal value for any of these LPs is less than 0, the corresponding utility profile identifies a violated constraint in the original LP. Thus, the original LP can be solved in polynomial time using the ellipsoid method.

The main contribution of Boutilier et al. [13], therefore, is not polynomial-time computability, but rather that they provide a single LP with $O(nm^2)$ variables and $O(nm^2)$ constraints to be solved, which is much faster than solving the above LP with uncountably many constraints using a separation oracle that solves m LPs each with $O(nm)$ variables and constraints in each iteration.

Our main contribution in this section is to devise a much simpler LP with only $O(nm)$ variables and $O(nm)$ constraints, and without using sophisticated techniques such as LP duality. The bedrock of our improvement is a novel structural characterization of the worst-case utility profile for a distribution p on a preference profile $\vec{\sigma}$, which may be of independent interest. We also use this characterization in Section 4.

3.1 Worst-Case Utility Profiles

We prove that the distortion of any distribution p on any preference profile $\vec{\sigma}$ is attained at some utility profile $\vec{u} \in \mathcal{C}(\vec{\sigma})$ with the following dichotomous structure.

Definition 1 (Dichotomous Utilities)**.** The (unit-sum) *dichotomous utility function* with respect to ranking σ over A and $r \in [m]$ is

$$\mathbb{1}_{\sigma,r}(a) = \begin{cases} 1/r & \text{if } \sigma^{-1}(a) \leqslant r, \\ 0 & \text{o.w.}; \end{cases}$$

that is, the agent is indifferent among her top r alternatives and has zero utility for the remaining alternatives. We say that a utility profile $\vec{u}$ is dichotomous if the utility function u_i of each agent $i \in N$ is dichotomous with respect to σ_i and some $r_i \in [m]$.

We are ready to prove our structural insight.

Theorem 1. *For any distribution $p \in \Delta(A)$ and preference profile $\vec{\sigma}$, there exists a dichotomous utility profile $\vec{u}^* \in \mathcal{C}(\vec{\sigma})$ where p attains its worst distortion (i.e., $\mathsf{dist}(p, \vec{u}^*) = \mathsf{dist}(p, \vec{\sigma})$).*

Proof. Let $d = \mathsf{dist}(p, \vec{\sigma})$. Then,

$$\begin{aligned}
\max_{\vec{u}\in\mathcal{C}(\vec{\sigma})} \frac{\max_{a^*\in A} \mathsf{sw}(a^*, \vec{u})}{\mathsf{sw}(p, \vec{u})} = d & \qquad \iff \\
\max_{\vec{u}\in\mathcal{C}(\vec{\sigma})} \max_{a^*\in A} \mathsf{sw}(a^*, \vec{u}) - d \cdot \mathsf{sw}(p, \vec{u}) = 0 & \qquad \iff
\end{aligned}$$

$$\max_{a^*\in A} \max_{\vec{u}\in\mathcal{C}(\vec{\sigma})} \sum\nolimits_{i\in N} (u_i(a^*) - d\cdot u_i(p)) = 0 \qquad \iff$$
$$\max_{a^*\in A} \sum\nolimits_{i\in N} \max_{u_i\in\mathcal{C}(\sigma_i)} (u_i(a^*) - d\cdot u_i(p)) = 0, \tag{4}$$

where in the last transition, the maximum can be taken over each agent separately due to the expression being linear over the agents.

It remains to show that, for a fixed alternative a^* and agent i, $u_i(a^*) - d\cdot u_i(p)$ is maximized at some dichotomous utility function $u_i^* \in \mathcal{C}(\sigma_i)$. Note that dichotomous utility functions with respect to σ_i are linearly independent and span the space $\mathcal{C}(\sigma_i)$ of unit-sum utility functions consistent with σ_i. Hence,

$$\begin{aligned}&\max_{u_i\in\mathcal{C}(\sigma_i)} u_i(a^*) - d\cdot u_i(p)\\ &= \max_{\vec{\alpha}\in\Delta^m} \left(\sum\nolimits_{r\in[m]} \alpha_r \cdot \mathbb{1}_{\sigma_i,r}(a^*) - d\cdot \sum\nolimits_{r\in[m]} \alpha_r\cdot \mathbb{1}_{\sigma_i,r}(p)\right)\\ &= \max_{\vec{\alpha}\in\Delta^m} \sum\nolimits_{r\in[m]} \alpha_r \cdot (\mathbb{1}_{\sigma_i,r}(a^*) - d\cdot \mathbb{1}_{\sigma_i,r}(p)),\end{aligned}$$

where $\Delta^m = \{\vec{\alpha}\in[0,1]^m : \sum_{r\in[m]}\alpha_r = 1\}$ is the $(m-1)$-simplex. Due to the final expression being linear in $\vec{\alpha}$, the maximum is attained at α^* where $\alpha_r^* = 1$ (i.e., $u_i^* = \mathbb{1}_{\sigma_i,r}$) for some $r\in[m]$. □

Given Theorem 1, we can replace the maximum over all $u_i\in\mathcal{C}(\sigma_i)$ in Equation (4) with a maximum over dichotomous utility functions with respect to σ_i to obtain:

$$\mathsf{dist}(p,\vec{\sigma}) = d \qquad \iff$$
$$\max_{a^*} \sum_{i\in A} \max_{r\in[m]} \frac{1}{r}\left(\mathbb{I}\left[\mathsf{rank}_i(a^*) \leqslant r\right] - d\cdot \sum_{\ell=1}^{r} p_{\sigma_i(\ell)}\right) = 0. \tag{5}$$

We revisit Equation (5) later to derive our results.

3.2 A Faster Polynomial-Time Algorithm

Building on the novel characterization of worst-case utility profiles from Theorem 1, we design a simpler linear program for computing the instance-optimal rule. This is the main result of this section.

Theorem 2. *Given a preference profile $\vec{\sigma}$, there exists a linear program with $O(nm)$ variables, $O(nm)$ constraints, and $O(nm)$ size for computing the optimal distribution $p^* \in \arg\min_{p\in\Delta(A)} \mathsf{dist}(p,\vec{\sigma})$ and its distortion $\mathsf{dist}(p^*,\vec{\sigma})$.*

Proof. We derive the final linear program gradually via a number of insights and transformations. Let $\vec{\sigma}$ be the given preference profile. First, note that the distortion $\mathsf{dist}(p,\vec{\sigma})$ of a distribution p is the minimum value d for which Equation (5) holds. This yields the following optimization problem which computes the optimal distribution p and its corresponding distortion d.

$$\begin{aligned}\min\quad & d\\ \text{s.t.}\quad & \delta_{i,a} \geqslant \frac{1}{r}\left(\mathbb{I}[\mathsf{rank}_i(a)\leqslant r] - d\cdot \sum\nolimits_{\ell=1}^{r} p_{\sigma_i(\ell)}\right)\end{aligned}$$

$$\forall i \in N, a \in A, r \in [m]$$

$$\begin{aligned} &\textstyle\sum_{i\in N} \delta_{i,a} \leqslant 0 && \forall a \in A \\ &\textstyle\sum_{a\in A} p_a = 1 \\ &p_a \geqslant 0 && \forall a \in A. \end{aligned}$$

Here, $\delta_{i,a}$ upper bounds the expression from Equation (5) corresponding to a fixed $a^* = a$ and $i \in N$, so the constraint $\sum_{i\in N} \delta_{i,a} \leqslant 0$ for all $a \in A$ implements precisely Equation (5).

Linearization. However, this is not a linear program due to the multiplication of d with p_a-s in the first constraint. To linearize it, we introduce a new variable $\widehat{p}_a = d \cdot p_a$ for each $a \in A$. Since $\sum_{a\in A} \widehat{p}_a = d$, the above program is equivalent to

$$\begin{aligned} \min \quad & \textstyle\sum_{a\in A} \widehat{p}_a \\ \text{s.t.} \quad & \delta_{i,a} \geqslant \frac{1}{r}\left(\mathbb{I}[\mathsf{rank}_i(a) \leqslant r] - \textstyle\sum_{\ell=1}^{r} \widehat{p}_{\sigma_i(\ell)}\right) \\ & \qquad\qquad\qquad \forall i \in N, a \in A, r \in [m] \\ & \textstyle\sum_{i=1}^{n} \delta_{i,a} \leqslant 0 \qquad\qquad \forall a \in [m] \\ & \widehat{p}_a \geqslant 0 \qquad\qquad\qquad \forall a \in [m] \end{aligned}$$

This is now a linear program, whose optimal solution $\widehat{p}$ yields both the optimal distortion $\sum_{a\in A} \widehat{p}_a$ and the optimal distribution given by $p_a = \widehat{p}_a / \sum_{b\in A} \widehat{p}_b$ for all $a \in A$. While it already has $O(nm)$ variables, an improvement over $O(nm^2)$ variables of Boutilier et al. [13], it still has $O(nm^2)$ constraints, the same as them.

Optimizing the number of constraints. To reduce the number of constraints to $O(nm)$, the key observation is the following reconstruction of the constraints that bound $\delta_{i,\sigma_i(r)}$. For $i \in N$ and $r \in [m]$, let $s_{i,r} := \sum_{\ell=1}^{r} \widehat{p}_{\sigma_i(\ell)}$. Then, the first constraint in the above program can be written as

$$\delta_{i,\sigma_i(r)} \geqslant \max\left\{ \max_{\ell\in[r-1]} -\frac{1}{\ell} \cdot s_{i,\ell},\ \max_{\ell\in[r,m]} \frac{1}{\ell} \cdot (1 - s_{i,\ell}) \right\}. \tag{6}$$

Define $\alpha_{i,r} = \max_{\ell\in[r]} -\frac{1}{\ell} \cdot s_{i,\ell}$ and $\beta_{i,r} = \max_{\ell\in[r,m]} \frac{1}{\ell} \cdot (1 - s_{i,\ell})$. Then, we can bound $\delta_{i,\sigma_i(r)}$ using only two constraints:

$$\delta_{i,\sigma_i(r)} \geqslant \alpha_{i,r-1}, \quad \text{and} \quad \delta_{i,\sigma_i(r)} \geqslant \beta_{i,r}.$$

The α's and β's can be set using $O(nm)$ constraints as in the final linear program below.

Linear program $\mathcal{P}$ consists of $O(nm)$ and $O(nm)$ constraints. The third constraint type involves n variables and there are m of such constraints. All other constraints involve at most 3 variables. Thus, the size (number of non-zero coefficients) of $\mathcal{P}$ is also $O(nm)$. □

Linear Program $\mathcal{P}$

$$\begin{aligned}
\min \quad & \sum_{a\in[m]} \widehat{p}_a & \\
\text{s.t.} \quad & \delta_{i,\sigma_i(r)} \geqslant \alpha_{i,r-1} & \forall i \in N, r \in [2, m-1]\\
& \delta_{i,\sigma_i(r)} \geqslant \beta_{i,r} & \forall i \in N, r \in [m]\\
& \textstyle\sum_{i=1}^{n} \delta_{i,a} \leqslant 0 & \forall a \in [m]
\end{aligned}$$

Partial sums:

$$\begin{aligned}
& s_{i,1} = \widehat{p}_{\sigma_i(1)} & \forall i \in N\\
& s_{i,r} = s_{i,r-1} + \widehat{p}_{\sigma_i(r)} & \forall i \in N, r \in [2, m]
\end{aligned}$$

Top partial maximums:

$$\begin{aligned}
& \alpha_{i,r} \geqslant \alpha_{i,r-1} & \forall i \in N, r \in [2, m-1]\\
& \alpha_{i,r} \geqslant \frac{1}{r} \cdot (-s_{i,r}) & \forall i \in N, r \in [m-1]
\end{aligned}$$

Bottom partial maximums:

$$\begin{aligned}
& \beta_{i,r} \geqslant \beta_{i,r+1} & \forall i \in N, r \in [m-1]\\
& \beta_{i,r} \geqslant \frac{1}{r} (1 - s_{i,r}) & \forall i \in N, r \in [m]
\end{aligned}$$

Variable ranges:

$$\begin{aligned}
& \widehat{p}_a \geqslant 0 & \forall a \in [m]\\
& \delta_{i,\sigma_i(r)}, \alpha_{i,r}, \beta_{i,r} \in \mathbb{R} & \forall i \in N, r \in [m]
\end{aligned}$$

3.3 Distortion of a Rule on a Preference Profile

While the LP above computes the instance-optimal rule f^* (i.e., computes the optimal distribution p^* on a given preference profile $\vec{\sigma}$), in practice one may wish to implement a different voting rule f due to, for example, its normative properties or the status quo. In such cases, one may wish to know the distortion of the distribution $f(\vec{\sigma})$ it returns on $\vec{\sigma}$, which may not be the optimal distribution p^*.

To the best of our knowledge, the only approach to solve this problem prior to our work was to use the linear program of Boutilier et al. [13] and fixing the selection probabilities to match $f(\vec{\sigma})$ (instead of letting them be variables). Adapting this approach to our optimized linear program $\mathcal{P}$ already provides a faster algorithm. However, known algorithms to solve a linear program with $\Theta(nm)$ variables and $\Theta(nm)$ size still take $\Omega((nm)^3)$ time, which can be very time consuming. Using insights from Theorem 1, we develop an $O(nm \log(nm))$ time algorithm for this problem which avoids solving linear programs.

Theorem 3. *There is an algorithm to compute the distortion* $\mathsf{dist}(p, \vec{\sigma})$ *of distribution* p *on preference profile* $\vec{\sigma}$ *in* $O(nm \log(nm))$ *time.*

The first step is to design a fast subroutine for checking whether $\mathsf{dist}(p, \vec{\sigma}) \leqslant d$ for a given threshold d.

Lemma 1. *There is an algorithm that given a preference profile* $\vec{\sigma}$*, distribution* p*, and a real number* d *checks if distortion of* p *w.r.t* $\vec{\sigma}$ *is at most* d *in linear time* $O(nm)$*. If not, it finds a witness utility profile for which distortion of* p *is more than* d*.*

We defer the proof of this lemma result to Appendix B, but the key idea is to quickly compute Equation (5) as both p and d are given. For each agent i and alternative a, we compute the index $h_{i,a}(d)$ such that the dichotomous utility function corresponding to σ_i and $h_{i,a}(d)$ maximizes the inner expression in Equation (5):

$$h_{i,a}(d) = \arg\max_{r \in [m]} \left\{ \frac{1}{r} \cdot \left(\mathbb{I}\left[\mathsf{rank}_i(a) \leqslant r\right] - d \cdot \sum\nolimits_{\ell=1}^{r} p_{\sigma_i(\ell)} \right) \right\}. \tag{7}$$

This allows evaluating the left hand side of Equation (5): if it is non-positive, the distortion is indeed at most d, and if it is positive, the utility profile where each agent i has the aforementioned dichotomous utility function forms the sought witness.

Lemma 1 immediately implies that one can perform a binary search for the desired distortion $d^* := \mathsf{dist}(p, \vec{\sigma}) \in [1, \infty]$ and get ε-close in time $nm \log(d^*/\varepsilon)$. However, we can prove that, instead of searching for d^* in a continuous range, we can selectively focus on at most nm potential distortion values. This is because $h_{i,a}(d)$ defined in Equation (7) changes at only a limited number of "pivotal" values of d. However, computing these pivotal values across all $i \in N$ and $a \in A$ in only $O(nm)$ time requires additional ideas based on convex hulls (specifically, Graham's Scan [1]). The proof of the next result is relegated to Appendix B.

Lemma 2. *There exists a set* $D = \{d_0 = 0, d_1, d_2, \ldots, d_{|D|-1} = \infty\}$ *of* $O(nm)$ *values* $d_0 \leqslant d_1 \leqslant \ldots \leqslant d_{|D|-1}$*, computable in* $O(nm)$ *time, such that for all* $\ell \in [|D| - 1]$ *and all* $d' \in [d_{\ell-1}, d_\ell)$*, it holds that* $h_{i,a}(d') = h_{i,a}(d_{\ell-1})$ *for all* $i \in N$ *and* $a \in A$*.*

Using Lemmas 1 and 2, we can prove Theorem 3.

Proof of Theorem 3. By Lemma 2, we can find a set D of $O(nm)$ values such that for any two consecutive values $d_{\ell-1}, d_\ell \in D$, the utility profile maximizing Equation (5) stays unchanged across all $d \in [d_{\ell-1}, d_\ell]$. By performing a binary search on this set of values, we can find the maximum $d_{\ell-1}$ such that $\mathsf{dist}(p, \vec{\sigma}) \geqslant d_{\ell-1}$ by invoking Lemma 1. Then, we would know that $d^* \in [d_{\ell-1}, d_\ell)$. Since there is a unique worst-case utility profile $\vec{u}$ in this range, the actual distortion can simply be computed as $\mathsf{dist}(p, \vec{u})$. The $O(nm \log(nm))$ running time is due to $O(\log(nm))$ calls to the $O(nm)$ time algorithm from Lemma 1 as part of the binary search; the rest of the steps amount to $O(nm)$ time. □

4 Distortion of Randomized Positional Scoring Rules

In the previous section, we were concerned with computing the distortion of a given distribution or the optimal distribution on a *given* preference profile. However, the (overall) distortion of a rule requires computing its worst-case distortion across all preference profiles — a significantly more complex task, for which prior work provides no algorithms to the best of our knowledge. Our main contribution in this section is to devise such an algorithm for the following well-studied family of rules.

Randomized Positional Scoring Rules.

Given a scoring vector $\vec{s} = (s_1, s_2, \dots, s_m)$, the *positional scoring rule* (PSR) assigns a score of s_j to an alternative each time it appears at the j-th position in the ranking of any agent (for each j), and (deterministically) selects an alternative with the highest total score. A *randomized positional scoring rule* (RPSR) $f_{\vec{s}}$ assigns scores to alternatives in the same way, but selects each alternative with probability proportional to its score. On preference profile $\vec{\sigma}$, this selects each alternative $a \in A$ with probability $\mathsf{P}_{\vec{s}}(a, \vec{\sigma}) := \Pr[f_{\vec{s}}(\vec{\sigma}) = a] = \frac{1}{n}\sum_{i \in N} s_{\mathsf{rank}_i(a)}$.

These rules, also called point-voting schemes [9], are known due to their strategyproofness [20]. Ebadian et al. [17] provide asymptotic distortion bounds for many rules in this family.

In this section, we first devise an algorithm for computing the (exact) distortion of a given RPSR. Then, we iteratively use our algorithm, together with an appropriately constructed linear program, to find the RPSR with the smallest distortion for a given number of alternatives m. Finally, we present the bounds achieved via our techniques for up to $m = 5$ alternatives, for both the best RPSR and commonly used RPSRs from the literature.

4.1 Worst-Case Utility and Preference Profiles

Let $A = \{a_1, a_2, \dots, a_m\}$. Let us define three properties of a pair of preference profile $\vec{\sigma}$ and utility profile $\vec{u} \in \mathcal{C}(\vec{\sigma})$.

(P1) $\vec{u}$ is dichotomous.

(P2) $\mathsf{sw}(a_j, \vec{u}) \geqslant \mathsf{sw}(a_{j+1}, \vec{u})$ for all $j \in [m-1]$.

(P3) For each agent $i \in N$ and alternatives $a_j, a_{j'} \in A$ with $j < j'$, if $u_i(a_j) = u_i(a_{j'})$, then $a_{j'} \succ_i a_j$.

Property (P1) is our insight from Theorem 1 that the worst-case utility profile $\vec{u}$ is dichotomous WLOG. Property (P2) uses neutrality of RPSRs (i.e., that they do not depend on the names of the alternatives) to further restrict the search space for $\vec{u}$. Unlike in Section 3 where the preference profile $\vec{\sigma}$ was given, here we seek to identify its worst case, for which Property (P3) is our novel insight.

Let $W_m = \{(\sigma_1, u_1), \dots, (\sigma_t, u_t)\}$ denote the set of possible pairs of preference ranking σ_i and utility function $u_i \in \mathcal{C}(\sigma_i)$ that any agent can have in any pair of preference profile $\vec{\sigma}$ and utility profile $\vec{u} \in \mathcal{C}(\vec{\sigma})$ satisfying Properties (P1) to (P3); note that $t := |W_m|$ and from here on, with slight abuse of notation, we will use i to index a pair (agent type) in W_m instead of denoting an individual agent. The next lemma shows that the three properties above significantly reduce the search space.

Lemma 3. *$|W_m| \leqslant 2^m - 2$.*

Table 3 in Appendix C shows the $2^3 - 2 = 6$ possible pairs of preference ranking and utility function that we need to consider for $m = 3$ alternatives. Finally, we show that every RPSR achieves its worst distortion on a pair $(\vec{\sigma}, \vec{u})$ satisfying Properties (P1) to (P3). The proofs of both these lemmas are in Appendix C.1.

Lemma 4. *For every randomized positional scoring rule $f_{\vec{s}}$, there exists a pair $(\vec{\sigma}, \vec{u})$ of preference and utility profiles satisfying Properties (P1) to (P3) at which $f_{\vec{s}}$ attains its worst distortion.*

4.2 Computing the Distortion of an RPSR

Fix any RPSR $f_{\vec{s}}$. For $i \in [t]$, define q_i to be the fraction of agents who have preference ranking and utility function $(\sigma_i, u_i) \in W_m$. Note that the $1 \times t$ vector $q = (q_1, \ldots, q_t)$ satisfies $\sum_{i\in[t]} q_i = 1$ and can capture any worst-case instance $(\vec{\sigma}, \vec{u})$ satisfying Properties (P1) to (P3), for any number of agents n.

We want to understand the distortion of $f_{\vec{s}}$ at a given q and then write a program to optimize over q. Let $U_{m\times t}$ be a matrix where $U_{j,i} = u_i(a_j)$, and $P_{m\times t}$ be a matrix where $P_{j,i}$ is the score that candidate a_j gets from any agent with preference ranking σ_i, i.e. $P_{j,i} = s_{\mathsf{rank}_i(a_j)}$. Also, define $C_{t\times t} = U^\intercal P$. For example, when $m = 3$, matrices U and P corresponding to the set W_m from Table 3 are shown in Appendix C.

From the above, we can see that if q represents the instance $(\vec{\sigma}, \vec{u})$, then $\mathsf{sw}(a_j, \vec{u}) = n \cdot q \cdot U_j^\intercal$ and $\mathsf{P}_{\vec{s}}(a_j, \vec{\sigma}) = q \cdot P_j^\intercal$, where U_j and P_j denote the j-th row of U and P, respectively. This means

$$\mathsf{dist}(f_{\vec{s}}) = \frac{q \cdot U_1^\intercal}{\sum_{j\in[m]}(q \cdot U_j^\intercal)(q \cdot P_j^\intercal)} = \frac{q \cdot U_1^\intercal}{q\left(\sum_i U_j^\intercal \cdot P_j\right) q^\intercal} = \frac{qU_1^\intercal}{qCq^\intercal}.$$

Hence, we arrive at the following optimization program to compute the distortion of the given RPSR $f_{\vec{s}}$.

$$\begin{aligned}
\max \quad & \frac{qU_1^\intercal}{qCq^\intercal} & \\
\text{s.t.} \quad & q \cdot U_j^\intercal \leqslant q \cdot U_1^\intercal & \forall j \in [m] \\
& \textstyle\sum_{i\in[t]} q_i = 1 & \\
& q_i \geqslant 0 & \forall i \in [t].
\end{aligned}$$

To transform this optimization program to a form more amenable to solving via standard solvers, we add a variable D to capture the inverse of the distortion, i.e., $1/\mathsf{dist}(f_{\vec{s}})$. The new objective is to minimize D, and we add a quadratic constraint $D \cdot (q^\intercal U_1) \geqslant q^\intercal C q$ to ensure that D remains an upper bound on the inverse of the distortion. The resulting optimization program has a linear objective, a set of linear constraints, and a single quadratic constraint.

The (inverse of the) solution to Program $\mathcal{Q}$ yields the distortion $\mathsf{dist}(f_{\vec{s}})$ of a given RPSR $f_{\vec{s}}$ as well as the instance $(\vec{\sigma}, \vec{u})$ where the worst distortion is attained; the latter part will be useful in our algorithm in Section 4.3 for finding the best RPSR.

We remark that Program $\mathcal{Q}$ is not convex, and thus not known to be polynomial-time solvable, even to a given precision. Still, for relatively small values of m (e.g., up to $m = 5$), standard solvers are able to provide globally optimal solutions within reasonable running times as we show in Section 4.4.

Quadratic Program $\mathcal{Q}$

$$\begin{array}{lll} \text{min.} & D & \\ \text{s.t.} & D \sum_{i\in[t]} q_i \cdot U_{1,i} \geqslant \sum_{i\in[t]} \sum_{k\in[t]} q_i \cdot q_k \cdot C_{i,k} & \\ & \sum_{i\in[t]} q_i \cdot U_{j,i} \leqslant \sum_{i\in[t]} q_i \cdot U_{1,i} & \forall j \in [m] \\ & \textstyle\sum_{i\in[t]} q_i = 1 & \\ & q_i \geqslant 0 & \forall i \in [t]. \end{array}$$

4.3 Finding the Best Possible RPSR

In this section, we make use of our quadratic program $\mathcal{Q}$ to find the RPSR that achieves the minimum distortion (and that distortion itself) for a given number of alternatives m. The idea is to use program $\mathcal{Q}$ repeatedly to search over the space of all RPSRS. Since this is a vast search space and solving the non-convex program $\mathcal{Q}$ takes time, we employ the technique of Filos-Ratsikas and Miltersen [18], which uses zero-sum games (and as a result, linear programs) to aid the search and quickly converge to the optimal solution without exploring too many points in the search space.

Lower bounds via zero-sum games

Filos-Ratsikas and Miltersen [18, Theorem 2] show that any RPSR for m alternatives is a convex combination of rules F_m^k for $k \in [m]$: rule F_m^k (a) selects an agent uniformly at random and (b) select one of her k most-preferred alternatives uniformly at random. This result of [18] is in fact almost a direct corollary of a result of Barbera [9], which was in turn obtained from the characterization of Gibbard [20] of truthful rules.

By applying Yao's minimax principle [29], we can effectively transform the design of the optimal RPSR into a (series of) zero-sum game(s) $\mathcal{G}$. In a game $\mathcal{G}$, the pure strategies of the maximizer are the rules F_m^k for $k \in [m]$ and the pure strategies of the minimizer are a set of instances $\mathcal{I}$ (the construction of this set is explained in the next paragraph), where an instance is a pair of preference and utility profiles $(\vec{\sigma}, \vec{u})$ such that $\vec{u} \in \mathcal{C}(\vec{\sigma})$. When the maximizer chooses rule F_m^j and the minimizer chooses instance $(\vec{\sigma}, \vec{u}) \in \mathcal{I}$, the reward in the corresponding cell of the game is the inverse of the corresponding distortion, i.e., $1/\mathsf{dist}(F_m^j(\vec{\sigma}), \vec{u})$. From the characterization of Filos-Ratsikas and Miltersen [18], the set of mixed strategies of the maximizer is precisely the set of RPSRs. By computing an optimal mixed strategy for the maximizer, we obtain an RPSR with the smallest distortion in the worst case over all instances in $\mathcal{I}$ (and this distortion value is the inverse of the value of game $\mathcal{G}$). Game $\mathcal{G}$ can be solved by a standard formulation of zero-sum games as linear programs.

An iterative algorithm

Throughout our algorithm, we maintain (a) the current distortion upper bound d^u, (b) the current distortion lower bound d^ℓ, (c) a set of "bad" instances $\mathcal{I}$, and (d) a candidate RPSR f_c. Initially $d^u = \infty$, $d^\ell = 0$, $\mathcal{I} = \emptyset$, and f_c is any RPSR.

In each iteration, we run the quadratic program $\mathcal{Q}$ on the rule f_c, which returns both $\mathsf{dist}(f_c)$ and the worst-case instance $(\vec{\sigma}, \vec{u})$ at which this distortion is attained. We update $d^u \leftarrow \min(d^u, \mathsf{dist}(f_c))$ and add $(\vec{\sigma}, \vec{u})$ to $\mathcal{I}$. Then, we construct the matrix game $\mathcal{G}$ described above with the pure strategies of the maximizer being the rules F_m^k and the pure strategies of the minimizer being the instances in $\mathcal{I}$. We update d^ℓ to be the inverse of the value of this game $\mathcal{G}$ and f_c to be the optimal mixed strategy of the maximizer in $\mathcal{G}$. We repeat this process until a stopping criterion has been met, which is that $d^u - d^\ell \leqslant \varepsilon$ for a sufficiently small ε; the choice of ε in our experiments is described in the next section. During this process, we keep track of the RPSR that achieves the distortion of d^u, and at termination, output that as our estimate of the best RPSR. Note that d^u is the exact distortion of this rule, and thus, a valid upper bound on the distortion of the best RPSR. Similarly, d^ℓ is also a valid lower bound on the distortion of any (and thus the best) RPSR.

4.4 Bounds for a Small Number of Alternatives

By deploying the techniques presented in the previous two subsections, we are able to compute the exact distortion of several well-known RPSRs and of the best RPSRs for $m \in \{2, 3, 4, 5\}$ alternatives. In more detail, we employ the quadratic program $\mathcal{Q}$ that we develop in Section 4.2 to compute the exact distortion of the following well-known RPSRs:

- *Randomized Plurality*, with scoring vector $\vec{s} = (1, 0, \dots 0)$,
- *Randomized Borda*, with scoring vector $\vec{s} = (m-1, \dots, 1, 0)$,
- *Randomized k-Approval*, with scoring vector $\vec{s} = (1, \dots, 1, 0, \dots, 0)$, with k ones, for $k = 2$ and $k = 3$,
- *Randomized Veto*, with scoring vector $\vec{s} = (1, 1, \dots 1, 0)$,
- *Randomized Harmonic*, with scoring vector $\vec{s} = (1, 1/2, \dots, 1/m)$,
- the *"Golden Rule"* [13] with scoring vector $\vec{s} = (1, 1/2, \dots, 1/m) + (1/m, \dots, 1/m)$. The Golden Rule has the smallest *asymptotic* distortion among all RPSRs [11].

Using the iterative algorithm described in Section 4.3, with stopping criterion $d^u - d^\ell \leqslant 0.005$, we also obtain (essentially) tight bounds on the distortion of the best RPSR for m alternatives. To solve the quadratic program $\mathcal{Q}$ we use the Gurobi Optimization Solver. We remark that the main computational bottlneck of this technique is solving the quadratic program; the aided search of Section 4.3 terminated in at most 9 iterations in all cases.

We complement these results with bounds on the best (deterministic) PSRs to demonstrate the advantages of randomization. To calculate these bounds, we first show that the (exact) distortion of a given PSR can be computed by a linear program, which we present in Appendix D. Here, we cannot use the zero-sum-game-aided search from Section 4.2, so we perform a simple grid search on the space of PSRs to find the best PSR; this is still fast enough because we are now solving linear (rather than quadratic) programs to compute the distortion of PSRs; see Appendix D for details.

To assess how close the best RPSR is to the best rule overall, we also present lower bounds on the distortion of *any* randomized voting rule. To compute these, we employ a gradient-descent style search over the space of preference profiles, employing our algorithm in Section 3.3 to find the best distortion on each preference profile (see Algorithm 2 in Appendix B). More details are

↓ Rule, m →	2	3	4	5
Best Randomized LB	1.500	1.835	2.092	2.303
Best RPSR LB	1.500	1.870	2.188	2.473
Best RPSR UB	1.500	1.870	2.188	2.474
Harmonic Rule	1.556	1.987	2.354	2.682
Randomized 2-Approval	2.000	2.414	2.743	3.078
Randomized Borda	1.522	2.000	2.556	3.148
Golden Rule	1.714	2.276	2.751	3.173
Randomized Plurality	1.522	2.155	3.000	4.000
Randomized 3-Approval	-	3.000	3.621	4.098
Randomized Veto	1.522	2.414	3.621	4.828
Best PSR UB	3	5.5	8.333	11.418

Table 1: The exact distortion of well-known RPSRs, and distortion bounds for the best RPSR, the best PSR, and the best randomized rule.

presented in Appendix E. While the algorithm does not return the global optimum (which would be the precise distortion of the instance-optimal rule), the optimal distortion it finds on any preference profile serves as a valid lower bound on the distortion of any rule.

Our results are summarized in Table 1. The benefits of using randomization are evident, as the distortion of deterministic PSRs are notably worse than those of even common RPSRs. On the other side of the spectrum, the best RPSRs are quite close (exactly matching for $m = 2$) to the best randomized rules in general. We provide the best RSPRs achieving the displayed bounds in Table 4 in Appendix D. Among common RPSRs, most are near-optimal for $m = 2$, but the differences between them are more pronounced for $m \geqslant 3$. Interestingly, with the exception of $m = 2$, the Randomized Harmonic rule achieves the lowest distortion among all common RPSRs, even though the Golden Rule has a better asymptotic distortion (and the best asymptotic distortion among all RPSRs) [17]. This showcases that the hidden constants in the asymptotic bounds can make a difference for small values of m.

5 Experiments

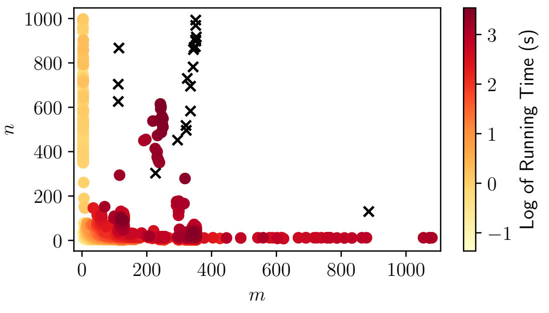


Figure 1: Performance of BCHL+ on Preflib data.

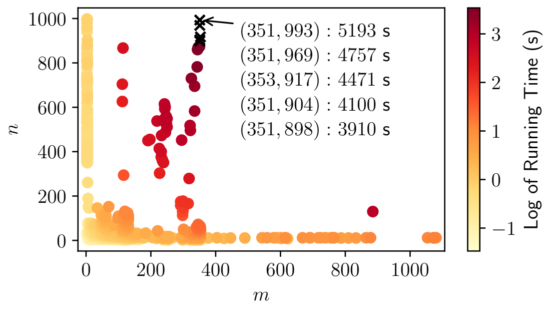
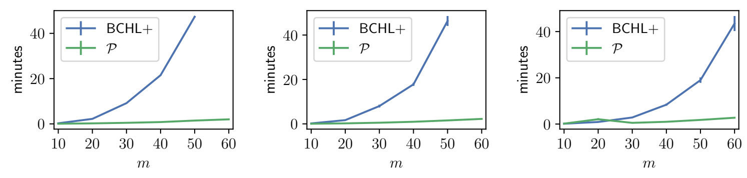


Figure 2: Performance of $\mathcal{P}$ on Preflib data.



(a) Mallows w. $\phi = 0.5$ (b) Plackett-Luce (P.L.) (c) Single-Peaked (S.P.)

Figure 3: Running time of $\mathcal{P}$ and BCHL+ on synthetic data ($n = 1000$).

m	ϕ=0.1	ϕ=0.5	ϕ=0.8	S.P	P.L.
10	3.74	2.07	3.1	0.77	3.29
20	9.71	5.13	8.08	0.43	7.28
30	19.6	9.09	15.79	5.48	15.42
40	26.8	23.37	20.18	8.54	18.91
50	32.28	31.06	28.89	10.61	29.84

Table 2: Speed-up of $\mathcal{P}$ over BCHL+ .

In this section, we present a comparative analysis between our linear program $\mathcal{P}$ and the LP of Boutilier et al. [13] (referred to as BCHL+) for computing the optimal distribution (and its corresponding distortion) on a given preference profile. We test on preference profiles generated synthetically as well as drawn from real-world datasets. To solve the LPs, we use Gurobi [22], with each run utilizing 4 cores and 50 GB of memory.

5.1 Synthetic Data

We generated instances from three statistical models: the *Mallows* model with parameter $\phi \in \{0.1, 0.2, 0.5, 0.8, 1\}$, the *Plackett-Luce* model, and random[2] *Single-Peaked* preferences; see [28] for a description of these models. All instances were comprised of $n = 1000$ agents. For $m \in \{10, 20, 30, 40, 50\}$ alternatives and each model, we generated 10 instances and calculated the average running time along with the standard error.

Results. Figure 3 and Table 2 provide a summary of the running time comparison between the two LPs on synthetic data. Notably, BCHL+ failed to terminate in 3 hours on every instance with $m = 60$ under every model, except for the single-peaked model (Figure 3c), whereas $\mathcal{P}$ completed in less than 150 seconds on each such instance. The speed-up (ratio of average running times) achieved by $\mathcal{P}$ is detailed in Table 2. For the larger values of $m \in \{30, 50\}$, $\mathcal{P}$ consistently outperformed BCHL+ across all statistical models, achieving as much as 10x to 30x speed-up, with the performance gap widening with m increasing. The only case where BCHL+ performed better was small instances ($m \in \{10, 20\}$) generated using the Single-Peaked model. Our experiments demonstrate a significant improvement in the efficiency for $\mathcal{P}$ over BCHL+ .

5.2 Preflib Data

We utilized the 7742 real-world preference profiles from Preflib [26] of type *SOC* (strict order - complete order). The largest instances had as many as $n = 14081$ voters and as many as $m = 1080$ alternatives.

Results The running times of BCHL+ and $\mathcal{P}$ on Preflib data are illustrated in Figures 1 and 2, with color intensities representing $\log_{10}$ of the running time. Here are the various percentiles of the running times of the two LPs across the Preflib instances.

[2]We sampled the positions of agents and alternatives iid in $[0, 1]$ and derived preferences based on distances.

Rule	50%	90%	99%	99.5%	mean
BCHL+	51.63	653.83	1737.71	2582.31	218.28†
$\mathcal{P}$	0.84	4.6	21.84	257.16	9.42

† excluding 20 instances where BCHL+ did not finish in an hour

Our LP $\mathcal{P}$ was successfully solved in under a minute in 99% of the instances. On average across all instances, $\mathcal{P}$ completed in under 10 seconds, marking a speed-up of more than 20x over BCHL+ . Additionally, instances denoted by $\times$ on Figures 1 and 2 are the ones where the algorithms failed to terminate in an hour. These were 20 instances for BCHL+ versus 5 instances for $\mathcal{P}$; $\mathcal{P}$ concluded on these 5 instances within 1.5 hours, while BCHL+ still did not terminate on those 20 instances after 3 hours.

6 Discussion and Future Work

An interesting question is whether we can extend our results in Section 4.4 to larger values of m. The bottleneck is the running time of the quadratic program $\mathcal{Q}$. Could there be a method that computes the distortion of a given RPSR faster? We conjecture that the associated computational problem is NP-hard, and therefore a polynomial-time algorithm should not be expected. Proving this conjecture is an interesting avenue for future work. Finally, could we extend our approach in Section 4.1 to voting rules beyond RPSRs? One candidate class of rules would be *support voting schemes*, which, together with RPSRs complete the class of truthful randomized rules on ordinal preferences [9, 20].

We studied single winner selection, where the goal is to choose a single alternative as the winner. Some of our techniques may be applicable to related settings such as multiwinner selection, participatory budgeting, matching and assignment problems, incomplete preferences, domain restrictions, and metric distortion problems.

References

[1] An efficient algorithm for determining the convex hull of a finite planar set. *Info. Proc. Lett.*, 1:132–133, 1972.

[2] Georgios Amanatidis, Georgios Birmpas, Aris Filos-Ratsikas, and Alexandros A Voudouris. Peeking behind the ordinal curtain: Improving distortion via cardinal queries. *Artificial Intelligence*, 296:103488, 2021.

[3] Georgios Amanatidis, Georgios Birmpas, Aris Filos-Ratsikas, and Alexandros A Voudouris. A few queries go a long way: Information-distortion tradeoffs in matching. *Journal of Artificial Intelligence Research*, 74:227–261, 2022.

[4] Nima Anari, Moses Charikar, and Prasanna Ramakrishnan. Distortion in metric matching with ordinal preferences. In *Proceedings of the 24th ACM Conference on Economics and Computation*, pages 90–110, 2023.

[5] Elliot Anshelevich and Wennan Zhu. Ordinal approximation for social choice, matching, and facility location problems given candidate positions. *ACM Transactions on Economics and Computation (TEAC)*, 9(2):1–24, 2021.

[6] Elliot Anshelevich, Onkar Bhardwaj, Edith Elkind, John Postl, and Piotr Skowron. Approximating optimal social choice under metric preferences. *Artificial Intelligence*, 264:27–51, 2018.

[7] Elliot Anshelevich, Aris Filos-Ratsikas, Nisarg Shah, and Alexandros A Voudouris. Distortion in social choice problems: The first 15 years and beyond. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence Survey Track*, pages 4294–4301, 2021.

[8] Haris Aziz. Justifications of welfare guarantees under normalized utilities. *ACM SIGecom Exchanges*, 17(2):71–75, 2020.

[9] Salvador Barbera. Nice decision schemes. *Decision theory and social ethics*, pages 101–117, 1978.

[10] Gerdus Benade, Swaprava Nath, Ariel D Procaccia, and Nisarg Shah. Preference elicitation for participatory budgeting. *Management Science*, 67(5):2813–2827, 2021.

[11] Umang Bhaskar, Varsha Dani, and Abheek Ghosh. Truthful and near-optimal mechanisms for welfare maximization in multi-winner elections. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pages 925–932, 2018.

[12] Allan Borodin, Daniel Halpern, Mohamad Latifian, and Nisarg Shah. Distortion in voting with top-t preferences. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence (IJCAI)*, page 4, 2022.

[13] Craig Boutilier, Ioannis Caragiannis, Simi Haber, Tyler Lu, Ariel D Procaccia, and Or Sheffet. Optimal social choice functions. *Artificial Intelligence*, 227(C):190–213, 2015.

[14] Ioannis Caragiannis, Swaprava Nath, Ariel D Procaccia, and Nisarg Shah. Subset selection via implicit utilitarian voting. *Journal of Artificial Intelligence Research*, 58:123–152, 2017.

[15] Ioannis Caragiannis, Aris Filos-Ratsikas, Søren Kristoffer Stiil Frederiksen, Kristoffer Arnsfelt Hansen, and Zihan Tan. Truthful facility assignment with resource augmentation: An exact analysis of serial dictatorship. *Mathematical Programming*, pages 1–30, 2022.

[16] Soroush Ebadian, Anson Kahng, Dominik Peters, and Nisarg Shah. Optimized distortion and proportional fairness in voting. In *Proceedings of the 23rd ACM Conference on Economics and Computation*, pages 563–600, 2022.

[17] Soroush Ebadian, Aris Filos-Ratsikas, Mohamad Latifian, and Nisarg Shah. Explainable and efficient randomized voting rules. In *Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS), to appear*, 2023.

[18] Aris Filos-Ratsikas and Peter Bro Miltersen. Truthful approximations to range voting. In *Proceedings of the 10th Conference on Web and Internet Economics*, pages 175–188. Springer, 2014.

[19] Aris Filos-Ratsikas, Srøen Kristoffer Stiil Frederiksen, and Jie Zhang. Social welfare in one-sided matchings: Random priority and beyond. In *Proceedings of the 7th Symposium on Algorithmic Game Theory*, pages 1–12, 2014.

[20] Allan Gibbard. Manipulation of schemes that mix voting with chance. *Econometrica: Journal of the Econometric Society*, pages 665–681, 1977.

[21] Mohak Goyal, Sukolsak Sakshuwong, Sahasrajit Sarmasarkar, and Ashish Goel. Low sample complexity participatory budgeting. *arXiv preprint arXiv:2302.05810*, 2023.

[22] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2023. URL https://www.gurobi.com.

[23] Zhihao Jiang, Kamesh Munagala, and Kangning Wang. Approximately stable committee selection. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 463–472, 2020.

[24] Debmalya Mandal, Ariel D Procaccia, Nisarg Shah, and David Woodruff. Efficient and thrifty voting by any means necessary. *Advances in Neural Information Processing Systems*, 32, 2019.

[25] Debmalya Mandal, Nisarg Shah, and David P Woodruff. Optimal communication-distortion tradeoff in voting. In *Proceedings of the 21st ACM Conference on Economics and Computation*, pages 795–813, 2020.

[26] Nicholas Mattei and Toby Walsh. Preflib: A library for preferences http://www. preflib. org. In *International conference on algorithmic decision theory*, pages 259–270. Springer, 2013.

[27] Ariel D Procaccia and Jeffrey S Rosenschein. The distortion of cardinal preferences in voting. In *Cooperative Information Agents X: 10th International Workshop, CIA 2006 Edinburgh, UK, September 11-13, 2006 Proceedings 10*, pages 317–331. Springer, 2006.

[28] Stanisław Szufa, Piotr Faliszewski, Piotr Skowron, Arkadii Slinko, and Nimrod Talmon. Drawing a map of elections in the space of statistical cultures. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems*, pages 1341–1349, 2020.

[29] Andrew Chi-Chin Yao. Probabilistic computations: Toward a unified measure of complexity. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pages 222–227. IEEE Computer Society, 1977.

Appendix

In this supplementary material (referred to as "Appendix" in the main text), we include proofs and details.

A The LP of Boutilier et al. [13]

We include the LP of Boutilier et al. [13] here for completeness. For details about the logic behind this linear program and its correctness, we refer the reader to the original paper.

Let $\vec{\sigma}$ be the queried preference profile, for which we want to find $\min_p \mathsf{dist}(p, \vec{\sigma})$. For each $a \in A$, define a variable p_a, and let $D = (\min_p \mathsf{dist}(p, \vec{\sigma}))^{-1}$ be a variable in the LP that its inverse corresponds to the optimal distortion. The LP of Boutilier et al. [13] is defined as follows.

Linear Program of Boutilier et al. [13]

$$\begin{aligned}
\text{max.} \quad & D & \\
\text{s.t.} \quad & \text{INEQ}(\{p_a\}_{a \in A}, D, a^*) \text{ is satisfied} & \forall a^* \in A \\
& \sum_{a \in A} p_a = 1 & \\
& p_a \geqslant 0 & \forall a \in A,
\end{aligned}$$

$\text{INEQ}(\{p_a\}_{a \in A}, D, a^*)$:

$$\begin{aligned}
& \sum_{i \in N} y_{i,a^*} \geqslant 0 & \\
& D + \sum_{a \in A} x_{a,a^*} \leqslant 0 & \\
& p_a + x_{a,a^*} - y_{i,a^*} - z_{1,i,a^*} \geqslant 0 & \forall i \in N, a = \sigma_i(1) \\
& p_a + x_{a,a^*} - y_{i,a^*} - z_{r,i,a^*} + z_{r-1,i,a^*} \geqslant 0 & \\
& \quad \forall i \in N, a = \sigma_i(r) : r \in [2, m-1] & \\
& p_a + x_{a,a^*} - y_{i,a^*} + z_{m-1,i,a^*} \geqslant 0 & \forall i \in N, a = \sigma_i(m) \\
& x_{a,a^*} \geqslant 0 & \forall a \in A \setminus \{a^*\} \\
& z_{r,i,a^*} \geqslant 0 & \forall r \in [m-1], i \in N
\end{aligned}$$

For each $a \in A$, the number of variables and constraints in $\text{INEQ}(p, D, a^*)$ is $O(nm)$. Thus, the total number of variables and constraints is $O(nm^2)$.

B Omitted Details from Section 3

B.1 Proof of Lemma 1

Lemma 1. *There is an algorithm that given a preference profile $\vec{\sigma}$, distribution p, and a real number d checks if distortion of p w.r.t $\vec{\sigma}$ is at most d in linear time $O(nm)$. If not, it finds a witness utility profile for which distortion of p is more than d.*

Proof. We demonstrate that Algorithm 1 accurately answers the query. According to Equation (5), $\mathsf{dist}(p, \vec{\sigma}) \leqslant d$ if and only if

$$\max_a \sum_{i \in A} \max_{r \in [m]} \frac{1}{r} \left(\mathbb{I}\left[\mathsf{rank}_i(a) \leqslant r\right] - d \cdot \sum_{\ell=1}^{r} p_{\sigma_i(\ell)} \right) \leqslant 0.$$

Our objective is to assess the left-hand side of the above equation. This is feasible if we have the value for each agent i and alternative a defined as

$$\delta_{i,a} = \max_{r \in [m]} \frac{1}{r} \left(\mathbb{I}\left[\mathsf{rank}_i(a) \leqslant r\right] - d \cdot \sum_{\ell=1}^{r} p_{\sigma_i(\ell)} \right). \tag{8}$$

Hence,

$$\mathsf{dist}(p, \vec{\sigma}) \leqslant d \iff \max_a \sum_{i \in N} \delta_{i,a} \leqslant 0.$$

The objective above is separable by agents. Therefore, it suffices to prove that for each voter i, we can compute all $\delta_{i,a} a \in A$ in $O(m)$ time. This is accomplished in lines 2-9 of Algorithm 1. We calculate these values similar to the linear program $\mathcal{P}$. Let $si, r = \sum_{\ell \in [r]} p_{\sigma_i(r)}$. These values can be computed in $O(m)$ time, as shown in line 4. Next, let

$$\alpha_{i,r} = \max_{\ell \in [r]} -\frac{1}{\ell} \cdot d \cdot s_{i,r}.$$

These values can also be computed in $O(m)$ time, as depicted in line 5. Each $\alpha_{i,r}$ is the maximum of the preceding one $\alpha_{i,r-1}$ and the new term $-\frac{1}{\ell} \cdot d \cdot s_{i,r}$. Similarly, define

$$\beta_{i,r} = \max_{\ell \in [r]} \frac{1}{\ell} - \frac{1}{\ell} \cdot d \cdot s_{i,r},$$

and compute the series in $O(m)$, as shown in line 6. With these values available, Equation (8) can be computed in $O(1)$ for each $a \in A$ as follows:

$$\delta_{i,a} = \max\{\alpha_{i,\mathsf{rank}_i(a)-1}, \beta_{i,\mathsf{rank}_i(a)-1}\},$$

as executed in line 7. Consequently, Algorithm 1 accurately determines whether $\mathsf{dist}(p, \vec{\sigma}) \leqslant d$ or not. If it exceeds d, we identify the candidate $a^* \in A$ that maximizes $\sum_{i \in N} \delta_{i,a}$ in $O(nm)$ time (line 11). Having a^* fixed, for each agent i, we can identify the dichotomous utility vector maximizing Equation (8) in $O(m)$ (using the auxiliary variable $s_{i,r} = \sum_{\ell=1}^{r} p_{\sigma_i(\ell)}$) and return the witness utility profile. □

Algorithm 1: CheckDistortion(of p, w.r.t. $\tilde{\sigma}$, is at most d)

```
Input: Preference Profile $\vec{\sigma}$, distribution $p$
Output: Yes, No with a witness utility profile
1  for $i \in N, a \in A$ do $\delta_{i,a} \leftarrow 0$;
2  for $i \in N$ do
3      $s_{i,0} \leftarrow 0$, $\alpha_{i,0} \leftarrow -\infty$, $\beta_{i,m+1} \leftarrow -\infty$;
4      for $r = 1$ to $m$ do   $s_{i,r} \leftarrow s_{i,r-1} + p_{\sigma_i(r)}$;
5      for $r = 1$ to $m$ do   $\alpha_{i,r} \leftarrow \max\{ \alpha_{i,r-1}, -\frac{1}{r} \cdot d \cdot s_{i,r} \}$ ;
6      for $r = m$ to $1$ do   $\beta_{i,r} \leftarrow \max\{ \beta_{i,r+1}, \frac{1}{r} \cdot (1 - d \cdot s_{i,r}) \}$;
7      for $r = 1$ to $m$ do   $\delta_{i,\sigma_i(r)} \leftarrow \max\{\alpha_{i,r-1}, \beta_{i,r}\}$ ;
8  if $\max_{a \in A} \sum_i \delta_{i,a} \leqslant 0$ then
9      return Yes
10 else
11     $a^* \leftarrow \arg\max_{a \in A} \sum_i \delta_{i,a}$;
12     for $i \in N$ do
13         $r_i^* \leftarrow \arg\max_{r \in [m]} \frac{1}{r} (\mathbb{I}[\mathsf{rank}_i(a^*) \leqslant r] - s_{i,r})$;
14         $u_i \leftarrow \mathbb{1}_{i,r_i^*}$;                         // Respective dichotomous utility
15     return No, due to utility profile $\{u_i\}_{i \in N}$
```

B.2 Proof of Lemma 2

Lemma 2. *There exists a set $D = \{d_0 = 0, d_1, d_2, \ldots, d_{|D|-1} = \infty\}$ of $O(nm)$ values $d_0 \leqslant d_1 \leqslant \ldots \leqslant d_{|D|-1}$, computable in $O(nm)$ time, such that for all $\ell \in [|D| - 1]$ and all $d' \in [d_{\ell-1}, d_\ell)$, it holds that $h_{i,a}(d') = h_{i,a}(d_{\ell-1})$ for all $i \in N$ and $a \in A$.*

Proof. Fix an agent i. We show that there are at most m pivotal values of d for i. Suppose $a = \sigma_i(r)$. As in Equation (6), break the maximum of $h_{i,a}(d)$ into two cases of $h_{i,a}(d) < r$ or $h_{i,a}(d) \geqslant r$. For the first case, the maximum is attained at one index independent of the value of d, i.e., $\arg\max_{\ell \in [r-1]} \left\{ -\frac{1}{\ell} \cdot d \cdot \sum_{q=1}^{\ell} p_{\sigma_i(\ell)} \right\}$ is a constant function w.r.t. d ($d \geqslant 0$). For the latter case, take $f_{i,r}(d) = \max_{\ell \in [r,m]} \left(\frac{1}{\ell} - d \cdot \left(\frac{1}{\ell} \sum_{q=1}^{\ell} p_{\sigma_i(q)} \right) \right)$. Note that $f_{i,r}(d)$ is the convex hull of lines $l_{i,\ell}(d) = \frac{1}{\ell} - d \cdot \left(\frac{1}{\ell} \sum_{q=1}^{\ell} p_{\sigma_i(q)} \right)$. The convex hull has at most $m - r$ transition points (pivotal d's) between the lines $L_{i,r} = \{l_{i,\ell}\}_{\ell \in [r,m]}$.

Next, we show that the convex-hull of $L_{i,r}$, denoted by c-hull($L_{i,r}$) can be computed from c-hull($L_{i,r+1}$) by adding line $l_{i,r}$ and at most one new point (pivotal d) will be realized. At $d = 0$, $l_{i,r}$ has the maximum value of $\frac{1}{r}$ among $L_{i,r}$. Thus, it will be included in c-hull($L_{i,r}$) at $d = 0$ (already in D). We can find the maximum d that $l_{i,r}$ is in the convex-hull by iterating over the lines in c-hull($L_{i,r}$) similar to the the Graham's Scan algorithm [1] and add this point (pivotal d). Therefore, the algorithm correctly finds the values of d such that between two consecutive ones, $h_{i,a}(d)$ does not change for all a. The amortized running time per agent is $O(m)$ since each line is only added and removed once from the convex-hull. □

Algorithm 2: MeasureDistortion(of f, w.r.t. $\tilde{\sigma}$)

```
Input: Preference Profile $\vec{\sigma}$, voting rule $f$
Output: Distortion value of $f(\vec{\sigma})$ w.r.t. to $\vec{\sigma}$
1  $p \leftarrow f(\vec{\sigma})$, $D \leftarrow \{0, \infty\}$;
2  for $i \in N$ do
3      $s_{i,0} \leftarrow 0$; for $r = 1$ to $m$ do $s_{i,r} \leftarrow s_{i,r-1} + p_{\sigma_i(r)}$;
       /* Find the upper convex hull of lines $l_{i,r}(d) = \frac{1}{r} - \frac{1}{r} \cdot s_{i,r} \cdot d$.   */
4      upper-hull$_i \leftarrow \emptyset$;
5      for $r = m$ to $1$ do
6          upper-hull$_i \leftarrow$ Add line $l_r(d) = \frac{1}{r} - \frac{1}{r} \cdot s_{i,r} \cdot d$ to upper-hull$_i$ using Graham's Scan
             algorithm;
7          $d'_{i,r} \leftarrow$ the maximum $d$ such that $l_{i,r}(d)$ appears in upper-hull$_i$ ; // $d'_{i,r}$ can be $\infty$
8          $D \leftarrow D \cup \{d'_{i,r}\}$;
9  Sort values of $D$ in increasing order;
10 Using the checker of Lemma 1, do binary search to find the maximum $d \in D$ that
     distortion of $p$ w.r.t. $\vec{\sigma}$ is more than $d$, and find the corresponding utility profile $\vec{u}^*$;
11 Find the final distortion value $d^*$ of $p$ for $\vec{u}^*$;
12 return $d^*$;
```

C Omitted Details from Section 4

C.1 Proof of Lemma 3

Lemma 3. $|W_m| \leqslant 2^m - 2$.

Proof. Since utility profiles are dichotomous, each u_i partition the alternatives into two sets with equal utility. We know that members of each of these two sets appear in the reverse order of $\succ$. That means if we know the members of each set then σ_i is unique. So the number of different pairs of (σ_i, u_i) is equal to the number of ways in which we can partition the alternatives into two sets. In addition we have to make sure that the first set (alternatives with $1/r$ utility) is non empty. That means we have at most $2^m - 1$ such pairs. In addition we know that if a^* is the only member of the bottom set (the only one with zero utility), then by changing the utility function from $\mathbb{1}_{m-1}$ to $\mathbb{1}_m$ we get a higher distortion. That means the set of all possible (σ_i, u_i) includes at most $2^m - 2$ members. □

C.2 Proof of Lemma 4

Lemma 4. *For every randomized positional scoring rule $f_{\vec{s}}$, there exists a pair $(\vec{\sigma}, \vec{u})$ of preference and utility profiles satisfying Properties (P1) to (P3) at which $f_{\vec{s}}$ attains its worst distortion.*

Proof. First by Theorem 1 we know that a worst-case instance with dichotomous utility profile exists. We can compute the social welfare of the alternatives in that instance and relabel the alternatives to match the ordering $\succ$. That means we have a worst case instance that satisfies Properties (P1) and (P2).

We want to prove that there exists an instance that satisfies Property (P3). For the sake of contradiction assume that such instance does not exist. Consider a worst-case instances $(\vec{u}, \vec{\sigma})$ that satisfy Properties (P1) and (P2) but not Property (P3). Among all those instance consider the one with the minimum number of agents $i \in N$ and pairs $a \succ b$ that $u_i(a) = u_i(b)$ but $a \succ_i b$. If such pair exists in an agent's ranking then a consecutive pair of alternatives in the ranking with such condition also exists. So w.l.o.g. assume that $\mathsf{rank}_i(a) = \mathsf{rank}_i(b) - 1$. Let this instance be $(\vec{u}, \vec{\sigma})$. We want to create instance $(\vec{u}', \vec{\sigma}')$ with (at least) the same distortion with less number of such pairs.

Define $r_a := \mathsf{rank}_i(a)$ and $r_b := \mathsf{rank}_i(b)$. If we switch the position of a and b in σ_i to form $\vec{\sigma}'_i$, we do not change the social welfares and we have

$$\begin{aligned}
\mathsf{dist}(f_{\vec{s}}(\vec{\sigma}'), \vec{u}') &= \frac{\mathsf{sw}(a^*, \vec{u}')}{\sum\limits_{a' \in A} \mathsf{P}_{\vec{s}}(a', \vec{\sigma}')\mathsf{sw}(a', \vec{u}')} \\
&= \frac{\mathsf{sw}(a^*, \vec{u})}{\sum\limits_{a' \in A} \mathsf{P}_{\vec{s}}(a', \vec{\sigma}')\mathsf{sw}(a', \vec{u}) + (s_{r_b} - s_{r_a})\mathsf{sw}(a, \vec{u}) + (s_{r_a} - s_{r_b})\mathsf{sw}(b, \vec{u})} \\
&= \frac{\mathsf{sw}(a^*, \vec{u})}{\sum\limits_{a' \in A} \mathsf{P}_{\vec{s}}(a', \vec{\sigma}')\mathsf{sw}(a', \vec{u}) + (s_{r_a} - s_{r_b})(\mathsf{sw}(b, \vec{u}) - \mathsf{sw}(a, \vec{u}))} \\
&\geqslant \mathsf{dist}(f_{\vec{s}}(\vec{\sigma}), \vec{u}).
\end{aligned}$$

This has one less "bad" pair since we do not change the ranking of other voters and the ranking of other alternatives in voter i's ranking. $\square$

C.3 Example W_m, U, and P

	(σ_1, u_1)	$(\sigma_2, u_)$	(σ_3, u_3)	(σ_4, u_4)	(σ_5, u_5)	(σ_6, u_6)
σ_i	$a_1 \succ a_2 \succ a_3$	$a_2 \succ a_1 \succ a_3$	$a_2 \succ a_3 \succ a_1$	$a_3 \succ a_1 \succ a_2$	$a_3 \succ a_2 \succ a_1$	$a_2 \succ a_3 \succ a_1$
u_i	$[1,\ 0,\ 0]$	$[\frac{1}{2},\ \frac{1}{2},\ 0]$	$[1,\ 0,\ 0]$	$[\frac{1}{2},\ \frac{1}{2},\ 0]$	$[1,\ 0,\ 0]$	$[\frac{1}{3},\ \frac{1}{3},\ \frac{1}{3}]$

Table 3: The set $W_3 = \{(\sigma_1, u_1), \ldots, (\sigma_6, u_6)\}$ from Lemma 3.

$$U = \begin{bmatrix} 1 & \frac{1}{2} & 0 & \frac{1}{2} & 0 & \frac{1}{3} \\ 0 & \frac{1}{2} & 1 & 0 & 0 & \frac{1}{3} \\ 0 & 0 & 0 & \frac{1}{2} & 1 & \frac{1}{3} \end{bmatrix} \text{ and } P = \begin{bmatrix} s_1 & s_2 & s_3 & s_2 & s_3 & s_3 \\ s_2 & s_1 & s_1 & s_3 & s_2 & s_1 \\ s_3 & s_3 & s_2 & s_1 & s_1 & s_2 \end{bmatrix}.$$

D Computing the distortion of deterministic PSRs

To find the distortion of a given PSR we take the same route as in Section 4 by identifying some properties for the worst-case instances.

Lemma 5. *For any (deterministic) positional scoring rule $f_{\vec{s}}$ and alternatives a^*, a^+ there exists a pair of preference profile $\vec{\sigma}$ and utility profile $\vec{u} \in \mathcal{C}(\vec{\sigma})$ that satisfies the following properties and has the maximum possible distortion for that rule.*

(i) a^* is the optimal alternative and a^+ is the winner.

(ii) For $i \in N$ if $a^* \succ_i a^+$ then $u_i = \mathbb{1}_{\mathsf{rank}_i(a^*)}$, and $\mathsf{rank}_i(a^+) = \mathsf{rank}_i(a^*) + 1$.

(iii) For $i \in N$ if $a^+ \succ_i a^*$ then $u_i = \mathbb{1}_m$, and $\mathsf{rank}_i(a^+) = 1$.

In addition, the number of possible pairs of σ, u that appear in one of such preference and utility profiles is at most $2(m-1)!$.

Proof. For any worst-case profile the winner and the optimal are different alternatives. Relabel the alternatives to make a^* the optimal and a^+ the winner. Let $(\sigma, \vec{u})$ be such worst-case instance. We want to build $(\sigma', \vec{u}')$ with the desired properties.

Consider any $i \in N$ where $a^* \succ_i a^+$. The distortion is

$$\frac{\mathsf{sw}(a^*, \vec{u})}{\mathsf{sw}(a^+, \vec{u})},$$

so to maximize the distortion we can set $u'_i = \mathbb{1}_{\mathsf{rank}_i(a^*)}$. After this if we move a^+ up to be just below a^* we do not increase it's social welfare, and do not decrease it's score. That means winner remains the same and the distortion does not change, and hence we have a worst-case instance with Property (ii).

Consider any $i \in N$ where $a^+ \succ_i a^*$. The distortion is

$$\frac{\mathsf{sw}(a^*, \vec{u})}{\mathsf{sw}(a^+, \vec{u})},$$

so to maximize the distortion we can set $u'_i = \mathbb{1}_m$. After this if we move a^+ to the top of the list. By this change we do not increase it's social welfare, and do not decrease it's score. That means winner remains the same and the distortion does not change, and hence we have a worst-case instance with Property (iii).

Now consider any agent in $(\sigma', \vec{u}')$. There are $(m-1)!$ rankings over $A \setminus \{a^+\}$. For each of them either $a^* \succ_i a^+$, in which case by Property (ii), $u_i = \mathbb{1}_{\mathsf{rank}_i(a^*)}$, and $\mathsf{rank}_i(a^+) = \mathsf{rank}_i(a^*) + 1$, or $a^+ \succ_i a^*$ then by Property (iii), $u_i = \mathbb{1}_m$, and $\mathsf{rank}_i(a^+) = 1$. That means we have at most 2 possible total ordering, and utility function for each of the $(m-1)$ partial orderings over $A \setminus \{a^+\}$. □

Now let $W_m = \{(\sigma_i, u_i)\}$ be the set of all possible preferences and utility functions in a worst-case instance that satisfy the properties of Lemma 5, and define q to be a vector of size $w := t$, where q_i is in proportion to the fraction of the agents with preference ranking σ_i, and utility function u_i. We can use the following linear program to find the distortion of this rule.

m	Optimal DPSR	Optimal DPSR Distortion	Optimal RPSR	Optimal RPSR Distortion	# Iteratio
2	$[1, 0]$	3	$[0.843, 0.157]$	1.5	4
3	$[1, 0, 0]$	5.5	$[0.776, 0.189, 0.035]$	1.870	6
4	$[0.65, 0.26, 0.09, 0.0]$	8.333	$[0.7247, 0.2025, 0.069, 0.0038]$	2.188	7
5	$[0.634, 0.22, 0.082, 0.036, 0.028]$	11.418	$[0.6645, 0.2121, 0.0881, 0.0353, 0.0]$	2.474	9

Table 4: Optimal distortion of PSR and RPSR with their corresponding scoring vectors for m up to 5.

Linear Program $\mathcal{P}$'

$$\begin{aligned}
\max \quad & \sum_{i\in[w]} q_i u_i(a^*) & \\
\text{s.t.} \quad & \sum_{i\in[w]} q_i s_{\mathsf{rank}_i(a)} \leqslant \sum_{i\in[w]} q_i s_{\mathsf{rank}_i(a^+)} & \forall a \in A \\
& \sum_{i\in[w]} q_i u_i(a) \leqslant \sum_{i\in[w]} q_i u_i(a^*) & \forall a \in A \\
& \sum_{i\in[w]} q_i u_i(a^+) = 1 & \\
& q_i \geqslant 0 & \forall i \in [w]
\end{aligned}$$

To find the best possible scoring vector we perform a grid search on the space of possible PSRs; while this search space is very large (in fact continuous), the fact that we only have to solve a linear program for each candidate point allows us to use a rather fine grid for our search. This technically only gives us upper bounds, but if there is a better rule, that should be on a point which not on our grid. Of course, we have not proved that the distortion function is continuous, although all of our empirical observations seem to suggest that. In that case, we would expect our upper bounds to be within ε of the distortion of the best possible PSR, where ε would depend on the size of the grid; we take the grid granularity to be 0.001, so the differences in the distortion between the lower and upper bounds would be insignificant. It would be interesting to show the continuity of the distortion objective, but since it is only relevant for this rather complementary part of our main work, we leave that as a future direction.

We remark that the best possible bounds are achieved by multiple scoring vectors. In fact, for $m = 2$, *any* scoring vector achieves the best distortion of 3. The proven distortion and one of the scoring vectors that achieve that distortion for up to $m = 5$ are shown in Table 4.

E Details for Gradient-Based Search in Section 4.4

To find a lower bound on the distortion of instance optimal rule we run a gradient decent style search. As defined in Section 4.4, each worst-case instance is identified by q, a unit sum vector of size t. We start with a random starting point q that represents instance $(\vec{\sigma}, \vec{u})$ and $\varepsilon = 0.1$. In each iteration we define the set of all possible next steps as $S = \cup_{i\in[t]}\{(q_1, \ldots, q_i + \varepsilon, \ldots, q_t), (q_1, \ldots, q_i - \varepsilon, \ldots, q_t)\}$. We re-normalize all the vectors in S to make them unit sum and compute their distortions using Linear program $\mathcal{P}$. Then if there exists $q' \in S$ that represents instance $(\vec{\sigma}', \vec{u}')$, where $\mathsf{dist}(f^*(\vec{\sigma}'), \vec{u}') > \mathsf{dist}(f^*(\vec{\sigma}), \vec{u})$ we randomly jump to one of such vectors for the next iteration, and if non of them has a distortion greater than the distortion of q then we decrease the value of ε and repeat the procedure. We keep iterating while $\varepsilon \geqslant \tau_\varepsilon$ and report the distortion of instance optimal rule on the final instance as a lower bound on the distortion.

To make sure that we have a better bound we try different random starting points and report the maximum output.

F Comparing with the quadratic program of [18]

As we mentioned in Section 1.2, Filos-Ratsikas and Miltersen [18] devised a quadratic program for finding the distortion of a given RPSR, when the utilities of the agents are normalized according to the unit-range normalization.[3] In this normalization, for every agent $i \in N$, we have that $\max_{a\in A} u_i(a) = 1$, $\min_{a\in A} u_i(a) = 0$, and $u_i(a) \in [0, 1]$ for all $a \in A$.

The quadratic programming formulation of [18] is enabled by a structural characterization of the worst-case utility profiles, similar in spirit to that of our Theorem 1. In particular, they prove that the worst-case distortion is attained in what they refer to as "quasi-combinatorial" utility profiles, i.e., essentially utility profiles where the utility of an agent for any alternative is either 1 or 0. This enables them to restrict the set of possible utility vectors to a set of size $(m-1) \cdot m!$. Then, in a conceptually similar way to our approach in Section 4.2, they define variables x_i for the fraction of agents that have each of those utility vectors, and use those as variables of their optimization program. That program has linear constraints and a non-convex quadratic objective function, but the number of constraints is $(m-1)m!$. For $m = 3$, in the unit-range case, that amounts to 12 constraints, and Filos-Ratsikas and Miltersen solve their quadratic program using active set methods, by enumerating over all possible subsets of active constraints.

For their approach to be even conceivably applicable to the unit-sum case, one would need first to prove a similar structural result for the worst-case utility profiles. Our theorem 1 clearly provides such a result. Still, for a given m, the number of possible k-dichotomous utility vectors is in fact $m \cdot m!$. While at first glance this might not sound much different from the $(m-1)m!$ used by Filos-Ratsikas and Miltersen for unit-range, we remark that for $m = 3$, this increases the number of variables (and constraints in their quadratic program) from 12 to 18. Solving their quadratic program with 18 constraints is clearly computationally infeasible, and hence their technique is inapplicable for unit-sum, even when $m = 3$.

Our crucial technical contribution here is the refinement of the worst-case characterization by means of Properties (P2) and (P3). In turns, this allows us to devise a (different) quadratic program

[3]Filos-Ratsikas and Miltersen [18] only present their program in the context of $m = 3$, but its definition is valid for any number of alternatives.

with much fewer constraints, enabling us to handle up to at least 5 agents in reasonable running time.