Computational Aspects of Distortion

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

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

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

Nisarg Shah
University of Toronto
Toronto, Canada
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, wefi nd 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.

KEYWORDS

Computational Social Choice; Voting; Distortion

ACM Reference Format:
Soroush Ebadian, Aris Filos-Ratsikas, Mohamad Latifian, and Nisarg Shah. 2024. Computational Aspects of Distortion. In *Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), Auckland, New Zealand, May 6 – 10, 2024*, IFAAMAS, 14 pages.

1 INTRODUCTION

The area of computational social choice aims tofi nd 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, 5, 11–13, 15, 24, 25], participatory budgeting [10, 21], and matching [3, 4, 7, 14, 18].

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 veryfi rst 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. [17] 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. [16] 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 [16] 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 [16].

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, raising the following question:

1. Given a vector of agents' preferences, how can we compute the best possible distortion on this vector? How can wefi nd 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

Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), N. Alechina, V. Dignum, M. Dastani, J.S. Sichman (eds.), May 6 – 10, 2024, Auckland, New Zealand. © 2024 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

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 wefi nd 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.

ForQu estion 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 thefirst 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.

ForQu estion 2. We formulate the problem offi nding 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 thefi rst 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. All the appendices and missing proofs are available in the full version. [1]

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. [6].

Forfi nding 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 programfi nds 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. [17], ours seems conceptually simpler, as it is based on linear programs. In contrast, the rule in [17] 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 [19], 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.*[2] 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 [19], has much fewer constraints and can thus easily handle instances with $m = 5$ alternatives; we provide more details about the comparison with [19] in Appendix F.

Tofi nd 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 [19], 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 [19]), and can also be seen as an application of Yao's minimax principle [29].

2 PRELIMINARIES

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

[1]Full version: https://www.cs.toronto.edu/~nisarg/papers/distortion-computation.pdf

[2]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 [6, 8].

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 $a \succ_i b \Leftrightarrow \mathsf{rank}_i(a) < \mathsf{rank}_i(b)$. These rankings collectively form a preference profile $\vec{\sigma}$. 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 $C(\sigma_i)$ denote the set of utility functions consistent with σ_i. We say that utility profile $\vec{u}$ is consistent with $\vec{\sigma}$ if $u_i \in C(\sigma_i)$ for each agent $i \in N$; let $C(\vec{\sigma})$ denote the set of utility profiles consistent with $\vec{\sigma}$.

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, $\mathsf{dist}(p, \vec{u}) := \frac{\max_{a \in A} \mathsf{sw}(a, \vec{u})}{\mathsf{sw}(p, \vec{u})}$. Its distortion on a preference profile $\vec{\sigma}$ is its worst-case distortion on any utility profile $\vec{u} \in C(\vec{\sigma})$: $\mathsf{dist}(p, \vec{\sigma}) := \sup_{\vec{u} \in C(\vec{\sigma})} \mathsf{dist}(p, \vec{u})$; note that this supremum is attained at some $\vec{u} \in C(\vec{\sigma})$ 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, theyfi rst 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.

If polynomial-time computability was the sole goal, one could write the following straightforward LP given a preference profile $\vec{\sigma}$.

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

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 $\vec{u} \in C(\vec{\sigma})$. 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, andfi nding 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.

Therefore, an important contribution of Boutilier et al. [13] is 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 $\vec{u} \in C(\vec{\sigma})$ 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.

Theorem1. *For any distribution $p \in \Delta(A)$ and preference profile $\vec{\sigma}$, there exists a dichotomous utility profile $\vec{u}^* \in C(\vec{\sigma})$ where p attains its worst distortion (i.e., $\text{dist}(p, \vec{u}^*) = \text{dist}(p, \vec{\sigma})$).*

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

$$\max_{\vec{u} \in C(\vec{\sigma})} \frac{\max_{a \in A} \text{sw}(a, \vec{u})}{\text{sw}(p, \vec{u})} = d \iff$$

$$\max_{\vec{u} \in C(\vec{\sigma})} \left(\max_{a \in A} \text{sw}(a, \vec{u}) - d \cdot \text{sw}(p, \vec{u}) \right) = 0 \iff$$

$$\max_{a \in A} \max_{\vec{u} \in C(\vec{\sigma})} \sum\nolimits_{i \in N} (u_i(a) - d \cdot u_i(p)) = 0 \iff$$

$$\max_{a \in A} \sum\nolimits_{i \in N} \max_{u_i \in C(\sigma_i)} (u_i(a) - d \cdot u_i(p)) = 0, \tag{4}$$

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

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

where $\Delta^m = \{\vec{\alpha} \in [0,1]^m : \sum_{r \in [m]} \alpha_r = 1\}$ is the $(m-1)$-simplex. Thanks to thefi nal 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 $u_i \in C(\sigma_i)$ in Equation (4) with a maximum over dichotomous utility functions with respect to σ_i to obtain:

$$\text{dist}(p, \vec{\sigma}) = d \iff$$

$$\max_{a \in A} \sum_{i \in N} \max_{r \in [m]} \frac{1}{r} \left(\mathbb{I}\left[\text{rank}_i(a) \leqslant r\right] - d \cdot \sum_{\ell=1}^{r} p_{\sigma_i(\ell)} \right) = 0. \tag{5}$$

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.

Theorem2. *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 $p^* \in \arg\min_{p \in \Delta(A)} \text{dist}(p, \vec{\sigma})$ and its distortion $\text{dist}(p^*, \vec{\sigma})$.*

Proof. We derive thefi nal linear program gradually via a number of insights and transformations. Let $\vec{\sigma}$ be the given preference profile. First, note that the distortion $\text{dist}(p, \vec{\sigma})$ 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.

$$\begin{aligned} \min \quad & d \\ \text{s.t.} \quad & \delta_{i,a} \geqslant \frac{1}{r} \left(\mathbb{I}[\text{rank}_i(a) \leqslant r] - d \cdot \sum\nolimits_{\ell=1}^{r} p_{\sigma_i(\ell)} \right) \\ & \qquad\qquad\qquad \forall i \in N, a \in A, r \in [m] \\ & \sum\nolimits_{i \in N} \delta_{i,a} \leqslant 0 \qquad \forall a \in A \\ & \sum\nolimits_{a \in A} p_a = 1 \\ & p_a \geqslant 0 \qquad \forall a \in A. \end{aligned}$$

Here, $\delta_{i,a}$ upper bounds the expression from Equation (5) corresponding to afi xed $a^* = a$ 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 thefi rst 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

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

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, thefi rst constraint in the above program can be written as

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

Define $\alpha_{i,r} = \max_{\ell \in [r]} -\frac{1}{\ell} \cdot s_{i,\ell}$ and $\beta_{i,r} = \max_{\ell \in [r,m]} \frac{1}{\ell} \cdot \left(1 - s_{i,\ell}\right)$. 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 thefinal 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 $\mathcal{P}$

$$\begin{aligned}
\min \quad & \sum_{a\in[m]} \widehat{p}_a & \\
\text{s.t.} \quad & \delta_{i,\sigma_i(r)} \geqslant \alpha_{i,r-1} & \forall i\in N, r\in[2,m-1]\\
& \delta_{i,\sigma_i(r)} \geqslant \beta_{i,r} & \forall i\in N, r\in[m]\\
& \textstyle\sum_{i=1}^{n} \delta_{i,a} \leqslant 0 & \forall a\in[m]\\
\text{Partial sums:} & & \\
& s_{i,1} = \widehat{p}_{\sigma_i(1)} & \forall i\in N\\
& s_{i,r} = s_{i,r-1} + \widehat{p}_{\sigma_i(r)} & \forall i\in N, r\in[2,m]\\
\text{Top partial maximums:} & & \\
& \alpha_{i,r} \geqslant \alpha_{i,r-1} & \forall i\in N, r\in[2,m-1]\\
& \alpha_{i,r} \geqslant \frac{1}{r}\cdot(-s_{i,r}) & \forall i\in N, r\in[m-1]\\
\text{Bottom partial maximums:} & & \\
& \beta_{i,r} \geqslant \beta_{i,r+1} & \forall i\in N, r\in[m-1]\\
& \beta_{i,r} \geqslant \frac{1}{r}\left(1-s_{i,r}\right) & \forall i\in N, r\in[m]\\
\text{Variable ranges:} & & \\
& \widehat{p}_a \geqslant 0 & \forall a\in[m]\\
& \delta_{i,\sigma_i(r)}, \alpha_{i,r}, \beta_{i,r} \in \mathbb{R} & \forall i\in N, r\in[m]
\end{aligned}$$

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] andfi xing 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, but linear programming based algorithms are still slow. Using insights from Theorem 1, we develop an $O(nm\log(nm))$ time combinatorial algorithm for this problem which avoids solving linear programs. The proof and the algorithm appears in the full version.

Theorem3. *There is an algorithm to compute the distortion* $\text{dist}(p,\vec{\sigma})$ *of distribution p on preference profile* $\vec{\sigma}$ *in* $O(nm\log(nm))$ *time.*

Thefi rst step is to design a fast subroutine for checking whether $\text{dist}(p,\vec{\sigma}) \leqslant d$ 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, itfi nds a witness utility profile for which distortion of p is more than d.*

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):

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

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 $d^* := \text{dist}(p,\vec{\sigma}) \in [1,\infty]$ and get ε-close in time $O(nm\log(d^*/\varepsilon))$. 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]). In the full version, we discuss, in details, how to compute the set of all pivotal values in $O(nm)$. At a high-level, Theorem 3 follows by performing binary search over the pivotal values d, and invoking Lemma 1 for at most $O(\log nm)$ times, wefi nd the worst-case utility profile and compute its distortion in time $O(nm\log(nm))$.

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, and we do so by solving a non-convex quadratic program. Note that such programs, including ours, are not known to be solvable in polynomial time.

Randomized Positional Scoring Rules. Given a scoring vector $\vec{s} = (s_1, s_2, \ldots, s_m)$, 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 $\text{P}_{\vec{s}}(a,\vec{\sigma}) := \Pr[f_{\vec{s}}(\vec{\sigma}) = a] = \frac{1}{n|\vec{s}|}\sum_{i\in N} s_{\text{rank}_i(a)}$.

These rules, also called point-voting schemes [9], are known due to their strategyproofness [20]. Ebadian et al. [16] provide asymptotic distortion bounds for many rules in this family.

In this section, wefi rst 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, tofi nd 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 $A = \{a_1, a_2, \ldots, a_m\}$. Let us define three properties of a pair of preference profile $\vec{\sigma}$ and utility profile $\vec{u} \in C(\vec{\sigma})$.

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

(P2) $\mathrm{sw}(a_j, \vec{u}) \geqslant \mathrm{sw}(a_{j+1}, \vec{u})$ 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 $W_m = \{(\sigma_1, u_1), \ldots, (\sigma_t, u_t)\}$ denote the set of possible pairs of preference ranking σ_i and utility function $u_i \in C(\sigma_i)$ that any agent can have in any pair of preference profile $\vec{\sigma}$ and utility profile $\vec{u} \in C(\vec{\sigma})$ 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 2. $|W_m| \leqslant 2^m - 2$.

Table 4 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 3. *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. $P_{j,i} = s_{\mathrm{rank}_i(a_j)}$. Also, define $C_{t \times t} = U^\top P$. For example, when $m = 3$, matrices U and P corresponding to the set W_m from Table 4 are shown in Appendix C.

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

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

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

$$\begin{array}{lll} \max & \dfrac{qU_1^\top}{qCq^\top} & \\ \text{s.t.} & q \cdot U_j^\top \leqslant q \cdot U_1^\top & \forall j \in [m] \\ & \sum_{i \in [t]} q_i = 1 & \\ & q_i \geqslant 0 & \forall i \in [t]. \end{array}$$

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., $1/\mathrm{dist}(f_{\vec{s}})$. The new objective is to minimize D, and we add a quadratic constraint $D \cdot (q^\top U_1) \geqslant q^\top Cq$ 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 Q yields the distortion $\mathrm{dist}(f_{\vec{s}})$ 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 forfi nding the best RPSR.

We remark that Program 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 Q

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

4.3 Finding the Best Possible RPSR

In this section, we make use of our quadratic program Q tofind the RPSR that achieves the minimum distortion (and that distortion itself) for a given number of alternatives m. The idea is to use program Q repeatedly to search over the space of all RPSRS. Since this is a vast search space and solving the non-convex program Q takes time, we employ the technique of Filos-Ratsikas and Miltersen [19], 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 [19, 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 [19] 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 $\vec{u} \in C(\vec{\sigma})$. 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., $1/\text{dist}(F_m^j(\vec{\sigma}), \vec{u})$. From the characterization of Filos-Ratsikas and Miltersen [19], 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 Q on the rule f_c, which returns both dist(f_c) and the worst-case instance $(\vec{\sigma}, \vec{u})$ at which this distortion is attained. We update $d^u \leftarrow \min(d^u, \text{dist}(f_c))$ 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 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 Q we use the Gurobi Optimization Solver. We remark that the main computational bottleneck of this technique is solving the quadratic program; the aided search of Section 4.3 terminated in at most 9 iterations in all cases. All computations were performed using an Apple M2 CPU with 24 GB of Ram.

↓ 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.

We complement these results with bounds on the best (deterministic) PSRs to demonstrate the advantages of randomization. To calculate these bounds, wefi rst 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 tofi nd 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 tofi nd the best distortion on each preference profile (see Algorithm 2 in Appendix B). More details are 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 itfi nds 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 5 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) [16]. This showcases that the hidden constants in the asymptotic bounds can make a difference for small values of m.

5 EXPERIMENTS

In this section, we present a comparative analysis between our linear program $\mathcal{P}$ and the LP of Boutilier et al. [13] (referred to

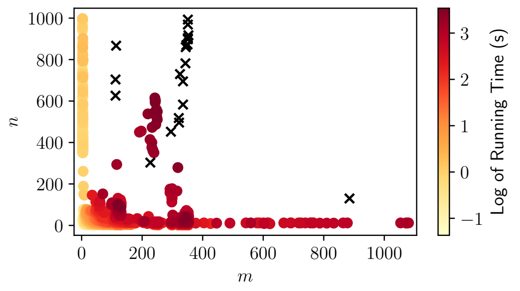


Figure 1: Performance of BCHLPS on Preflib data.

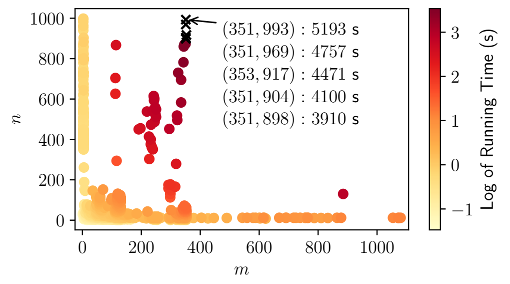
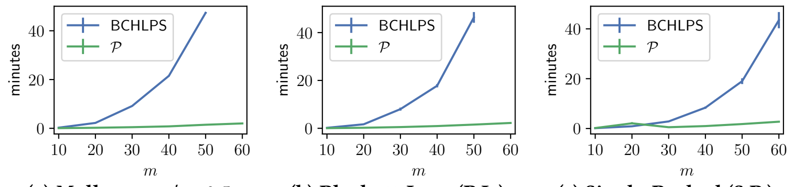


Figure 2: Performance of $\mathcal{P}$ on Preflib data.



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

Figure 3: Running time of $\mathcal{P}$ and BCHLPS 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 BCHLPS.

as BCHLPS) 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. [3]

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[4] *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, BCHLPS 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 BCHLPS 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 BCHLPS 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 BCHLPS.

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.

[3]Code: https://github.com/latifian/Computational-Aspects-of-Distortion
[4]We sampled the positions of agents and alternatives iid in $[0, 1]$ and derived preferences based on distances.

Results. The running times of BCHLPS and $\mathcal{P}$ on Preflib data are illustrated in Figures 1 and 2, with color intensities representing $\log_{10}$ of the running time. Table 3 shows various percentiles of the running times of the two LPs across the Preflib instances.

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

† excluding 20 instances where BCHLPS did notfi nish in an hour

Table 3: Order statistics of running times on Preflib

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 BCHLPS. 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 BCHLPS versus 5 instances for $\mathcal{P}$; $\mathcal{P}$ concluded on these 5 instances within 1.5 hours, while BCHLPS still did not terminate on those 5 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] 1972. An efficient algorithm for determining the convex hull of afi nite planar set. *Info. Proc. Lett.* 1 (1972), 132–133.

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

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

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

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

[6] Elliot Anshelevich, Aris Filos-Ratsikas, Nisarg Shah, and Alexandros A Voudouris. 2021. Distortion in Social Choice Problems: The First 15 Years and Beyond. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence Survey Track.* 4294–4301.

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

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

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

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

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

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

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

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

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

[16] Soroush Ebadian, Aris Filos-Ratsikas, Mohamad Latifian, and Nisarg Shah. 2023. Explainable and Efficient Randomized Voting Rules. In *Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS), to appear.*

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

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

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

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

[21] Mohak Goyal, Sukolsak Sakshuwong, Sahasrajit Sarmasarkar, and Ashish Goel. 2023. Low Sample Complexity Participatory Budgeting. *arXiv preprint arXiv:2302.05810* (2023).

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

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

[24] Debmalya Mandal, Ariel D Procaccia, Nisarg Shah, and David Woodruff. 2019. 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. 2020. Optimal communication-distortion tradeoff in voting. In *Proceedings of the 21st ACM Conference on Economics and Computation.* 795–813.

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

[27] Ariel D Procaccia and Jeffrey S Rosenschein. 2006. 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.* Springer, 317–331.

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

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