Preference Elicitation as Average-Case Sorting

Dominik Peters and **Ariel D. Procaccia**
Harvard University
{dpeters, arielpro}@seas.harvard.edu

Abstract

Many decision making systems require users to indicate their preferences via a ranking. It is common to elicit such rankings through pairwise comparison queries. By using sorting algorithms, this can be achieved by asking at most $O(m \log m)$ adaptive comparison queries. However, in many cases we have some advance (probabilistic) information about the user's preferences, for instance if we have a learnt model of the user's preferences or if we expect the user's preferences to be correlated with those of previous users. For these cases, we design elicitation algorithms that ask fewer questions in expectation, by building on results for average-case sorting. If the user's preference are drawn from a Mallows phi model, $O(m)$ queries are enough; for a mixture of k Mallows models, $\log k + O(m)$ queries are enough; for Plackett–Luce models, the answer varies with the alternative weights. Our results match information-theoretic lower bounds. We also provide empirical evidence for the benefits of our approach.

1 Introduction

A sizable body of literature deals with the problem of eliciting a decision maker's preferences, represented as a ranking over a set of m alternatives, via pairwise comparison queries ("which of these is better: x or y?"). In its most basic form, this is just the standard sorting problem, which requires $\Theta(m \log m)$ queries in the worst case.

Our point of departure is that we assume access to some probabilistic information about the ranking. Suppose we know (or suspect) that the ranking is drawn from some distribution $\mathcal{D}$ over all $m!$ possible rankings. We can then use this prior to guide our queries, potentially resulting in an average-case number of queries that is significantly lower than the worst-case bound. Specifically, we ask:

> *Given a "typical" distribution $\mathcal{D}$ over rankings, what is the expected number of pairwise comparisons queries needed to identify a ranking drawn from $\mathcal{D}$?*

From this viewpoint, preference elicitation becomes closely intertwined with *average-case sorting*, and we will indeed show that the connection is technically fruitful. But first, let us motivate the research question by describing two domains in which it is potentially possible to apply informed elicitation schemes, which rely on knowledge of $\mathcal{D}$ to achieve lower query complexity.

A better UI for eliciting rankings Social choice practitioners have implemented a number of web tools that allow users to set up polls that are run using the best voting methods taken from the literature. To use these tools, each user needs to input a ranking of the alternatives. Typically, the user interfaces of these tools allow this either by presenting a random ranking and letting the user to correct it via Drag and Drop (e.g., *RoboVote*, *Pnyx*, or *OPRA*) or by presenting a list of alternatives and asking the user to click on them in order of preference, thereby building the ranking (e.g., *Whale*). A version of the latter method is also used in Australian elections for the House of Representatives, where voters number candidates (1, 2, 3, ...) in order of preference on a paper ballot. All these methods require significant effort, something like $\Theta(m^2)$ mental operations, because the list of alternatives needs to be scanned repeatedly. But we know that we can do it using any of the common sorting algorithms in $O(m \log m)$ queries!

In larger elections, we could obtain a prior via polls or early votes. Using informed elicitation schemes, this should, on average, lead to a reduction in the number of queries.

Virtual democracy In some situations, a collection of decision makers need to repeatedly make similar decisions. Running a vote on each occasion may be tedious. Instead, the emerging paradigm of *virtual democracy* proposes that for each decision maker, we train a machine learning model of their preferences (for example by learning personalized weights of various attributes). When a new decision needs to be made, we use these models to predict each voter's preference ranking, and then use a voting rule to make a decision. Variants of this scheme have been proposed for ethical decision making in self-driving cars (Noothigattu et al. 2018) and for prioritization decision in kidney exchange (Freedman et al. 2020). It has been deployed in practice at a Pittsburgh charity that transports food donations to nonprofits (Lee et al. 2019).

While decision makers may be happy to outsource preference determination to the model most of the time, there may be instances of increased importance to a voter, when they may wish to vote themselves or check the quality of the model output. An informed elicitation scheme can help speed up this process, since we already have a probabilistic model of what the voter is going to say.

Our Approach and Results The first decision we must make is how to represent the prior $\mathcal{D}$ over rankings. We consider the most common preference models — also known as *cultures* (Regenwetter et al. 2006) or as *random noise models* — namely Mallows ϕ model, a mixture of Mallows models, and the Plackett–Luce model. Although these models may be seen as stylized, the Mallows model has been argued to be a particularly good fit for virtual democracy applications (Kahng et al. 2019), and so our results should be directly applicable in that domain. In addition, as we shall see in Section 6, some of these models are realistic enough to lead to significant gains in practice.

For our main results we present adaptive elicitation schemes that use a minimum number of queries in the average case (taken with respect to $\mathcal{D}$). These elicitation schemes are based on insertion sort together with a binary search that is biased using information from $\mathcal{D}$. This method matches the information-theoretic lower bound; in particular by a result of Moran and Yehudayoff (2016), we use at most $H(\mathcal{D}) + 2m$ queries, where $H(\mathcal{D})$ is the Shannon entropy of $\mathcal{D}$. For each of the models $\mathcal{D}$ we study, we estimate this entropy, and find that often $O(m)$ queries are enough in expectation, improving upon the lower bound of $\Omega(m \log m)$ queries if we do not have prior information. Notably, a linear number of queries is enough even for mixtures of Mallows models; so our positive results apply to a very flexible class of distributions.

In is worth noting that we are able to obtain strikingly decisive answers to our research question by building on the literature on average-case sorting. These techniques, and more broadly approaches based on entropy, have not played a role in computational social choice before (to the best of our knowledge), and so we view these connections as a central part of our technical contribution.

At the end of the paper, we test the utility of our schemes on real-world preference data from PrefLib. Conveniently, given a profile of rankings, we can simulate query answers from each voter, and thus easily count how many queries we would have asked had the rankings be elicited by our method. In our experiments, we put voters in a random sequence, and elicit one-by-one. Before eliciting the next ranking, we fit a Mallows or Plackett–Luce model to the set of rankings we have already elicited, and use the resulting model to guide the next elicitation. We find that this method saves 10.1% of queries on average over all datasets compared to eliciting with an uninformative prior (impartial culture).

Related Work Average-case sorting has predominantly been studied with respect to the uniform distribution over all rankings. The canonical treatment is Knuth's (1998, Section 5.3.1). The main focus is to find exactly optimum sorting algorithm for small m, which are now known up to $m = 10$ (Knuth 1998; Césary 1968; Kollár 1986; AbouEisha, Chikalov, and Moshkov 2016). Finding optimal algorithms for other distributions and small m could be interesting.

The field of *adaptive sorting* designs sorting algorithms with good worst-case performance under the assumption that the input string is partially "presorted". The literature is surveyed by Estivill-Castro and Wood (1992), and has studied a large variety of notions of presortedness. For most of these notions, algorithms are known whose worst-case performance asymptotically matches the information-theoretic lower bound, and some of the algorithms are universal in the sense of matching the lower bound for several types of presorted input. Using adaptive sorting algorithms for the Kendall-tau distance, we can obtain an alternative proof that a Mallows model can be elicited in $O(m)$ queries: One can show that the expected Kendall-tau distance to the reference ranking of a Mallows model is $O(m)$ (Fligner and Verducci 1986, Eq. 3.2), and there are many sorting algorithms whose number of queries does not exceed $O(m + \mathrm{KT})$, where KT is the distance to being sorted. However, adaptive sorting algorithms are not immediately useful for richer classes of random noise models, such as mixtures of Mallows models or Plackett–Luce models.

Surprisingly little work in computational social choice has explicitly studied preference elicitation via pairwise comparisons. An influential counterexample is a paper by Conitzer (2009), which shows that one can beat $O(m \log m)$ queries if preferences are known to have additional structure. Specifically, Conitzer (2009) shows that when preferences are single-peaked, there exists an adaptive elicitation protocol that requires $O(m)$ queries in the worst case; under certain conditions $O(m)$ queries suffice even if one does not know the underlying positions of alternatives. When preferences are known to be one-dimensional Euclidean (a stronger assumption than single-peakedness), Conitzer (2009) shows that $O(\log m)$ queries are enough, since one can find the voter's position by a binary search. Jamieson and Nowak (2011) show that one can elicit d-dimensional Euclidean preferences in $O(d \log m)$ queries on average, under a uniform distribution. Worst-case query complexity results have been obtained for several other kinds of structured preferences (Dey and Misra 2016a,b; see also Elkind, Lackner, and Peters 2017).

If preferences of several decision-makers are elicited with the goal of using a voting rule to make a decision, we may not need to elicit every voter's entire preference ranking. Thus, fewer than $O(nm \log m)$ queries may suffice. For example, in a plurality election we only need to know each voter's top choice, doable in $O(nm)$ queries. Boutilier and Rosenschein (2016) survey work on various elicitation schemes, often using queries other than pairwise comparisons. Using communication complexity, Conitzer and Sandholm (2005) give lower bounds for many voting rules on the number of queries required by any elicitation scheme. They also give a protocol using pairwise comparison queries to find a Condorcet winner in $O(nm)$ queries (see also Procaccia 2008).

In statistics and machine learning there is a significant body of work on learning and estimation from pairwise comparisons; see, for example, the surveys by Marden (1996), Catelan (2012) and Xia (2019). With few exceptions (Xia 2019, Chapter 6), this literature focuses on estimating a ground-truth model (parametric or non-parametric) using given non-adaptive samples, with the goal of having the estimate converge to the ground truth as quickly as possible. By contrast, we aim to precisely recover the ground-truth ranking using adaptive queries.

2 Distribution-Aware Insertion Sort

We begin our discussion by describing a variant of insertion sort due to Moran and Yehudayoff (2016) which can be instantiated for any distribution over rankings, and almost achieves the information-theoretic lower bound.

We write $[i] = \{1, \dots, i\}$. Let $A = \{c_1, \dots, c_m\}$ be a set of alternatives, and write $A!$ for the set of rankings of A. Given a ranking, we write $c_i \succ c_j$ if c_i is ranked higher than c_j. Let $\mathcal{D}$ be a probability distribution over $A!$. Now consider Algorithm 1. Its task is to recover an unknown ranking $\sigma \in A!$ by asking pairwise comparison queries, i.e. queries of the form "is c_i ranked higher than c_j in σ?". The algorithm builds up the correct ranking step-by-step: after having found the correct ranking of the alternatives $c_1, \dots, c_i$, it then determines where c_{i+1} needs to be inserted. It discovers this position using a binary search over the possible positions, but this binary search is biased according to the distribution $\mathcal{D}$. Specifically, the algorithm calculates, for each of the $i+1$ possible insertion positions of c_{i+1}, the probability that $\mathcal{D}$ assigns to this position conditional on the already-discovered ranking of $c_1, \dots, c_i$. It then partitions the interval $[0, 1]$ into $i+1$ pieces whose lengths correspond to those probabilities, and then runs binary search on $[0, 1]$.

Let $H(\mathcal{D}) = -\sum_{\sigma \in A!} \mathcal{D}(\sigma) \log \mathcal{D}(\sigma)$ be the Shannon entropy of $\mathcal{D}$. (All logarithms in this paper are base 2.) As is well-known, no algorithm asking only binary queries can succeed using fewer than $H(\mathcal{D})$ queries in expectation (even allowing fully general binary queries, not just pairwise comparisons). Remarkably, Algorithm 1 matches this lower bound up to an additive term of $2(m-1)$.

Theorem 1 (Moran and Yehudayoff 2016)**.** *Suppose a ranking σ is drawn from $\mathcal{D}$. Algorithm 1 correctly elicits σ, using at most $H(\mathcal{D}) + 2(m-1)$ queries in expectation.*

Since there are distributions $\mathcal{D}$ over $A!$ with $H(\mathcal{D}) = \Theta(m \log m)$ (for example the uniform distribution), this additive term is quite small, and we will see later that Algorithm 1 comes very close to the $H(\mathcal{D})$ optimum for the distributions we care about here. The proof idea behind Theorem 1 is that $H(\mathcal{D})$ is the sum of the entropies of the conditional distributions over insertion positions. The probability-based binary search on distribution $\mathbf{p}$ needs at most $H(\mathbf{p}) + 2$ queries. As we run the binary search $m-1$ times, the result follows.

In order to apply Algorithm 1 in our setting, there are two things we need to do. First, to understand the algorithm's guarantees we must bound the entropy of the distributions we are interested in. Second, we need some way of calculating the conditional insertion probabilities for a given distribution. The latter task can always be done by iterating through all $m!$ rankings and filtering out those that are compatible with the comparisons we have already elicited. However, this is extremely time-consuming, so it will be important to identify more computationally efficient procedures for this task that work for specific families of distributions. The two tasks will be our focus in the next sections.

Algorithm 1 Distribution-Aware Insertion Sort

$\sigma \leftarrow (c_1)$, initialize sorted sequence
for $i = 2, \dots, m$ **do**
  // insert c_i into σ
  For $j \in [i]$, calculate the probability p_j that c_i needs to be inserted in position j, by conditioning $\mathcal{D}$ on σ
  For $j \in [i]$, let $m_j = \sum_{t=1}^{j-1} p_t + p_j/2$
  $L \leftarrow 0, R \leftarrow 1$
  while $|\{j \in [i] : L \leqslant m_j \leqslant R\}| > 1$ **do**
    $M \leftarrow (L+R)/2$, $j^* \leftarrow \max\{j \in [i] : m_j \leqslant M\}$
    query c_i vs. c_{j^*}
    if $c_i \succ c_{j^*}$ **then**
      $R \leftarrow M$
    else
      $L \leftarrow M$
  Insert c_i into σ so that c_i is in position j, where $j \in [i]$ is the unique choice for which $L \leqslant m_j \leqslant R$
return σ

3 Mallows ϕ Model

Mallows (1957) introduced an influential family of distributions over rankings, one that is particularly appropriate in cases where there exists an underlying ground truth. A Mallows model (aka Mallows ϕ model) is specified by a *reference ranking*, which we take as $c_1 \succ c_2 \succ \cdots \succ c_m$ for convenience, and a *dispersion parameter* $\phi \in (0, 1]$. For a given ranking $\sigma \in A!$, we write $\mathrm{KT}(\sigma) = |\{c_i, c_j \in A : i < j \text{ but } c_j \succ_\sigma c_i\}|$ for the Kendall-tau distance between σ and the reference ranking (that is, the number of alternative pairs on which the two rankings disagree). Then the Mallows model $\mathcal{D}_\phi$ is the distribution with

$$\mathcal{D}_\phi(\sigma) = \tfrac{1}{Z}\phi^{\mathrm{KT}(\sigma)} \quad \text{for every } \sigma \in A!,$$

where $Z = (1+\phi)(1+\phi+\phi^2)\cdots(1+\phi+\cdots+\phi^{m-1})$ is a normalization constant. Thus, the probability of seeing a ranking σ in the Mallows model is exponentially decreasing with its Kendall-tau distance to the reference ranking, which in itself is the most likely ranking. For smaller ϕ, the model is more concentrated around the reference, and for $\phi = 1$, the Mallows model becomes the uniform distribution over $A!$.

It turns out that the Mallows model is a particularly good fit for the insertion sort algorithm we discussed, because a closed form for the conditional insertion probabilities is known. Doignon, Pekeč, and Regenwetter (2004) established that every Mallows model can be described as a *Repeated Insertion Model*, which in particular means that if in Algorithm 1 we insert the alternatives in order of the reference ranking (c_1 then c_2 then c_3 etc.), then the insertion probabilities for c_{i+1} are *independent* of the ordering of previous alternatives! Indeed, the probabilities are

$$p_1 = \tfrac{1}{W}\phi^i, p_2 = \tfrac{1}{W}\phi^{i-1}, \dots, p_{i+1} = \tfrac{1}{W}\phi^0,$$

where $W = 1 + \phi + \cdots + \phi^i$ is a normalization constant.

This formula not only allows us to run Algorithm 1 very efficiently, but it also allows us to better estimate the expected number of comparison queries, by estimating the entropy of a Mallows model. The first step is to estimate the entropy of the insertion probabilities above. For notational convenience, for a vector $\mathbf{x} \in \mathbb{R}^n_{\geqslant 0}$, let us write $H(\mathbf{x})$ for the Shannon entropy of the normalized vector $\mathbf{x}/\|\mathbf{x}\|$, where $\|\mathbf{x}\|$ is the

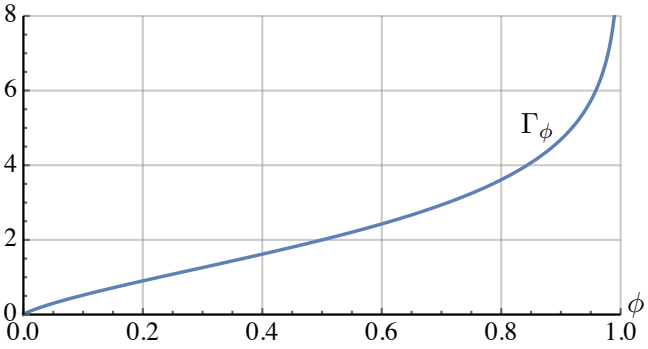

Figure 1: The function Γ_ϕ, defined in Lemma 2, which is an upper bound on the per-alternative entropy of a Mallows model with dispersion ϕ. It diverges to infinity as $\phi \to 1$.

sum of the entries of $\mathbf{x}$. The following result can be obtained using well-known formulas for geometric progressions.

Lemma 2. *Let $\phi \in (0,1)$ and let $\mathbf{x} = (1, \phi, \phi^2, \ldots, \phi^{j-1})$ for some j. Then*

$$H(\mathbf{x}) = \log\left(\frac{1-\phi^j}{1-\phi}\right) - \left(\frac{\phi}{1-\phi} - \frac{j\phi^j}{1-\phi^j}\right)\log(\phi)$$
$$\leqslant \frac{1}{1-\phi}\, H(\phi, 1-\phi) =: \Gamma_\phi.$$

Proof. The following identity is well-known:

$$1 + \phi + \cdots + \phi^{j-1} = \frac{1-\phi^j}{1-\phi}. \tag{1}$$

Differentiating and multiplying by ϕ, we deduce

$$\phi + 2\phi^2 + \cdots + (j-1)\phi^{j-1} = \frac{(j-1)\phi^{j+1} - j\phi^j + \phi}{(1-\phi)^2}. \tag{2}$$

Now, using properties of logarithms,

$$H(\mathbf{x}) = -\sum_{i=0}^{j-1} \frac{\phi^i}{\frac{1-\phi^j}{1-\phi}} \log\left(\frac{\phi^i}{\frac{1-\phi^j}{1-\phi}}\right)$$
$$= \log\left(\frac{1-\phi^j}{1-\phi}\right) - \frac{(1-\phi)}{1-\phi^j}\sum_{i=0}^{j-1} \phi^i \log\left(\phi^i\right)$$
$$= \log\left(\frac{1-\phi^j}{1-\phi}\right) - \frac{(1-\phi)\log(\phi)}{1-\phi^j}\sum_{i=0}^{j-1} i \cdot \phi^i$$
$$\overset{(2)}{=} \log\left(\frac{1-\phi^j}{1-\phi}\right) - \frac{(j-1)\phi^{j+1} - j\phi^j + \phi}{(1-\phi)(1-\phi^j)}\log(\phi)$$
$$= \log\left(\frac{1-\phi^j}{1-\phi}\right) - \left(\frac{\phi}{1-\phi} - \frac{j\phi^j}{1-\phi^j}\right)\log(\phi).$$

Next, we check that this value is increasing as j increases. Clearly, the first summand is increasing (see (1)). Since $\log(\phi)$ is negative, we now need to check that

$$\frac{\phi}{1-\phi} - \frac{j\phi^{j+1}}{1-\phi^{j+1}} \geqslant \frac{\phi}{1-\phi} - \frac{j\phi^j}{1-\phi^j},$$

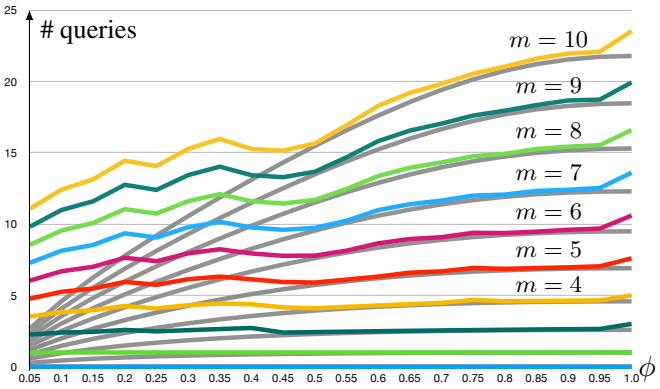

Figure 2: For each $m = 1, \ldots, 10$ and each $\phi \in (0,1)$ (in multiples of 0.05), we evaluate the number of queries used by Algorithm 1 when tested on rankings sampled from $\mathcal{D}_\phi$; these are plotted as colored lines. For each of these parameter values, we also calculate the entropy $H(\mathcal{D}_\phi)$ and plot is as a gray line (which is a lower bound for the colored line).

which follows because $x \mapsto \frac{x}{1-x}$ is increasing on $(0,1)$, its derivative being $\frac{1}{(1-x)^2} > 0$. The bound $H(\mathbf{x}) \leqslant \Gamma_\phi$ now follows by letting $j \to \infty$ in our expression for $H(\mathbf{x})$. □

The upper bound Γ_ϕ is plotted in Figure 1. For $\phi = 1$, we have $H(\mathbf{x}) = \log j$, and thus there cannot be a constant upper bound, hence $\Gamma_\phi \to \infty$ as $\phi \to 1$. One can check that $\Gamma_{0.5} = 2$ exactly, and as we will see in Figure 2 below, there appears to be a regime change at $\phi = 0.5$ in the performance of insertion sort on Mallows models.

As developed in the paper of Moran and Yehudayoff (2016), the entropy of a distribution over rankings is just the sum over i of the conditional insertion probabilities at step i. Hence, Lemma 2 immediately implies an estimate for the entropy of a Mallows model. In particular, if we treat $\phi < 1$ as a constant, then its entropy is linear in m.

Theorem 3. *Let $\mathcal{D}_\phi$ be Mallows distribution with $\phi \in (0,1)$. Then $H(\mathcal{D}_\phi) \leqslant \Gamma_\phi \cdot m = O(m)$.*

Figure 2 shows the actual performance of Algorithm 1 when run using a Mallows model $\mathcal{D}_\phi$ and evaluated by repeatedly sampling rankings from $\mathcal{D}_\phi$ and calculating the average number of queries. The figure shows that for $\phi \geqslant 0.5$, Algorithm 1 gets extremely close to the entropy lower bound, except for a sudden uptick at $\phi = 1$. For $\phi \leqslant 0.5$, the gap to entropy grows; one can check that as $\phi \to 0$, Algorithm 1 makes about m queries, while the entropy goes to 0.

4 Mixture of Mallows Models

The Mallows model is concentrated around a single reference ranking, which is a somewhat plausible assumption when there is a ground truth ranking which the voter has noisy access to. But there is a much broader range of situations where voters come in different types, such that different reference rankings are appropriate for the different types. For example, in a polarized political election, there might be a

typical ranking of the candidates for a right-wing voter, and a different typical ranking for a left-wing voter.

To capture these situations, it makes sense to consider a *mixture* of Mallows models. Suppose that $\mathcal{D}_1, \ldots, \mathcal{D}_k$ are Mallows models with reference rankings $\sigma_1, \ldots, \sigma_k$ and dispersion parameters $\phi_1, \ldots, \phi_k$. Let $\alpha_1, \ldots, \alpha_k \geqslant 0$ with $\sum_{i=1}^k \alpha_i = 1$ be weights. Then $\mathcal{D} = \alpha_1\mathcal{D}_1 + \cdots + \alpha_k\mathcal{D}_k$ is a mixture of Mallows models.

Such mixtures are often used to generate synthetic datasets for experimental evaluation of preference-based systems. They are also used as a flexible model for learning to rank (Lu and Boutilier 2014; Awasthi et al. 2014; Chierichetti et al. 2015; Liu and Moitra 2018).

To estimate the entropy of a mixture of Mallows model, we need to recall some elementary properties of entropy. An intuitive statement of the first property is via a two-step random processes. Suppose we first flip a biased coin with probability p of coming up heads. If it comes up heads, we then take a sample from distribution $\mathbf{q}_1$; otherwise we take a sample from distribution $\mathbf{q}_2$. We assume that $\mathbf{q}_1$ and $\mathbf{q}_2$ are defined on disjoint supports. What is the entropy of this process? It is $H(p, 1-p) + pH(\mathbf{q}_1) + (1-p)H(\mathbf{q}_2)$, that is, the entropy of the coin flip, plus the p-weighted average of the entropies of $\mathbf{q}_1$ and $\mathbf{q}_2$. Here is a formal and more general statement for k-sided coins.

Fact 4. *Let $\mathbf{x}_1, \ldots, \mathbf{x}_k$ be vectors of non-negative numbers, and let $\mathbf{x}$ be the concatenation of these vectors. Then*

$$H(\mathbf{x}) = H(\|\mathbf{x}_1\|, \ldots, \|\mathbf{x}_k\|) + \sum_{i=1}^{k} \frac{\|\mathbf{x}_i\|}{\|\mathbf{x}\|} H(\mathbf{x}_i).$$

This property was one of the defining characteristics of the entropy function in the derivation of Shannon (1948); see also Ash (1965, pp. 5–8) who calls it the *grouping axiom*.

A second intuitive fact says that entropy decreases if we merge outcomes: if we throw away information, entropy ought to go down. The formal statement below follows immediately from the concavity of $\log$.

Fact 5. *Let $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n_{\geqslant 0}$, and consider $\mathbf{x}' = (x_1 + x_2, x_3, \ldots, x_n)$. Then $H(\mathbf{x}) \geqslant H(\mathbf{x}')$.*

Using these two facts, we can show that a mixture of k distributions has entropy that is at most $\log k$ higher than the highest-entropy component. Thus, we can bound the entropy of a mixture of Mallows models.

Theorem 6. *Let $\mathcal{D} = \alpha_1\mathcal{D}_1 + \cdots + \alpha_k\mathcal{D}_k$ be a mixture of k Mallows models, so $\sum_{i=1}^k \alpha_i = 1$ and each $\mathcal{D}_i$ is a Mallows distribution with some reference ranking σ_i and dispersion parameter $\phi < 1$. Then $H(\mathcal{D}) \leqslant \log k + O(m)$.*

Proof. Consider the distribution $\mathcal{D}^+$ over $[k] \times A!$ defined by $\mathcal{D}^+(i, \sigma) = \alpha_i \cdot \mathcal{D}_i(\sigma)$. Thus $\mathcal{D}^+$ is like the mixture distribution $\mathcal{D}$, except that we are told the extra information from which $\mathcal{D}_i$ the ranking was drawn. From Fact 5, we know that $H(\mathcal{D}) \leqslant H(\mathcal{D}^+)$. But from Fact 4, we see that $H(\mathcal{D}^+) = H(\alpha_1, \ldots, \alpha_k) + \sum_{i=1}^k \alpha_i H(\mathcal{D}_i) \leqslant \log k + O(m)$, where the inequality follows because the maximum entropy of a length-k vector is $\log k$ and from Theorem 3. □

This result is very encouraging. It implies that Algorithm 1 can elicit a ranking drawn from a mixture model almost as cheaply as from a single Mallows model. Strikingly, even if we mix 2^m different Mallows models (with bounded ϕ parameters), a linear number of queries is still enough.

An interesting example of such a mixture arises from single-peaked preferences. Suppose we know that the alternative set has a natural one-dimensional structure, so we expect rankings to be single-peaked. There may be slight deviations from single-peakedness, but it is likely that the ranking is within a few swaps of being single-peaked. Since there are only 2^{m-1} rankings that are single-peaked, we can model this by mixing 2^{m-1} Mallows models, where each single-peaked ranking is the reference ranking of one of these models. Even after allowing noise around single-peakedness, the entropy estimate suggests we can elicit in $O(m)$ comparison queries, asymptotically identical to the $O(m)$ query elicitation procedure of Conitzer (2009) for noiseless single-peaked rankings, though the latter bound holds even in the worst case.

However, there is an obstacle to implementing the promise of Theorem 6: a mixture of Mallows models is not a Random Insertion Model anymore, and we do not have a closed form for the conditional insertion probabilities. Suppose Algorithm 1 has inserted $c_1, \ldots, c_{i-1}$, and we now need to calculate the conditional insertion probabilities for c_i, which is obtained by mixing the insertion probabilities of the Mallows components. The reference rankings of most of these components will not start with $c_1, \ldots, c_{i-1}$, and hence the insertion probabilities $1, \phi, \ldots, \phi^{i-1}$ are not valid.

Here is a way to obtain the correct probabilities. For a partial linear order of the alternatives $c_{i_1} \succ \cdots \succ c_{i_r}$, suppose we had a way to compute the probability

$$\Pr_{\mathcal{D}_\phi}(\succ) = \sum_{\sigma \in A! \,:\, \sigma \text{ extends } \succ} \mathcal{D}_\phi(\sigma).$$

Then we could generate all $i+1$ partial linear order obtained by inserting c_i into the already-ranked alternatives $c_1 \ldots, c_{i-1}$, and compute their probabilities, then normalize. Thus, we have obtained conditional insertion probabilities. We would do this for each Mallows model in the mixture, then take a weighted average.

Unfortunately, calculating the probability a Mallows model assigns to an arbitrary partial order is #P-hard (Lu and Boutilier 2014, Prop. 10). While there are known polynomial-time solvable classes of partial orders (Kenig et al. 2018), partial linear orders are not among them, so it is unclear how to run Algorithm 1 for Mallows mixtures.

A naïve way to sidestep this problem is to just elicit each of the Mallows components separately: run k instantiations of Algorithm 1 simultaneously, one for each component of the mixture, by interleaving the operations of the algorithm runs. In this way, by the analysis of Section 3, we can elicit a Mallows mixture in $O(km)$ queries, which is linear if we treat k as a constant.

However, we can do better, and match the promise of Theorem 6 in practice. Specifically, we can adapt some techniques of Lu and Boutilier (2014) to get a good estimate of the conditional insertion probabilities by using an efficient

Algorithm 2 Approximate Mallows Posterior (AMP)

Input: a Mallows model with parameter ϕ, a partial order $\succ$
Output: a random ranking ρ that extends $\succ$

$\sigma \leftarrow (c_1)$
multiplier $\leftarrow 1$
for $i = 2, \ldots, m$ **do**
  Determine all positions $j_1, \ldots, j_\ell \in [i]$ into which c_i could be inserted while keeping σ compatible with $\succ$
  Choose one of these positions j^* at random, where j_t should be chosen with probability proportional to ϕ^{i-j_t}
  Insert c_i into position j^* in σ
  multiplier $\leftarrow$ multiplier $\cdot \frac{\phi^{j_1}+\cdots+\phi^{j_\ell}}{1+\phi+\cdots+\phi^{i-1}}$
return σ, multiplier

sampler. Their Approximate Mallows Posterior (AMP) Algorithm operates on a Mallows model with reference ranking $c_1 \succ \cdots \succ c_m$ and dispersion parameter ϕ, and is given a partial order $\succ$ on A. The algorithm returns a random ranking that extends $\succ$ (i.e., whenever $a \succ b$ then a is ranked higher than b in the returned ranking). The algorithm is designed so that a ranking extending $\succ$ is returned with approximately the same probability as it would be returned by the Mallows model, conditioned on $\succ$.

We now discuss the AMP algorithm in detail, and show how it can be adjusted to output probability estimates. Using this algorithm, we can approximate the required insertion probabilities for each Mallows component. Additionally we can use it to calculate the likelihoods needed to find posteriors for the weights $\alpha_1, \ldots, \alpha_k$ for mixing the insertion probabilities.

The AMP algorithm is simple: it builds the output ranking up alternative-by-alternative, at each step inserting alternatives into positions with probabilities as suggested by the Random Insertion Model formulation of Mallows models. However, it restricts (conditions) this distribution over insertion positions to only those positions that do not conflict with the partial order $\succ$. Because partial orders are transitive, it is easy to check that the set of allowed positions is always an interval, and that all rankings consistent with $\succ$ can be obtained in this manner.

Unfortunately, the sampler does not output rankings with the same probabilities as a conditioned Mallows model.

Example (Lu and Boutilier 2014, Example 2)**.** *Suppose $\phi = 1$ (so the Mallows model is a uniform distribution over $A!$), and let $\succ$ be the partial linear order $c_2 \succ c_3 \succ \cdots \succ c_m$. The Mallows model conditional on $\succ$ is the uniform distribution over the m ways to insert c_1 into $\succ$, but Algorithm 2 places c_1 in position i with probability $1/2^i$ and position m with probability $1/2^{m+1}$.*

Conveniently, one can write down a formula for the factor by which the probability of a ranking produced by the AMP algorithm differs from the true Mallows probability. Algorithm 2 calculates this as the multiplier. It is obtained by keeping track of the relative probability mass of the allowed insertion positions at each step.

We can now see how to use Algorithm 2 to approximate the conditional insertion probabilities we need to run Algorithm 1. We initialize $q_1, \ldots, q_i \leftarrow 0$. We run the AMP algorithm many times, conditioned on the partial linear order on $\{c_1, \ldots, c_{i-1}\}$ that we have already elicited. For each output σ, multiplier), we make a note of the position t of c_i among $\{c_1, \ldots, c_{i-1}\}$ in the ranking σ, and set $q_t \leftarrow q_t +$ multiplier. By our discussion, once we normalize $q_1, \ldots, q_i$, we obtain an approximation of the conditional insertion probabilities.

Lu and Boutilier (2014) report that in experiments, the AMP algorithm performs very well, and in our version with the multiplier, it converges to the correct distribution with sufficiently many samples. The example above shows that exponentially many samples may be needed in the worst case; but this should not be a problem in practice. In particular, the experimental results of Lu and Boutilier (2014) suggest that the AMP algorithm will perform much better than naïve rejection sampling. In a real-world implementation, we could run the AMP algorithm while waiting for the user to reply to our query, and interrupt the sampling when we need to produce the next query.

5 Plackett–Luce Models

Another popular ranking model is the Plackett–Luce Model, introduced by Plackett (1975) and Luce (1977), but going back as far as Zermelo (1929) and Bradley and Terry (1952). Xia (2019) provides an excellent overview of recent work on this model, with a focus on learning its parameters.

In the Plackett–Luce model, each alternative c_i is assigned a weight w_i. Given these weights, a ranking is constructed using the following random process: repeatedly and without replacement, draw an alternative that is not yet ranked, with probability proportional to its weight, and place it at the bottom of the partial ranking. Hence, for the weight vector $\mathbf{w} = (w_1, \ldots, w_m)$, Plackett–Luce induces distribution $\mathcal{D}_\mathbf{w}$, where for each $\sigma \in A!$ with $\sigma = c_{i_1} \succ \cdots \succ c_{i_m}$ we have

$$\mathcal{D}_\mathbf{w}(\sigma) = \frac{w_{i_1}}{w_{i_2} + \cdots + w_{i_m}} \cdot \frac{w_{i_2}}{w_{i_3} + \cdots + w_{i_m}} \cdots \frac{w_{i_m}}{w_{i_m}}.$$

A useful property of the Plackett–Luce model is that it satisfies what is known as *Luce's choice axiom*.

Fact 7. *Consider a Plackett–Luce model $\mathcal{D}_\mathbf{w}$ with weights $\mathbf{w}$. Let $A' \subseteq A$. Then the induced distribution on rankings of A' is a Plackett–Luce model with weights $\mathbf{w}|_{A'}$.*

Luce's choice axiom gives us a simple way to calculate conditional insertion probabilities. Suppose we have already placed $c_1, \ldots, c_{i-1}$, and now need to place c_i. By Luce's choice axiom, the induced ranking on $\{c_1, \ldots, c_i\}$ itself follows a Plackett–Luce distribution. For the $i+1$ possible rankings obtained by inserting c_i, we can calculate their probability in the restricted Plackett–Luce model, and then just normalize these probabilities to obtain insertion probabilities.

Let us now try to estimate the entropy of Plackett–Luce models for some classes of weights. Normalize $\mathbf{w}$ so that $\sum_{i=1}^m w_i = 1$. We can directly apply Fact 4 to the Plackett–Luce process, and find that

$$H(\mathcal{D}_\mathbf{w}) = H(\mathbf{w}) + \textstyle\sum_{i=1}^m w_i H(\mathcal{D}_{\mathbf{w}|_{A\setminus\{c_i\}}}). \qquad (3)$$

For example, from this formula, it immediately follows that if $\mathbf{w} = (1, 1, \ldots, 1)$, then $H(\mathcal{D}_\mathbf{w}) = \Theta(m \log m)$ (as expected

since $\mathcal{D}_{\mathbf{w}}$ is in fact the uniform distribution over A!). It turns out that the entropy of the Plackett–Luce model stays high even for weights that are linearly increasing, such as $\mathbf{w} = (1, 2, \ldots, m)$. First, we need a lemma.

Lemma 8. *Let $\mathbf{w} \in \mathbb{R}^n_{\geqslant 0}$ satisfy $\max \mathbf{w} \leqslant C \cdot \min \mathbf{w}$ for some constant $C \geqslant 1$, then $H(\mathbf{w}) \geqslant \frac{1}{C} \log \frac{n}{C} = \Omega(\log n)$. Moreover, for such $\mathbf{w}$, $H(\mathcal{D}_{\mathbf{w}}) = \Omega(m \log m)$.*

Proof. Note that $\frac{1}{C} n \cdot \max \mathbf{w} \leqslant \|\mathbf{w}\| \leqslant nC \cdot \min \mathbf{w}$. Now

$$H(\mathbf{w}) = \sum_i \frac{w_i}{\|\mathbf{w}\|} \log \frac{\|\mathbf{w}\|}{w_i}$$
$$\geqslant \sum_i \frac{\min \mathbf{w}}{nC \cdot \min \mathbf{w}} \log \frac{\frac{1}{C} n \cdot \max \mathbf{w}}{\max \mathbf{w}} = \frac{1}{C} \log \frac{n}{C},$$

which proves the first part. Next, note that if $\mathbf{w}$ satisfies the condition of the statement, then so does any vector obtained from $\mathbf{w}$ by deleting elements (since deleting elements decreases the maximum and increases the minimum). Thus, we can apply (3) inductively, and find that $H(\mathcal{D}_{\mathbf{w}}) \geqslant \sum_{j=1}^{m} \Omega(\log(j)) = \Omega(\log(m!))$. □

Theorem 9. *Let $\mathbf{w} = (1, 2, \ldots, m)$. Then $H(\mathcal{D}_{\mathbf{w}}) = \Omega(m \log m)$.*

Proof. Consider the distribution of the middle half of alternatives. By Luce's choice axiom, this is a Plackett–Luce distribution on $\mathbf{w}' = (\lceil \frac{1}{4} m \rceil, \ldots, \lfloor \frac{3}{4} m \rfloor)$. By Lemma 8 with $C = 3$, $H(\mu_{\mathbf{w}'}) = \Omega(m \log m)$. By Fact 5, throwing away information decreases entropy, and so $H(\mathcal{D}_{\mathbf{w}}) \geqslant H(\mathcal{D}_{\mathbf{w}'})$. □

The same proof applies to any $\mathbf{w}$ that is an arithmetic progression, since for $\mathbf{w} = (p+q, p+2q, \ldots, p+mq)$ we can again consider the middle half $\mathbf{w} = (p+\frac{1}{4}mq, \ldots, p+\frac{3}{4}mq)$ and apply Lemma 8 with $C = 3 + 4p/q$. Thus, to have any hope of finding Plackett–Luce models with lower entropy (and therefore ones that can be elicited faster), we need to use weights $\mathbf{w}$ that grow faster, or in other words are more uneven. We can show that an exponentially increasing $\mathbf{w}$ gives rise to a Plackett–Luce model with $O(m)$ entropy. Again, we need to establish some properties of entropy first.

Lemma 10. *Consider $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n_{\geqslant 0}$ with $\|\mathbf{x}\| = 1$, and let $\mathbf{x}' = (x_2, \ldots, x_n)$. Suppose $y \in (0, 1)$ is such that $H(\mathbf{x}) \leqslant H(y, 1-y)/y$ and $x_1 \leqslant y$. Then $H(\mathbf{x}') \leqslant H(\mathbf{x})$.*

Proof. Using the grouping axiom (Fact 4), we have

$$H(\mathbf{x}) = H(x_1, 1 - x_1) + (1 - x_1) H(\mathbf{x}').$$

Rearranging,

$$H(\mathbf{x}') = \frac{H(x) - H(x_1, 1 - x_1)}{1 - x_1}.$$

The right-hand side is at most $H(\mathbf{x})$ iff $H(\mathbf{x}) \leqslant H(x_1, 1 - x_1)/x_1$. By our assumption, $H(\mathbf{x}) \leqslant H(y, 1-y)/y$. But the function $t \mapsto H(t, 1-t)/t$ is decreasing on $(0, 1)$, so if $x_1 \leqslant y_1$, then $H(\mathbf{x}) \leqslant H(y, 1-y)/y \leqslant H(x_1, 1-x_1)/x_1$. This implies $H(\mathbf{x}') \leqslant H(\mathbf{x})$, as desired. □

Lemma 11. *Let $\phi \in (0, \frac{1}{2}]$, and let $\mathbf{w} = (1, \phi, \ldots, \phi^j)$. Suppose $\mathbf{w}'$ is obtained from $\mathbf{w}$ by removing a number of entries. Then $H(\mathbf{w}') \leqslant H(\mathbf{w})$.*

Proof. From Lemma 2, $H(\mathbf{w}) \leqslant H(\phi, 1-\phi)/(1-\phi)$. Thus, by Lemma 10, removing any entry whose value is at most $1 - \phi$ from $\mathbf{w}$ decreases its entropy. However, since $\phi \leqslant \frac{1}{2}$, this is true for all entries of $\mathbf{w}$ except 1. Since entropy decreases, we can repeatedly apply Lemma 2 to find that for all sublists $\mathbf{w}'$ of $\mathbf{w}$ that contain 1, we have $H(\mathbf{w}') \leqslant H(\mathbf{w})$.

Now suppose that $\mathbf{w}' = (\phi^{i_1}, \ldots, \phi^{i_t})$ is a sublist that does not contain 1 (so $1 \leqslant i_1 < \cdots < i_t \leqslant j$). Notice that the function H satisfies $H(\alpha \mathbf{x}) = H(\mathbf{x})$ for any scalar $\alpha > 0$. Hence $H(\mathbf{w}') = H(\phi^{-i_1} \mathbf{w}') \leqslant H(\mathbf{w})$, where the inequality follows because $\phi^{-i_1} \mathbf{w}'$ is a sublist of $\mathbf{w}$ that contains 1. This proves the statement. □

We can now show that the entropy of a Plackett–Luce model is linear if each weight is at most half the next higher weight.

Theorem 12. *Let $\phi \in (0, \frac{1}{2}]$ be a constant, and let $\mathbf{w} = (1, \phi, \ldots, \phi^{m-1})$. Then $H(\mathcal{D}_{\mathbf{w}}) = O(m)$.*

Proof. From Lemma 2, $H(\mathbf{w}) \leqslant \Gamma_\phi$, a constant. From Lemma 11, all weight vectors that we will encounter in the Plackett–Luce process will have entropy at most Γ_ϕ. Applying (3) repeatedly, we find $H(\mathcal{D}_{\mathbf{w}}) \leqslant \Gamma_\phi \cdot m = O(m)$. □

We conjecture that the result remains true for all $\phi \in (0, 1)$.

6 Experiments

Suppose we need to elicit the preferences of many decision-makers, all over the same set of alternatives. This will be the case when our aim is to use a voting rule, or to otherwise aggregate the preferences. If we imagine that voters log on to a website to input their preferences, then we can model this as a sequential process where voters vote one after another.[1] If we plan to elicit rankings via adaptive pairwise comparisons, the sequential nature allows us to apply the results of this paper: After a voter finishes submitting preferences, we fit a preference model (such as Mallows) to the collection of all votes elicited thus far. When the next voter arrives, we elicit their vote using Algorithm 1 under the assumption that this vote is drawn from the model we have learned. In many cases (especially those with a ground truth), this will be a reasonable assumption, and we can thus expect to save on the number of queries by employing this procedure.

How can we estimate the magnitude of the speed-up in practice? Well, if we are given any dataset consisting of a collection of preference rankings, we can simulate the sequential voting process described above: We place the voters in a random order, and then elicit their preferences by simulating their responses to pairwise comparison queries, which we can do because we already know their entire ranking.

We apply this idea to all datasets from PrefLib (Mattei and Walsh 2013) that contain complete strict rankings. For each dataset, we remove all votes that contain indifferences, and use the resulting profile if the number of remaining votes is

[1] In practice, several voters may submit their votes simultaneously, but everything that follows can be adapted to allow for this.

Dataset	$\lvert A\rvert$	$\lvert N\rvert$	IC	Mallows	P–L
Dublin North	12	4259	30.98	27.99	28.38
Dublin West	9	4810	19.85	17.64	17.97
Meath	14	3166	38.95	35.56	35.84
Debian 2003	5	399	7.48	5.80	5.81
Debian 2005	7	377	13.50	10.56	10.92
Debian 2006	8	323	16.38	12.34	13.09
Debian 2007	9	335	19.84	17.53	17.24
Debian 2010	5	377	7.46	6.29	5.85
Burlington 2006	6	2603	10.69	7.71	7.92
Burlington 2009	6	2853	10.52	7.53	7.90
AGH 2003	9	146	19.91	17.05	17.75
AGH 2004	7	153	13.64	10.75	10.97
Sushi	10	5000	23.41	20.96	21.05
Aspen Mayor	5	1183	7.05	5.72	5.67
Jester Jokes	19	8169	59.47	55.03	54.55

Table 1: Performance of Algorithm 1 on estimated distributions on PrefLib datasets, showing the average number of queries per voter. Each row averages 10 runs with random voter orderings. IC = Impartial Culture, P–L = Plackett–Luce.

at least $10 \cdot |A|$ and if $|A| \geqslant 5$. (Thus, we skip small profiles and profiles with few rankings over large alternative spaces where we do not have a good opportunity to learn a model.) This leaves 68 preference profiles. In addition, we use the Jester dataset (Goldberg et al. 2001) of numerical ratings of jokes; we take the 19 jokes rated most frequently and obtain rankings from the 8169 users who rated all of them.

We make 10 copies of each dataset and in each copy we randomly shuffle its votes. We initialize $\mathcal{D}$ as the uniform distribution over $A!$. Then we elicit votes using Algorithm 1, and after each step update $\mathcal{D}$ with a maximum likelihood estimate (MLE) of either a Mallows model or a Plackett–Luce model.[2] We count the average number of queries per voter needed under either the Mallows or the Plackett–Luce regime, and compare it to the number of queries used by Algorithm 1 if run using the uniform distribution (impartial culture) throughout. For Plackett–Luce, since estimation of the MLE is very slow, we only elicit the first 250 votes in the first shuffle for our experiments. In each run, we elicit the first 10 voters using the uniform distribution and only start using the learned model for the 11th voter onwards, to avoid overfitting at the beginning.

Table 1 shows results for a selection of datasets. In all datasets, Mallows and Plackett–Luce outperform impartial culture. The two types of models perform roughly similarly well, though the Plackett–Luce data need to be interpreted with some care, since these are based on fewer data points due to computational limitations. Appendix A shows results for all datasets from PrefLib. There, we also give the standard deviation of the number of queries asked under Impartial Culture and under Mallows. We find that the number of queries varies more across voters under the Mallows scheme; intuitively, some voters are unlucky and have preferences far away from the reference ranking, and those voters need to answer many queries.

[2]We use Gurobi to calculate the Kemeny ranking needed for a Mallows MLE (Young 1995) using a standard ILP formulation (Conitzer, Davenport, and Kalagnanam 2006), and scipy to calculate the best-fit ϕ parameter. We calculate Plackett–Luce MLEs using the choix package (https://github.com/lucasmaystre/choix).

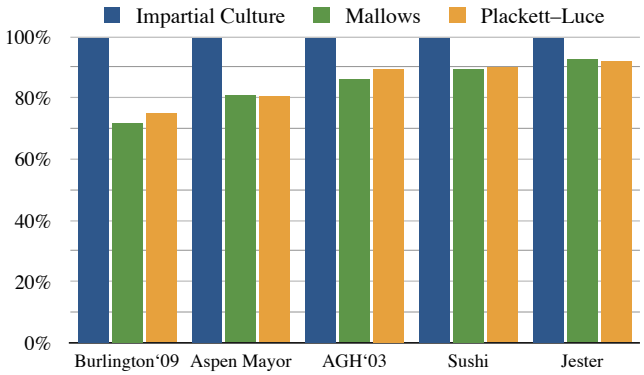

Figure 3: Chart of selected datasets, normalizing the average queries asked under impartial culture to 100%.

Figure 3 gives another view of the data, focusing on 5 representative datasets and showing the improvement compared to impartial culture, which we normalize to 100%. As this figure makes clear, our approach is more successful on some datasets than others, but this is to be expected. Indeed, real preferences and opinions do not necessarily conform to theoretical noise models (Mao, Procaccia, and Chen 2013); the better the fit, the greater the improvement we would expect from using our approach. While none of the datasets we consider has a ground truth, we would expect some datasets to exhibit a greater degree of consensus among voters and less polarization. Such datasets would have a good fit to a Mallows model with a low dispersion parameter or to a Plackett–Luce model with fast-increasing weights, and so, by our theoretical analysis, would lead to low query complexity. In Figure 4, we show the relationship between the dispersion of the MLE of Mallows and the number of queries we asked when eliciting based on the Mallows model. As expected, these are positively correlated.

7 Discussion

Our work draws connections between preference elicitation, average-case sorting, and learning to rank. Using our method, improvements in the latter fields can be used directly to improve preference elicitation. In particular, as we develop better ways to learn and predict users' preferences, preference elicitation can directly benefit from such advancements.

With any system that includes predictions about the user, there are worries that the user would "go along" with the prediction to save effort. In our case, it could become apparent from the queries what preferences the system is expecting us to have (e.g., because we insert alternatives in order of the learned reference ranking), and this could bias respondents to report preferences that are closer to the reference ranking than they would otherwise have reported. It would be worthwhile to study whether this effect occurs in a lab study.

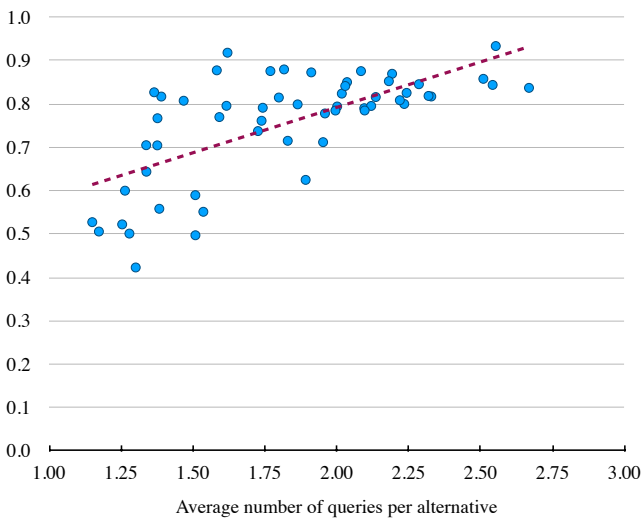

Figure 4: A scatterplot, where each dataset is shown as a point, where the vertical axis corresponds to the dispersion parameter ϕ of the MLE of a Mallows model fit to the entire dataset, and the horizontal axis shows the average number of queries per alternative that were asked during elicitation. The least-squares fit gives an R^2 of 0.447.

Technically, some open questions are raised by our work, such as a more complete characterization of the entropy of the Plackett–Luce model as a function of the weight vector, and the extension of our method to other preference distributions.

References

AbouEisha, H.; Chikalov, I.; and Moshkov, M. 2016. Decision trees with minimum average depth for sorting eight elements. *Discrete Applied Mathematics* 204: 203–207.

Ash, R. B. 1965. *Information Theory*. Interscience Tracts in Pure and Applied Mathematics. John Wiley & Sons.

Awasthi, P.; Blum, A.; Sheffet, O.; and Vijayaraghavan, A. 2014. Learning mixtures of ranking models. In *Advances in Neural Information Processing Systems*, 2609–2617.

Boutilier, C.; and Rosenschein, J. 2016. Incomplete Information and Communication in Voting. In Brandt, F.; Conitzer, V.; Endriss, U.; Lang, J.; and Procaccia, A. D., eds., *Handbook of Computational Social Choice*, chapter 10. Cambridge University Press.

Bradley, R. A.; and Terry, M. E. 1952. Rank analysis of incomplete block designs: I. The method of paired comparisons. *Biometrika* 39(3/4): 324–345.

Catelan, M. 2012. Models for paired comparison data: A review with emphasis on dependent data. *Statistical Science* 27(3): 412–433.

Césary, Y. 1968. Questionnaire, codage et tris.

Chierichetti, F.; Dasgupta, A.; Kumar, R.; and Lattanzi, S. 2015. On learning mixture models for permutations. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science (ITCS)*, 85–92.

Conitzer, V. 2009. Eliciting single-peaked preferences using comparison queries. *Journal of Artificial Intelligence Research* 35: 161–191.

Conitzer, V.; Davenport, A.; and Kalagnanam, J. 2006. Improved Bounds for Computing Kemeny Rankings. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI)*, 620–627.

Conitzer, V.; and Sandholm, T. 2005. Communication Complexity of Common Voting Rules. In *Proceedings of the 6th ACM Conference on Electronic Commerce (ACM EC)*, 78–87.

Dey, P.; and Misra, N. 2016a. Elicitation for Preferences Single Peaked on Trees. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*, 215–221.

Dey, P.; and Misra, N. 2016b. Preference Elicitation for Single Crossing Domain. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*, 222–228.

Doignon, J.-P.; Pekeč, A.; and Regenwetter, M. 2004. The repeated insertion model for rankings: Missing link between two subset choice models. *Psychometrika* 69(1): 33–54.

Elkind, E.; Lackner, M.; and Peters, D. 2017. Structured Preferences. In Endriss, U., ed., *Trends in Computational Social Choice*, chapter 10, 187–207. AI Access.

Estivill-Castro, V.; and Wood, D. 1992. A survey of adaptive sorting algorithms. *ACM Computing Surveys (CSUR)* 24(4): 441–476.

Fligner, M. A.; and Verducci, J. S. 1986. Distance based ranking models. *Journal of the Royal Statistical Society: Series B* 48(3): 359–369.

Freedman, R.; Borg, J. S.; Sinnott-Armstrong, W.; Dickerson, J. P.; and Conitzer, V. 2020. Adapting a kidney exchange algorithm to align with human values. *Artificial Intelligence* 283.

Goldberg, K.; Roeder, T.; Gupta, D.; and Perkins, C. 2001. Eigentaste: A constant time collaborative filtering algorithm. *information Retrieval* 4(2): 133–151.

Jamieson, K. G.; and Nowak, R. 2011. Active ranking using pairwise comparisons. In *Advances in Neural Information Processing Systems*, 2240–2248.

Kahng, A.; Lee, M. K.; Noothigattu, R.; Procaccia, A.; and Psomas, C.-A. 2019. Statistical foundations of virtual democracy. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 3173–3182.

Kenig, B.; Ilijasic, L.; Ping, H.; Kimelfeld, B.; and Stoyanovich, J. 2018. Probabilistic Inference Over Repeated Insertion Models. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI)*, 1897–1904.

Knuth, D. E. 1998. *The Art of Computer Programming. Volume 3: Sorting and Searching*. Addison–Wesley.

Kollár, L. 1986. Optimal sorting of seven element sets. In *International Symposium on Mathematical Foundations of Computer Science*, 449–457. Springer.

Lee, M. K.; Kusbit, D.; Kahng, A.; Kim, J. T.; Yuan, X.; Chan, A.; See, D.; Noothigattu, R.; Lee, S.; Psomas, A.; et al. 2019. WeBuildAI: Participatory framework for algorithmic governance. *Proceedings of the ACM on Human-Computer Interaction* CSCW: 1–35.

Liu, A.; and Moitra, A. 2018. Efficiently learning mixtures of Mallows models. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, 627–638. IEEE.

Lu, T.; and Boutilier, C. 2014. Effective sampling and learning for Mallows models with pairwise-preference data. *Journal of Machine Learning Research* 15: 3783–3829.

Luce, R. D. 1977. The choice axiom after twenty years. *Journal of Mathematical Psychology* 15(3): 215–233.

Mallows, C. L. 1957. Non-Null Ranking Models. *Biometrika* 44(1/2): 114–130.

Mao, A.; Procaccia, A. D.; and Chen, Y. 2013. Better Human Computation Through Principled Voting. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI)*, 1142–1148.

Marden, J. I. 1996. *Analyzing and Modeling Rank Data*. CRC Press.

Mattei, N.; and Walsh, T. 2013. PrefLib: A Library for Preference Data. In *Proceedings of the 3rd International Conference on Algorithmic Decision Theory (ADT)*, 259–270.

Moran, S.; and Yehudayoff, A. 2016. A note on average-case sorting. *Order* 33(1): 23–28.

Noothigattu, R.; Gaikwad, S.; Awad, E.; Dsouza, S.; Rahwan, I.; Ravikumar, P.; and Procaccia, A. 2018. A Voting-Based System for Ethical Decision Making. *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI)* 1587–1594.

Plackett, R. L. 1975. The Analysis of Permutations. *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 24(2): 193–202.

Procaccia, A. 2008. A Note on the Query Complexity of the Condorcet Winner. *Information Processing Letters* 108(6): 390–393.

Regenwetter, M.; Grofman, B.; Marley, A. A. J.; and Tsetlin, I. M. 2006. *Behavioral Social Choice: Probabilistic Models, Statistical Inference, and Applications*. Cambridge University Press.

Shannon, C. E. 1948. A mathematical theory of communication. *Bell System Technical Journal* 27(3): 379–423.

Xia, L. 2019. Learning and decision-making from rank data. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 13(1): 1–159.

Young, H. P. 1995. Optimal Voting Rules. *Journal of Economic Perspectives* 9(1): 51–64.

Zermelo, E. 1929. Die Berechnung der Turnier-Ergebnisse als ein Maximumproblem der Wahrscheinlichkeitsrechnung. *Mathematische Zeitschrift* 29(1): 436–460.

A Data on all PrefLib datasets

Here, we give a expanded version of Table 1, showing results for all PrefLib datasets meeting our criteria. In the table, datasets are identified by abbreviated PrefLib filenames (omitting redundant zeros). We also show standard deviations ("± 0.80") after averages obtained for Impartial Culture and Mallows. (For Plackett–Luce, we do not show standard deviations because this data is based on small runs only.) We see that the variance for elicitation using a Mallows prior is consistently higher than when using Impartial Culture. Intuitively, this is because some votes are far away from the predicted reference ranking, and those voters will need to answer more queries. One the other hand, for Impartial Culture, each candidate insertion runs a balanced binary search which always induces roughly the same number of queries.

Name	Filename	$\lvert A\rvert$	$\lvert N\rvert$	Impartial Culture		Mallows		Plackett–Luce
Dublin North	ED-01-01.toc	12	4259	30.98	±0.99	27.99	±2.44	1.67
Dublin West	ED-01-02.toc	9	4810	19.85	±0.64	17.64	±1.80	0.93
Meath	ED-01-03.toc	14	3166	38.95	±1.05	35.56	±3.22	2.83
Debian 2003	ED-02-02.toc	5	399	7.48	±0.50	5.80	±1.26	3.64
Debian 2005	ED-02-03.toc	7	377	13.50	±0.50	10.56	±2.43	7.24
Debian 2006	ED-02-04.toc	8	323	16.38	±0.48	12.34	±3.31	10.13
Debian 2007	ED-02-05.toc	9	335	19.84	±0.66	17.53	±2.12	12.87
Debian 2010	ED-02-06.toc	5	377	7.46	±0.50	6.29	±1.71	3.88
Burlington 2006	ED-05-01.toc	6	2603	10.69	±0.46	7.71	±1.83	0.76
Burlington 2009	ED-05-02.toc	6	2853	10.52	±0.50	7.53	±1.29	0.69
ERS	ED-07-09.toc	13	184	35.00	±1.05	33.22	±1.68	33.13
ERS	ED-07-10.toc	6	375	10.61	±0.49	9.56	±0.82	6.43
ERS	ED-07-11.toc	7	333	13.52	±0.50	12.07	±1.73	8.81
ERS	ED-07-13.toc	5	1809	7.63	±0.48	6.70	±1.11	0.93
ERS	ED-07-14.toc	8	271	16.55	±0.50	15.32	±1.39	14.08
ERS	ED-07-26.toc	5	50	7.61	±0.49	7.19	±0.85	7.24
ERS	ED-07-31.toc	10	626	23.32	±0.80	21.28	±2.02	8.47
ERS	ED-07-32.toc	13	149	34.98	±1.13	32.61	±3.00	32.94
ERS	ED-07-33.toc	16	320	46.64	±1.15	42.92	±3.93	33.84
ERS	ED-07-36.toc	10	131	23.55	±0.85	21.95	±1.63	21.95
ERS	ED-07-37.toc	10	175	23.55	±0.83	21.83	±1.63	22.05
ERS	ED-07-38.toc	11	163	27.06	±0.98	25.13	±2.11	25.56
ERS	ED-07-42.toc	7	71	13.66	±0.47	12.53	±1.55	12.82
ERS	ED-07-43.toc	7	298	13.56	±0.50	12.38	±1.07	10.44
ERS	ED-07-44.toc	9	170	20.00	±0.67	18.68	±1.38	18.72
ERS	ED-07-57.toc	8	229	16.61	±0.49	14.70	±2.18	14.79
ERS	ED-07-59.toc	5	207	7.56	±0.50	6.92	±0.84	6.93
ERS	ED-07-62.toc	5	130	7.64	±0.48	6.78	±1.50	6.72
ERS	ED-07-67.toc	7	215	13.58	±0.49	12.20	±1.39	12.39
ERS	ED-07-68.toc	11	217	26.87	±0.92	24.60	±2.25	24.51
ERS	ED-07-71.toc	7	222	13.55	±0.50	12.12	±1.47	12.04
ERS	ED-07-73.toc	5	139	7.49	±0.50	6.92	±0.92	6.88
ERS	ED-07-76.toc	5	110	7.53	±0.50	6.84	±1.25	6.83
ERS	ED-07-77.toc	12	1312	30.78	±0.99	27.98	±2.48	5.32
ERS	ED-07-83.toc	7	160	13.58	±0.49	12.58	±0.99	12.49
ERS	ED-07-84.toc	6	169	10.59	±0.49	9.72	±0.86	9.69
Glasgow	ED-08-01.toc	9	593	19.90	±0.64	18.19	±1.46	7.74
Glasgow	ED-08-02.toc	11	535	26.96	±0.98	24.65	±2.72	11.69
Glasgow	ED-08-03.toc	10	363	23.56	±0.86	21.67	±2.10	15.08
Glasgow	ED-08-04.toc	11	419	27.00	±0.98	24.81	±2.44	15.00
Glasgow	ED-08-05.toc	10	718	23.30	±0.89	21.45	±2.20	7.49
Glasgow	ED-08-06.toc	10	556	23.42	±0.87	21.22	±2.47	9.61
Glasgow	ED-08-07.toc	13	284	34.95	±1.08	31.80	±3.30	28.30
Glasgow	ED-08-08.toc	10	559	23.34	±0.89	20.98	±2.33	9.65
Glasgow	ED-08-09.toc	11	411	27.11	±0.96	24.74	±2.74	15.19
Glasgow	ED-08-10.toc	9	818	19.82	±0.68	17.98	±1.99	5.54

Name	Filename	$\|A\|$	$\|N\|$	Impartial Culture		Mallows		Plackett–Luce
Glasgow	ED-08-11.toc	10	630	23.53	±0.80	21.45	±1.74	8.47
Glasgow	ED-08-12.toc	8	1040	16.57	±0.49	14.86	±1.41	3.61
Glasgow	ED-08-13.toc	11	494	26.91	±0.97	24.84	±2.30	12.68
Glasgow	ED-08-14.toc	8	1071	16.61	±0.49	14.91	±1.26	3.52
Glasgow	ED-08-15.toc	9	648	19.83	±0.67	18.06	±1.90	7.04
Glasgow	ED-08-16.toc	10	690	23.33	±0.89	20.94	±2.63	7.79
Glasgow	ED-08-17.toc	9	962	19.77	±0.67	17.87	±1.81	4.66
Glasgow	ED-08-18.toc	9	767	19.90	±0.65	18.37	±1.55	6.05
Glasgow	ED-08-19.toc	11	405	26.90	±0.95	24.49	±2.91	15.15
Glasgow	ED-08-20.toc	9	726	19.84	±0.71	18.28	±1.67	6.35
Glasgow	ED-08-21.toc	10	365	23.52	±0.86	21.17	±2.38	14.83
AGH	ED-09-01.soc	9	146	19.91	±0.59	17.05	±3.13	17.75
AGH	ED-09-01.soc	9	146	19.90	±0.58	17.04	±3.11	17.75
AGH	ED-09-02.soc	7	153	13.64	±0.48	10.75	±2.71	10.86
AGH	ED-09-02.soc	7	153	13.67	±0.47	10.76	±2.71	11.07
NES	ED-13-07.toc	6	100	10.59	±0.49	9.70	±1.16	9.66
NES	ED-13-13.toc	6	105	10.52	±0.50	9.48	±1.47	9.41
NES	ED-13-14.toc	6	100	10.46	±0.50	8.38	±2.54	8.84
NES	ED-13-15.toc	5	156	7.42	±0.49	6.40	±1.92	6.44
NES	ED-13-17.toc	5	290	7.61	±0.49	6.91	±0.94	6.11
Sushi	ED-14-01.soc	10	5000	23.41	±0.82	20.96	±1.97	1.05
Aspen	ED-16-02.toc	5	1183	7.05	±0.22	5.72	±0.95	1.20