Learning Real-Life Approval Elections

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

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

Andrzej Kaczmarczyk
Department of Computer Science,
The University of Chicago
Chicago, USA
akaczmarczyk@uchicago.edu

Marcin Kurdziel
AGH University
Kraków, Poland
kurdziel@agh.edu.pl

Grzegorz Pierczyński
AGH University
Kraków, Poland
g.pierczynski@gmail.com

Stanisław Szufa
CNRS, LAMSADE, Université Paris
Dauphine – PSL
Paris, France
s.szufa@gmail.com

ABSTRACT

We study the independent approval model (IAM) for approval elections, where each candidate has its own approval probability and is approved independently of the other ones. This model generalizes, e.g., the impartial culture, the Hamming noise model, and the resampling model. We propose algorithms for learning IAMs and their mixtures from data, using either maximum likelihood estimation or Bayesian learning. We then apply these algorithms to a large set of elections from the Pabulib database. In particular, we find that single-component models are rarely sufficient to capture the complexity of real-life data, whereas their mixtures perform well.

KEYWORDS

elections; approvals; learning; Pabulib; MLE; Bayesian methods

ACM Reference Format:
Piotr Faliszewski, Łukasz Janeczko, Andrzej Kaczmarczyk, Marcin Kurdziel, Grzegorz Pierczyński, and Stanisław Szufa. 2025. Learning Real-Life Approval Elections. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 9 pages.

1 INTRODUCTION

The goal of this paper is to design algorithms that take an approval election as input and produce simple probabilistic models for generating similar elections (models of generating random elections are often called statistical cultures). We form such algorithms using maximum likelihood estimation (MLE) and Bayesian learning approaches, and evaluate them on the data from the Pabulib collection of real-life participatory budgeting elections [8]. Consequently, we also get an insight into the nature of Pabulib data.[1]

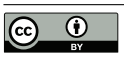
More formally, an approval election consists of a set of candidates and a collection of voters. Each voter indicates which candidates he or she finds appealing, i.e., which ones he or she approves, and which ones he or she does not. One particularly natural model of generating such elections is to provide for each candidate c a probability p_c, so that each voter approves c with this probability, independently from the other candidates and voters. We refer to this model as the independent approval model (IAM). For a positive integer t, by a t-parameter IAM we mean an IAM where each candidate has one of at most t different approval probabilities, and we often write *full IAM* when no such restriction applies. While full IAM is not widely used in computational social choice—indeed, it does not appear in the recent overview of Boehmer et al. [3]—its restricted variants are quite popular. For example, 1-IAM—where each candidate is approved independently with some probability p—is simply the p-Impartial Culture model (p-IC), one of the most popular statistical cultures for approval elections [3], and 2-IAM is equivalent to the resampling model of Szufa et al. [20]. We also note that the Hamming noise model—analyzed, e.g., by Caragiannis et al. [5], is a restricted variant of 2-IAM.

We provide algorithms that given an approval election and a number t, find a t-parameter IAM that maximizes the probability of generating this election. These algorithms are simple for impartial culture, Hamming noise model, and full IAM, while resampling and general t-parameter IAMs require more effort. We also show how the classic expectation-maximization (EM) and Bayesian learning algorithms can be used to learn mixtures of t-parameter IAMs, albeit with limited guarantees. In this case, we focus on the Hamming noise model, resampling, and full IAM.

There are two main reasons why such learning algorithms are useful. The first one is that using them we can get a strong insight into the nature of the elections that we learn. In our case, we consider all 271 approval elections from Pabulib [8] with up to tens of thousands of voters and between a few to 200 candidates. For each of these elections we learn each of our models. We find that while single IAMs are sufficient for some of the instances, in most cases it is necessary to consider mixtures of at least a few IAM components. In addition, we find that some elections are inherently difficult to learn, irrespectively how strong would our models be.

The second reason why our learning algorithms—especially those for mixture models—are useful, is that they provide models which generate fairly realistic synthetic data. In this sense, such models are more realistic than basic, stylized models often used in the literature (see the overview of Boehmer et al. [3]), but still offer a strong level of control over the generated data. For example, we can generate as many votes as we like.

[1] We disregard the costs of the projects (candidates) present in participatory budgeting.



Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Y. Vorobeychik, S. Das, A. Nowé (eds.), May 19 – 23, 2025, Detroit, Michigan, USA. © 2025 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

Related Work. So far, learning approval elections did not receive much attention in computational social choice [3]. However, we mention a paper that analyzes the number of approval votes that we need to sample to learn an underlying ground truth [6]. Further, Rolland et al. [19] consider learning profiles where each voter assigns a score to each candidate that either comes from a continuous domain or from a discrete one. For the discrete case, their setting generalizes ours, but they consider quite different distributions and learning approaches. On the other hand, learning models of ordinal elections, where each voter ranks the candidates, is well-represented. For example, there are algorithms for learning the classic Mallows model [1, 2, 12, 14], or the Plackett–Luce model, which is similar in spirit to our IAMs [13, 16, 21, 22]. There is also literature on learning voting rules, but it is quite distant from our work and we only mention a single paper on this topic [4].

2 PRELIMINARIES

For a positive integer t, by $[t]$ we mean the set $\{1, \dots, t\}$. Given two numbers a and b, we write $[a; b]$ to denote a closed interval between a and b. For some probabilistic event X, we write $\mathbb{P}(X)$ to denote the probability that it occurrs. Given a random variable X and some probability distribution D, we write $X \sim D$ to indicate that X is distributed according to D. In particular, we use the following standard distributions:

(1) $U(a, b)$ is the uniform distribution over interval $[a; b]$.
(2) Under *Bernoulli*(p) we draw 1 with probability p and 0 with probability $1 - p$.
(3) Under the categorical distribution $Cat(p_1, \dots, p_k)$, for each $i \in [k]$ the probability of drawing i is p_i (hence we require all p_i values to be nonnegative and to sum up to 1).

Elections. An *(approval) election* is a pair $E = (C, V)$, where $C = \{c_1, c_2, \dots, c_m\}$ is a set of *candidates* and $V = (v_1, \dots, v_n)$ is a collection of voters. Each voter v_i has a *vote* $A(v_i) \subseteq C$ (also called an *approval ballot*), i.e., a set of candidates that this voter approves. By $v_i(c_j)$ we mean the 1/0 value indicating whether c_j is included in $A(v_i)$ or not. For each candidate $c \in C$, we write $V(c)$ to denote the set of voters that approve c. The value $|V(c)|$ is known as the approval score of c; $|V(c)|/n$ is the probability that a random voter approves c. Given two votes, $X, Y \subseteq C$, their Hamming distance is $\mathrm{ham}(X, Y) = |X \setminus Y| + |Y \setminus X|$, i.e., it is the number of candidates approved in exactly one of them.

Probabilistic Models of Elections. For a set of candidates C, we write $\mathcal{D}(C)$ to denote the family of probability distributions over the subsets of C, i.e., over the votes with candidates from C. For a distribution $D \in \mathcal{D}(C)$ and a vote $X \subseteq C$, $\mathbb{P}(X \mid D)$ is the probability of generating vote X under D. For an election $E = (C, V)$, where $V = (v_1, \dots, v_n)$, $\mathbb{P}(E \mid D)$ is the probability of generating E, provided that each of its votes is drawn from D independently:

$$\mathbb{P}(E \mid D) = \prod_{i \in [n]} \mathbb{P}(A(v_i) \mid D). \tag{1}$$

$\mathbb{P}(E \mid D)$ is called the *likelihood* of generating E under D. We will also be interested in $\ln(\mathbb{P}(E \mid D))$, i.e., the log-likelihood of generating E.

Remark 2.1. *We view the voters as non-anonymous. To see what this entails, consider elections $E' = (C, V')$ and $E'' = (C, V'')$, with $C = \{a, b\}$, $V' = (v'_1, v'_2)$, and $V'' = (v''_1, v''_2)$, where:*

$$A(v'_1) = \{a, b\},\ A(v'_2) = \{b\}, \quad \text{and} \quad A(v''_1) = \{b\},\ A(v''_2) = \{a, b\}.$$

In our model, these two elections are distinct, but they would be equal if one viewed the voters as anonymous (it would only matter how many particular votes were cast, and not who cast them). The choice of the voter model does not affect our results: The problems of maximizing the probability of generating a given election under both models are equivalent (the respective probabilities only differ by a product of some binomial coefficents that only depend on the election's votes).

For a candidate set $C = \{c_1, \dots, c_m\}$ and a number of voters n, a *statistical culture* is a probability distribution over elections with this candidate set and n voters. By a small abuse of notation, we will also refer to the distributions from $\mathcal{D}(C)$ as statistical cultures. Indeed, given $D \in \mathcal{D}(C)$ we can always sample an election by drawing the votes from D independently. The following cultures from $\mathcal{D}(C)$ are particularly relevant:

p-Impartial Culture (p-IC). Under p-IC, for each voter v and candidate c we have that $v(c) \sim$ *Bernoulli*(p), i.e., each voter approves each candidate with probability p.

ϕ-Hamming. This distribution is parameterized by a central vote $U \subseteq C$ and a parameter $\phi \in [0; 1]$. The probability of generating voter v's vote is proportional to $\phi^{\mathrm{ham}(U, A(v))}$. ϕ-Hamming is sometimes referred to as the ϕ-noise model [20].

(p, ϕ)-Resampling. The resampling model is parameterized by a central vote $U \subseteq C$, resampling probability $\phi \in [0; 1]$, and approval probability $p \in [0; 1]$. To generate a voter's v, vote $A(v) \subseteq C$, we do as follows: First, we let $A(v)$ be equal to U. Then, independently for each $c \in C$, with probability ϕ we replace value $v(c)$ with one sampled from *Bernoulli*(p).

Impartial culture and the Hamming model are part of the folk knowledge, although the Hamming model was recently studied by Caragiannis et al. [5] and Szufa et al. [20]. The resampling model is due to Szufa et al. [20]. The Hamming model is analogous to the classic Mallows model from the world of ordinal elections [15], albeit Szufa et al. [20] advocate using the resampling model instead.

Mixture Models for Elections. Let C be some set of candidates. Given a family of K distributions $D_1, \dots, D_K \in \mathcal{D}(C)$ and probabilities $p_1, \dots, p_K$ (where $\sum_{k=1}^{K} p_k = 1$), we can form the following *mixture model*: (1) We draw a number $k \sim Cat(p_1, \dots, p_K)$ and, then, (2) we draw a vote $X \sim D_k$. We call $D_1, \dots, D_K$ the components of this model, and K is their number.

3 INDEPENDENT APPROVAL MODEL

In this section we introduce the independent approval model and argue that it generalizes all the statistical cultures from Section 2. Let $C = \{c_1, \dots, c_m\}$ be the candidate set:

$(p_1, \dots, p_t)$-Independent Approval Model. In this model, abbreviated as $(p_1, \dots, p_t)$-IAM, the candidate set is partitioned into t disjoint groups, $C_1, \dots, C_t$, and for each group C_j, each candidate $c_i \in C_j$ is approved independently, with probability p_j. We use the name *t-parameter IAM* when we disregard specific probability values.

The $(p_1, \dots, p_m)$-IAM, where every candidate has his or her individual approval probability, is a particularly natural special case

of the independent approval model, which we refer to as *full IAM*. Further, p-IC and the (p, ϕ)-resampling models also are special cases of IAM. For the former, simply take a single candidate group with approval probability p. For the latter, note that (p, ϕ)-resampling with central vote U is equivalent to the (p_1, p_2)-IAM with:

$$p_1 = (1 - \phi) + \phi \cdot p, \quad \text{and} \quad p_2 = \phi \cdot p,$$

where $C_1 = U$ and $C_2 = C \setminus U$. In the other direction, (p_1, p_2)-IAM with candidate groups C_1 and C_2, where $p_1 \geq p_2$, is equivalent to (p, ϕ)-resampling with $\phi = 1 - (p_1 - p_2)$, $p = p_2/1-(p_1-p_2)$, and central vote $U = C_1$ (note that as $p_1 \geq p_2$, we have that p and ϕ are guaranteed to be between 0 and 1). Consequently, resampling and 2-parameter IAM are equivalent.

The ϕ-Hamming model is a special case of 2-parameter IAM. Putting it in our language, Caragiannis et al. [5] have shown that for an approval probability $p \in [0.5; 1]$, $(p, 1-p)$-IAM—with candidate groups C_1 and C_2—is equivalent to ϕ-Hamming with $\phi = \frac{1-p}{p}$ and central vote $U = C_1$. Similarly, ϕ-Hamming with central vote U is equivalent to (p_1, p_2)-resampling with candidate groups $C_1 = U$ and $C_2 = C \setminus U$, $p_1 = \frac{1}{1+\phi}$ and $p_2 = 1 - p_1 = \frac{\phi}{1+\phi}$.

Altogether, we have the following hierarchy of expressivity of the IAM models (we view our models as sets of distributions, for all possible choices of parameters; e.g., by p-IC we mean all the impartial culture distributions for all approval probabilities p):

$$p\text{-IC} \subset \phi\text{-Hamming} \subset (p, \phi)\text{-Resampling} = (p_1, p_2)\text{-IAM} \subset (p_1, p_2, p_3)\text{-IAM} \subset \cdots \subset (p_1, \ldots, p_m)\text{-IAM}.$$

Remark 3.1. *Later on we consider mixture models based on IAM variants. For example, by 2-full-IAM we mean a mixture model with two full-IAM components. This should not be confused with 2-parameter IAM, which is a single-component resampling model.*

4 LEARNING ALGORITHMS

Let us now focus on the following task: We are given an election $E = (C, V)$ with candidate set $C = \{c_1, \ldots, c_m\}$, and voter collection $V = (v_1, \ldots, v_n)$. We also have a family of distributions from $\mathcal{D}(C)$. Our goal is to find a distribution from this family that maximizes the probability of generating election E. We will first solve this problem for each of the special cases of IAM from the previous two sections, and then we will consider IAM mixture models.

4.1 Learning a Single IAM Model

Let $E = (C, V)$ be an election, as described above. For each set of candidates $B \subseteq C$, let $\text{app}_E(B) = \sum_{c \in B} |V(c)|$ be the total number of approvals that members of B receive, and let $\text{prob}_E(B) = \frac{\text{app}(B)}{n|B|}$ be the probability that a random voter approves a random candidate from B. By $E(B)$, we mean election E restricted to the candidate set B. Consider a partition of C into sets X and Y, and let $D_X \in \mathcal{D}(X)$ and $D_Y \in \mathcal{D}(Y)$ be two IAMs with t_1 and t_2 parameters, respectively. Further, let $D_{XY} \in \mathcal{D}(C)$ be the $(t_1 + t_2)$-parameter IAM that generates approvals for candidates from X according to D_X and for those from Y according to D_Y. We have:

$$\mathbb{P}(E \,|\, D_{XY}) = \mathbb{P}(E(X) \,|\, D_X) \cdot \mathbb{P}(E(Y) \,|\, D_Y). \tag{2}$$

For each $t \in [|C|]$ and each partition of C into $C_1, \ldots, C_t$, we write $\text{IAM}(C_1, \ldots, C_t)$ to refer to the t-parameter IAM that uses this partition and for each $i \in [t]$, the probability of approving a candidate from C_i is $p_i = \text{prob}_E(C_i)$. As per Equation (2), we have:

$$\mathbb{P}(E \,|\, \text{IAM}(C_1, \ldots, C_t)) = \textstyle\prod_{i \in [t]} \mathbb{P}(E(C_i) \,|\, \text{prob}_{E(C_i)}(C_i)\text{-IC}).$$

Intuitively, if we want a t-parameter IAM that maximizes the probability of generating a given election, it suffices to use $\text{IAM}(C_1, \ldots, C_t)$ for an appropriate partition of the candidate set. Next we show this fact formally and argue how to find optimal partitions (for the ϕ-Hamming model we use a different approach).

4.1.1 *Impartial Culture and the Full IAM Model.*

For impartial culture, finding the parameter that maximizes the probability of generating a given election E is easy: It suffices to use p-IC with p equal to the proportion of approvals in the election (this is a standard fact from statistics, often expressed in different contexts).

Proposition 4.1. *Let $E = (C, V)$ be an approval election. Probability $\mathbb{P}(E \,|\, q\text{-IC})$ is maximized for $p = \text{prob}_E(C)$.*

Consequently, to learn the parameters of an IAM for a given election, it suffices to find a partition of the candidates.

Proposition 4.2. *Let $E = (C, V)$ be an election. For each $t \in [|C|]$ there is a partition of C into $C_1, \ldots, C_t$ such that $\text{IAM}(C_1, \ldots, C_t)$ maximizes the probability of generating E among t-parameter IAMs.*

Consequently, the full IAM that maximizes the probability of generating a given election uses parameters where each candidate is approved with probability equal to the fraction of its approvals.

Corollary 4.3. *Let $E = (C, V)$ be an election, where $C = \{c_1, \ldots, c_m\}$ and n voters. Probability $\mathbb{P}(E \mid (p_1, \ldots, p_m)\text{-IAM})$ is maximized if for each $i \in [m]$ we have $p_i = |V(c)|/n$.*

4.1.2 *Hamming Model.*

For each $\phi \in [0; 1]$ and each vote U, we let $\phi\text{-Ham}(U)$ denote the ϕ-Hamming model with central vote U. The probability of generating vote $A(v)$ under $\phi\text{-Ham}(U)$ is:

$$\mathbb{P}\big(A(v) \,|\, \phi\text{-Ham}(U)\big) = \tfrac{1}{(1+\phi)^m} \phi^{\text{ham}\left(U, A(v)\right)}$$

(the normalizing constant is derived, e.g., by Caragiannis et al. [5]), and the probability of generating election E is:

$$f_u(\phi) = \mathbb{P}\big(E \,|\, \phi\text{-Ham}(U)\big) = \tfrac{1}{(1+\phi)^{mn}} \phi^{\sum_{i=1}^n \text{ham}\left(U, A(v_i)\right)}. \tag{3}$$

For each fixed ϕ, this value is maximized when the exponent, $\sum_{i=1}^n \text{ham}\big(U, A(v_i)\big)$, is minimized. This happens for central vote U such that for each candidate c_i, c_i belongs to U if and only if at least half of the voters approve c_i. We refer to such a central vote as *majoritarian*. Let us fix U to be majoritarian and let $h = \sum_{i=1}^n \text{ham}\big(U, A(v_i)\big)$. By definition, we have $h \leq {}^{mn}/_2$, and we assume that $h > 0$ (otherwise it suffices to take $\phi = 0$ to maximize the probability of generating E). The derivative of $f_u(\phi)$ (with respect to ϕ) is:

$$f_u'(\phi) = \frac{h\phi^{h-1}(1+\phi)^{mn} - mn(1+\phi)^{mn-1}\phi^h}{(1+\phi)^{2mn}} = \frac{\phi^{h-1}(1+\phi)^{mn-1}}{(1+\phi)^{2mn}}(h(1+\phi) - mn\phi).$$

By analyzing the final term, one can verify that its value is 0 exactly for $\phi = \frac{h}{mn-h}$, for smaller ϕ it is positive, and for larger ϕ it is negative. Hence, we have the following result.

Proposition 4.4. *Let $E = (C, V)$ be an approval election with m candidates and n voters, where $V = (v_1, \ldots, v_n)$. Let U be the majoritarian central vote for E and let $h = \sum_{i=1}^{n} \mathrm{ham}(u, v_i)$. The probability of generating E using ϕ-Hamming model is maximized for the majoritarian central vote and $\phi = h/mn-h$.*

4.1.3 Resampling and Other IAMs. Next, let us consider learning the parameters of 2-parameter IAMs (or, equivalently, of the resampling models). The solution is intuitive, but requires a more careful proof. In particular, the next theorem restricts the parameter space that we need to analyze.

Theorem 4.5. *Let $E = (C, V)$ be an election with candidate set $C = \{c_1, \ldots, c_m\}$, and voter collection $V = (v_1, \ldots, v_n)$, such that $|V(c_1)| \geq |V(c_2)| \geq \cdots \geq |V(c_m)|$. Then $\mathbb{P}(E \mid (p_1, p_2)\text{-IAM})$ is maximized for some $m' \in [m]$ and:*

$$p_1 = |V(c_1)|+\cdots+|V(c_{m'})|/nm', \quad C_1 = \{c_1, \ldots, c_{m'}\},$$
$$p_2 = |V(c_{m'+1})|+\cdots+|V(c_m)|/n(m-m'), \quad C_2 = C \setminus C_1.$$

Intuitively, Theorem 4.5 says that if we want to find the parameters of a 2-parameter IAM that maximize the probability of generating a given election $E = (C, V)$, then there are only polynomially many options to try. Namely, using the notation from Theorem 4.5, it suffices to try all $O(m)$ choices of m', each giving a different candidate partition, and—among those—select the one that leads to maximizing the probability of generating E.

Corollary 4.6. *There is a polynomial-time algorithm that given an election $E = (C, V)$ finds the parameters of a 2-parameter IAM that maximizes the probability of generating E.*

Using the ideas from the proof of Theorem 4.5, we also obtain an analogous result for IAMs with arbitrary number t of parameters.

Theorem 4.7. *Let $E = (C, V)$ be an election with m candidates and n voters. For each $t \in [m]$, $\mathbb{P}(E \mid (p_1, p_2, \ldots, p_t)\text{-IAM})$ is maximized for $p_1, \ldots, p_t$ and a partition of C into $C_1, \ldots, C_t$ such that for each $i \in [t]$ we have $p_i = \sum_{c \in C_i} |V(c)|/n|C_i|$ and for each $i \in [t-1]$ and every two candidates $a \in C_i$ and $b \in C_{i+1}$ we have $|V(a)| \geq |V(b)|$.*

Based on Theorem 4.7, we derive a dynamic-programming algorithm that computes a t-parameter IAM that maximizes the probability of generating a given election.

Theorem 4.8. *There is a polynomial-time algorithm that given an election $E = (C, V)$ and an integer $t \in [|C|]$ finds the parameters of a t-parameter IAM that maximizes the probability of generating E.*

4.2 Mixture Models: Expectation Maximization

In this and the next section we consider two different approaches of learning mixtures of IAMs. Here we start with one of the most standard and widely adopted machine learning algorithms used for estimating the values of parameters in statistical models, the Expectation-Maximization (EM) algorithm [7]. The algorithm is useful when there are some missing or unobserved latent variables in the model. In the context of learning mixtures of distributions, these latent variables correspond to the assignment of the data points to the mixture components that generated them. The algorithm first guesses the parameters of the model and then iteratively improves the estimation trying to maximize the log-likelihood of generating the observed data. Each iteration consists of two steps:

- **E-step (Expectation),** where the algorithm computes the posterior probability (soft assignment) that each data point was generated by each mixture component.
- **M-step (Maximization),** where the algorithm updates the parameters of the mixture model to maximize the expected log-likelihood given the probabilities from the E-step.

Intuitively, after each iteration the likelihood of generating the observed data under the current model parameters increases. The algorithm alternates between the E- and M-steps until convergence (i.e., until two consecutive iterations return the same model parameters up to some negligible ϵ), obtaining a local maximum of the log-likelihood.

Let us now consider how the EM algorithm can be used for learning mixtures of IAM models. Let us fix some election E and the number K of components. For $k \in [K]$, the k-th component has parameters $\theta_k = (\alpha_k, (p_1, p_2, \ldots, p_m))$, where α_k is the weight of the component ($\sum_{k \in [K]} \alpha_k = 1$) and $p_1, \ldots, p_m \in [0; 1]$ are the probabilities of approving candidates $c_1, \ldots, c_m$ (these probabilities are independent only for the full IAM model; however, potential dependencies between them are not relevant for the E-step). Given a k-th mixture component and its parameters θ_k, the probability of generating vote $A(v)$ by this component is:

$$\mathbb{P}(A(v)|\theta_k) = \alpha_k \cdot \left(\Pi_{j \in [m]: c_j \in A(v)} p_j\right) \left(\Pi_{j \in [m]: c_j \notin A(v)} (1 - p_j)\right).$$

Let us denote the posterior probability of generating votes from E, computed in the E-step of the algorithm—further called the *soft assignment* of votes from E—as $\gamma = \{\gamma_{v,k}\}_{v \in V, k \in [K]}$. For each voter v and the k-th component, $\gamma_{v,k}$ is equal to the conditional probability of generating v by the k-th component subject to the fact that v has been generated by *some* component. Formally:

$$\gamma_{v,k} = \frac{\mathbb{P}(A(v)|\theta_k)}{\sum_j \mathbb{P}(A(v)|\theta_j)}.$$

For clarity of the notation, for each $k \in [K]$ let us denote the total weight of the votes assigned to the k-th component as $\gamma_k = \sum_{v \in V} \gamma_{v,k}$. Note that $\sum_{k \in [K]} \gamma_k = n$.

For the M-step, we need to find the parameters $\theta_1, \theta_2, \ldots, \theta_K$ maximizing the expected complete log-likelihood up to the computed soft assignment, given by the following formula:

$$\sum_{v \in V} \sum_{k \in [K]} \gamma_{v,k} \ln(\mathbb{P}(A(v)|\theta_k)).$$

Note that in the above formula the gamma variables should be viewed as constants; their values depend on theta values computed in the previous iteration of the algorithm, not on the ones we currently search for.

Let us first focus on the α_k parameter of each component $k \in [K]$. We have the following result here:

Proposition 4.9. *For each mixture of K IAM distributions, the expected log-likelihood of generating the observed data is maximized if for each $k \in [K]$ it holds that $\alpha_k = \gamma_k/n$.*

Let us now focus on optimizing the remaining parameters of IAM components. Note that since we know the optimal weights of the components and the other parameters are independent between different components, we can now optimize the parameters of each IAM component separately.

Let us fix component number $k \in [K]$ and compute the probabilities maximizing the expected log-likelihood of generating a

corresponding soft assignment γ using the k-th component. Note that for each $v \in V$ we have that $\gamma_{v,k}$ is a rational number, i.e., it is a quotient of two integers. Let Q be the least common multiple of all the denominators of these quotients. We now consider an election $E_{\gamma,k}$ obtained from the initial election E and the given soft assignment γ by multiplying each voter v, $Q \cdot \gamma_{v,k}$ times (so that for each two votes v and v', the proportion between the numbers of their copies is equal to $\gamma_{v,k}/\gamma_{v',k}$), which we will further call an election *induced by* E, γ and the k-th component. We will show that maximizing the log-likelihood of generating the part of γ corresponding to the considered k-th component is equivalent to maximizing the probability of generating $E_{\gamma,k}$.

Proposition 4.10. *Let $E = (C, V)$ be an approval election with m candidates and n voters and let γ be the soft assignment of votes to some K mixture components. Let $E_{\gamma,k}$ be the election induced by E, γ and some k-th component for $k \in [K]$. Then finding the k-th mixture component parameters maximizing the log-likelihood of generating the assignment is equivalent to finding the parameters maximizing the probability of generating $E_{\gamma,k}$.*

We also obtain direct formulas for optimal parameters for other models from the preceding section (see the full version of the paper).

4.3 Mixture Models: Bayesian Learning

Consider an election $E = (C, V)$, with candidates $C = \{c_1, \ldots, c_m\}$ and voters $V = (v_1, \ldots, v_n)$. So far we focused on learning parameter values that maximize the probability of generating E. Once learned, model's parameters in these settings are, essentially, fixed constants. Alternatively, we can view the parameters themselves as random variables. We can then estimate a distribution over the parameters that is compatible with the election E. Learning the distribution over model's parameters conditioned on the observed data is the core concept in Bayesian statistics.

One simple way to formalize a Bayesian model for an approval election is to postulate a *generative process* for the votes, i.e., a sampling procedure that describes our *prior assumptions* about the distribution over the votes. For example, consider the full IAM model parametrized by the vector of approval probabilities: $(p_1, p_2, \ldots p_m)$. The generative process in this case starts with sampling each approval probability p_i from a prior distribution over its possible values. In this work we do not assume any a priori knowledge about the approval probabilities. The prior distribution for approval probabilities is therefore the uniform distribution over the $[0, 1]$ interval. After sampling the parameters, the generative process samples the votes conditioned on the parameter values. In the IAM model this conditional distribution is simply the Bernoulli distribution parametrized by the approval probability. Together, these two steps give the following process:

(1) For all $c \in C$, sample the approval probability, $p_c \sim U(0, 1)$.
(2) For all $v_i \in V, c \in C$, sample the vote outcome, $v_i(c) \sim Bernoulli(p_c)$.

The generative process fixes the prior over model's parameters $p(p_1, \ldots, p_m)$ and the data likelihood $p(V \mid p_1, \ldots, p_m)$. The Bayes theorem then gives a principled formula for the *posterior distribution* over the model's parameters: $p(p_1, \ldots, p_m \mid V)$. This distribution summarizes our knowledge about values of the parameters, once we observed a set of votes V.

The Bayesian framework provides a flexible way to specify more complex generative processes. In particular, we can easily write a generative process for a mixture of K full IAM components:

(1) Sample component probabilities, $(\alpha_1, \ldots, \alpha_K) \sim Dirich(\mathbf{1}^K)$.
(2) For all $c \in C$, $k \in [K]$, sample the k-th component's approval probability for the candidate c, $p_{c,k} \sim U(0, 1)$.
(3) For all $v_i \in V$:
   (a) Sample the component index, $z \sim Cat(\alpha_1, \ldots, \alpha_K)$.
   (b) For all $c \in C$, sample $v_i(c) \sim Bernoulli(p_{c,z})$.

Here, $Dirich(\mathbf{1}^K)$ is the Dirichlet distribution with unit concentration parameters, while $Cat(\alpha_1, \ldots, \alpha_K)$ is the categorical distribution parametrized by components' probabilities. Note that the Dirichlet prior in our IAM mixture is, again, uninformative: Dirichlet distribution with unit concentrations is the uniform distribution over the $K - 1$ dimensional probability simplex. Using similar prior distributions we can also formulate Bayesian models for other models (see the full version of the paper).

The flexibility of Bayesian models comes with a price: due to the intractable normalization constant in the Bayes rule, it is typically impossible to evaluate the posterior distribution exactly. That said, there are efficient, general-purpose algorithms that can be used to draw samples from the posterior distribution. We generate posterior samples using the No-U-turn sampler [10] with variable elimination [17] for the component assignments and Gibbs sampling for the central votes. To this end, we implement and estimate our models in the NumPyro probabilistic programming language [18].

After sampling from the posterior distribution, we approximate posterior means of model's parameters by averaging across sampled values. We then use these mean estimates in downstream analyses. Note that our models use so-called exchangeable priors: the model specification and, consequently, the posterior density is invariant to permutation of component labels. In a naive implementation, samples from such models may differ in the ordering of components, leading to incorrect mean estimates. We remedy this issue by using a standard *identifiability constraint* technique [11]. In particular, we restrict the Dirichlet prior on the components' probabilities to the polytope that satisfy the constraint: $\alpha_1 > \alpha_2 > \cdots > \alpha_K$, and put zero prior probability mass elsewhere. This constraint uniquely identifies one out of $K!$ equivalent component labellings. In practice, the constraint can be enforced post sampling, by reordering the components in the samples [11].

5 EXPERIMENTS

Our experiments focus on learning variants of IAMs, as well as their mixtures, on elections from the Pabulib database [8]. Specifically, we considered all 271 approval-based Pabulib elections that include at least 2 000 voters. For each election $E = (C, V)$ from this set, and each considered algorithm $\mathcal{A}$ (for a given IAM variant) we executed the following procedure $t_{\text{try}} = 5$ times (each time using independent coin tosses; for the experiment described in Section 5.2 we used $t_{\text{try}} = 20$):

(1) We formed elections E_{learn} and E_{eval}, where the latter consisted of randomly selected $n_{\text{eval}} = 1000$ votes from E, and the former consisted of the remaining votes. If E_{learn}

ended up with more than 20 000 voters, then we kept only $n_{\text{sample}} = 20\,000$ of them, selected uniformly at random (to bound the computation time). We refer to E_{learn} as the *learning* election and to E_{eval} as the *evaluation* one.

(2) We run $\mathcal{A}$ on E_{learn} and obtained distribution $D \in \mathcal{D}(C)$.
(3) We computed the log-likelihood of obtaining E_{eval} using D, as well as a few other metrics (see Section 5.1).

We only performed 5 runs of this procedure because we found that the variance of the results that we get (i.e., variance of the metrics that we computed) was typically several orders of magnitude lower than the results themselves.

For each of our elections, we ran all the algorithms from Section 4.1, i.e., the single-component algorithms for IC, Hamming, Resampling, full IAM and all the t-parameter IAM models for t ranging from 1 to the number of candidates in the given election (with a step of 1). Then, we applied Bayesian learning (Section 4.3) to compute mixture models with 2, 3, and 4 components, where each of the components was either a Hamming model, a resampling model, or full IAM. Finally, we applied the EM algorithm (Section 4.2) to learn mixture models of 2, 3, and 4 full IAM components (we omitted Hamming and resampling models due to computation cost).

5.1 Evaluation Metrics

While we could use log-likelihoods of the models that we learn to evaluate their quality, this has drawbacks. For example, it is difficult to compare log-likelihood values across different elections. Thus, we use metrics based on the voter-anonymous variant of the Hamming distance, defined below.

Definition 5.1. *Let $E = (C, V)$ and $F = (C, U)$ be two elections over the same candidate set C, with voter collections $V = (v_1, \ldots, v_n)$ and $U = (u_1, \ldots, u_n)$ of equal size. Their voter-anonymous Hamming distance is as follows (S_n is the set of permutations of $[n]$):*

$$\text{va-ham}(E, F) = \frac{1}{n} \min_{\sigma \in S_n} \sum_{i=1}^{\ell} \text{ham}\big(A(v_i), A(u_{\sigma(i)})\big).$$

In other words, va-ham(E, F) is the average Hamming distance between the votes from E and F, matched in such a way as to minimize the final result. Note that our definition is similar to the definitions of isomorphic distances of Faliszewski et al. [9] and Szufa et al. [20], except that we consider election with equal candidate sets. In particular, voter-anonymous Hamming distance is invariant to reordering the voters and is normalized by the number of voters.

Baseline Distance. Let E be an election from the subset of Pabulib that we consider. We define E's baseline distance as the expected value of the random variable defined as va-ham(E_1, E_2), where E_1 and E_2 are subelections of E, each with n_{eval} voters, selected uniformly at random up to the condition that E_1 and E_2 do not have any voters in common.[2] Intuitively, baseline distance is a measure of an election's internal diversity. For example, if its value is 2 then if we take two random, disjoint subelections of E (each with n_{eval} voters), it would be possible, on average, to match their votes so that two matched votes differ on two candidates (e.g., each of them may include a single candidate not present in the other one). In practice, we compute baseline distance of an election by drawing 5 pairs of elections and averaging their voter-anonymous Hamming distance (typically, the variance is orders of magnitude lower than the value of the average, so considering 5 pairs of elections is justified).

[2]E.g., it means that if some voter v from E is included in E_1 then he or she is certainly not included in E_2. However, E_2 may contain other voter u with $A(v) = A(u)$.

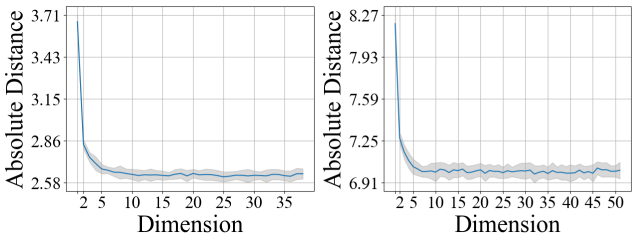


Figure 1: Absolute distance between the Amsterdam 289 election (38 candidates, left) or Warszawa 2020 Ochota election (51 candidates, right) and single-component t-parameter IAMs, as a function of t, from 1 to the number of candidates.

Absolute and Relative Distances. Given a Pabulib election $E = (C, V)$ and a learning algorithm $\mathcal{A}$, we compute their *absolute distance* as follows: For each evaluation election E_{eval} that we computed for E and $\mathcal{A}$, we take distribution $D \in \mathcal{D}(C)$ obtained by $\mathcal{A}$ on E_{learn}, generate election E_D by drawing n_{eval} votes independently from D, and compute va-ham(E_{eval}, E_D). We obtain five numbers and we output their average value. We define the *relative distance* between E and $\mathcal{A}$ as their absolute distance divided by E's baseline. In other words, relative distance normalizes the absolute one by E's inherent diversity (E's baseline is, essentially, its absolute distance from a distribution that samples E's votes uniformly at random so, intuitively, it bounds achievable absolute distance).

5.2 Impact of the Number of IAM Parameters

Let us now focus on single-component IAMs and the influence that the number of parameters has on their ability to learn Pabulib elections. In particular, in Figure 1 we plot the absolute distance between elections generated using t-parameter IAM models learned (on two example Pabulib elections) using the algorithm from Theorem 4.8. We find that for IC ($t = 1$) we get a significantly higher absolute distance than for the resampling model ($t = 2$), which itself is somewhat higher than the absolute distance for full-IAM (t equal to the number of candidates). The plots for other elections are very similar in spirit.

Our conclusion from this experiment is that impartial culture performs notably worse than the other IAM variants, but models with two parameters and more achieve fairly similar results (even if there is still a visible difference between the results for the resampling model and full IAM). In the following experiments we limit our attention to the Hamming, resampling, and full IAM models, as they are simple to learn, give good results, and the former two can be specified using much less information than full IAMs.

5.3 General Analysis of Learning Results

In this section, we provide a high-level overview of the Pabulib dataset and the performance of our learning methods. In Figure 2 we illustrate the relationship between the average vote length in a given election (i.e., the average number of candidates approved by a

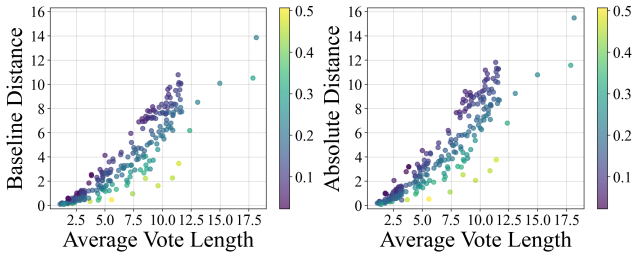


Figure 2: The relation between the average vote length and baseline distance (left plot), and absolute distances from the best learned model (right plot). Each dot depicts a single Pabulib instance. The color gives the profile's saturation (i.e., the average vote length divided by the number of candidates).

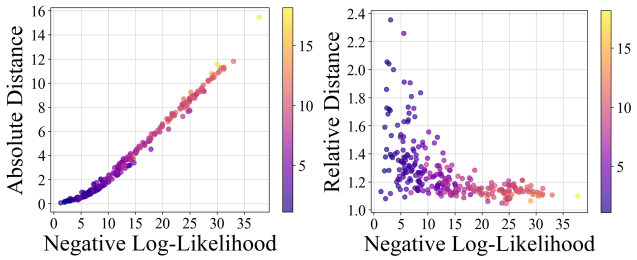


Figure 3: The relation between the (negative) log-likelihood and the absolute distance (left plot), and relative distance (right plot). Each dot depicts a single Pabulib instance. The color corresponds to the average vote length.

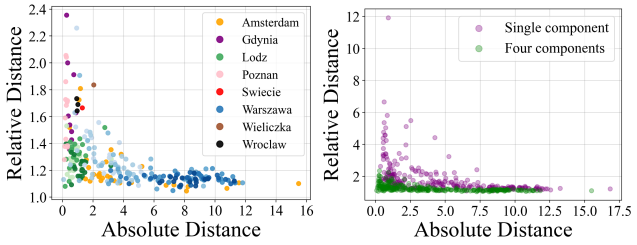


Figure 4: The comparison of the absolute and relative distances. The left plot shows from which city each instance originates (for Lodz and Warszawa we use different shades of green and blue, respectively, for different years). The right plot compares the single- and multi-component approaches. Each dot depicts a single Pabulib instance (the number of dots doubles in the right plot due to two approaches used).

single voter) and the baseline distance of this election (left plot) or its absolute distance from the best-learned model (right plot). Each dot represents a single Pabulib instance, with the color corresponding to the profile saturation (i.e., the average vote length divided by the total number of candidates). The left plot reflects the self-similarity of the Pabulib instances. We see that while some Pabulib elections have low baseline distance (i.e., are close to the x axis) for many of them this is not the case. Indeed, the baseline distance seems to have

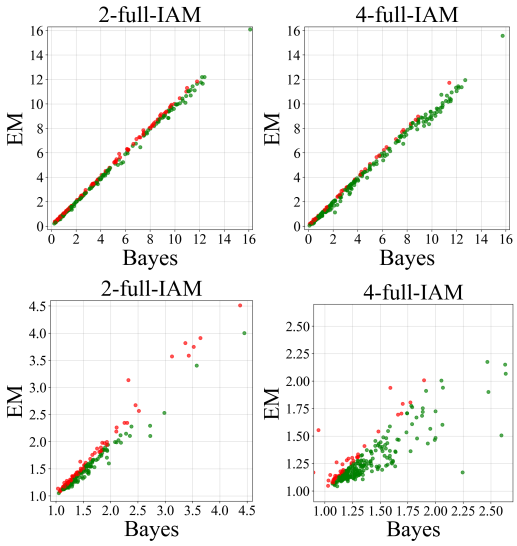


Figure 5: Comparison of Bayes and EM learning methods. The upper plots show the absolute distances for both methods, and the lower ones show the relative distances. By red (green) color we mark the cases where Bayes (EM) achieved smaller distance. Each dot depicts a single Pabulib instance.

a close-to-linear dependence on the average vote length; the more candidates the voters approve, the more diverse are the votes that they cast. The close resemblance between the two plots in Figure 2 indicates that, in most cases, our learning algorithms perform well and achieve absolute distance similar to the baseline one (we discuss this in more detail later, when analyzing Figure 4). Regarding the plot on the right-hand side, we observe two key trends: First, as the average vote length increases, the absolute distance also increases. Second, for a given average vote length, higher saturation tends to correspond to a smaller absolute distance.

In Figure 3 we explore the relationship between the (negative) log-likelihood and the absolute distance (left plot) and the relative distance (right plot), both for the best-learned model (i.e., the one that achieves the lowest absolute distance). While the log-likelihood is strongly correlated with the absolute distance (having Pearson correlation coefficient equal to 0.993), it is barely (negatively) correlated with the relative distance (having PCC=-0.567). Our conclusion here is that by using distance-based metrics of quality we gain interpretability of our results (as discussed in Section 5.1) without losing much of statistical significance of log-likelihoods (as absolute distances are, in essence, negative log-likelihoods in disguise).

Figure 4 compares the absolute and relative distances between each election and its best-learned model (i.e., the one that achieves lowest absolute distance). The left plot uses color to show the city of origin for each instance, revealing that different cities tend to occupy distinct regions of the plot. This suggests significant variation in the nature of elections across cities and is a strong argument to use data with different origins in experiments based on Pabulib.

We note that absolute and relative distances tell us quite different stories about the quality of a learned model. For instance, we may

view relative distance below 1.2 as quite good, but it may still correspond to the absolute distance of, say, 10. For the case where, on average, each voter in the considered election also approves 10 candidates, this means that our algorithm learned a very good model as compared to the baseline distance, but the input election is internally so diverse that the generated votes will still largely differ from those present in the actual data. Similarly, we may view relative distance equal to 2, as rather unsatisfactory, but it might still mean an absolute distance of 0.5 for the baseline of 0.25. Even though the relative distance is large, the absolute one is objectively small and the learned distribution produces votes that can be seen as very similar to those present in the considered election. Finally, there also are some elections for which both distances are quite high. Then, we have to concede that either IAM mixtures, or our algorithms, are simply insufficient to learn these elections well.

Let us now consider the right-hand plot of Figure 4 (note different scales on the axes as compared to the left-hand one). This plot contrasts best-learned single-component models (in this case these always are the full-IAM ones) and best-learned mixture models (these often are 4-full-IAMs learned using the EM algorithm, but sometimes also 4-resampling or 4-full-IAM ones learned using the Bayesian approach). As expected, mixture models perform much better: Their points have lower absolute and relative distance coordinates. While the fact that mixture models are more expressive than single-component ones is hardly surprising, knowing the extent of their advantage is useful. Indeed, we see that to generate realistic elections similar to those in Pabulib, using mixture models of several IAMs gives notably better results that using single-component full-IAMs (not to mention even simpler single-component models).

In Figure 5 we compare the EM and Bayesian approaches. As the plots show, particularly in the case of the 4-full-IAM model, the EM approach consistently outperforms the Bayesian method. Indeed, in this case the Bayesian approach often finds it difficult to identify four components and outputs models that perform even worse than the 3-full-IAM models that it identifies. While we believe that one could improve on this by engineering the priors used in Bayesian models, we did not pursue this direction and view it as a follow-up work. The main conclusion from Figure 5 is that both EM and the Bayesian approach achieve similar results, even though the former explicitly minimizes the negative log-likelihood and the latter does not. This reinforces our view that both algorithms—based on so different principles—identify meaningful components. As component analysis can be challenging, we find this finding valuable.

5.4 Closer Look on a Few Instances

Next, we zoom in on a few selected instances to describe our observations in more detail. To improve understanding, we use histograms which show the number of votes of a given length in an election. In Figure 6, for selected Pabulib elections we lay over histograms of the respective E_{learn} and of the learned single-component and 4-component resampling models (results for Hamming and full-IAM are similar, we chose resampling for variety).

As mentioned in Section 5.3, nearly always multiple-component models yield elections with significantly smaller distances to the original one. There are two main reasons to explain this observation: First, single-component models are prone to "the flaw of average"

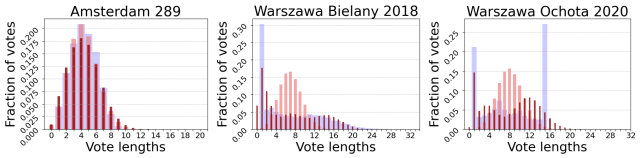


Figure 6: Superposition of three histograms of vote lengths for three Pabulib instances. Each picture shows histograms of the training election (blue) and learned single-component (pale red) and four-component (dark red) resampling models. For clarity, we removed bars for fractions smaller than 10^{-10}.

of the vote lengths. It is evident for elections with bimodal vote length distributions, like that of election Warszawa Ochota 2020 in Figure 6: A single learned component mostly generates average-length votes, which are dissimilar from those of either of the peaks.

Second, single-component models can only produce votes with a relatively limited variance of their length. Hence, it is frequently counterproductive to apply single-component models to learn elections where this variance is high. It includes elections with a vote length distribution that is asymmetric, uni-modal, and has a "heavy tail" (such as Warszawa Bielany 2018 depicted in Figure 6) or that have a bimodal distribution of vote lengths (like Warszawa Ochota 2020 in Figure 6). The histograms clearly show how multiple components help in dealing with such distributions.

On the positive side, we want to stress that single-component models can perform well on some real-life elections. In particular, this holds true for elections with Gaussian-like distributions of vote lengths; for an example, see election Amsterdam 289 in Figure 6.

6 SUMMARY

To the best of our knowledge, we performed the first comprehensive analysis of Pabulib elections by learning them using both single-component and mixture models. We found that some elections can be captured with simple models such as resampling or full IAM, but typically using mixtures of a few components is preferable.

ACKNOWLEDGMENTS

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 101002854), from the French government under the management of Agence Nationale de la Recherche as part of the France 2030 program, reference ANR-23-IACL-0008. In part, A. Kaczmarczyk acknowledges support from NSF CCF-2303372 and ONR N00014-23-1-2802. In part, S. Szufa was supported by the Foundation for Polish Science (FNP). In part, M. Kurdziel was supported by Poland's National Science Centre (NCN) grant no. 2023/49/B/ST6/01458. We gratefully acknowledge Polish high-performance computing infrastructure PLGrid (HPC Center: ACK Cyfronet AGH) for providing computer facilities and support within computational grant no. PLG/2024/017160.

REFERENCES

[1] N. Betzler, R. Bredereck, and R. Niedermeier. 2014. Theoretical and empirical evaluation of data for exact Kemeny Rank Aggregation. *Auton. Agent. Multi-Ag.* 28, 5 (2014), 721–748.

[2] N. Boehmer, R. Bredereck, E. Elkind, P. Faliszewski, and S. Szufa. 2022. Expected Frequency Matrices of Elections: Computation, Geometry, and Preference Learning. In *Proceedings of NeurIPS-2022*.

[3] N. Boehmer, P. Faliszewski, L. Janeczko, A. Kaczmarczyk, G. Lisowski, G. Pierczynski, S. Rey, D. Stolicki, S. Szufa, and T. Was. 2024. Guide to Numerical Experiments on Elections in Computational Social Choice. In *Proceedings of IJCAI-2024*. 7962–7970.

[4] I. Caragiannis and K. Fehrs. 2022. The Complexity of Learning Approval-Based Multiwinner Voting Rules. In *Proceedings of AAAI-2022*. 4925–4932.

[5] I. Caragiannis, C. Kaklamanis, N. Karanikolas, and G. Krimpas. 2022. Evaluating Approval-Based Multiwinner Voting in Terms of Robustness to Noise. *Auton. Agent. Multi-Ag.* 36, 1 (2022), Article 1.

[6] I. Caragiannis and E. Micha. 2017. Learning a Ground Truth Ranking Using Noisy Approval Votes. In *Proceedings of IJCAI-2017*. 149–155.

[7] AP Dempter. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of Royal Statistical Society* 39 (1977), 1–22.

[8] P. Faliszewski, J. Flis, D. Peters, G. Pierczyński, P. Skowron, D. Stolicki, S. Szufa, and N Talmon. 2023. Participatory Budgeting: Data, Tools and Analysis. In *Proceedings of IJCAI-2023*. 2667–2674.

[9] P. Faliszewski, P. Skowron, A. Slinko, S. Szufa, and N. Talmon. 2019. How Similar Are Two Elections?. In *Proceedings of AAAI-2019*. 1909–1916.

[10] M. D. Hoffman and A. Gelman. 2014. The No-U-turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *J. Mach. Learn. Res.* 15, 1 (2014), 1593–1623.

[11] A. Jasra, C. C. Holmes, and D. A. Stephens. 2005. Markov Chain Monte Carlo Methods and the Label Switching Problem in Bayesian Mixture Modeling. *Statist. Sci.* 20, 1 (2005), 50–67.

[12] A. Liu and A. Moitra. 2018. Efficiently Learning Mixtures of Mallows Models. In *Proceedings of FOCS-2018*. 627–638.

[13] A. Liu, Z. Zhao, C. Liao, P. Lu, and L. Xia. 2019. Learning Plackett-Luce Mixtures from Partial Preferences. In *Proceedings of AAAI-2019*. 4328–4335.

[14] T. Lu and C. Boutilier. 2014. Effective Sampling and Learning for Mallows Models with Pairwise-Preference Data. *Journal of Machine Learning Research* 15, 1 (2014), 3783–3829.

[15] C. Mallows. 1957. Non-null ranking models. *Biometrica* 44 (1957), 114–130.

[16] D. Nguyen and A. Zhang. 2023. Efficient and Accurate Learning of Mixtures of Plackett-Luce Models. In *Proceedings of AAAI-2023*. 9294–9301.

[17] F. Obermeyer, E. Bingham, M. Jankowiak, N. Pradhan, J. T. Chiu, A. M. Rush, and N. D. Goodman. 2019. Tensor Variable Elimination for Plated Factor Graphs. In *Proceedings of ICML-2019*. 4871–4880.

[18] D. Phan, N. Pradhan, and M. Jankowiak. 2019. Composable Effects for Flexible and Accelerated Probabilistic Programming in NumPyro. *arXiv preprint arXiv:1912.11554* (2019).

[19] A. Rolland, J.-B. Aubin, I. Gannaz, and S. Leoni. 2024. Probabilistic Models of Profiles for Voting by Evaluation. *Social Choice and Welfare* 63, 2 (2024), 377–400.

[20] S. Szufa, P. Faliszewski, Ł. Janeczko, M. Lackner, A. Slinko, K. Sornat, and N. Talmon. 2022. How to Sample Approval Elections?. In *Proceedings of IJCAI-2022*. 496–502.

[21] Z. Zhao, P. Piech, and L. Xia. 2016. Learning Mixtures of Plackett-Luce Models. In *Proceedings of ICML-2016*. 2906–2914.

[22] Z. Zhao and L. Xia. 2019. Learning Mixtures of Plackett-Luce Models from Structured Partial Orders. In *Proceedings of NeurIPS-2019*. 10143–10153.