Maximally Random Sortition

GABRIEL DE AZEVEDO, Cornell University, USA
PAUL GÖLZ, Cornell University, USA

Citizens' assemblies are a form of democratic innovation in which a randomly selected panel of constituents deliberates on questions of public interest. We study a novel goal for the selection of panel members: maximizing the entropy of the distribution over possible panels.

We design algorithms that sample from maximum-entropy distributions, potentially subject to constraints on the individual selection probabilities. We investigate the properties of these algorithms theoretically, including in terms of their resistance to manipulation and transparency. We benchmark our algorithms on a large set of real assembly lotteries in terms of their intersectional diversity and the probability of satisfying unseen representation constraints, and we obtain favorable results on both measures. We deploy one of our algorithms on a website for citizens' assembly practitioners.

Contents

1 Introduction

In a time of democratic backsliding [45], many countries are experimenting with *citizens' assemblies* to complement and reinvigorate their democracies [12, 46]. These assemblies gather a random sample of constituents to propose policy on a given topic. Recent examples include assemblies in Ireland on same-sex marriage and abortion [13], which led to constitutional change, the French citizens' convention on climate change [30], and permanently instituted assemblies in Brussels and Paris [47].

How exactly to do the random selection of panel members, called *sortition*, is not obvious. While practitioners and political theorists praise the virtues of random selection [11, 17, 20, 40], they consider an idealized sortition process in which the panel members are directly drawn (uniformly and without replacement) from the population, which is unrealistic since many people will not agree to participate. In practice, sortition proceeds in two stages: a large random sample of constituents is invited, and those willing to participate opt into a *pool* of volunteers; then, a *selection algorithm* randomly selects the panel from among the pool. To be representative, this panel must satisfy *quotas* imposed by the assembly organizers — say that, between 20 and 23 out of the 100 participants should have a college degree, exactly 50 should be female, and so forth.

Hence, the question is: *Which probability distribution over the possible panels should the selection algorithm implement, to approximate idealized sortition?* Flanigan et al. [22] gave a first answer in *Nature*: since one desirable property of idealized sortition is that all constituents are equally likely to be selected [11, 20, 21], their algorithmic framework implements a lottery such that the pool members' selection probabilities are as close to equal as possible. They refer to this objective as *fairness*. Algorithms within this framework[1] are now widely used in practice, through an open source tool[2] and *panelot.org*.

Though equalizing individual selection probabilities has clear appeal, the *column generation* technique used by their algorithms comes with several drawbacks. In particular, the resulting distributions' support consists of only few panels, which can cause extreme correlations in the selection of several pool members. A wider problem is that many distributions over panels induce the same individual selection probabilities, leaving the choice of distributions to implementation details. These limitations present an opening for new algorithms with complementary strengths.

In this paper, we design selection algorithms attaining a different objective: maximizing the randomness, specifically the *entropy* [50], of the probability distribution. If the only constraints are the practitioners' quotas, this means choosing the panel uniformly from the set of all possible panels. If organizers prescribe the pool members' selection probabilities to ensure fairness as above, our algorithms sample from the maximum-entropy distribution subject to this constraint.

Sampling from maximum-entropy distributions is attractive for several reasons:

- In a straight-forward way, the maximum entropy distribution makes the sortition **as random as possible** given the constraints. It is well known that the maximum entropy distribution minimizes the Kullback–Leibler divergence to the uniform distribution over all subsets of the desired panel size [14, Eq. (2.94)], so our distribution is **as close to a uniform selection** without replacement from the pool as possible given the constraints.
- Our approach follows the **principle of maximum entropy** of Jaynes [34, 35], who argues that, among all distributions satisfying known constraints, the maximum-entropy distribution is the only one that avoids introducing "arbitrary assumption of information" [34]. He aims to estimate the expected value of a function on the random variable, which captures the central

[1] These algorithms are MaxiMin, its refinement LexiMin [22], and Goldilocks [5], which differ in their optimized measure of closeness to equality.

[2] https://github.com/sortitionfoundation/stratification-app/

question of how much representation in the panel a group not protected by quotas should have (on average). Jaynes' [34] reassures us that, by drawing from the maximum-entropy distribution, we represent all such groups according to the best guess given our information.
- We found our choice of distribution **simple to explain** to practitioners and politicians since it can be defined by the following rejection procedure: uniformly draw the desired number of pool members,[3] check if it happens to satisfy all quotas, and repeat this until a valid panel is found. Though this process is hopelessly slow, and our selection algorithm quite non-trivial, our algorithm only implements a speed-up of this common-sense, intuitively fair procedure.
- Dowlen [17] and Stone [53], influential philosophers of sortition, value randomness as a tool to decide between equally acceptable options **without any influence by illicit reasons**, which is what we do when choosing uniformly between panels satisfying the quotas. In contrast to the goal of equalizing selection probabilities, maximizing entropy defines a unique probability distribution, which means that the distribution entirely emerges from the input (the quotas, which practitioners already must justify). This leaves **no room for discretion by the organizers or algorithm** that could raise a "suspicion of bias" [11]. A maximum-entropy distribution also makes the result of the sortition as unpredictable as possible, which is seen as an obstacle to undue outside influence such as bribery [53].

1.1 Our Techniques and Results

Sampling from a maximum-entropy distribution over valid panels is clearly a hard algorithmic problem because even deciding if a valid panel exists is NP-complete [22]. What allows us to nevertheless solve the problem is a heavily optimized code implementation and a sequence of algorithmic optimizations exploiting the structure of practical sortition problems.

At the heart of our algorithm is a dynamic program (DP) for counting valid panels, similar to Papadimitriou's [48] pseudo-polynomial algorithm for integer linear programming with a constant number of constraints. For a typical medium-sized assembly, there are 20 to 30 quotas, which is limited but hopeless to include in this DP given its exponential running-time dependency. By exploiting symmetries between pool members, going through pool members and quotas in a smart order, and aggressively pruning states, we scale the DP to as many quotas as possible. Though this rarely scales to all quotas, it allows us to uniformly sample from the candidate panels that satisfy the incorporated quotas and to enforce the remaining quotas with rejection sampling. The resulting algorithm, MaxEntropy, can uniformly sample among all feasible panels for 78 out of 86 real-world sortition problems, leaving out only instances with unusually many quotas.

To ensure fair selection probabilities, our algorithm FairMaxEntropy can take in target selection probabilities, which might be computed by column generation. We formulate finding the maximum-entropy probability distribution as a convex problem with a constraint on the selection probabilities and consider its dual, which has fewer variables. Each solution of the dual corresponds to an exponential-family distribution over panels, from which we can sample by using a variant of the DP above as an oracle. By estimating the pool members' selection probabilities through sampling, we obtain an estimator for the gradient of the dual objective and optimize it using stochastic gradient descent. We prove that FairMaxEntropy always produces a maximum-entropy distribution, whose selection probabilities converge to the targets at a rate of $O(1/\sqrt{T})$ in the number of iterations T.

Next, we study the properties of our selection algorithms, theoretically (Section 5) and empirically (Section 6). From a theoretical perspective, MaxEntropy and FairMaxEntropy coincide with stratified sampling if the quotas are over disjoint strata and satisfy a notion of equal treatment of

[3] If there are constraints on selection probabilities, each pool member has an adjusted probability of being selected in this stage, as we describe in Section 4. The rest of the process is unchanged.

equals not satisfied by column generation. We prove that MaxEntropy satisfies asymptotically optimal bounds on resistance to misrepresentation of features as defined by Flanigan et al. [25] and how our algorithms can be implemented as a physical lottery to transparently demonstrate its selection probabilities [24].

Empirically, we evaluate our algorithms on a dataset of real-world sortition problems that is several times larger than the datasets of earlier papers, and compare them to existing selection algorithms. In terms of fairness, pure MaxEntropy lies between the fairness of Legacy (an algorithm developed by the Sortition Foundation) and the column generation approaches. Already few iterations of gradient descent seem to clearly increase fairness. We find that the panels produced by our algorithms tend to have higher intersectional diversity than those produced by column generation. If we hold out one category of quotas from the quotas (say, age), our algorithms are more likely to satisfy the held-out quotas by chance than column generation algorithms, primarily because column generation satisfies a fair fraction of features with zero probability.

Finally, we describe in Section 7 how we are deploying MaxEntropy on *panelot.org*, to facilitate its adoption in real-world assemblies.

1.2 Related Work

We already touched on several works on sortition algorithms [5, 22, 24, 25]. By providing a practical algorithm for maximum-entropy sortition, our paper resolves a question left open by Flanigan et al. [22], who also discuss potential strengths of maximum-entropy sortition in resisting outside influence. Similarly, Flanigan et al. [23] call for "selection algorithms for goals beyond individual fairness". Other works on sortition algorithms aim to equalize the probabilities of *constituents* to be selected rather than those of pool members [26] and study how to replace exiting panel members [4]. Do et al. [16] study the recruitment of assembly members by random dialing (as done in French assemblies), which makes selection an online problem, and Gölz et al. [31] study how to compose the pool if population registers are located in municipalities (as in German assemblies).

Two recently proposed selection algorithms are *Diversimax* by Matar [41] and *simulated annealing* by Gerwin et al. [28]. Diversimax only selects a deterministic panel, maximizing some notion of diversity by mixed integer linear programming, which is why it cannot be compared in terms of its random properties. The simulated annealing algorithm, on the other hand, does not respect quotas, which is why we can also not compare it with other algorithms on equal terms. Gerwin et al. [28] stress as a key advantage over LexiMin that, among 10,000 sampled panels, almost all are distinct; MaxEntropy provides an even wider range of panels while additionally satisfying quotas.

Finally, several papers study sortition when assembly members can be directly picked from the population, and how well the assembly represents the underlying population. This setting is studied in terms of representation of features without quotas [7], the welfare of majority votes performed by the assembly [42], and representativeness measures in a metric space [10, 18, 19].

On a technical level, our work is related to papers on sampling from combinatorial structures based on uniform and general maximum-entropy distributions [2, 3, 36]. Our paper uses a well-known reduction from sampling and counting, made formal by Jerrum et al. [37]. Specifically for maximum-entropy distributions over combinatorial sets, Singh and Vishnoi [51] establish the equivalence between counting and optimizing via the ellipsoid method. Other approaches for counting combinatorial structures include Markov Chain Monte Carlo [36, 38, 43] and generating functions [6, 15].

2 Preliminaries

Sortition. An instance of the sortition problem consists of a set of *pool members* $N = [n] = \{1, \dots, n\}$, a set of features F, the values of these features V, the features' lower and upper quotas

ℓ, u, and a desired panel size k. Each pool member is characterized by their value for all the features $f \in F$. For instance, one feature $f \in F$ might be *gender*, and its possible values V_f be {female, male, other}. We write each feature as a function $f : N \to V_f$, so that $f(i)$ gives pool member i's gender. For each *feature-value pair* $f, v \in FV \coloneqq \{(f, v) : f \in F, v \in V_f\}$, we are given integer quotas $0 \leq \ell_{f,v} \leq u_{f,v} \leq k$ specifying the minimum and maximum allowed number of selected panel members with feature f taking value v.

At times, it will be convenient to consider pool members' attributes in the shape of a characteristic vector. Let $w : N \to \{0,1\}^{FV}$ be this mapping from pool members to *feature-value vectors*, so that $w(i)_{f,v} = \mathbb{1}[f(i) = v]$ for all $f, v \in FV$. We denote the set of distinct feature-value vectors in the pool by $\mathcal{W}$. For any set of feature-value vectors $W \subseteq \{0,1\}^{FV}$, we denote the subset of pool members with features in W by $N|_W \coloneqq \{ i \in N : w(i) \in W\}$.

A *panel* is a set of k pool members such that, for every feature-value pair $f, v \in FV$, the number of pool members with feature f taking value v is between $\ell_{f,v}$ and $u_{f,v}$. We denote the set of all panels by $\mathcal{P}$, and the set of panels containing some pool member $i \in N$ by $\mathcal{P}(i)$. A *selection algorithm* takes in the instance, and randomly selects a panel.

Given a *panel distribution* $\lambda \in \Delta(\mathcal{P})$, pool member i's *selection probability* is $\pi_\lambda(i) \coloneqq \mathbb{P}_{P\sim\lambda}[i \in P] = \sum_{P \in \mathcal{P}(i)} \lambda(P)$. When λ is clear from the context, we drop it from the subscript. Note that sum of selection probabilities $\sum_{i \in N} \pi_\lambda(i) = k$ is constant for all panel distributions.

A *fairness measure* is an objective intended to quantify how evenly the selection probability is distributed across pool members. Formally, a fairness measure is a concave function $g : [0,1]^N \to \mathbb{R} \cup \{-\infty\}$ that maps a vector of selection probabilities to a *fairness score*. A panel distribution maximizes a fairness measure if its induced selection probabilities maximize the fairness measure.

Convex analysis. We review some standard notions from convex analysis, which we will use for the development of FairMaxEntropy in Section 4 see Rockafellar [49] for a comprehensive treatment. Given a set $U \subseteq \mathbb{R}^d$, its *affine hull* $\mathrm{aff}(U)$ is the set of all affine combinations of elements of U. For a $x \in \mathbb{R}^d$, we define the ℓ_∞ *ball* as $B_\infty(x, \eta) \coloneqq \{ y \in \mathbb{R}^d : \|x - y\|_\infty \leq \eta\}$. The *$\eta$-relative interior* of a set U is defined as $\mathrm{rI}_\eta(U) \coloneqq \{ x \in U : B_\infty(x, \eta) \cap \mathrm{aff}(U) \subseteq U\}$.

Given a distribution $\gamma \in \Delta(S)$ over a finite set S, we define its *(Shannon) entropy* as $H(\gamma) \coloneqq -\sum_{s \in S} \gamma(s) \log \gamma(s)$. Since entropy is strictly concave, it has a unique maximizer over any convex compact set of distributions, so the maximum-entropy distribution is uniquely defined, with or without additional constraints on selection probabilities.

3 MaxEntropy: Sampling Uniformly among Possible Panels

When the only restrictions on the sortition are the quotas, the maximum-entropy distribution coincides with the uniform distribution. In this section, we describe how our algorithm MaxEntropy samples from this distribution. We defer all proofs to Appendix A, where we directly describe a generalization of this algorithm. This generalization samples panels with different probabilities based on some pool member-specific weights and will be useful for our development of fair maximum-entropy algorithms in Section 4.

To sample uniformly from the feasible panels, MaxEntropy needs to solve a problem on practical instances that is in general intractable. Indeed, sampling panels is clearly at least as hard as finding one, and deciding whether a panel exists is NP-hard, even for only three features by reduction from exact 3-cover [22].[4] Our quota constraints are so expressive that MaxEntropy sortition is equivalent to uniformly sampling from the solutions of a quite general class of 0–1 integer linear programs, which underscores the difficulty of the problem. General tools for this problem, like

[4] Even with the promise that a panel exists, no algorithm can find one in polynomial time unless P = NP, and no polynomial-time algorithm can find one with constant probability unless RP = NP [22].

LattE [15] scale badly in the number of variables (i.e., pool members). The one piece of good news from complexity theory is that the number of quotas tends to be somewhat bounded in practice, and that 0–1 integer linear programs with a constant number of constraints can be solved in polynomial-time [48]. Indeed, a similar dynamic program will be the starting point of our algorithm, its running time on even a smaller instance would be at least on the order of $k^{|FV|} \approx 30^{20} \approx 3 \cdot 10^{29}$, so hopelessly inefficient.

3.1 Counting Panels by Dynamic Programming

It is well known that uniform sampling over combinatorial structures can be reduced to counting the number of solutions [37, 44]. We will begin by formulating a naïve dynamic program for counting the number of panels, and then describe three optimizations that exploit the structure of real-world sortition problems to speed up the computation. In addition to an optimized C++ implementation with careful memory management (see Section 6.1), we need all these optimizations, in addition to the later rejection sampling stage, to be able to sample from highly constrained real-world sortition instances.

For a set of pool members $U \subseteq N$, define its *(feature-value) profile* as $w(U) \coloneqq \sum_{i \in U} w(i)$. We denote the set of *quota-compliant profiles* by

$$\overline{\mathcal{Z}} \coloneqq \left\{ z \in \mathbb{N}^{FV} : \ell_{f,v} \leq z_{f,v} \leq u_{f,v} \text{ for all } f, v \in FV \text{ and } \sum_{v \in V_{f_0}} z_{f_0,v} = k \text{ for some fixed } f_0 \in F \right\},$$

where the latter constraint uses the fact that $\sum_{v \in V_f} w(i)_{f,v} = 1$ for any pool member i and feature f to count the number of pool members. The set U constitutes a panel if and only if $w(U) \in \overline{\mathcal{Z}}$, though some profiles in $\overline{\mathcal{Z}}$ may be unrealizable. Our dynamic program calculates the following *counting function*:

$$\varphi : [n] \times \mathbb{N}^{FV} \to \mathbb{N}, \quad (i, z) \mapsto |\{ U \subseteq \{i, \dots, n\} : z + w(U) \in \overline{\mathcal{Z}} \}|,$$

which counts the number of ways to complete a partial profile z to a quota-compliant profile using only a suffix of pool members $\{i, \dots, n\}$. In particular, $\varphi(1, \vec{0}) = |\mathcal{P}|$ computes all possible panels. We may compute the counting function according to the following recursion, based on the simple idea that the ways for completing z with the pool members $\{i, \dots, n\}$ can be partitioned into those that choose i and those that do not.

Proposition 3.1. *For every $i \in N$ and $z \in \mathbb{N}^{FV}$, we have that*

$$\varphi(i, z) = \varphi\big(i + 1, z + w(i)\big) + \varphi(i + 1, z),$$

with the base case $\varphi(n + 1, z) = \mathbb{1}[z \in \overline{\mathcal{Z}}]$.

Since the number of profiles that can possibly be completed to be quota-compliant is at most $(k + 1)^{|FV|}$ and since we do a constant amount of operations per state, we may bound the running time by $O\big(n\,(k + 1)^{|FV|}\big)$, which as discussed above is hopelessly large.

Processing identical pool members simultaneously. When several pool members have the same feature-value vector and we place them next to each other in the ordering of N, we can perform the recurrence for these pool members in a single step, in which we decide how many of them to choose. Specifically, if pool members $i, i + 1, \dots, i + n_\omega - 1$ have the feature-value vector $\omega = w(i) = \cdots = w(i + n_\omega - 1)$, where n_ω is the number of pool members with feature-value vector ω, we can write

$$\varphi(i, z) = \sum_{d=0}^{n_\omega} \binom{n_\omega}{d} \cdot \varphi(i + n_\omega, z + d \cdot \omega) \quad \text{for each } z \in \mathbb{N}^{FV}.$$

While this does not improve the theoretical complexity, and instances with many quotas have few entirely identical pool members, it will pay off in combination with a later optimization.

Decreasing necessary state space for one feature. To further reduce the running time, we sort the pool members by their value for some chosen feature f^*. The benefit of this approach is that, in the recurrence step for the last pool member with some value $v \in V_{f^*}$, we can immediately decide whether the quotas for f^*, v are violated (so the count is zero) or satisfied (so this quota will be satisfied for all completions). As a result, the dynamic program only needs to keep track of the profile for the current value of f^* rather than all of them, which reduces the running time to $O\big(n\,(k+1)^{|FV|-|V_{f^*}|+1}\big)$.

This optimization is highly effective because many sortition problems include a feature for geographical location, which can have a large number of values. If there are, say, 10 regions in the country or 10 neighborhoods in a city, processing the pool members by geography reduces the running times by a considerable factor of $(k+1)^9$.

Pruning states based on subsets of the features. In our experience, a large majority of states cannot be completed to a quota-compliant profile, even when only a portion of the quotas is considered. One reason for this is that practitioners often set the lower and upper quotas with a narrow gap between them (often around the census percentages [29, 46]). Moreover, some feature-values, especially those of populations that opt into the pool at lower rates, cannot vary entirely independently on each other.

To avoid spending large amounts of memory and computation on states that cannot be completed, we iteratively compute versions of the dynamic program that include more and more features. Fix a subset of features $F' \subseteq F$, set $z|_{F'}$ for the restriction of a profile z to the features in F', and define the counting function φ' for only the features F' as above. The key insight is that the counting function for F' gives an upper bound on the full counting function:

Proposition 3.2. *For all $i \in N$, $z \in \mathbb{N}^{FV}$, and $F' \subseteq F$, we have that $\varphi(i,z) \le \varphi'(i, z|_{F'})$.*

Hence, we can prune all states (i,z) with $\varphi'(i, z|_{F'}) = 0$.

We found this pruning procedure to be a large improvement over direct pruning steps that do not take into account the interdependency of features. Our approach of iteratively increasing the number of features also makes the first optimization pay off to a much greater degree because much more pool members appear identical when only considering a subset of the features.

This approach also raises the question of in which order features should be added to the dynamic program. We use an intelligent heuristic which, in each iteration, prioritizes features that are rarely satisfied among a large number of panels sampled from the previous iteration's dynamic program. The goal of this heuristic is to rule out a large number of dynamic programming states, and to prioritize the features that cannot be addressed well through rejection sampling as described below.

3.2 Uniformly Sampling a Panel

After having computed the dynamic program for any subset of the features, we can use it to uniformly sample from all sets of k pool members that satisfy the incorporated features. For exposition, we describe this process for the naïve dynamic program without the optimizations.

To sample using the dynamic program, we follow the recurrence of the dynamic program, building our panel person by person. We start at the state $(1, \vec{0})$ and with an empty panel. At each state (i,z), we add pool member i to the panel with probability

$$\frac{\varphi(i+1, z+w(i))}{\varphi(i,z)} = \frac{\varphi(i+1, z+w(i))}{\varphi(i+1, z+w(i)) + \varphi(i+1,z)},$$

which gives the ratio of completions that include pool member i. If i is included in the panel, the algorithm recurses to state $(i+1, z+w(i))$, otherwise, to state $(i+1, z)$. The algorithm stops when it reaches a base state, at which point the panel is guaranteed to be feasible by Proposition 3.1.

Enforcing remaining features with rejection sampling. If we can compute the dynamic program for all features, the above is sufficient for our goal of uniform sampling. Most of the time, however, this is only feasible for a subset of the features.

As we iteratively build the dynamic program for larger subsets $F' \subseteq F$ of features, we keep sampling from each dynamic program, uniformly among all pool member sets of size k that satisfy the quotas for F'. This sampling helps us choose the next feature to add, but it can also reach the point where a sample satisfies all remaining quotas. In this case, this panel is a uniform sample from the set of all panels, and can be returned:

Theorem 3.3. *Given an instance of a sortition problem, MaxEntropy samples a panel uniformly from the set of all panels.*

4 FairMaxEntropy: Sampling with Prescribed Selection Probabilities

We see the MaxEntropy algorithm as a serious contender for use in practical citizens' assemblies, in particular due to its ease of explanation. But many practitioners use algorithms that optimize the fairness of selection probabilities and may be reluctant to give up their optimal fairness properties.

With these practitioners in mind, we develop a selection algorithm called FairMaxEntropy, which targets selection probabilities prescribed by the practitioners and draws from the maximum-entropy distribution subject to this constraint. If practitioners derive these marginals from the output of LexiMin or Goldilocks, they retain the optimal fairness guarantees provided by these algorithms, without suffering from the limitations of column generation (small support, lack of unique specification) we describe in the introduction.

Optimization formulation and structure. To find these maximum-entropy distributions, we rely on convex optimization. Fix a sortition instance, and let π denote the set of target selection probabilities. The goal of FairMaxEntropy is to sample panels from the distribution λ_π^* with maximum entropy satisfying this constraint. To achieve this, we compute a solution to the following pair of primal and dual entropy-regularized convex programs (see [8]):

$$\begin{aligned} &\max H(\lambda) && \\ &\text{s.t. } \pi_\lambda = \pi && (\mathrm{P}(\pi)) \\ &\quad \lambda \in \Delta(\mathcal{P}). && \end{aligned} \qquad\qquad \begin{aligned} &\min \ \mathrm{LSE}(A^\intercal \theta) - \langle \theta, \pi \rangle && \\ &\text{s.t. } \theta \in \mathbb{R}^N. && (\mathrm{D}(\pi)) \end{aligned}$$

In the dual, A is the *incidence matrix* of $\mathcal{P}$, that is, $A \in \{0,1\}^{N \times \mathcal{P}}$ where $A_{i,P} = \mathbb{1}[i \in P]$ for all $i \in N$ and $P \in \mathcal{P}$, and LSE denotes the *log-sum-exp* function

$$\mathrm{LSE}(A^\intercal \theta) = \log\left(\sum_{P \in \mathcal{P}} \exp\left(\sum_{i \in P} \theta(i)\right)\right).$$

Whereas the primal program has an exponentially large number of variables, the dual has only n and might therefore be more amenable to optimization. We begin by relating the primal and dual optimal solutions of the convex program.

Proposition 4.1. *A panel distribution λ such that $\pi_\lambda = \pi$ and a vector $\theta \in \mathbb{R}^N$ are an optimal primal–dual pair iff*

$$\lambda(P) = \frac{\exp(\sum_{i \in P} \theta(i))}{\sum_{Q \in \mathcal{P}} \exp\left(\sum_{i \in Q} \theta(i)\right)} \quad \text{for all } P \in \mathcal{P}. \tag{1}$$

Whenever π is contained in the relative interior of the vector of selection probabilities.

This proposition shows that the maximum-entropy distribution λ_π^* has a compact representation in exponential-family form when parameterized by the optimal dual variables (see [8, 51] for a proof of the statement). It also shows that, for any $\theta \in \mathbb{R}^N$, the exponential-family distribution λ defined by Equation (1) is an optimal primal solution for the primal (P(π_λ)), i.e., maximizes entropy among all distributions with the same selection probability vector. It will be natural to write these exponential-family distributions in a multiplicative form, as

$$\lambda_\mu(P) \coloneqq \frac{\prod_{i\in P} \mu(i)}{\sum_{Q\in\mathcal{P}} \prod_{i\in Q} \mu(i)} \quad \text{for all } P \in \mathcal{P}$$

for given weights $\mu \in \mathbb{R}^N_{>0}$, where $\mu(i)$ plays the role of $e^{\theta(i)}$.

This characterization shows that any exponential-family distribution can still be described by a variant of the rejection sampling procedure for uniform sampling we described in the introduction. Define a probability distribution over the pool members, in which pool member i has mass proportional to $\mu(i)$. We draw k pool members with replacement from this distribution; check if they are all distinct and happen to satisfy the quotas; and repeat this process until we find a valid panel. Clearly, each panel P's probability of being sampled in this process is proportional to $\prod_{i\in P} \mu(i)$, which shows that the procedure implements λ_μ. This means that, if assembly organizers can explain the different weights $\mu(i)$ for the pool members (say, giving higher weight to those appearing on few panels), the resulting probability distributions are still aligned with common sense.

Selection algorithm. If we know $\mu \in \mathbb{R}^N_{>0}$, a generalization of our algorithm in Section 3 can sample panels from its associated distribution λ_μ. To do so, our dynamic program should no longer simply count the number of panels, but compute a weighted sum $\sum_{P\in\mathcal{P}} \prod_{i\in P} \mu(i)$, which is easy to change (see Appendix A.1).[5] So all that remains is to find the optimal dual weights μ^* that give us the correct selection probabilities.

The dual (D(π)) has few variables, but the log-sum-exp term in its objective contains a sum ranging over all exponentially many panels. What helps us optimize it is that the gradient of the dual objective at a point θ is simply

$$\left(\frac{\sum_{P\in\mathcal{P}(i)} \exp(\sum_{j\in P} \theta(j))}{\sum_{P\in\mathcal{P}} \exp(\sum_{j\in P} \theta(j))} - \pi(i)\right)_{i\in N} = \left(\sum_{P\in\mathcal{P}(i)} \lambda_\mu(P) - \pi(i)\right)_{i\in N} = \pi_{\lambda_\mu} - \pi$$

for the weights $\mu(i) = e^{\theta(i)}$.

It is a remarkable coincidence that computing the gradients reduces to the one thing we know how to do — sampling many panels according to the exponential-family distribution λ_μ and using them to estimate the pool members' selection probabilities. Specifically, the empirical frequency $\overline{\pi}(i)$ of drawn panels that include pool member i is an unbiased estimator for $\pi_{\lambda_\mu}(i)$. By subtracting the known vector π from $\overline{\pi}$, we obtain an unbiased estimator for the gradient and can optimize the dual using stochastic gradient descent.

Convergence bounds. Let $\mathcal{M} \coloneqq \{\, \pi_\lambda : \lambda \in \Delta(\mathcal{P}) \}$ be the *polytope of selection probabilities*. Assume for now that the vector of target selection probabilities π lies in $\mathrm{rI}_\eta(\mathcal{M})$ for some $\eta > 0$ so that strong duality holds. (Further below, we discuss the necessity of this assumption and what can be done when it does not hold.) We can then give following convergence bound.

[5] Here, we ignore questions of numerical precision in the dynamic program, and treat it as an oracle. In our practical implementation, we round the $\mu(i)$ to be able to represent the values of the dynamic program with bounded integers, which entails some error.

Theorem 4.2. *Given a sortition instance and target selection probabilities $\pi \in \mathrm{rI}(\mathcal{M})$, run* FairMaxEntropy *for T iterations, and let λ be the distribution obtained from the last iterate. Then, λ is an optimal solution for (P(π_λ)) (i.e., the maximum-entropy distribution subject to its own selection probabilities) and $\mathbb{E}\left[\|\pi_\lambda - \pi\|_2^2\right] \leq O(1/\sqrt{T})$.*

Proof. We define FairMaxEntropy by minimizing (D(π_λ)), using stochastic gradient descent with the step-size schedule of Jain et al. [33], using the dynamic program as an oracle for the gradient. (This step-size rule ensures that we can give guarantees for the last iterate, rather than standard bounds which hold on average over the iterates.)

To apply their convergence result, we must show that the dual objective $d(\theta) \coloneqq \mathrm{LSE}\,(A^\top\theta) - \langle\theta, \pi\rangle$ is convex, Lipschitz continuous, that there is a dual optimal solution with bounded norm, and that the estimated gradients have bounded norm. The dual objective d is a convex because the log-sum-exp function is convex. Its gradient has bounded norm $\|\pi_{\lambda_\mu} - \pi\|_2 \leq \|\pi_{\lambda_\mu}\|_2 + \|\pi\|_2 \leq 2\sqrt{k}$, and so does any estimated gradient $\|\overline{\pi} - \pi\|_2 \leq 2\sqrt{k}$. Finally, the η-interior condition implies that there is a dual optimum solution θ^* with $\|\theta^*\|_2 \leq k\log(n)/\eta$ [51, Thm. 4.2]. Taking θ as the solution obtained at iteration T, Theorem 1 of Jain et al. [33] implies that

$$\mathbb{E}\left[d(\theta) - d(\theta^*)\right] \leq O(1/\sqrt{T}).$$

The dual function is k-smooth [9, Lem. 3.10], which together with convexity implies that the norm of the gradient decays proportionally to the suboptimality gap. Formally, for any $\theta' \in \mathbb{R}^N$, it holds that $\|\nabla d(\theta')\|_2^2 \leq 2k(d(\theta') - d(\theta^*))$. Therefore, taking $\lambda \coloneqq \lambda_{e^\theta}$ as the distribution obtained from the last iterate, we conclude that

$$\mathbb{E}\left[\|\pi_\lambda - \pi\|_2^2\right] = \mathbb{E}\left[\|\nabla d(\theta)\|_2^2\right] \leq 2k\,\mathbb{E}\left[d(\theta) - d(\theta^*)\right] \leq O(1/\sqrt{T}). \qquad \square$$

This $O(1/\sqrt{T})$ convergence rate is the best we can hope for our optimization problem since the dual objective is not strongly convex [1]. If π indeed lies in the relative interior, the theorem shows that FairMaxEntropy can sample from a maximum-entropy distribution with selection probabilities that are arbitrarily close to the desired vector π.

Discussion of relative-interior assumption. Recall that our theorem assumes that the target marginals π lie in the relative interior of $\mathcal{M}$. If, instead, π lies on its boundary, then any distribution λ^* must lie on a facet of $\Delta(\mathcal{P})$, because $\pi = \pi_{\lambda^*} = A\,\lambda^*$, and the linear function A (i.e., the incidence matrix defined below the convex programs) maps the interior of $\Delta(\mathcal{P})$ to the relative interior of $\mathcal{M}$. This means that, if π is not in the relative interior, some panels have zero mass in λ^*, which is not possible for any exponential-family distribution and can only be approximated by having some $\theta(i)$ become very negative. As a result, θ diverges in the optimization.

Given target selection probabilities π on the boundary of $\mathcal{M}$, we can fix this issue by conducting an interiorization step. Let λ_{unif} be the uniform distribution over panels, which clearly lies in the relative interior of $\Delta(\mathcal{P})$. As we argued above, its marginals must lie in the relative interior of $\mathcal{M}$. Hence, one can introduce a small perturbation to π in the direction of $\pi_{\lambda_{\text{unif}}}$ to obtain a π' that is in $\mathrm{rI}(\mathcal{M})$. Since $\pi_{\lambda_{\text{unif}}}$ is not known, we can only estimate it by sampling from MaxEntropy, which might find a point in the relative interior, but is not guaranteed to.

Approaching arbitrary target probabilities. The good news is that FairMaxEntropy can actually approach any target vector π — even ones that are not realizable (i.e., not in $\mathcal{M}$), in which case the selection probabilities of FairMaxEntropy approach the projection of π on the feasible set $\mathcal{M}$. This follows from very recent results by Hirai and Sakabe [32], who show that gradient descent converges to the minimizer of the gradient norm (in our case, $\|\pi_\lambda - \pi\|_2$, where λ is the primal distribution for θ) in the limit, even when the function is unbounded. Though no meaningful

bounds are known for the speed of convergence of the gradient, this means that our algorithm is robust to marginals on the boundary of $\mathcal{M}$, or to errors in the target probabilities π.

Since gradient descent optimizes $\|\pi_\lambda - \pi\|_2$, one might be tempted to set $\pi = (\frac{k}{n}, \dots, \frac{k}{n})$, to let FairMaxEntropy find the fairest selection probabilities without prior computation. This works, in a way, and maximizes the fairness measure defined as the negative ℓ_2 norm of selection probabilities from the point π of perfectly equal probabilities. Unfortunately, this happens to be a rather poor fairness measure, which tends to select many pool members with zero probability. This has been previously observed by [25, App. D.7] for a relaxation of the sortition problem.

5 Theoretical Properties

In the introduction, we gave arguments that make maximizing entropy an attractive objective on a conceptual level. In this section, we add desirable properties satisfied by MaxEntropy and FairMaxEntropy requiring more mathematical elaboration and discuss our algorithms according to properties of selection algorithms studied in earlier papers.

5.1 Simplicity of Distributions

If a citizens' assembly is to legitimately speak for a population, it is important that the sortition process choosing its members be publicly trusted [27]. This poses a challenge because all selection algorithms, by virtue of solving an NP-hard problem, are somewhat complex and hard for a layperson to understand. This difficulty, however, can be ameliorated by showing that selection algorithms work in a simple way in combinatorially straight-forward instances, or by stressing natural properties of the distributions.

One sampling algorithm that is broadly understood (and referenced in practitioners' explanations of sortition [40]) is *stratified sampling*. Stratified sampling only applies to a subsetting of the sortition problem, in which there is a single feature and there are no gaps between lower and upper quotas. For instance, one might stratify by age group, with the values "18–29", "30–44", "45–64", and "65+" and respective targets 6, 8, 7, and 4, for a total of 25 members. The set of pool members with a given value is called a *stratum*. To produce the panel, stratified sampling draws, from each stratum, a uniform subset of the targeted size and returns their union.

In this restricted subsetting, we show that MaxEntropy coincides with stratified sampling. The fact that MaxEntropy generalizes a common-sense selection procedure to more general settings supports the notion that this algorithm itself implements a natural panel distribution. By contrast, column generation algorithms do not coincide with stratified sampling, which can be seen from the fact that their support captures only a vanishing fraction of possible panels.

Proposition 5.1. *MaxEntropy is equivalent to stratified sampling in the case of a single feature and tight quotas.*

Proof. Consider a sortition instance with a single feature f, values $V_f = \{v_1, \dots, v_m\}$, and targets $k_{v_j} = \ell_{f,v_j} = u_{f,v_j}$ for $j \in [m]$. Set $N_j \coloneqq \{\, i \in N : f(i) = v_j \}$ for the stratum belonging to v_j. For this instance to be feasible, it must hold that $k = \sum_{j=1}^{m} k_j$. It suffices to observe that

$$\mathcal{P} = \{\, P \subseteq N : |P| = k \text{ and } |N_i \cap P| = k_i \text{ for all } i \in [m] \}$$

also describes the set of all panels that can be selected by stratified sampling, and that each of these is selected with an equal probability, namely

$$\prod_{j=1}^{m} 1 / \binom{|N_i|}{k_i}.$$

□

One attractive property of stratified sampling is that it treats pool members with the same value perfectly symmetrically. Not only do two such pool members have the same probability of being selected but they, for example, also have the same probability of being selected jointly with a third pool member. We generalize this observation in the following axiom.

Definition 5.2. A panel distribution λ satisfies *higher-order equal treatment of equals* if, for any two pool members i, j with identical feature-value vectors and, if applicable, equal target selection probabilities, the distribution λ is invariant under swapping i and j. That is, if $\tau_{i,j}P$ denotes the swapping of i for j and j for i in P, and if $\tau_{i,j}\lambda$ is the distribution defined by $(\tau_{i,j}\lambda)(P) \coloneqq \lambda(\tau_{i,j}P)$ for all $P \in \mathcal{P}$, it holds that $\lambda = \tau_{i,j}\lambda$.

Note that this notion is substantially stronger than the *equal treatment of equals* defined by Flanigan et al. [22, App. 15.3], who only require i and j to have equal selection probabilities.

Theorem 5.3. *Given a sortition instance, the probability distributions of* MaxEntropy *and* FairMaxEntropy *(when optimized to optimality) satisfy higher-order equal treatment of equals.*

Proof. For MaxEntropy, this is immediate since all panels have an equal probability (and swapping pool members with identical feature-value vectors preserves quota satisfaction).

Now, let λ^* denote the optimal panel distribution for FairMaxEntropy, i.e., the optimal solution to (P(π)). Fix two pool members i, j with equal feature-value vectors and target probabilities $\pi(i) = \pi(j)$, and consider $\tau_{i,j}\lambda^*$ as defined above. Clearly, $\pi_{\tau_{i,j}\lambda^*}(i) = \pi_{\lambda^*}(j) = \pi(j) = \pi(i)$, $\pi_{\tau_{i,j}\lambda^*}(j) = \pi_{\lambda^*}(i) = \pi(i) = \pi(j)$, and all other pool members' selection probabilities are also unchanged. Hence, $\tau_{i,j}\lambda^*$ is also a feasible solution to (P(π)). Since we just permuted the probability masses of panels, $\tau_{i,j}\lambda^*$ must furthermore have the same, maximal entropy as λ^*. Since the entropy objective is strictly concave, the maximizer is unique, and hence $\lambda^* = \tau_{i,j}\lambda^*$. □

5.2 Resistance to Manipulation

When recruiting the members for an assembly, the pool members report their own feature-value vectors when opting into the pool. This process might be vulnerable to misreporting by a group of malicious agents who seek to strategically change their own and others' selection probabilities.

Under the assumptions made by Flanigan et al. [25], who first studied this issue theoretically, MaxEntropy achieves optimal manipulability across three measures. *Internal manipulability* measures the largest increase in selection probability that can be achieved for one of the members of the manipulating coalition; *external manipulability* measures the largest decrease in selection probability that can be imposed on a non-coalition member; and *composition manipulability* measures the largest amount of seats that a malicious group can, in expectation, misappropriate in favor of some group.

Flanigan et al. [25] showed that both LexiMin and Nash are arbitrarily manipulable; that is, as the pool grows, malicious agents can raise their selection probabilities by almost 1. They also show that minimizing the ℓ_∞ distance to the uniform vector of selection probabilities achieves optimal manipulability. There are two drawbacks here. First, minimizing the ℓ_∞ distance typically results in many pool members having zero selection probability, making it a poor choice of algorithm. Second, their analysis is in a continuous relaxation, meaning that their algorithms only satisfy ex-ante targets rather than ex-post quotas.

We show that, on the same set of assumptions, adapted to the non-relaxed case with ex-post quotas, MaxEntropy also achieves optimum manipulability for all three measures, and it does so without introducing the degeneracies of the ℓ_p norm.

Theorem 5.4 (Resistance to Manipulation, Informal statement). *Given a sortition instance satisfying an equivalent set of assumptions as the ones described in [25], MaxEntropy achieves optimum manipulability.*

We defer a formal treatment of this result to Appendix B.1. Goldilocks also achieves a notion of optimum manipulability [5], but under stronger assumptions on this instance, which makes these results hard to compare.

5.3 Transparency

Now we turn our attention to the *transparency* of the random selection. To show that pool members are indeed selected with their fair selection probabilities (rather than included based on the whims of the organizers), Flanigan et al. [22, 24] perform part of the panel selection in a public event, in what they call a *transparent lottery*. Specifically, having generated a panel distribution with column generation, they produce from this lottery a list of m (say, 1000) panels, possibly with duplicates. Then, they draw the final panel uniformly from these m options, using a publicly observable process such as a lottery machine.

Flanigan et al. [24] propose a rounding algorithm that, given an explicit representation of the distribution with small support, obtains a list of m panels, such that the selection probabilities when drawing from this list does not deviate much from the selection probabilities of the original distribution, say by at most $O(k/m)$ for each pool member. Their approach does not apply to the distributions of MaxEntropy and FairMaxEntropy, since it relies on a distribution that is explicitly given and has small support. The best we can do is to sample m panels independently from our distribution, in which case a straight-forward concentration bound yields:

Proposition 5.5. *Let λ be any panel distribution, and create a transparent lottery consisting of m independently drawn panels. Then, the selection probabilities $\hat{\pi}$ for the transparent lottery satisfy*

$$\|\hat{\pi} - \pi_\lambda\|_\infty \leq \sqrt{\frac{\ln(2n) + \ln(\delta^{-1})}{2m}}.$$

with probability at least $1 - \delta$, for any $0 < \delta < 1$.

This bound is substantially worse than those of Flanigan et al. [24], since it scales like $O(1/\sqrt{m})$ rather than in $O(1/m)$ in the lottery size m, and it only holds probabilistically. Nevertheless, a larger number of panels m can ensure that selection probabilities are faithfully represented with high probability; for example, in a pool of size 1,000, a transparent lottery of size 10,000 yields a maximum deviation of 2.5% with a failure probability of at most 1%.

Note that, since the entropy of any choice over a list of size m is at most $\log(m)$, there is little chance to preserve the high entropy (which requires large support) in a transparent lottery (which should draw from a manageable number of panels). One upside is that drawing m independent samples and choosing one uniformly among them is of course equivalent to drawing a single sample. As a result, creating a transparent lottery does not reduce the entropy (or any other properties) of the overall sortition.

6 Empirical Evaluation

Now we evaluate the empirical properties of our algorithms on 86 real-world sortition instances provided to us by several organizations, mainly the *Sortition Foundation* (United Kingdom), *MASS LBP* (Canada), and the *newDemocracy Foundation* (Australia). This dataset is several times larger than those used in previous papers on sortition algorithms [22, 28], which allows us to gain more robust insights into the breadth of problems encountered in practice. For a list of all instances

and their sizes, see Table 5 in Appendix C.1. Instance names (for example, sf_b_20) refer to the organization ("sf" for Sortition Foundation) and the panel size (20).

We benchmark our algorithms against three prior ones: LexiMin [22], Goldilocks [5], and Legacy[6] (a heuristic used by the Sortition Foundation before their adoption of column generation). For Goldilocks, we use a smooth version of the algorithm that replaces the ℓ_∞ norm [5] with the ℓ_{100} norm, which makes the optimal selection probabilities uniquely determined and is the version that has been used in practical assemblies. For both LexiMin and Goldilocks, we use column generation to obtain an explicit distribution, and then target the resulting selection probabilities with FairMaxEntropy. We refer to the respective runs of FairMaxEntropy as Maximum-Entropy Leximin and Maximum-Entropy Goldilocks.

Due to the large number of experiments, we impose timeouts for the runs. For column generation, the timeouts were 15 minutes, 20 minutes for MaxEntropy, and 30 minutes for FairMaxEntropy (see Appendix C.2 for more details).

Statistics. Since the column generation versions of LexiMin and Goldilocks yield explicit distributions, selection probabilities and other empirical quantities for them are exact rather than estimates. For the other algorithms, we estimate selection probabilities from the empirical frequencies of inclusion over a number of sampled panels (10^5 for MaxEntropy and Legacy and 10^4 for FairMaxEntropy). For each pool member, the number of panels in which they appear is distributed binomially with an unknown success probability. We compute a Jeffreys 95% confidence interval for each selection probability, which yields the error bars in the plots. We follow the same approach for estimating the probability that a random panel satisfies a feature's quotas.

6.1 Implementation and Running Times

Implementing our algorithms was a challenging engineering task. The number of states in the dynamic programming algorithm grows rapidly, which makes memory consumption a bottleneck. Our implementation involves specialized data structures for store the dynamic programming states and a careful memory layout to even make medium-sized instances viable.

The algorithm trades off memory and execution time by deciding which features to satisfy explicitly using dynamic programming and which to satisfy via rejection sampling, which we can do in several threads in parallel. After each feature is added, the algorithm samples 10^5 panels satisfying the current set of features to estimate the satisfaction probabilities of the unenforced features. Then, the algorithm either attempts to add the feature with the lowest satisfaction probability to the dynamic program or defers it to rejection sampling.

In Table 1, we give the running times of MaxEntropy on the set of instances studied by Flanigan et al. [22]. These running times were computed on a 2023 MacBook Pro M3 with 18 GB of memory, with 5 threads allocated for sampling. The exception is sf_e_110, for which we used a cloud machine with 64 GB of memory and 5 threads allocated for sampling; the peak memory usage was 50 GB. We were unable to successfully sample a panel from obf_30, even on the cloud machine.

On average, MaxEntropy was able to sample a panel from 78 out of 86 instances and sample more than 10^5 panels on 68 of them. The eight instances for which MaxEntropy failed to sample a panel within the 20 minute timeout all have at least 8 features, the largest number among the 10 sortition instances studied by Flanigan et al. [22], and most have a large panel size.

Extending MaxEntropy to FairMaxEntropy imposes additional challenges. At each iteration of the algorithm, it (1) computes the weighted counts with the dynamic program, and (2) samples enough panels to estimate selection probabilities. For (1), a simple but powerful observation is

[6] https://github.com/sortitionfoundation/stratification-app/

Table 1. Running times of MaxEntropy on the instances studied by Flanigan et al. [22]. For all instances except cca_75 and sf_e_110, the sampling times are averaged over 10^4 samples; for cca_75, over 100 samples; and for sf_e_110, a single sample.

Instance	Pool size	Features	Values	Counting (s)	Sampling (ms)
cca_75	825	4	66	226.5	136,398.3
hd_30	239	7	33	448.6	30.53
mass_a_24	70	5	11	1.8	0.0045
nexus_170	342	5	33	229.1	18.9
obf_30*	321	8	38	Timeout	Timeout
sf_a_35	312	6	22	69.9	13.8
sf_b_20	250	6	20	6.1	0.074
sf_c_44	161	7	20	478.4	8.9
sf_d_40	404	6	19	6.2	1.7
sf_e_110*	1727	7	31	623.2	369,374.2

Table 2. Running times of FairMaxEntropy on LexiMin selection probabilities, on instances studied by Flanigan et al. [22] without timeout.

Instance	Pool size	Features	Values	Time (s)
mass_a_24	70	5	11	18.8
sf_a_35	312	6	22	940.7
sf_b_20	250	6	20	115
sf_c_44	161	7	20	1416.6
sf_d_40	404	6	19	607.2

that the non-zero states of the dynamic programming are the same for every (positive) weight vector μ. Thus, steps past the first immediately have access to a perfectly pruned dynamic program, which speeds up the reweighting considerably. For (2), as the optimization routine progresses, weights yielded by the dual solution increase in magnitude, which increases the computational cost of sampling and would risk numeric instability with floating-point numbers. We solve this by rounding the weights to integers in the range $[1, 10^{12}]$ and performing the computations with exact integer precision.

Finally, we use ADAM [39] as the underlying solver for practical performance. In Table 2, we present selected running times for FairMaxEntropy on LexiMin selection probabilities for those instances from Table 1 for which it does not time out. In total, we were able to solve 48 instances with LexiMin selection probabilities and 43 with those of Goldilocks. Since Goldilocks is, in general, slower than LexiMin, the column generation timed out more often, which caused some distributions to be missing for FairMaxEntropy to optimize on. (See Table 5 for more information).

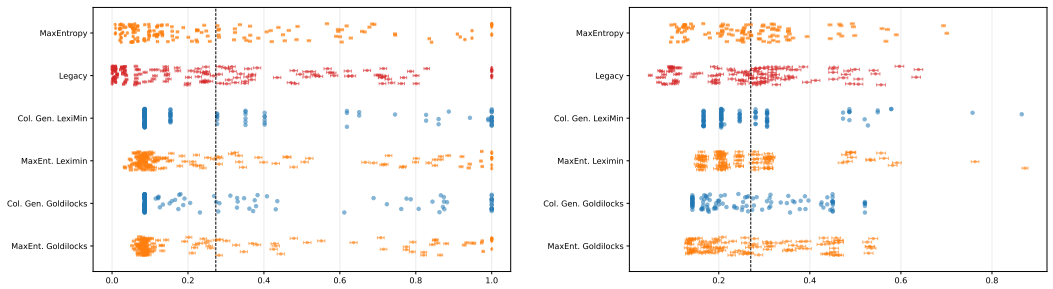
6.2 Fairness in Selection Probabilities

We start our discussion of empirical properties by comparing the fairness of selection probabilities across the selection algorithms. In general, there are no fairness guarantees for MaxEntropy, as the selection probabilities rely on the structure of the pool, and even if there is a distribution that equalizes all selection probabilities, MaxEntropy might not sample from it.

We give two representative examples in Figure 1. Regarding minimum selection probabilities, no algorithm can perform better than Leximin, since it optimizes that measure. Goldilocks is often quite close, but might loosen the lower probabilities in favor of reducing the upper probabilities.

Both MaxEntropy and Legacy tend to have some pool members with very low selection probabilities, with Legacy substantially closer to zero [22]. Regarding the maximum selection probability, MaxEntropy sits halfway between the best performer Goldilocks and the worst, Leximin, which has the tendency to assign very high selection probabilities to some pool members. As illustrated in the plot for sf_c_44, whenever there is a pool member that is contained in all panels, Goldilocks tends to assign high selection probabilities to multiple members, whereas MaxEntropy does not.

We use the *Gini inequality coefficient* and *geometric mean* as global measures of fairness that go beyond minimum and maximum selection probabilities, the results are contained in Table 3. On those metrics, MaxEntropy sits again between the unfairer Legacy and fairer column-generation optimizing a notion of fairness.



(a) Selection probabilities for sf_c_44.

(b) Selection probabilities for sf_k_30.

Fig. 1. Strip plots for the selection probabilities, each point represents a pool member and its position represents the selection probability. Vertical lines mark the equalized selection probabilities $(\frac{k}{n}, \ldots, \frac{k}{n})$.

Table 3. Aggregate fairness metrics (average and median) across all 47 common instances. Metrics are based on sample averages for algorithms other than column generation. "▲" means "higher is better".

	MaxEntropy		Legacy		Leximin		Goldilocks	
	Avg.	Med.	Avg.	Med.	Avg.	Med.	Avg.	Med.
Gini (▼)	0.478	0.482	0.505	0.512	0.441	0.448	0.443	0.448
Geometric mean (▲)	0.122	0.077	0.096	0.062	0.129	0.083	0.129	0.083
Minimum probability (▲)	0.034	0.008	0.021	0.001	0.080	0.052	0.077	0.050
Maximum probability (▼)	0.909	1.000	0.876	1.000	0.975	1.000	0.958	1.000

The FairMaxEntropy distributions in Figure 1a only completed two iterations of gradient descent in the allocated time. Even though the distributions have clearly not converged to their column-generation targets in this time, they have made significant progress in that direction from their starting point, i.e., the MaxEntropy distribution, in particular in terms of the minimum probability. This is encouraging, since it shows that even a few iterations of gradient descent may be worthwhile to avoid low selection probabilities.

6.3 Intersectional Diversity

One of the central ideals of sortition is *representativeness*: if the panel was a truly uniform sample from the population, the selected panel approximates the population composition across all dimensions of interest [11, 20]. The quotas imposed by practitioners enforce representation across the

quota-protected features, but they do not directly constrain the representation of *joint categories* such as *Education* × *Income* × *Region*. This limitation is a central concern in the literature on intersectionality, which emphasizes that single-axis categories can obscure systematic disadvantages and patterns of underrepresentation that arise at their intersections [29, 46, 56]. Imposing quotas on these intersectional groups is generally not possible, since there are so many of them, and since the population share of complex intersectional groups is often not known.

Due to these obstacles, we focus on intersectional diversity rather than representation [52], by which we mean that it is desirable to see a variety of distinct intersectional groups and a lack of unnecessary correlations between features within the same panel. We summarize intersectional outcomes using three metrics: two focusing on all features at once, and one focusing on pairs of features in isolation. We will call *vector count* the expected number of distinct feature-value vectors present in a sampled panel, and apply the information-theoretic measure of *total correlation* [55] to quantify how much a panel's feature-value vector distribution differs from a product distribution over all features. Since much of the focus is on intersections of two features, we measure, for each pair of features, the *normalized mutual information*. That is, knowing the panel and drawing a random person from it, how much does revealing this person's value for f_1 reduce the information needed to communicate their value for f_2? (Larger values mean more correlation.) The global metrics are contained in Table 4 and the plots for the pairwise mutual information comparisons are in Appendix C.3. All values are normalized to facilitate the comparison between algorithms. We restricted the analysis to the set of 39 instances where all selection algorithms where able to produce a distribution.

Table 4. Average of expected vector count, average of total correlation, and median normalized mutual information for each selection algorithm. "▲" means "higher is better".

			LexiMin fairness		Goldilocks fairness	
	MaxEntropy	Legacy	col. generation	max. entropy	col. generation	max. entropy
Vector Count (▲)	0.914	0.908	0.823	0.910 (10.4% ↑)	0.830	0.908 (9.1% ↑)
Total Corr. (▼)	0.599	0.602	0.655	0.601 (8.2% ↓)	0.651	0.602 (7.5% ↓)
Median NMI (▼)	0.143	0.146	0.168	0.145 (13.6% ↓)	0.166	0.144 (13.2% ↓)

Table 4 shows that, out of the box, maximum-entropy produces the most diverse panels according to all three measures, whereas column generation performs worst, and similarly bad for both fairness measures. Given that FairMaxEntropy achieves almost the same diversity as pure MaxEntropy, this is not a consequence of the fairness measure. We conjecture that the pricing problem, used by column generation to generate new panels, tend to have optimal solutions that have below-average diversity. MaxEntropy and Legacy show similar results across all metrics, which also supports the assumption that low diversity is an artifact of column generation. If one wants to generate panels with even higher vector count, this could be easily incorporated in the dynamic program to sample with maximum entropy from only maximal vector-count panels.

6.4 Generalization to Unprotected Features

An often-cited ideal [21, 54] is that the citizens' assembly should be "in miniature an exact portrait of the people at large" (in the words of John Adams). No matter which features assembly organizers choose to set quotas on, there will invariably be many more, relevant to the deliberation, not protected by quotas. In this section, we compare selection algorithms based on their ability to satisfy an unknown feature, that is, one for which there are no quotas; we denote this by the

generalization probability. Flanigan et al. [23] have argued that selection algorithms with fairer selection probabilities may tend be more representative in this situation.

To test this scenario with the available data, we leave out quotas. That is, for each instance and feature, we pretend that this feature were not part of the instance. Then, we run each selection algorithm and compute its probability of satisfying the held-out feature. Since the total number of instances in this experiment was 554, we imposed a stricter timeout for the selection algorithms (10 minutes for column-generation algorithms, 15 minutes for MaxEntropy and Legacy with 10^5 samples, and 25 minutes for FairMaxEntropy with 10^4 samples).

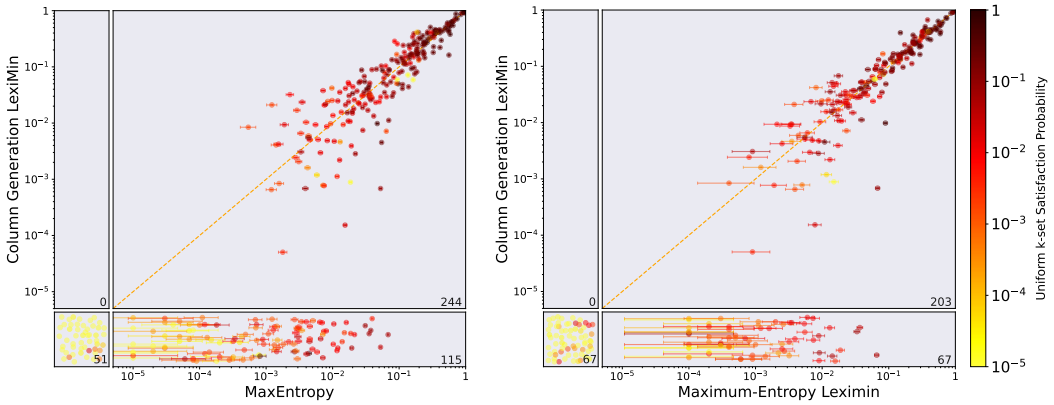


Fig. 2. Generalization probabilities. Points are colored according to the probability that a uniform set of k pool members satisfies the held-out feature. Points with probabilities/empirical means less than $5 \cdot 10^{-6}$ are moved into side boxes, see footnote.

We display two representative comparisons in Figure 2 and defer the others to Appendix C.4. Both cases illustrate the general finding that column generation has lower satisfaction probabilities for the held-out feature than other approaches. While many feature-instance pairs have similar satisfaction probabilities in, say, MaxEntropy and column generation LexiMin, there are many that are satisfied with zero probability by the column generation algorithm.[7]

This is not a simple artifact of the fairness measure, as maximum-entropy LexiMin obtains higher generalization probabilities. In both plots, when column-generation matches the maximum-entropy, it typically happens on the instance-feature combinations that are already easily satisfiable by sampling k pool members uniformly from the pool, which suggests that those quotas are easy to satisfy. The pattern of zero generalization probabilities for column generation is not limited to any specific organization's instances, as we show in Figure 9.

In our view, the better generalization probabilities of our algorithms validate Jaynes' [34] justification of the maximum-entropy distribution. By restricting their support to a small subset of panels, column generation algorithms introduce a "bias" that arbitrarily commits to never representing some features in their correct ranges.

[7]Since column generation produces an explicit representation, we verify that these probabilities are indeed equal to zero, not just smaller than the cutoff. For MaxEntropy, the probability of satisfying the held-out feature is computed as a sample mean. Since there are panels satisfying the left-out feature and our algorithms put positive mass on all panels, these probabilities can never be zero, and the probabilities in the lower left corner are simply too small to show up in our samples.

7 Practical Deployment

Since, as we have argued, maximum-entropy algorithms has a range of desirable properties, we are making it available to practitioners. We release our implementation as open source on Github[8]. We believe that self-hosting our algorithm will be attractive to privacy-concerned organizations, which will be easier since our algorithm does not rely on proprietary optimization libraries.

To facilitate the algorithm's uptake by practitioners, we will also make it available as one of the algorithmic choices on *panelot.org*. The left panel Figure 3 shows the planned screen in which practitioners can choose between algorithms and learn about their distinct properties. The figure's right panel shows the execution page, which will update statistics from the run and pool in real time, providing a progress estimate and an illustration of the current state of the dynamic program. During execution, the interface presents how each successive feature shrinks the count of panels, and which remaining features are hard to satisfy. This may help practitioners become familiar with the algorithm and help them adapt their quotas if they cannot be solved by MaxEntropy. To complete, the algorithm will sample 10^4 panels to plot the selection probabilities and provide a physical lottery. Based on practitioner interests and improvements in algorithm performance, we might also make versions of FairMaxEntropy available in the future.

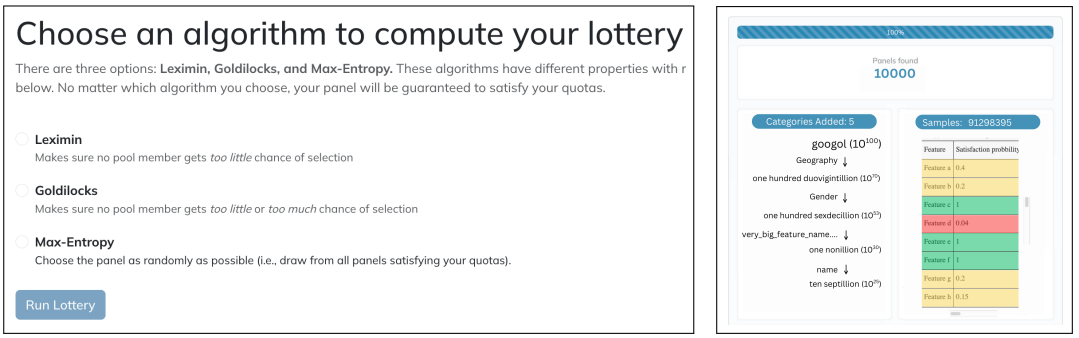


Fig. 3. Planned interface of Panelot for choosing the selection algorithm (left) and visualization of the state of MaxEntropy (right).

8 Conclusion

By providing a new selection algorithm with distinct advantages, we hope to contribute to practitioners' efforts to increase public trust in citizens' assemblies. We believe that this is timely, since higher-stakes uses for assemblies will likely lead to higher scrutiny as well.

Our empirical evaluation points to drawbacks of column generation algorithms, in terms of the diversity of their panels and their generalization to unknown features. At the same time, it is very encouraging to see that the FairMaxEntropy seemed not subject to these limitations despite converging to the same, fairness-optimal selection probabilities. These findings suggest that, when possible to compute, maximizing entropy is a worthwhile complementary objective even when the primary objective is fairness.

Acknowledgments

We thank Soroosh Shafiee and Noah Stephens-Davidowitz for helpful technical conversations. We thank the Center for Climate Assemblies, Healthy Democracy, MASS LBP, New Democracy,

[8]https://github.com/gabrielmorete/maximally-random-sortition

Nexus Institut, Of by For, and the Sortition Foundation for providing the sortition instances for our empirical analysis, and Carmel Baharav and Bailey Flanigan for sharing cleaned versions of the newDemocracy data. We thank Carmel Baharav, Anthony Bucci, and Bailey Flanigan for their contributions in implementing MaxEntropy in the upcoming release of Panelot.

References

[1] Alekh Agarwal, Martin J Wainwright, Peter Bartlett, and Pradeep Ravikumar. 2009. Information-theoretic lower bounds on the oracle complexity of convex optimization. In *Advances in Neural Information Processing Systems*, Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta (Eds.), Vol. 22. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2009/file/2387337ba1e0b0249ba90f55b2ba2521-Paper.pdf

[2] Arash Asadpour, Michel X. Goemans, Aleksander Mądry, Shayan Oveis Gharan, and Amin Saberi. 2017. An O (Log n/Log Log n)-Approximation Algorithm for the Asymmetric Traveling Salesman Problem. *Operations Research* 65, 4 (2017), 1043–1061. https://doi.org/10/gbq7fd

[3] Arash Asadpour and Amin Saberi. 2010. An Approximation Algorithm for Max-Min Fair Allocation of Indivisible Goods. *SIAM J. Comput.* 39, 7 (2010), 2970–2989. https://doi.org/10/d8p7wr

[4] Angelos Assos, Carmel Baharav, Bailey Flanigan, and Ariel Procaccia. 2025. Alternates, Assemble! Selecting Optimal Alternates for Citizens' Assemblies. In *Proceedings of the ACM Conference on Economics and Computation (EC)*. 719–738. https://doi.org/10.1145/3736252.3742614

[5] Carmel Baharav and Bailey Flanigan. 2024. Fair, Manipulation-Robust, and Transparent Sortition. In *Proceedings of the ACM Conference on Economics and Computation (EC)*. 756–775. https://doi.org/10.1145/3670865.3673606

[6] Alexander I. Barvinok. 1994. A Polynomial Time Algorithm for Counting Integral Points in Polyhedra When the Dimension Is Fixed. *Mathematics of Operations Research* 19, 4 (Nov. 1994), 769–779. https://doi.org/10.1287/moor.19.4.769

[7] Gerdus Benadè, Paul Gölz, and Ariel D. Procaccia. 2019. No Stratification Without Representation. In *Proceedings of the ACM Conference on Economics and Computation (EC)*. 281–314. https://doi.org/10/gqcq7x

[8] Stephen P. Boyd and Lieven Vandenberghe. 2004. *Convex Optimization*. Cambridge University Press, Cambridge, UK; New York.

[9] Peter Bürgisser, Yinan Li, Harold Nieuwboer, and Michael Walter. 2020. Interior-point methods for unconstrained geometric programming and scaling problems. arXiv:2008.12110 [math.OC] https://arxiv.org/abs/2008.12110

[10] Ioannis Caragiannis, Evi Micha, and Jannik Peters. 2024. Can a Few Decide for Many? The Metric Distortion of Sortition. https://doi.org/10.48550/ARXIV.2406.02400

[11] Lyn Carson and Brian Martin. 1999. *Random Selection in Politics*. Praeger.

[12] Committee of Ministers of the Council of Europe. 2023. *Recommendation CM/Rec(2023)6 of the Committee of Ministers to member States on deliberative democracy*. Recommendation CM/Rec(2023)6. Council of Europe. https://search.coe.int/cm?i=0900001680ac627a Adopted by the Committee of Ministers on 6 September 2023 at the 1473rd meeting of the Ministers' Deputies.

[13] Dimitri Courant. 2021. Citizens' Assemblies for Referendums and Constitutional Reforms: Is There an "Irish Model" for Deliberative Democracy? *Frontiers in Political Science* 2 (Jan. 2021). https://doi.org/10.3389/fpos.2020.591983

[14] Thomas M. Cover and Joy A. Thomas. 2006. *Elements of Information Theory* (second edition ed.). Wiley-Interscience, Hoboken, N.J. https://doi.org/10.1002/047174882X

[15] Jesús A. De Loera, Raymond Hemmecke, Jeremiah Tauzer, and Ruriko Yoshida. 2004. Effective lattice point counting in rational convex polytopes. *Journal of Symbolic Computation* 38, 4 (2004), 1273–1302. https://doi.org/10.1016/j.jsc.2003.04.003 Symbolic Computation in Algebra and Geometry.

[16] Virginie Do, Jamal Atif, Jérôme Lang, and Nicolas Usunier. 2021. Online Selection of Diverse Committees. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 154–160. https://doi.org/10/gn78qg

[17] Oliver Dowlen. 2008. *The Political Potential of Sortition: A Study of the Random Selection of Citizens for Public Office*. Imprint Academic.

[18] Soroush Ebadian, Gregory Kehne, Evi Micha, Ariel D. Procaccia, and Nisarg Shah. 2022. Is Sortition Both Representative and Fair? *Advances in Neural Information Processing Systems (NeurIPS)* 35 (2022), 3431–3443.

[19] Soroush Ebadian and Evi Micha. 2025. Boosting Sortition via Proportional Representation. In *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*. 667–675.

[20] Fredrik Engelstad. 1989. The Assignment of Political Office by Lot. *Social Science Information* 28, 1 (March 1989), 23–50. https://doi.org/10/cggrj2

[21] James S. Fishkin. 2009. *When the People Speak: Deliberative Democracy and Public Consultation*. Oxford University Press.

[22] Bailey Flanigan, Paul Gölz, Anupam Gupta, Brett Hennig, and Ariel D. Procaccia. 2021. Fair Algorithms for Selecting Citizens' Assemblies. *Nature* 596, 7873 (2021), 548–552. https://doi.org/10/gmz56v

[23] Bailey Flanigan, Paul Gölz, and Ariel Procaccia. 2023. *Mini-Public Selection: Ask What Randomness Can Do for You.* Policy Brief. Ash Center for Democratic Governance and Innovation, Harvard Kennedy School, Cambridge, MA. https://ash.harvard.edu/wp-content/uploads/2024/02/mini-public_selection_final_report.pdf

[24] Bailey Flanigan, Gregory Kehne, and Ariel D. Procaccia. 2021. Fair Sortition Made Transparent. In *Advances in Neural Information Processing Systems*, Vol. 34. 25720–25731. https://proceedings.neurips.cc/paper_files/paper/2021/file/d7b431b1a0cc5f032399870ff4710743-Paper.pdf

[25] Bailey Flanigan, Jennifer Liang, Ariel D. Procaccia, and Sven Wang. 2024. Manipulation-Robust Selection of Citizens' Assemblies. *Proceedings of the AAAI Conference on Artificial Intelligence* 38, 9 (mar 2024), 9696–9703. https://doi.org/10.1609/aaai.v38i9.28827

[26] Bailey G. Flanigan, Paul Gölz, Anupam Gupta, and Ariel D. Procaccia. 2020. Neutralizing Self-Selection Bias in Sampling for Sortition. (2020).

[27] Adela Gąsiorowska. 2023. Sortition and Its Principles: Evaluation of the Selection Processes of Citizens' Assemblies. *Journal of Deliberative Democracy* 19, 1 (Jan. 2023). https://doi.org/10.16997/jdd.1310

[28] Marcin Gerwin, Nikodem Mroek, Grzegorz Kuriata, Paulina Pospieszna, and Andalexander M. Geisler. 2025. Designing the Process of Random Selection of Citizens' Assemblies. *Journal of Sortition* 1, 1 (2025), 48–59.

[29] Nick Gill and Tom Lord. 2022. *Diversity, inclusion and intersectionality: analysing education attainment level and ethnicity in responses to citizens' assembly recruitment processes.* Technical Report. Sortition Foundation. https://assets.nationbuilder.com/sortitionfoundation/pages/906/attachments/original/1663663146/dii_study.pdf?1663663146=

[30] Louis-Gaëtan Giraudet, Bénédicte Apouey, Hazem Arab, Simon Baeckelandt, Philippe Bégout, Nicolas Berghmans, Nathalie Blanc, Jean-Yves Boulin, Eric Buge, and Dimitri Courant. 2022. "Co-construction" in Deliberative Democracy: Lessons from the French Citizens' Convention for Climate. *Humanities and Social Sciences Communications* 9, 1 (2022), 1–16.

[31] Paul Gölz, Jan Maly, Ulrike Schmidt-Kraepelin, Markus Utke, and Philipp C. Verpoort. 2025. City Sampling for Citizens' Assemblies. https://doi.org/10.48550/arXiv.2509.07557 arXiv:2509.07557 [cs]

[32] Hiroshi Hirai and Keiya Sakabe. 2025. Gradient descent for unbounded convex functions on Hadamard manifolds and its applications to scaling problems. arXiv:2404.09746 [math.OC] https://arxiv.org/abs/2404.09746

[33] Prateek Jain, Dheeraj M. Nagaraj, and Praneeth Netrapalli. 2021. Making the Last Iterate of SGD Information Theoretically Optimal. *SIAM Journal on Optimization* 31, 2 (Jan. 2021), 1108–1130. https://doi.org/10.1137/19M128908X

[34] E. T. Jaynes. 1957. Information Theory and Statistical Mechanics. *Physical Review* 106, 4 (May 1957), 620–630. https://doi.org/10.1103/PhysRev.106.620

[35] E. T. Jaynes. 1957. Information Theory and Statistical Mechanics. II. *Physical Review* 108, 2 (Oct. 1957), 171–190. https://doi.org/10.1103/PhysRev.108.171

[36] Mark Jerrum, Alistair Sinclair, and Eric Vigoda. 2004. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *J. ACM* 51, 4 (jul 2004), 671–697. https://doi.org/10.1145/1008731.1008738

[37] Mark R. Jerrum, Leslie G. Valiant, and Vijay V. Vazirani. 1986. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science* 43 (1986), 169–188. https://doi.org/10.1016/0304-3975(86)90174-X

[38] Ravi Kannan and Santosh Vempala. 1997. Sampling Lattice Points. In *Proceedings of the Annual ACM Symposium on Theory of Computing (STOC)*. 696–700. https://doi.org/10.1145/258533.258665

[39] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*. https://arxiv.org/abs/1412.6980

[40] MASS LBP. 2017. How to Run a Civic Lottery: Designing Fair Selection Mechanisms for Deliberative Public Processes.

[41] Oren Matar. 2025. Diversimax: Maximizing Intersectional Diversity in Sortition. https://orenmatar.github.io/diversimax_algorithm/ Accessed: 2026-02-09.

[42] Reshef Meir, Fedor Sandomirskiy, and Moshe Tennenholtz. 2021. Representative Committees of Peers. *Journal of Artificial Intelligence Research* 71 (July 2021), 401–429. https://doi.org/10/gqcq8m

[43] Ben Morris and Alistair Sinclair. 2004. Random Walks on Truncated Cubes and Sampling 0-1 Knapsack Solutions. *SIAM J. Comput.* 34, 1 (Jan. 2004), 195–226. https://doi.org/10.1137/S0097539702411915

[44] Albert Nijenhuis and Herbert S. Wilf. 1978. *Combinatorial Algorithms for Computers and Calculators* (2nd ed ed.). Academic press, Orlando San Diego San Francisco [etc.].

[45] Marina Nord, David Altman, Fabio Angiolillo, Tiago Fernandes, Ana Good God, and Staffan I. Lindberg. 2025. *Democracy Report 2025: 25 Years of Autocratization – Democracy Trumped?* Technical Report. V-Dem Institute, University of Gothenburg, Gothenburg, Sweden.

[46] Organisation for Economic Co-operation and Development. 2020. *Innovative Citizen Participation and New Democratic Institutions: Catching the Deliberative Wave.* OECD. https://doi.org/10.1787/339306da-en

[47] Organisation for Economic Co-operation and Development. 2021. *Eight Ways to Institutionalise Deliberative Democracy.* OECD Public Governance Policy Papers 12. https://doi.org/10.1787/4fcf1da5-en

[48] Christos H. Papadimitriou. 1981. On the Complexity of Integer Programming. *Journal of the ACM (JACM)* 28, 4 (1981), 765–768. https://doi.org/10/dmgs9t

[49] R. Tyrrell Rockafellar. 1970. *Convex Analysis.* Princeton Mathematical Series, Vol. 28. Princeton University Press, Princeton, NJ. xviii+451 pages.

[50] C. E. Shannon. 1948. A Mathematical Theory of Communication. *Bell System Technical Journal* 27, 3 (July 1948), 379–423. https://doi.org/10.1002/j.1538-7305.1948.tb01338.x

[51] Mohit Singh and Nisheeth K. Vishnoi. 2014. Entropy, Optimization and Counting. In *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing* (New York, New York) *(STOC '14).* Association for Computing Machinery, New York, NY, USA, 50–59. https://doi.org/10.1145/2591796.2591803

[52] Daniel Steel, Naseeb Bolduc, Kristina Jenei, and Michael Burgess. 2020. Rethinking Representation and Diversity in Deliberative Minipublics. *Journal of Deliberative Democracy* 16, 1 (Aug. 2020), 46–57. https://doi.org/10/gm7wnc

[53] Peter Stone. 2011. *The Luck of the Draw: The Role of Lotteries in Decision Making.* Oxford University Press.

[54] David Van Reybrouck. 2016. *Against Elections: The Case for Democracy* (first us edition ed.). Seven Stories Press, New York.

[55] Satosi Watanabe. 1960. Information Theoretical Analysis of Multivariate Correlation. *IBM Journal of research and development* 4, 1 (1960), 66–82.

[56] Marta Wojciechowska. 2019. Towards Intersectional Democratic Innovations. *Political Studies* 67, 4 (2019), 895–911. https://doi.org/10.1177/0032321718814165

APPENDIX

A Deferred Details for MaxEntropy: Sampling Uniformly among Possible Panels

A.1 The Generalized Counting Problem

In this appendix, we provide proofs and generalize the dynamic programming approach from Section 3. Throughout this section, fix a sortition instance $\mathcal{I}$ and weights $\mu \in \mathbb{R}^N_{>0}$. For $\mathcal{U} \subseteq \mathcal{P}$, define the *weighted count* of panels as

$$\text{count}_\mu(\mathcal{U}) \coloneqq \sum_{P \in \mathcal{U}} \prod_{i \in P} \mu_i.$$

In particular, $|\mathcal{P}| = \text{count}_{\mathbb{1}}(\mathcal{P})$. The *generalized counting problem* consists of computing $\text{count}_\mu(\mathcal{P})$.

Fix an ordering $1, \dots, n$ of pool members. We define the *weighted counting function* as follows:

$$\varphi_\mu(i, z) \coloneqq \sum_{\substack{U \subseteq \{i,\dots,n\}: \\ z + w(U) \in \overline{\mathcal{Z}}}} \prod_{t \in U} \mu_t.$$

Note that

$$\varphi_\mu(1, 0) = \sum_{\substack{U \subseteq N: \\ w(U) \in \overline{\mathcal{Z}}}} \prod_{t \in U} \mu_t = \sum_{P \in \mathcal{P}} \prod_{t \in P} \mu_t = \text{count}_\mu(\mathcal{P}), \tag{2}$$

where the second equality uses that $U \subseteq N$ is a panel if and only if $w(U) \in \overline{\mathcal{Z}}$.

Proof of Proposition 3.2. We restate Proposition 3.2 in weighted form.

Proposition A.1. *For all $i \in N$, $z \in \mathbb{N}^{FV}$, and $F' \subseteq F$, we have that $\varphi_\mu(i, z) \leq \varphi'_\mu(i, z|_{F'})$.*

Proof. Let $z' \coloneqq z|_{F'}$. Define the *set of valid extensions* as follows

$$\mathcal{U}(i, z) \coloneqq \{ U \subseteq \{i, \dots, n\} : z + w(U) \in \overline{\mathcal{Z}} \} \quad \text{and} \quad \mathcal{U}'(i, z') \coloneqq \{ U \subseteq \{i, \dots, n\} : z' + w'(U) \in \overline{\mathcal{Z}}' \}.$$

If $U \in \mathcal{U}(i, z)$, then $z + w(U) \in \overline{\mathcal{Z}}$. Thus, $(z + w(U))|_{F'} = z' + w'(U) \in \overline{\mathcal{Z}}'$. Since $\overline{\mathcal{Z}}'$ is obtained by restricting the constraints of $\overline{\mathcal{Z}}$. Hence, $\mathcal{U}(i, z) \subseteq \mathcal{U}'(i, z')$. Finally, since $\mu > 0$, we have that

$$\varphi_\mu(i, z) = \sum_{U \in \mathcal{U}(i,z)} \prod_{t \in U} \mu_t \leq \sum_{U \in \mathcal{U}'(i,z')} \prod_{t \in U} \mu_t = \varphi'_\mu(i, z'). \qquad \square$$

Proof of Proposition 3.1. Here, we consider more natural to state the DP in terms of the feature-vector vectors instead of pool members. Fix an ordering $\omega_1, \dots, \omega_{n_\mathcal{W}}$ of the distinct feature-value vectors in the pool. For each $j \in [n_\mathcal{W}]$ and $d = 0, \dots, n_{\omega_j}$, denote the *weighted combinations* of d members with feature-value vector ω_j by

$$C_\mu(j, d) \coloneqq \sum_{\substack{S \subseteq N|_{\omega_j}: \\ |S| = d}} \prod_{i \in S} \mu_i. \tag{3}$$

Equivalently, $C_\mu(j, d)$ is the coefficient of x^d in the polynomial $\prod_{i \in N|_{\omega_j}} (1 + \mu_i x)$. We restate Proposition 3.1 in weighted form.

Theorem A.2. *For every $j \in [n_\mathcal{W}]$ and $z \in \mathbb{N}^{FV}$, we have that*

$$\varphi_\mu(j, z) = \sum_{d=0}^{n_{\omega_j}} C_\mu(j, d)\ \varphi_\mu(j + 1, z + d\ \omega_j),$$

with base case $\varphi_\mu(n_\mathcal{W} + 1, z) = \mathbb{1}[z \in \overline{\mathcal{Z}}]$.

Proof. We prove the statement by induction. The base case follows from the definition of panel.

For each $j = 1, \dots, n_{\mathcal{W}}$, let N_j be the set of pool members with feature-value vectors $\omega_j, \dots, \omega_{n_{\mathcal{W}}}$. Fix $j \in [n_{\mathcal{W}}]$ and $z \in \mathbb{N}^{FV}$. Every $U \subseteq N_j$ decomposes uniquely as a disjoint union $U = S \dot\cup T$, where $S \subseteq N|_{\omega_j}$ and $T \subseteq N_{j+1}$. For all $i \in S$, since $w(i) = \omega_j$, we have $w(S) = |S|\omega_j$. Thus,

$$\begin{aligned}
\varphi_\mu(j, z) &= \sum_{S \subseteq N|_{\omega_j}} \prod_{i \in S} \mu_i \sum_{\substack{T \subseteq N_{j+1}: \\ z + w(S) + w(T) \in \overline{\mathcal{Z}}}} \prod_{i \in T} \mu_i \\
&= \sum_{d=0}^{n_{\omega_j}} \sum_{\substack{S \subseteq N|_{\omega_j}: \\ |S| = d}} \prod_{i \in S} \mu_i \varphi_\mu(j+1, z + d\omega_j) \\
&= \sum_{d=0}^{n_{\omega_j}} C_\mu(j, d) \varphi_\mu(j+1, z + d\omega_j). \qquad \square
\end{aligned}$$

As in the uniform case, the coefficients $C_\mu(j, d)$ can be precomputed for all j and all $d \le k$ in total time $O(nk)$, yielding the same complexity.

A.2 Sampling from Product Distributions

In this section, we give a formal description of the generalized sampling algorithm.

Throughout this section, fix a sortition instance $\mathcal{I}$ and weights $\mu \in \mathbb{R}^N_{>0}$. We prove our statements for the profile aggregated version of the counting function φ_μ. For each $j \in [n_{\mathcal{W}}]$, $z \in \mathbb{Z}^{FV}_{\ge 0}$, and $d \in \{0, \dots, n_{\omega_j}\}$ let $C_\mu(j, d)$ be as defined in (3). Define the following probability distributions:

$$p_{j,z}(d) \coloneqq \frac{C_\mu(j, d)\varphi_\mu(j+1, z + d\omega_j)}{\varphi_\mu(j, z)}, \quad \text{and} \quad q_{j,d}(S) \coloneqq \frac{\prod_{i \in S} \mu_i}{C_\mu(j, d)}, \quad S \subseteq N|_{\omega_j}, |S| = d.$$

The distribution $p_{j,z}$ describes how to move to the next state and $q_{j,d}$ defines a distribution of subsets of $N|_{\omega_j}$ of size d.

Algorithm 1: Exact Sampling

Input: A sortition instance $\mathcal{I}'$ and weights $\mu \in \mathbb{R}^N_{>0}$.
Output: A panel $P \sim \lambda'_\mu$.

Compute φ'_μ;
$(j, z) \leftarrow (1, 0)$;
$P \leftarrow \emptyset$;
while $j \le n_{\mathcal{W}}$ **do**
  Sample d according to $p_{j,z}$;
  Sample $S \subseteq N|_{\omega_j}$ according to $q_{j,d}$;
  $P \leftarrow P \cup S$;
  $(j, z) \leftarrow (j + 1, z + d\omega_j)$;
end
return P;

Algorithm 2: MaxEntropy

Input: A sortition instance $\mathcal{I}$ and weights $\mu \in \mathbb{R}^N_{>0}$.
Output: A panel $P \sim \lambda_\mu$.

Let $F' \subseteq F$ be a set of features;
repeat
  Sample $P \sim \lambda'_\mu$ using Algorithm 1;
until $P \in \mathcal{P}$;
return P;

Fix $F' \subseteq F$ and let $w' \coloneqq w|_{F'}$, and $\overline{\mathcal{Z}}'$ be the quota-compliant profiles restricted to F'. Let

$$\mathcal{P}' \coloneqq \left\{ P \subseteq N : w'(P) \in \overline{\mathcal{Z}}' \right\}$$

be the set of panels feasible with respect to F'. Define the distribution $\lambda'_\mu \in \Delta(\mathcal{P}')$ by

$$\lambda'_\mu(P) \coloneqq \frac{\prod_{i\in P'} \mu_i}{\text{count}_\mu(\mathcal{P}')}, \quad \text{for all } P' \in \mathcal{P}'.$$

Proposition A.3. *For every panel $P' \in \mathcal{P}'$, it holds that*

$$\Pr[\textit{Algorithm 1 outputs } P'] = \frac{\prod_{i\in P'} \mu_i}{\textit{count}_\mu(\mathcal{P}')}.$$

Proof. Fix $P' \in \mathcal{P}'$. For each $j \in [n_\omega]$, let $S_j \coloneqq P' \cap N|_{\omega_j}$ and $d_j \coloneqq |S_j|$. Define $z_1 \coloneqq 0$ and $z_{j+1} \coloneqq z_j + d_j\omega_j$. Then $z_{n_W+1} = w(P') \in \overline{\mathcal{Z}'}$. The algorithm outputs P if and only if it selects d_j at state (j, z_j) and then selects S_j at layer j. Thus, we have the following

$$\begin{aligned}\Pr[\text{Alg. outputs } P'] &= \prod_{j=1}^{n_W} \Pr[d = d_j \mid (j, z_j)] \prod_{j=1}^{n_W} \Pr[S = S_j \mid d_j, (j, z_j)] \\ &= \prod_{j=1}^{n_W} p_{j,z_j}(d_j) \prod_{j=1}^{n_W} q_{j,d_j}(S_j).\end{aligned}$$

Substituting the definitions of $p_{j,z}(d)$ and $q_{j,d}(S)$ gives

$$\begin{aligned}\Pr[\text{Alg. outputs } P'] &= \prod_{j=1}^{n_W} \frac{C_\mu(j, d_j)\varphi_\mu(j+1, z_{j+1})}{\varphi_\mu(j, z_j)} \prod_{j=1}^{n_W} \frac{\prod_{i\in S_j} \mu_i}{C_\mu(j, d_j)} \\ &= \frac{\varphi_\mu(n_W + 1, z_{n_W+1})}{\varphi_\mu(1, 0)} \prod_{j=1}^{n_W} \prod_{i\in S_j} \mu_i. \\ &= \frac{\prod_{i\in P'} \mu_i}{\text{count}_\mu(\mathcal{P}')}\end{aligned}$$

Where the last equality follows from Equation (2). □

Theorem A.4. *Given an instance $\mathcal{I}$ of a sortition problem and weights $\mu_{>0} \in \mathbb{R}^N$. MaxEntropy samples a panel according to λ_μ, that is,*

$$\Pr[\textit{Algorithm 2 outputs } P] = \frac{\prod_{i\in P} \mu_i}{\textit{count}_\mu(\mathcal{P})}, \quad \textit{for all } P \in \mathcal{P}.$$

Proof. Let $P'_1, P'_2 \ldots$ be the sequence of i.i.d samples obtained by sampling using Algorithm 1 on the restricted feature set F'. We define the acceptance probability

$$\alpha \coloneqq \Pr_{P'\sim\lambda'_\mu}[P' \in \mathcal{P}] = \sum_{P\in\mathcal{P}} \lambda'_\mu(P) = \frac{\text{count}_\mu(\mathcal{P})}{\text{count}_\mu(\mathcal{P}')} \tag{4}$$

We assume that $\mathcal{P} \neq \emptyset$ so $\alpha > 0$ and the algorithm terminates with high probability. Let $T \coloneqq \min\{\, t \geq 1 : P'_t \in \mathcal{P} \,\}$ be the stopping time, then, Algorithm 2 outputs P'_T.

Fix $P \in \mathcal{P}$. Then, since the samples are i.i.d, we have that

$$\begin{aligned}\Pr[P'_T = P] &= \sum_{t\geq 1} \Pr[P'_T = P \wedge T = t] \\ &= \sum_{t\geq 1} \Pr[P'_t = P, P'_1 \notin \mathcal{P}, \ldots, P'_{t-1} \notin \mathcal{P}],\end{aligned}$$

since the samples are i.i.d, we have that

$$= \sum_{t \geq 1} \Pr[P'_t = P] \prod_{s=1}^{t-1} \Pr[P'_s \notin \mathcal{P}]$$
$$= \lambda'_\mu(P) \sum_{t \geq 1} (1-\alpha)^{t-1}.$$

By the convergence of a geometric series, $\sum_{t \geq 1}(1-\alpha)^{t-1} = 1/\alpha$. Thus, bu (4), we obtain that $\Pr[P'_T = P] = \lambda_\mu(P)$. □

B Deferred Details for Theoretical Properties

B.1 Resistance to Manipulation

On this section, we present the formal statements and deferred proofs from Section 5.2.

Fix a sortition instance $\mathcal{I}$. A *coalition* is a set $C \subseteq N$ of pool members who jointly and strategically misreport their feature-value vectors. At all times, we assume that the coalition has full knowledge of the selection algorithm and pool composition. We denote the *reported feature-value vectors* by $\tilde{w}$, where $\tilde{w}(i) = w(i)$ for all $i \in N \setminus C$ and $\tilde{w}(i) \in \{0,1\}^{FV}$ for all $i \in C$. The resulting *manipulated instance* $\tilde{\mathcal{I}}$ is characterized by $(\mathcal{I}, C, \tilde{w}|_C)$. Let $\overline{\mathcal{W}}_C \in (\{0,1\}^{FV})^C$ be the set of all possible joint misreports for the coalition C. Given the size of the coalition $c \in \mathbb{N}$ and a sortition algorithm $\mathcal{A}$, we formally define the 3 measures of manipulability:

$$\mathrm{Manip}_{\mathrm{int}}(\mathcal{I}, c,) \coloneqq \max_{\substack{C \subseteq N \\ |C|=c}} \max_{\tilde{w}|_C \in \overline{\mathcal{W}}_C} \max_{i \in C} \tilde{\pi}(i) - \pi(i),$$

$$\mathrm{Manip}_{\mathrm{ext}}(\mathcal{I}, c, \mathcal{A}) \coloneqq \max_{\substack{C \subseteq N \\ |C|=c}} \max_{\tilde{w}|_C \in \overline{\mathcal{W}}_C} \max_{i \notin C} \pi(i) - \tilde{\pi}(i), \text{ and}$$

$$\mathrm{Manip}_{\mathrm{comp}}(\mathcal{I}, c, \mathcal{A}) \coloneqq \max_{\substack{C \subseteq N \\ |C|=c}} \max_{\tilde{w}|_C \in \overline{\mathcal{W}}_C} \max_{f, v \in FV} \sum_{\substack{i \in N \\ f(i)=v}} \tilde{\pi}(i) - \pi(i),$$

where π denotes the selection probabilities for feature vectors w and $\tilde{\pi}$ those for $\tilde{w}$ under $\mathcal{A}$, in our case MaxEntropy.

Without additional assumptions, there is no meaningful bound on manipulation. Consider n pool members and a single feature *Region = {Urban, Rural}*, where the quotas require $k-1$ urban panel members and 1 rural one. If the pool contains k urban pool members (and $n-k$ rural ones), any algorithm that satisfies equal treatment of equals will assign selection probability $1 - 1/k$ to urban pool members and $1/(n-k)$ to rural ones. As n grows, the probability of a rural pool member being selected goes to zero. Thus, if any rural pool members misreport their value, they increase their selection probability by a constant arbitrarily close to 1.

To rule out such trivial manipulability, Flanigan et al. [25] say that an instance is κ-*rich* if for some constant $\kappa > 0$ if there exists a subset W^* of the feature-value vectors $\mathcal{W}$ present in the pool such that

i) $\left|N|_w\right| \geq \kappa n + k$ for all $w \in W^*$; and
ii) there exists a panel $P_0 \in \mathcal{P}|_{W^*}$,

where $\mathcal{P}|_{W^*}$ is the set of panels only containing members in $N|_{W^*}$. Under this assumption, we have the following bound on the selection probabilities

Lemma B.1. *If $\mathcal{I}$ is κ-rich, then for all $i \in N$, we have that $\pi_{\lambda_{\mathrm{unif}}}(i) \leq \frac{k}{\kappa^k n}$.*

Proof. We first lower bound the number of feasible panels exclusively containing members with feature-value vectors in W^*. Since $\mathcal{I}$ is κ-rich, there exists $P_0 \in \mathcal{P}|_{W^*}$. For each $w \in W^*$, let

$p_0(w)$ be the number of members of P_0 with feature-value vector w, so $\sum_{w\in W^*} p_0(w) = k$. Since $|N|_w| \geq \kappa n + k$, we obtain

$$|\mathcal{P}|_{W^*}| \geq \prod_{w\in W^*} \binom{\kappa n + k}{p_0(w)} = \prod_{w\in W^*} \frac{\prod_{i=0}^{p_0(w)-1}(\kappa n + k - i)}{p_0(w)!} \geq \frac{(\kappa n)^k}{\prod_{w\in W^*} p_0(w)!} \geq \frac{(\kappa n)^k}{k!}, \tag{5}$$

where the last inequality follows from $(a+b)! \geq a!b!$ for $a, b \in \mathbb{Z}_{\geq 0}$. For each $i \in N$, the number of panels containing i is at most $\binom{n-1}{k-1}$. By Equation (5), we have that

$$\pi_{\lambda_{\text{unif}}}(i) = \frac{|\mathcal{P}(i)|}{|\mathcal{P}|} \leq \frac{\binom{n-1}{k-1}}{|\mathcal{P}|_{W^*}|} \leq \frac{n^{k-1}}{(k-1)!}\frac{k!}{(\kappa n)^k} = \frac{k}{\kappa^k n}. \qquad \square$$

To bound $\text{Manip}_{\text{comp}}$, we use the following observation due to Baharav and Flanigan [5].

Lemma B.2. *Fix a coalition $C \subseteq N$ of size c. Then, for any $f, v \in FV$ and $\lambda \in \Delta(\mathcal{P})$, we have that*

$$\sum_{\substack{i\in N:\\ f(i)=v}} \tilde{\pi}_\lambda(i) - \pi_\lambda(i) \leq \left(u_{f,v} - \ell_{f,v}\right) + c\max(\tilde{\pi}_\lambda).$$

Proof. Fix a feature-value pair $f, v \in FV$. Define $N_{f,v} \coloneqq \{\, i \in N : f(i) = v\}$ and $D_{f,v} \coloneqq \{\, i \in N_{f,v} : \tilde{f}(i) \neq v\}$. By feasibility, for any panel $P \in \mathcal{P}$, we have that, $\ell_{f,v} \leq \left|N_{f,v} \cap P\right| \leq u_{f,v}$. Thus,

$$\ell_{f,v} \leq \sum_{i\in N_{f,v}} \pi(i) = \sum_{P\in\mathcal{P}} \lambda(P)|N_{f,v} \cap P| \leq u_{f,v}.$$

The analogous result holds for $\tilde{\pi}$ over the set of reported feature-value vectors $N_{f,v} \setminus D_{f,v}$, as the manipulated instance is defined over the same quotas. We split the total post manipulation expected difference in assigned seats to truthful members and misreported members. By the expression above, we have that

$$\begin{aligned}\sum_{i\in N_{f,v}} \tilde{\pi}(i) - \pi(i) &= \sum_{i\in N_{f,v}\setminus D_{f,v}} \tilde{\pi}(i) + \sum_{i\in D_{f,v}} \tilde{\pi}(i) - \sum_{i\in N_{f,v}} \pi(i) \\ &\leq u_{f,v} - \ell_{f,v} + \sum_{i\in D_{f,v}} \tilde{\pi}(i) \leq u_{f,v} - \ell_{f,v} + c\max\tilde{\pi}.\end{aligned}$$

$\square$

This leads to the following theorem

Theorem B.3 (Resistance to manipulation, formal statement). *Let $\mathcal{I}$ be a κ-rich instance and let $c \leq \kappa' n$ for some constant $\kappa' \in [0, \kappa)$. Then, MaxEntropy satisfies the following:*

$$\begin{aligned}\text{Manip}_{\text{int}}(\mathcal{I}, c) &= O(k/n), \\ \text{Manip}_{\text{ext}}(\mathcal{I}, c) &= O(k/n), \text{ and} \\ \text{Manip}_{\text{comp}}(\mathcal{I}, c) &= \max_{f,v\in FV}\left(u_{f,v} - \ell_{f,v}\right) + O(ck/n).\end{aligned}$$

Proof. Fix any coalition C with $|C| = c$ and reported feature-value vectors $\tilde{w}$. Let $\beta \coloneqq \kappa - \kappa'$. Then, for any $w \in W^*$, we have that, $|\tilde{N}\big|_w| \geq |N|_w| - c \geq \beta n + k$. Let P_0 be the panel from the κ-rich definition and $p_0(w)$ be the number of members of P_0 with feature-value vector w. Since $p_0(w) \leq k$ for all $w \in W^*$ and $|\tilde{N}\big|_w| \geq k$, we can construct a panel $\tilde{P}_0$ in the manipulated instance by selecting $p_0(w)$ distinct members from each $\tilde{N}(w)$. Thus, $\tilde{\mathcal{I}}$ is β-rich.

Now, by Lemma B.1, $\tilde{\pi}(i) \leq k/\beta^k n$ for all $i \in N$. Thus, for any $i \in C$, we have that $\tilde{\pi}(i) - \pi(i) \leq \tilde{\pi}(i) \leq k/\beta^k n$, which proves the stated bound for $\text{Manip}_{\text{int}}$. The bound for $\text{Manip}_{\text{ext}}$ follows analogously from Lemma B.1 by imposing that $\pi(i) \leq k/\kappa^k n$.

Finally, the bound for $\text{Manip}_{\text{comp}}$ follows from taking the maximum over all feature-value pairs $f, v \in FV$ and applying Lemmas B.1 and B.2. □

Theorem B.3 is asymptotically tight when taking $n \to \infty$, as it matches the lower bound obtained in Flanigan et al. [25, Thm. 4.3] for $\text{Manip}_{\text{int}}$ and $\text{Manip}_{\text{ext}}$. Their bound also implies the optimality of $\text{Manip}_{\text{comp}}$ whenever the quotas are tight.

C Deferred Details for Empirical Evaluation

C.1 Instances

Table 5. List of instances in our dataset. The target panel size is the last subscript in the instance name. The runs were done in parallel on a cloud machine with 64 GB of memory and 32 cores. Runs marked with an * had a 90-minute timeout. "NO" means that the experiment timed out or failed.

					MaxEntropy		LexiMin fairness		Goldilocks fairness	
Instance	pool	features	values	Legacy	single sample	distribution	col. generation	max. entropy	col. generation	max. entropy
cca_75	825	4	66	NO	YES	NO	YES	NO	NO	NO
hd_30	239	7	33	YES	YES	YES	YES	NO	YES	NO
mass_a_24	70	5	11	YES	YES	YES	YES	YES	YES	YES
mass_b_37	349	8	32	YES	YES	YES	YES	YES	YES	YES
mass_c_35	118	5	13	YES	YES	YES	YES	YES	YES	YES
mass_d_31	168	9	36	NO	YES	YES	YES	NO	YES	NO
mass_e_36	322	4	13	YES	YES	YES	YES	YES	YES	YES
mass_f_33	158	3	9	YES	YES	YES	YES	YES	YES	YES
mass_g_35	379	8	33	YES	YES	YES	YES	YES	YES	YES
mass_h_36	204	5	14	YES	YES	YES	YES	YES	YES	YES
mass_i_36	106	8	30	YES	YES	YES	YES	YES	YES	YES
mass_j_28	421	7	22	YES	YES	YES	YES	YES	YES	YES
mass_k_36	57	7	17	YES	YES	YES	YES	YES	YES	YES
mass_l_34	133	5	29	YES	YES	YES	YES	YES	YES	YES
mass_m_26	72	13	33	NO	YES	YES	YES	YES	YES	YES
mass_n_32	1098	8	23	YES	YES	YES	YES	YES	NO	NO
mass_o_36	186	6	27	YES	YES	YES	YES	YES	YES	YES
mass_p_29	108	7	39	YES	YES	YES	YES	YES	YES	YES
mass_q_37	543	10	37	YES	YES	NO	YES	NO	YES	NO
mass_r_34	117	10	22	YES	YES	YES	YES	YES	YES	YES
mass_s_29	56	10	50	YES	YES	YES	YES	YES	YES	YES
mass_t_31	121	6	19	YES	YES	YES	YES	YES	YES	YES
mass_u_31	131	6	23	YES	YES	YES	YES	YES	YES	YES
mass_v_49	369	7	28	YES	YES	YES	YES	YES	NO	NO
mass_w_34	148	6	35	YES	YES	YES	YES	YES	YES	YES
mass_x_27	233	6	21	YES	YES	YES	YES	YES	YES	YES
mass_y_28	161	7	28	YES	YES	YES	YES	YES	YES	YES
mass_z_48	498	9	21	YES	YES	YES	YES	YES	YES	YES
nd_a_49	248	4	17	NO	YES	YES	YES	YES	YES	YES
nd_b_52	125	3	12	YES	YES	YES	YES	YES	YES	YES
nd_c_40	133	5	22	YES	YES	YES	YES	YES	YES	YES
nd_d_50	290	3	12	YES	YES	YES	YES	YES	NO	NO
nd_e_50	417	5	28	YES	YES	YES	YES	NO	YES	NO
nexus_170	342	5	33	YES	YES	YES	YES	NO	YES	NO
obf_30	321	8	38	NO	NO	NO	YES	NO	NO	NO
sf_a_35	312	6	22	YES	YES	YES	YES	YES	YES	YES
sf_aa_13	65	6	30	YES	YES	YES	YES	YES	YES	YES
sf_ab_13	37	6	30	YES	YES	YES	YES	YES	YES	YES
sf_ac_28	268	6	22	YES	YES	YES	YES	YES	YES	YES
sf_ad_28	303	6	22	YES	YES	YES	YES	YES	NO	NO
sf_ae_32	163	8	40	YES	YES*	NO	YES	NO	YES	NO
sf_af_40	125	7	19	YES	YES	YES	YES	YES	YES	YES
sf_ag_23	122	7	27	YES	YES	YES	YES	YES	YES	YES
sf_ah_42	179	8	34	NO	YES	NO	YES	NO	YES	NO

					MaxEntropy		LexiMin fairness		Goldilocks fairness	
Instance	pool	features	values	Legacy	single sample	distribution	col. generation	max. entropy	col. generation	max. entropy
sf_ai_20	182	8	41	NO	YES	YES	YES	YES	YES	YES
sf_aj_25	205	7	33	YES	YES	YES	YES	NO	YES	NO
sf_ak_15	58	6	22	YES	YES	YES	YES	YES	YES	YES
sf_al_82	515	8	32	NO	NO	NO	YES	NO	NO	NO
sf_am_82	533	8	32	YES	NO	NO	YES	NO	YES	NO
sf_an_82	428	8	32	NO	NO	NO	YES	NO	YES	NO
sf_ao_82	616	8	32	YES	NO	NO	YES	NO	NO	NO
sf_ap_82	548	8	32	NO	NO	NO	YES	NO	NO	NO
sf_aq_82	642	8	32	NO	NO	NO	YES	NO	NO	NO
sf_ar_23	283	8	33	NO	YES	NO	YES	NO	NO	NO
sf_as_22	248	8	33	YES	YES	NO	YES	NO	YES	NO
sf_at_25	360	8	33	YES	YES	NO	YES	NO	NO	NO
sf_au_20	231	8	33	YES	YES	NO	YES	NO	YES	NO
sf_av_22	133	8	34	YES	YES	YES	YES	NO	YES	YES
sf_aw_20	91	8	33	YES	YES	YES	YES	NO	YES	NO
sf_ax_25	140	8	33	NO	YES	YES	YES	NO	YES	NO
sf_ay_23	153	8	34	YES	YES	NO	YES	NO	YES	NO
sf_b_20	250	6	20	YES	YES	YES	YES	YES	YES	YES
sf_c_44	161	7	20	YES	YES	YES	YES	YES	YES	YES*
sf_d_40	404	6	19	YES	YES	YES	YES	YES	YES	YES
sf_e_110	1727	7	31	YES	YES*	NO	NO	NO	NO	NO
sf_f_21	126	5	27	YES	YES	YES	YES	YES	YES	YES
sf_g_21	152	5	27	YES	YES	YES	YES	YES	YES	YES
sf_h_20	129	5	20	YES	YES	YES	YES	YES	YES	YES
sf_i_55	362	5	16	YES	YES	YES	YES	YES	NO	NO
sf_j_40	302	5	21	YES	YES	YES	YES	YES	YES	YES
sf_k_30	111	3	15	YES	YES	YES	YES	YES	YES	YES
sf_l_23	256	8	34	YES	YES	YES	YES	NO	NO	NO
sf_m_20	197	8	34	YES	YES	YES	YES	NO	YES	NO
sf_n_25	298	8	34	YES	YES	YES	YES	NO	NO	NO
sf_o_22	250	8	34	YES	YES	YES	YES	NO	YES	NO
sf_p_30	85	8	34	NO	YES	YES	YES	NO	YES	NO
sf_q_30	151	10	38	NO	YES	NO	YES	NO	YES	NO
sf_r_30	137	10	37	NO	YES	YES	YES	NO	YES	NO
sf_s_30	167	10	39	NO	YES	NO	YES	NO	YES	NO
sf_t_30	131	9	36	NO	YES	YES	NO	NO	YES	NO
sf_u_30	114	9	36	NO	YES	YES	YES	YES	YES	YES
sf_v_30	112	6	26	YES	YES	YES	YES	YES	YES	YES
sf_w_30	143	7	31	NO	YES	YES	YES	NO	YES	NO
sf_x_30	113	10	38	NO	YES	YES	YES	NO	YES	NO
sf_y_30	98	9	34	NO	YES	YES	YES	NO	YES	NO
sf_z_30	152	10	53	NO	NO	NO	YES	NO	YES	NO

C.2 Details on Timeouts

The 30 minutes for FairMaxEntropy exclude the generation of marginals using column generation. After this, timeouts are handled as follows: We first run the stage of FairMaxEntropy that coincides with MaxEntropy, i.e., building the dynamic program and drawing uniform samples. Then, we start the iterative optimization. If this optimization reaches an ℓ_2 gradient norm below 0.001, we treat the optimization as completed and move on to sampling. If this threshold is not reached after 10 minutes of the iterative optimization, we check if at least two iterations have completed and, if so, move on to sampling, since nontrivial progress towards fairness has been made. If two iterations have not completed after 10 minutes, or this entire process exceeds the threshold of 30 minutes, we treat the experiment as a timeout and remove the instance from our analysis.

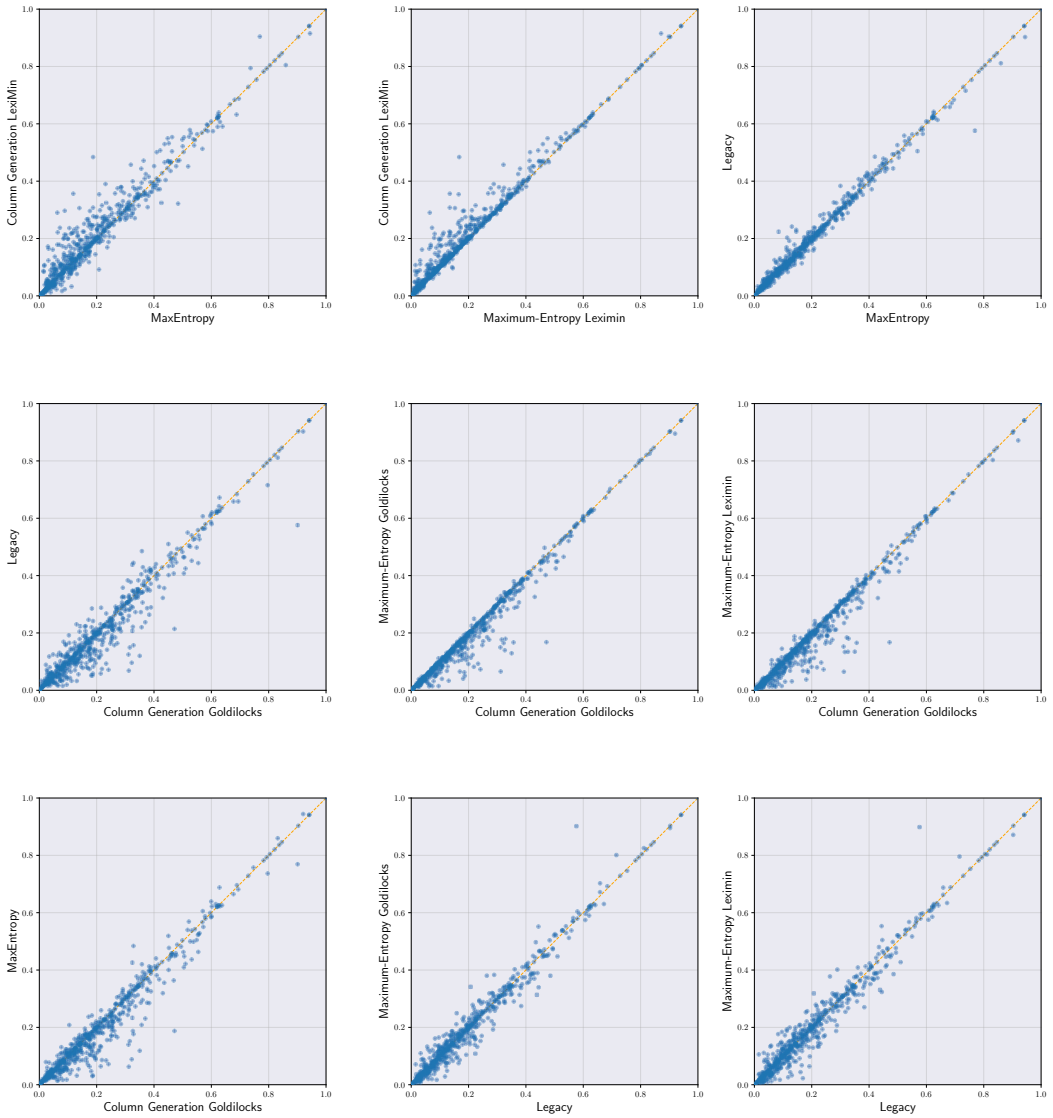
C.3 Intersectional Diversity

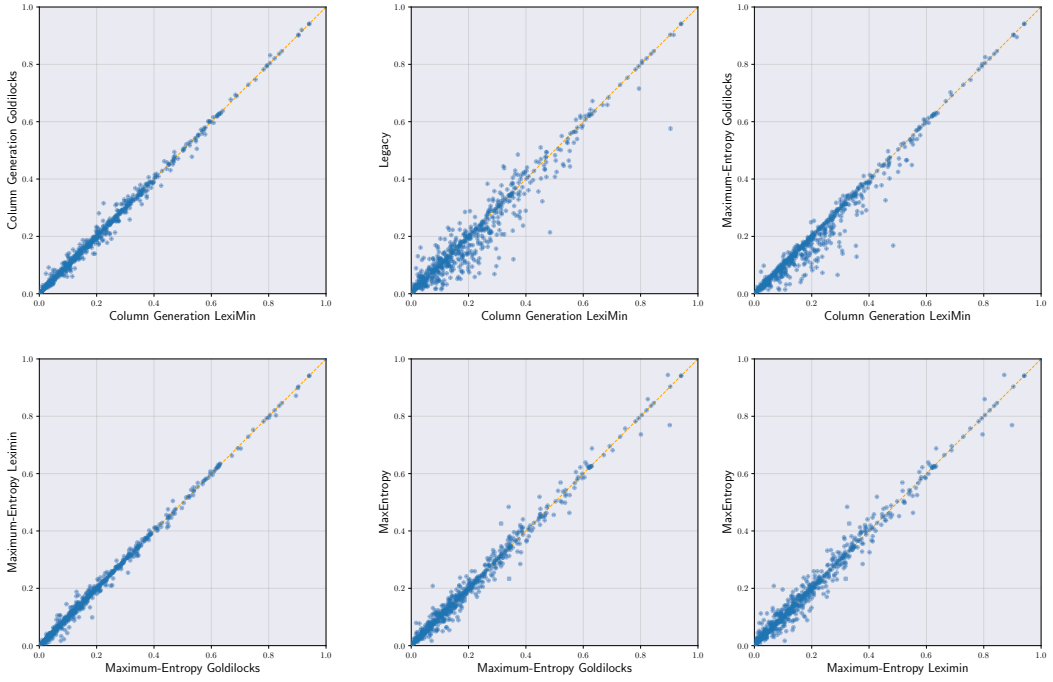
In this section, we compare the selection algorithms based on the pairwise mutual information between features. Formally, given a sortition instance $\mathcal{I}$ and a panel $P \in \mathcal{P}$, for each pair of features f, f', we define random variables X_f and $X_{f'}$ distributed according to the frequencies of each value

within the panel. Then, for each pair of features, we compute the normalized mutual information

$$\frac{I(X_f; X_{f'})}{H(X_f) + H(X_{f'})}.$$

To estimate measuring errors for the sampled selection algorithms, we compute a 95% confidence interval using the t-distribution. The metrics were computed across the 42 instances that were solved within the timeout for all selection algorithms (see Table 5). Across the board, the plots reinforce the conclusions from Section 6.3, namely that column-generation-based selection algorithms increase the correlation between features.




Column Generation Goldilocks
Column Generation LexiMin
Legacy
Column Generation LexiMin
Maximum-Entropy Goldilocks
Column Generation LexiMin
Maximum-Entropy Leximin
Maximum-Entropy Goldilocks
MaxEntropy
Maximum-Entropy Goldilocks
MaxEntropy
Maximum-Entropy Leximin

C.4 Generalization to Unprotected Features

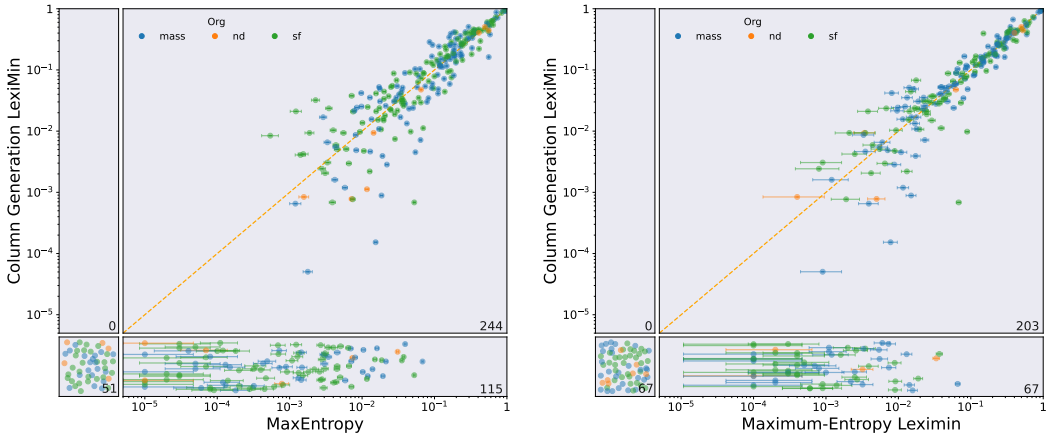
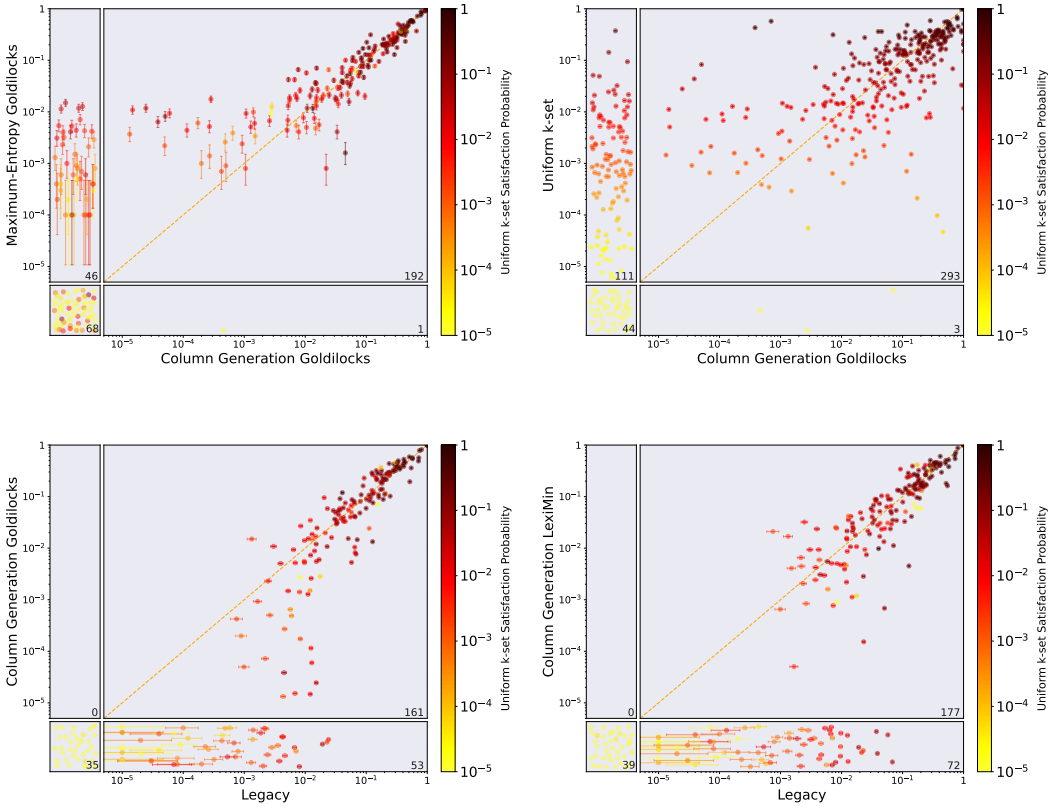
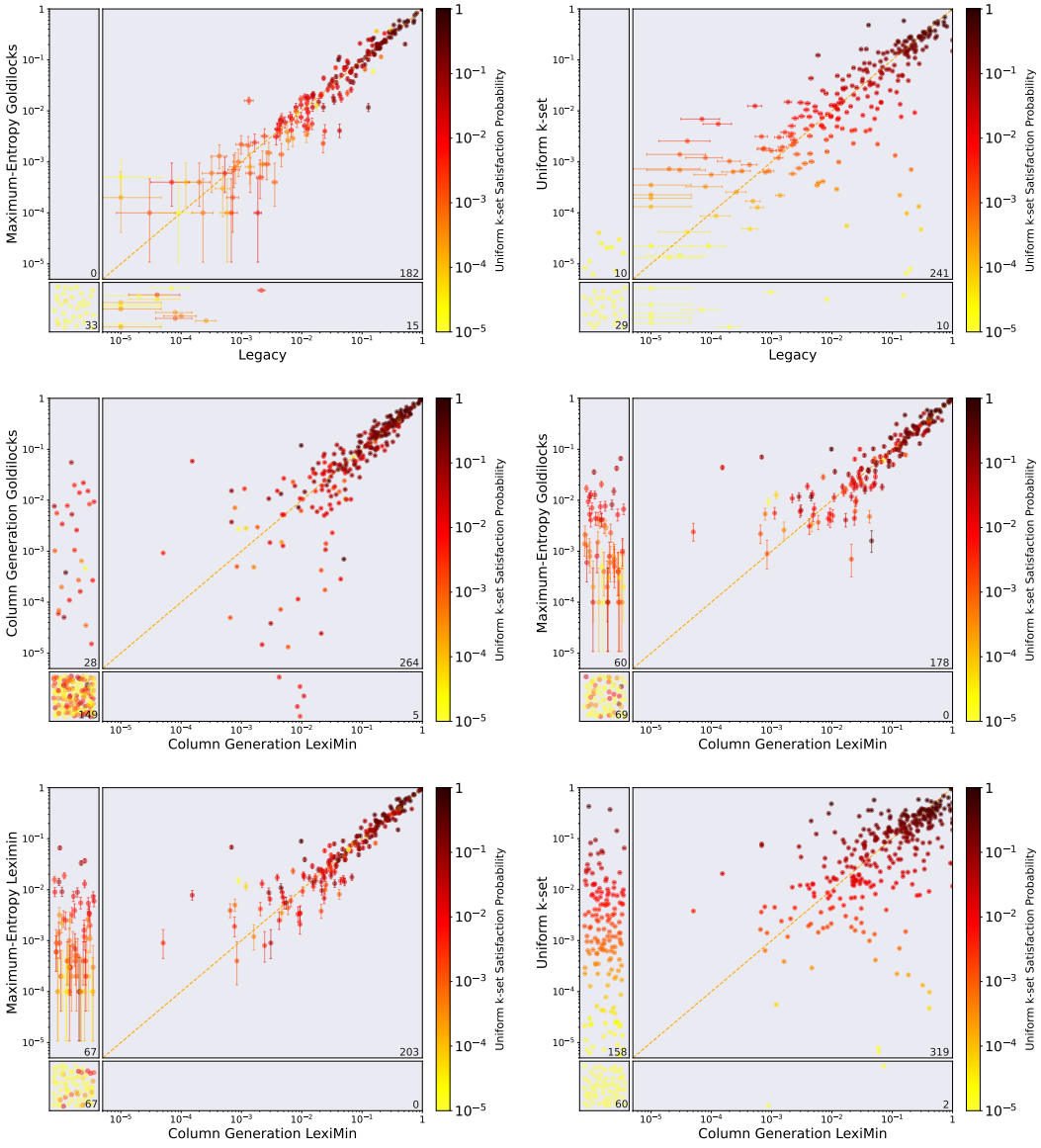
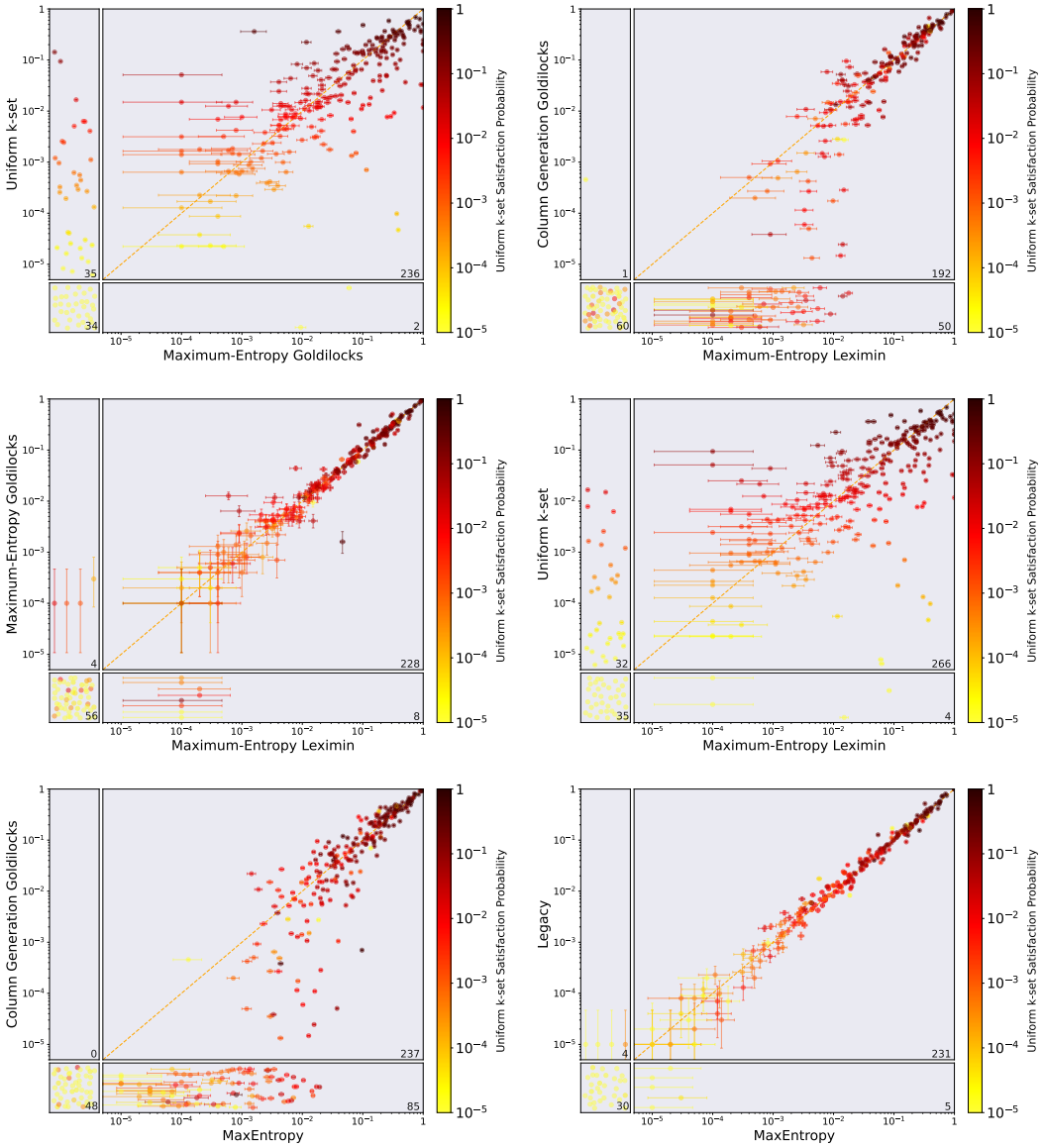
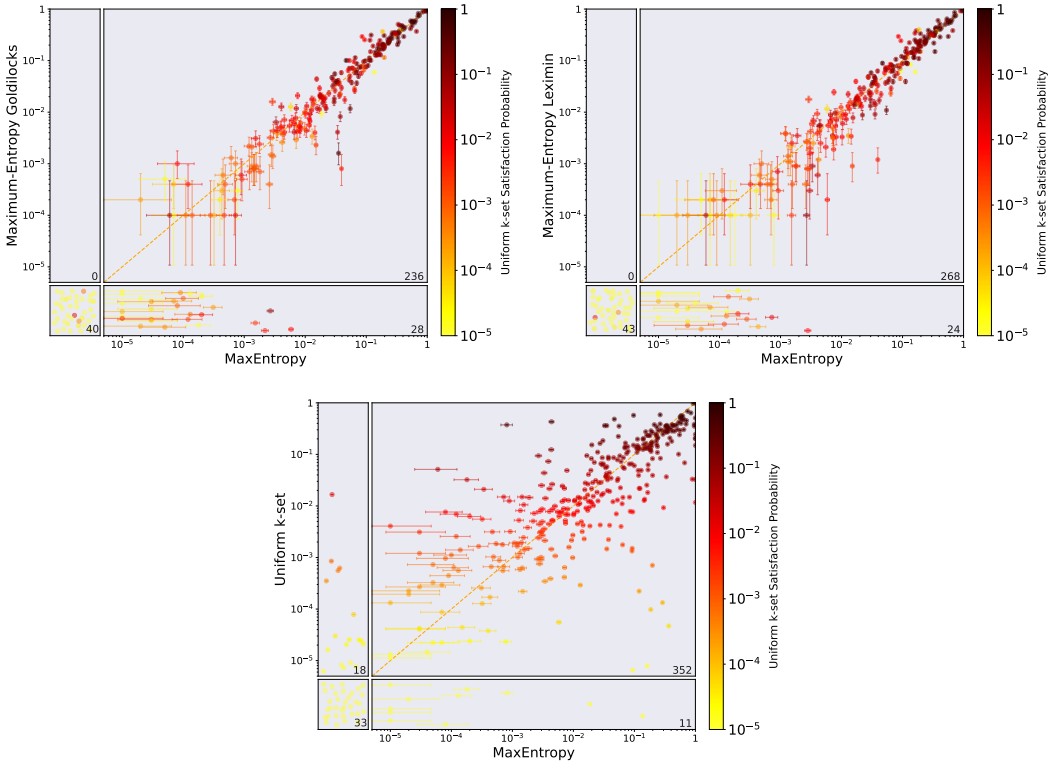


Fig. 9. Generalization probabilities colored by organization.




Maximum-Entropy Goldilocks
Legacy
Uniform k-set
Column Generation Goldilocks
Column Generation LexiMin
Maximum-Entropy Leximin
Uniform k-set Satisfaction Probability


Uniform k-set
Maximum-Entropy Goldilocks
Column Generation Goldilocks
Maximum-Entropy Leximin
Maximum-Entropy Goldilocks
Maximum-Entropy Leximin
Uniform k-set
Maximum-Entropy Leximin
Column Generation Goldilocks
MaxEntropy
Legacy
MaxEntropy
Uniform k-set Satisfaction Probability
35
236
34
2
1
192
60
50
4
228
56
8
32
266
35
4
0
237
48
85
4
231
30
5


Maximum-Entropy Goldilocks
Maximum-Entropy Leximin
Uniform k-set
MaxEntropy
Uniform k-set Satisfaction Probability
0
236
40
28
268
43
24
18
352
33
11