Properties of Position Matrices and Their Elections

Niclas Boehmer[1], **Jin-Yi Cai**[2], **Piotr Faliszewski**[3], **Austen Z. Fan**[2], **Łukasz Janeczko**[3], **Andrzej Kaczmarczyk**[3], **Tomasz Wąs**[3, 4]

[1] Algorithmics and Computational Complexity, Technische Universität Berlin
[2] University of Wisconsin-Madison
[3] AGH University
[4] Pennsylvania State University
niclas.boehmer@tu-berlin.de, jyc@cs.wisc.edu, faliszew@agh.edu.pl, afan@cs.wisc.edu, ljaneczk@agh.edu.pl, andrzej.kaczmarczyk@agh.edu.pl, twas@psu.edu

Abstract

We study the properties of elections that have a given position matrix (in such elections each candidate is ranked on each position by a number of voters specified in the matrix). We show that counting elections that generate a given position matrix is #P-complete. Consequently, sampling such elections uniformly at random seems challenging and we propose a simpler algorithm, without hard guarantees. Next, we consider the problem of testing if a given matrix can be implemented by an election with a certain structure (such as single-peakedness or group-separability). Finally, we consider the problem of checking if a given position matrix can be implemented by an election with a Condorcet winner. We complement our theoretical findings with experiments.

1 Introduction

Studies of voting and elections are at the core of computational social choice (Brandt et al. 2016). An (ordinal) election is represented by a set of candidates and a collection of voters who rank the candidates from the most to the least appealing one. Such preferences are sometimes shown in an aggregate form as a *position matrix*, which specifies for each candidate the number of voters that rank him or her on each possible position. Motivated by the connection of position matrices to the so-called maps of elections, and their similarity to weighted majority relations, we study the properties of elections with a given position matrix.

The idea of a map of elections, introduced by Szufa et al. (2020) and Boehmer et al. (2021b), is to collect a set of elections, compute the distances between them, and embed the elections as points in the plane, so that the Euclidean distance between points resembles the distance between the respective elections. Such maps are useful because nearby elections seem to have similar properties (such as, e.g., running times of winner determination algorithms, scores of winning candidates, etc.; see, e.g., the works of Szufa et al. (2020), Boehmer et al. (2021a), and Boehmer and Schaar (2022)). However, there is a catch. The positionwise distance, which is commonly used in these maps, views elections with the same position matrix as identical. Hence there might exist very different elections that, nonetheless, have identical position matrices and in a map are placed on top of each other. We want to evaluate to what extent this issue constitutes a problem for maps of elections.

The second motivation for our studies is that position matrices are natural counterparts of weighted majority relations, which specify for each pair of candidates how many voters prefer one to the other. While weighted majority relations provide sufficient information to determine winners of many Condorcet-consistent voting rules,[1] position matrices provide the information needed by positional scoring rules (i.e., rules where each voter gives each candidate a number of points that depends on this candidate's position in his or her ranking). Together with Condorcet-consistent rules, positional scoring rules are among the most widely studied single-winner voting rules. While weighted majority relations are commonly studied and analyzed (even as early as in the classic theorem of McGarvey (1953)), position matrices have not been studied as carefully.

Our contributions regard three main issues. First, we ask how similar are elections that have the same position matrix. To this end, we would like to sample elections with a given position matrix uniformly at random. Unfortunately, doing so appears to be challenging. In particular, a natural sampling algorithm requires the ability to count elections that generate a given position matrix, and we show that doing so is #P-complete. While, formally, there may exist a different approach, perhaps providing only an approximately uniform distribution, finding it is likely to require significant effort (indeed, researchers have been trying to solve related sampling problems for quite a while, without final success as of now; see, e.g., the works of Jacobson and Matthews (1996) and Hong and Miklós (2021)). We design a simpler sampling algorithm, without hard guarantees on the distribution, and use it to evaluate how different two elections with a given position matrix can be. The algorithm, albeit not central to our study, might be of independent interest when considering sampling various preference distributions (Regenwetter et al. 2006; Tideman and Plassmann 2012; Allen et al. 2017).

[1]Rules that can be computed using only the weighted majority relation are called C2 by Fishburn (1977); see also the overview of Zwicker (2015). A Condorcet winner is preferred to every other candidate by a majority of voters. Condorcet-consistent rules always select Condorcet winners when they exist. Some non-Condorcet-consistent rules are also C2 (e.g., the Borda rule).

Second, we consider structural properties of elections that generate a given position matrix (or its normalized variant, called a frequency matrix). Specifically, given a matrix we ask if there is an election that generates it and whose votes come from a given domain (such as the single-peaked domain (Black 1958), some group-separable domains (Inada 1964, 1969), or a domain given explicitly vote-by-vote as part of the input). We show polynomial-time algorithms that, given a frequency matrix and a description of a domain (e.g., via a single-peaked axis or by listing the votes explicitly), decides if there is an election with votes from this domain that generates this matrix. We apply these algorithms to test which frequency matrices from the map of elections can be generated from elections with a particular structure.[2]

Finally, we consider the problem of deciding for a given position matrix if there is an election that implements the matrix and has a Condorcet winner (i.e., a candidate who is preferred to every other one by a strict majority of voters). We evaluate experimentally which matrices from our map have such elections, provide a necessary condition for such elections to exist, and check how often this condition is effective on the map of elections. Additionally, for each matrix from the map we compute for how many different candidates there is an election that generates this matrix and where this candidate is a Condorcet winner.

With our theoretical and empirical analysis, we ultimately want to answer the question how much information is contained in a position matrix and how much flexibility is still left when implementing it.[3]

2 Preliminaries

For each $k \in \mathbb{N}_+$, by $[k]$ we denote the set $\{1, \dots, k\}$. Given a matrix X, by $X_{i,j}$ we mean its entry in row i and column j. For two equal-sized sets X and Y, by $\Pi(X, Y)$ we mean the set of one-to-one mappings from X to Y. S_n is a shorthand for $\Pi([n], [n])$, i.e., the set of permutations of $[n]$.

An *election* $\mathcal{E}$ is a pair $(\mathcal{C}, \mathcal{V})$ consisting of a set $\mathcal{C} = \{c_1, c_2, \dots, c_m\}$ of *candidates* and a collection $\mathcal{V} = (v_1, v_2, \dots, v_n)$ of *votes*, i.e., complete, strict orders over the candidates. These orders rank the candidates from the most to the least appealing one according to a given voter (we use the terms "vote" and "voter" interchangeably). If some voter v prefers candidate c over candidate c', then we write $c \succ_v c'$; we omit the subscript when it is clear from context. Given a vote $v_i : c_1 \succ c_2 \succ \cdots \succ c_m$, we say that v_i ranks c_1 on the first position, c_2 on the second one, and so on. For two votes u and v over the same candidate set, their swap distance, $\mathrm{swap}(u, v)$, is the smallest number of swaps of adjacent candidates necessary to transform u into v.

In an election $\mathcal{E} = (\mathcal{C}, \mathcal{V})$, a candidate $c \in \mathcal{C}$ is a *Condorcet winner* of the election if for every other candidate d more than half of the voters prefer c to d.

2.1 Position and Frequency Matrices

Let $\mathcal{E}$ be some election and assume that the candidates are ordered in some way. The *position matrix* of $\mathcal{E}$ (with respect to this order) is a non-negative, integral $m \times m$ matrix X such that for each $i, j \in [m]$, $X_{i,j}$ is the number of voters that rank the j-th candidate on the i-th position. By $P(\mathcal{E})$ we denote the set of all position matrices of $\mathcal{E}$ for all possible orderings of candidates. Note that the matrices in $P(\mathcal{E})$ only differ by the order of their columns.

For a position matrix $X \in P(\mathcal{E})$, where $\mathcal{E}$ is an election with n voters, the corresponding *frequency matrix* is $Y := \frac{1}{n} \cdot X$. In other words, frequency matrices are normalized variants of the position ones, where each value $Y_{i,j}$ gives the fraction of voters that rank the j-th candidate on the i-th position. Every frequency matrix is *bistochastic*, i.e., the elements in each row and in each column sum up to one. Hence, we often refer to bistochastic matrices as frequency matrices, and to integral square matrices with nonnegative entries, where each row and each column sums up to the same value, as position matrices.

We say that an election $\mathcal{E}$ *realizes* (or *generates*) a position matrix X (or, a frequency matrix Y) if $X \in P(\mathcal{E})$ (or, $n \cdot Y \in P(\mathcal{E})$, where n is the number of voters in $\mathcal{E}$). Boehmer et al. (2021b) showed that every position matrix X is realizable by some election (their result is a reinterpretation of an older result of Leep and Myerson (1999)). Yang and Guo (2016) also showed that position matrices are always realizable as part of a proof that they can be used to solve a Borda manipulation problem. Note that two distinct elections may generate the same position matrix.

Example 1. *Consider an election $\mathcal{E}$ with candidates a, b, c, and d and four votes shown below on the left. On the right we show a position matrix of this election (for the natural ordering of the candidates). Note that this is also a position matrix for an election with two votes $a \succ b \succ c \succ d$ and two votes $b \succ a \succ d \succ c$.*

$$\begin{array}{l} v_1 : a \succ b \succ c \succ d, \\ v_2 : b \succ a \succ d \succ c, \\ v_3 : a \succ b \succ d \succ c, \\ v_4 : b \succ a \succ c \succ d. \end{array} \qquad \begin{array}{c|cccc} & a & b & c & d \\ \hline 1 & 2 & 2 & 0 & 0 \\ 2 & 2 & 2 & 0 & 0 \\ 3 & 0 & 0 & 2 & 2 \\ 4 & 0 & 0 & 2 & 2 \end{array}$$

2.2 Structured Domains

We are interested in elections where the votes have some structure. For example, the single-peaked domain captures votes on the political left-to-right spectrum (and, more generally, votes focused on a single issue, such as those regarding the temperature in a room or the level of taxation).

Definition 1. *An election $\mathcal{E} = (\mathcal{C}, \mathcal{V})$ is* single-peaked *if there is an order $\rhd$ (the* societal axis*) over candidates $\mathcal{C}$ such that for each vote $v \in \mathcal{V}$ and for each $\ell \le |\mathcal{C}|$, the top ℓ candidates according to v form an interval with respect to $\rhd$.*

Intuitively, in a single-peaked election each voter first selects their favorite candidate and, then, extends his or her ranking step by step with either the candidate directly to the left or directly to the right (wrt. $\rhd$) of those already ranked.

[2]We form a map that is analogous to that used by Boehmer et al. (2021b), but which uses 8 candidates rather than 10 (using fewer candidates helps significantly with our computation times).

[3]Some proofs and details are deferred to the full version of the paper: https://arxiv.org/abs/2303.02538. The code for the experiments is available at: https://github.com/Project-PRAGMA/Position-Matrices-AAAI-2023.

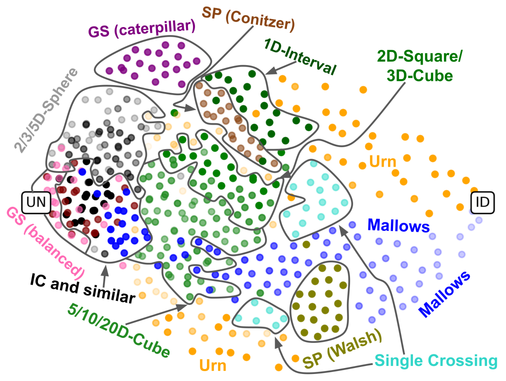


Figure 1: Map of elections visualizing the 8x80 dataset. Each dot represents an election and its color the statistical model used to generate it. xD-Cube/Sphere models are Euclidean models where the points of the candidates and voters are chosen uniformly at random from an x-dimensional hypercube/hypersphere (1D-Interval is 1D-Cube; 2D-Square is 2D-Cube). For Mallows and Urn elections the transparency of the coloring indicates the value of the used parameter.

Group-separability captures settings where the candidates have some features and the voters have hierarchical preferences over these features. Let $\mathcal{C}$ be a set of candidates and consider a rooted, ordered tree $\mathcal{T}$, where each leaf one-to-one corresponds to a candidate. A *frontier* of $\mathcal{T}$ is a vote that we obtain by reading the names of the candidates associated with the leaves of $\mathcal{T}$ from left to right. A vote is *compatible* with $\mathcal{T}$ if it can be obtained as its frontier by reversing for some nodes in $\mathcal{T}$ the order of their children. Intuitively, we view the internal nodes of $\mathcal{T}$ as features and a candidate has the features that appear on the path from it to the root.

Definition 2. *An election $\mathcal{E} = (\mathcal{C}, \mathcal{V})$ is* group-separable *if and only if there is a tree $\mathcal{T}$ over candidate set $\mathcal{C}$ such that each vote from $\mathcal{V}$ is compatible with $\mathcal{T}$.*

We focus on balanced trees (i.e., complete binary trees) and on caterpillar trees (i.e., binary trees where each non-leaf has at least one leaf as a child). If an election is group-separable for a balanced/caterpillar tree, then we say that this election is *balanced/caterpillar group-separable*.

Example 2. *The election from Example 1 is both single-peaked (for societal axis $c \rhd a \rhd b \rhd d$) and balanced group-separable (for a tree whose frontier is $a \succ b \succ c \succ d$).*

Single-peaked elections were introduced by Black (1958), and group-separable ones by Inada (1964, 1969). We mention that Inada's original definition is different from the one that we provided, but they are equivalent (Karpov 2019) and the tree-based one is algorithmically much more convenient. We point readers interested in structured domains to the recent survey of Elkind, Lackner, and Peters (2022).

2.3 Map of Elections

For our experiments, we use an *8x80* dataset that resembles those of Szufa et al. (2020), Boehmer et al. (2021b), and Boehmer et al. (2022). It contains 480 elections with 8 candidates and 80 votes generated using the same statistical models, with the same parameters, as the map of Boehmer et al. (2022). In particular, we used (i) impartial culture (IC), where each vote is equally likely, (ii) the Mallows and urn distributions, whose votes are more or less correlated, depending on a parameter, (iii) various Euclidean models, where candidates and voters are points in Euclidean spaces and the voters rank the candidates with respect to their geometric distance, and (iv) uniform distributions over balanced group-separable, caterpillar group-separable, and single-peaked elections (we refer to the uniform distribution of single-peaked elections as the Walsh model; we also use the model of Conitzer (2009) to generate single-peaked elections). See the full paper for exact descriptions. We repeated all our experiments from Sections 4 and 5 on analogously composed datasets with a varying number of candidates and voters. Specifically, we considered elections with either 4 or 8 candidates and either 40, 80, or 160 voters. The results on those datasets were similar to those for the 8x80 one.

We present our dataset as a map of elections, i.e., as points on a plane, where each point corresponds to an election (see Fig. 1). The Euclidean distances between the points resemble positionwise distances between the respective elections. For a definition of the positionwise distance, we point the reader to the work of Szufa et al. (2020) or to the full version; an important aspect of this distance is that for two elections $\mathcal{E}$ and $\mathcal{F}$ (with the same numbers of candidates and voters) it depends only on $P(\mathcal{E})$ and $P(\mathcal{F})$. Hence, we will also sometimes speak of the distance between position matrices.

Our maps include two special position matrices, the uniformity one (UN), which corresponds to elections where each candidate is ranked on each position equally often, and the identity one (ID), which corresponds to elections where all votes are identical. ID models "perfect order," whereas UN models "perfect chaos" (but note that there exist very structured elections whose position matrix is UN). UN and ID, as well as two other special points, were introduced by Boehmer et al. (2021b). For each two elections, their positionwise distance is at most as large as the distance between UN and ID (Boehmer et al. 2022).

3 Counting and Sampling Elections

Given a position matrix, it would be useful to be able to sample elections that realize it uniformly at random. Unfortunately, doing so seems challenging. Indeed, one of the natural sampling algorithms requires, among others, the ability to count elections that realize a given matrix, a task which we show to be #P-complete. While, formally, this does not preclude the existence of a polynomial-time uniform sampler (and, certainly, it does not preclude the existence of an approximately uniform one), we believe that it suggests that finding such algorithms would require deep insights; for closely related problems such insights are still elusive (Jacobson and Matthews 1996; Hong and Miklós 2021).

Formally, in the #REALIZATIONS problem we are given an $m \times m$ position matrix X (and a candidate set $\{c_1, \dots, c_m\}$, where, for each i, candidate c_i corresponds to the i-th column of X) and we ask for the number of elections that realize X. Two elections are distinct if their voter collections are distinct when viewed as multisets.

Theorem 1. #REALIZATIONS *is #P-complete even if the realizing elections contain three votes.*

3.1 Preparing for the Proof of Theorem 1

We first provide the necessary background for our proof of Theorem 1. Given a graph G, directed or undirected, a t-edge coloring is a function that associates each of its edges with one of t colors. Such a coloring is proper if for each vertex the edges that touch it have different colors. A graph is r-regular if each vertex touches exactly r edges (for directed graphs, both incoming and outgoing edges count). The #P-hardness of #REALIZATIONS follows by a reduction from the problem of counting proper 3-edge colorings of a given 3-regular bipartite (simple) graph. We refer to this problem as 3-REG.-BIPARTITE-3-EDGE-COLORING. We start by establishing that this problem is #P-hard. To prove this, we will give a reduction from a specific Holant problem, which we will call HOLANT-SPECIAL. In this problem we are given a planar, 4-regular, directed graph G, where each vertex has two incoming edges and two outgoing ones. Further, we have an embedding of this graph on the plane, which has the following property: As we consider the edges touching a given vertex in the counter-clockwise order, every other edge is incoming and every other one is outgoing. Let $\mathscr{C}$ be the set of all 3-edge-colorings of G. Given a vertex v, its four touching edges $e_1, \dots, e_4$ (listed in the counter-clockwise order, starting from some arbitrary one) and some coloring $\sigma \in \mathscr{C}$, we denote by $\sigma(v)$ the vector $(\sigma(e_1), \dots, \sigma(e_4))$. We define a function f so that:

1. $f(\sigma(v)) = 0$ if $\sigma(v)$ includes three different colors,
2. $f(\sigma(v)) = 2$ if all colors in $\sigma(v)$ are identical,
3. $f(\sigma(v)) = 1$ if $\sigma(v)$ includes two different colors and there are two consecutive edges in the counter-clockwise order that have the same color,
4. $f(\sigma(v)) = 0$ otherwise (i.e., if $\sigma(v)$ includes two different colors and each two consecutive edges in the counter-clockwise order have different colors).

The goal is to compute $\sum_{\sigma \in \mathscr{C}} \prod_{v \in V(G)} f(\sigma(v))$. Cai, Guo, and Williams (2016) have shown that doing so is #P-complete (their results are far more general than this; the problem we consider is a variant of their $\langle 2,1,0,1,0 \rangle$-HOLANT problem). The left-hand side of Fig. 2 shows an example input for HOLANT-SPECIAL.

Theorem 2. #3-REG.-BIPARTITE-3-EDGE-COLORING *is* #P-*hard.*

Proof. We give a reduction from HOLANT-SPECIAL to #3-REG.-BIPARTITE-3-EDGE-COLORING. The construction is inspired by those used by Cai, Guo, and Williams (2016). Let $G = (V, E)$ be the input graph and let the notation be as in the discussion preceding the theorem statement.

The high-level idea is to modify graph G by replacing each vertex $v \in V$ with a gadget, while keeping "copies" of edges from E. Then, the value of $f(\sigma(v))$ for some edge-coloring σ of the edges from E in G corresponds to the

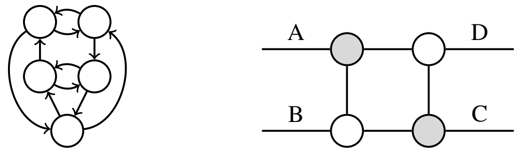


Figure 2: An example input graph (left) and the gadget used in the proof of Theorem 2. The letters label the gadget's dangling edges. The colors illustrate its bipartiteness.

number of proper 3-edge-colorings in the gadget for v assuming the "copies" of E in the constructed graph are colored according to σ. Specifically, we replace each vertex v by the gadget depicted in the right-hand side of Fig. 2. Its four dangling edges implement the original four edges of v. However, we need some care in deciding which of the dangling edges we connect to which vertices from the gadgets corresponding to the neighbors of v in G (we will return to this issue after we explain how the gadget works).

For each of our gadgets, we name the dangling edges A, B, C, and D, as shown in Fig. 2. It is now easy to see that if all dangling edges are of the same color, say 1, then there are two colorings of the remaining edges of the gadget resulting in a proper coloring: Both "vertical" edges need to have the same color (either 2 or 3), and both "horizontal" edges need to have the same color (the single remaining one). Similarly, if edges A and B have the same color, and edges C and D have the same color, then there is a unique proper coloring of the other edges. By symmetry, the same holds if both edges A and D and edges B and C have the same color. Finally, if edges A and C have the same color, and edges B and D have the same color (or, the dangling edges have three different colors) then there are no proper colorings of the remaining edges in the gadget. This way, for each vertex v and coloring σ, v's gadget implements the $f(\sigma(v))$ function.

Next we describe how we connect the dangling edges of the gadgets. If u and v are two vertices of G and there is a directed edge from u to v, then we merge one of the A and C dangling edges of u's gadget with one of the B and D dangling edges of v's gadget (which dangling edges we use is irrelevant for this proof). Since each vertex in G has two incoming and two outgoing edges, doing so is possible.

As the gadgets are bipartite themselves, and due to the way in which we connect their edges, the resulting graph G' is bipartite. It is also clear that it is 3-regular. Finally, due to the way in which 3-edge-colorings of G can be extended to proper 3-edge-colorings of G' (see the description of the gadgets), we see that the number of the latter is equal to the output of the HOLANT-SPECIAL for G. The reduction runs in polynomial-time and the proof is complete. $\square$

The above result also applies to 3-regular planar bipartite graphs. To see this, it suffices to appropriately arrange our gadgets in space (sometimes rotating them) and choose the dangling edges to connect more carefully.

3.2 The Proof of Theorem 1

The answer to an instance of #REALIZATIONS is the number of accepting paths of a non-deterministic Turing machine that constructs an election and then checks if it realizes the input matrix. As this machine works in (non-deterministic) polynomial time, #REALIZATIONS is in #P.

To show #P-hardness, we give a reduction from #3-REG.-BIPARTITE-3-EDGE-COLORING to #REALIZATIONS. Let $G = (U, V; E)$ be our input 3-regular bipartite graph, where U is the set of vertices on the left, V is the set of vertices on the right, and E is a set of edges. Since G is 3-regular, we have $|U| = |V|$. W.l.o.g., we let $U = \{u_1, \dots, u_m\}$ and $V = \{v_1, \dots, v_m\}$. We form an $m \times m$ matrix X, where each entry $X_{i,j}$ is either 1, if there is an edge between v_i and u_j, or 0, if there is no such edge. As G is 3-regular, X has exactly three ones in each row and in each column, so it is a position matrix and each election that realizes it contains three votes.

We now show that each proper 3-edge-coloring of G gives an election realizing matrix X. For a given coloring, the edges of the same color form a perfect matching in G. We interpret such a matching as a single vote. Specifically, we treat vertices from U as candidates and vertices from V as positions in the vote being constructed (e.g., if the matching contains an edge between v_i and u_j, then the vote ranks candidate u_j on position i). Hence, for each 3-coloring we get an election consisting of three votes, one for each matching associated with one color. Since all edges must be part of some matching and each edge corresponds to a single 1-entry in X, the resulting election realizes X.

Each election realizing matrix X corresponds to six 3-edge-colorings of G. Indeed, taking one 3-edge-coloring, each of the six permutations of the colors gives raise to the same election. This holds, because for a single 3-edge-coloring, each color forms an edge-disjoint matching (as opposed to graphs with parallel edges, where this would not be true). So our reduction preserves the number of solutions with a multiplicative factor of 6. This completes the proof.

3.3 Experiments

We checked experimentally how diverse are elections that generate the same position matrix. To do so, we used the isomorphic swap distance, due to Faliszewski et al. (2019).

Definition 3. *Let $\mathcal{E} = (\mathcal{C}, \mathcal{V})$ and $\mathcal{F} = (\mathcal{D}, \mathcal{U})$ be two elections, where $\mathcal{C} = \{c_1, \dots, c_m\}$, $\mathcal{D} = \{d_1, \dots, d_m\}$, $\mathcal{V} = (v_1, \dots, v_n)$, and $\mathcal{U} = (u_1, \dots, u_n)$. Their isomorphic swap distance is:*

$$d_{\mathrm{swap}}(\mathcal{E}, \mathcal{F}) = \min_{\sigma \in S_n} \min_{\pi \in \Pi(\mathcal{C}, \mathcal{D})} \sum_{i=1}^{n} \mathrm{swap}(\pi(v_i), u_{\sigma(i)}),$$

where $\pi(v_i)$ is the vote v_i where each candidate $c \in \mathcal{C}$ is replaced with candidate $\pi(c)$.

Intuitively, the isomorphic swap distance between two elections is the summed swap distance of their votes, provided we first rename the candidates and reorder the votes to minimize this value. Maps of elections could be generated using the isomorphic swap distance instead of the positionwise one, and they would be more accurate than those based on the positionwise distance (Boehmer et al. 2022), but the isomorphic swap distance is NP-hard to compute and challenging to compute in practice (Faliszewski et al. 2019); indeed, we use a brute-force implementation.

Boehmer et al. (2022) have shown that the largest isomorphic swap distance between two elections with m candidates and n voters is $\frac{1}{4}n(m^2 - m)$ (up to minor rounding errors; for this result, see their technical report). Whenever we give an isomorphic swap distance between two elections (with the same numbers of candidates and voters), we report it as a fraction of this value.

As we do not have a fast procedure for sampling (approximately) uniformly at random elections that realize a given matrix, we use the following naive approach (let X be an $m \times m$ position matrix):

1. We form an election $\mathcal{E} = (\mathcal{C}, \mathcal{V})$, where $\mathcal{C} = (c_1, \dots, c_m)$ and $\mathcal{V}$ is initially empty. For each $i \in [m]$, candidate c_i corresponds to the i-th column of the matrix.
2. We repeat the following until X consists of zeros only: We form a bipartite graph with vertices $c_1, \dots, c_m$ on the left and vertices $1, \dots, m$ on the right; there is an edge between c_j and i exactly if $X_{i,j} > 0$. We draw uniformly at random a perfect matching in this graph (it always exists; (Leep and Myerson 1999))—we generate it relying on the standard self-reducibility of computing perfect matchings and using the classic reduction to computing the permanent (Valiant 1979), which we compute using the formula of Ryser (1963).[4] Given such a matching, we form a vote v where each candidate $c_j \in \mathcal{C}$ is ranked on the position to which he or she is matched. We extend $\mathcal{E}$ with vote v and we subtract from X the position matrix of the election that contains v as the only vote.

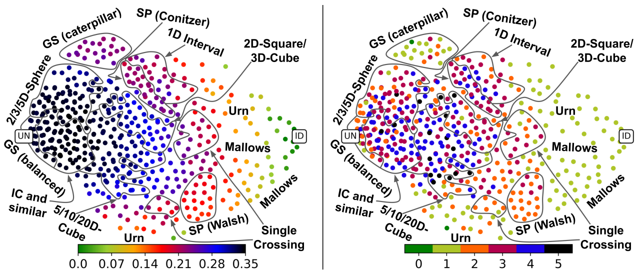
In essence, the above procedure is a randomized variant of the algorithm presented by Boehmer et al. (2021b) to show that every position matrix is realized by some election.

We performed the following experiment: (i) For each election from the 8x80 dataset we computed its position matrix, (ii) using the naive sampler, we generated 100 pairs of elections that realize it, and, (iii) for each pair of elections, we computed their isomorphic swap distance. We report the results in Figure 3a, where each dot has a color that corresponds to the farthest distance computed for the respective matrix.[5] For elections close to UN, this distance can be very large. Indeed, for about half of the elections (all located close to UN) this distance is larger than 20% of the maximum possible isomorphic swap distance. On the other hand, elections realizing position matrices in the vicinity of ID are much more similar to each other, which is quite natural.

While we used a naive sampling algorithm rather than a uniform one, the results are sufficient to claim that for many position matrices—in particular, those closer to UN than to ID—there are two elections that generate them, whose isomorphic swap distance is very large. If we had a uniform

[4]We used a python module called *permanent* (https://git.peteshadbolt.co.uk/pete/permanent) by Pete Shadbolt. In principle, we could have used an approximately uniform sampler that runs in polynomial time (Jerrum, Sinclair, and Vigoda 2004; Bezáková et al. 2008), but they are too slow in practice.

[5]We tried 100 pairs for two reasons. First, each computation is quite expensive. Second, even with testing 10 pairs the results were very similar to those for 100 pairs (if we reported average distances, the results also would not change very much).



(a) Swap distance. (b) Condorcet winners.

Figure 3: Maps with our experimental results for the 8x80 dataset. On the left, we show the maximum isomorphic swap distance found for elections realizing a given matrix. On the right, we show the number of candidates that can be Condorcet winners in elections realizing the matrices.

sampler, the distances could possibly increase, but the overall conclusion would not change. Indeed, we ran our experiment for an analogous dataset, but for elections with 4 candidates and 16 voters; in this case we computed maximum possible isomorphic swap distances by generating all elections realizing a given matrix. The results are analogous. (For this experiment we also counted how many elections realize a given matrix and the results were strongly correlated with the above described results for the maximum distance.)

4 Recognizing Structure

In this section we consider the problem of deciding if a given (arbitrary) position or frequency matrix can be realized by elections whose votes come from some domain (e.g., the single-peaked or group-separable one). Overall, we find that if a precise description of the domain is part of the input (e.g., if we are given the societal axis for the single-peaked domain), then for frequency matrices we can often solve this problem in polynomial time. For position matrices our results are less positive and less comprehensive. The reason why frequency matrices are easier to work with here is that they only specify fractions of votes where a given candidate appears on a given position, whereas position matrices specify absolute numbers of such votes and thus are less flexible.

Let us fix a candidate set $\mathcal{C}$. We consider sets $\mathcal{D}$ of votes, called domains, specified in one of the following ways:

1. *explicit*: $\mathcal{D}$ contains explicitly listed votes, or
2. *single-peaked*: $\mathcal{D}$ contains all votes that are single-peaked with respect to an explicitly given axis $\triangleright$, or
3. *group-separable*: $\mathcal{D}$ contains all group-separable votes that are compatible with a given rooted, ordered tree $\mathcal{T}$, where each leaf is associated with a unique candidate. We only consider balanced or caterpillar trees.

The next theorem is our main result of this section.

Theorem 3. *There is a polynomial-time algorithm that given a frequency matrix X and an explicit, single-peaked, or group-separable (balanced or caterpillar) domain $\mathcal{D}$, decides if there is an election that realizes X, and whose votes all belong to $\mathcal{D}$.*

For example, given a frequency matrix X and a societal axis $\triangleright$, we can check if there is an election that realizes X and is single-peaked with respect to $\triangleright$. A similar result for single-peakedness and a variant of weighted majority relations is provided by Spaanjard and Weng (2016).

The proof of Theorem 3 is quite involved and is available in the full version of the paper, but we mention two issues. First, some of our algorithms proceed by solving appropriate linear programs and, in principle, the elections that they discover might have exponentially many votes with respect to the length of the encoding of the input. This is not a problem as our algorithms do not build these elections explicitly. Second, while our algorithms need an explicit description of the domain, such as the societal axis or the underlying tree, for the balanced group-separable domain we can drop this assumption, and we can even deal with position matrices:

Theorem 4. *There is a polynomial-time algorithm that given a frequency (or position) matrix X decides if the matrix can be realized by a balanced group-separable election.*

Interestingly, if instead of taking the entire balanced group-separable domain (for a given tree) we only allow for an explicitly specified subset of its votes, the problem becomes NP-hard.

Theorem 5. *Given a set $\mathcal{D}$ of votes, listed explicitly, and a position matrix X, it is* NP*-hard to decide if there is an election that realizes X and whose votes are all from $\mathcal{D}$. This holds even if the votes in $\mathcal{D}$ are both single-peaked and balanced group-separable.*

Proof. We reduce from the NP-hard X3C problem, where we are given a set $\mathcal{U} = \{u_1, \dots, u_{3m}\}$ of $3m$ elements and a family $\mathcal{S} = \{S_1, \dots, S_n\}$ of size-3 subsets of $\mathcal{U}$. We ask if there are m sets from $\mathcal{S}$ whose union is $\mathcal{U}$.

Let I be our input instance of X3C. We form a $6m \times 6m$ position matrix X with values $m-1$ on the diagonal, and where for each odd column there is value 1 directly below the $m-1$ entry, and for each even column there is value 1 directly above the $m-1$ entry (all other entries are equal to 0). The matrix looks as follows:

$$\begin{bmatrix} m-1 & 1 & 0 & 0 & \cdots & 0 \\ 1 & m-1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & m-1 & 1 & \cdots & 0 \\ 0 & 0 & 1 & m-1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & 1 \\ 0 & 0 & 0 & 0 & \cdots & m-1 \end{bmatrix}.$$

We let the candidate set be $\mathcal{C} = \{c_1, \dots, c_{6m}\}$, where for each $i \in [6m]$, candidate c_i corresponds to the i-th column. For each set $S_\ell = \{u_i, u_j, u_k\}$ we include in the domain $\mathcal{D}$ a vote v_ℓ that is equal to $c_1 \succ c_2 \succ \cdots \succ c_{6m}$ except that c_{2i} and c_{2i+1} are swapped, c_{2j} and c_{2j+1} are swapped, and c_{2k} and c_{2k+1} are swapped. We claim that there is an election that realizes X and whose votes all belong to $\mathcal{D}$ if and only if I is a *yes*-instance of X3C.

Let us assume that there are m sets from $\mathcal{S}$ whose union is $\mathcal{U}$ and, w.l.o.g., that these sets are $S_1, \dots, S_m$. One can verify that election $(\mathcal{C}, (v_1, \dots, v_m))$ realizes X. Indeed, since $S_1, \dots, S_m$ cover $\mathcal{U}$, for each candidate c_i there are

$m-1$ votes where c_i is ranked on the i-th position, and a single vote where c_i is either ranked one position higher or one position lower, depending on the parity of i.

For the other direction, let us assume that there is an election $\mathcal{E}$ that realizes X and, w.l.o.g., that it contains votes $v_1, \dots, v_m$ (all the votes must be distinct as otherwise some non-diagonal entry of this election's position matrix would have value greater than 1). Since $\mathcal{E}$ realizes X, for each $i \in [3m]$ there is exactly one vote in $\mathcal{E}$ that ranks c_{2i} below c_{2i+1}. This means that for each $u_i \in \mathcal{U}$, there is exactly one set among $S_1, \dots, S_m$ that includes u_i. Hence, I is a *yes*-instance of X3C.

Finally, all votes in $\mathcal{D}$ are single-peaked with respect to societal axis $c_{6m-1} \succ c_{6m-3} \succ \cdots \succ c_3 \succ c_1 \succ c_2 \succ c_4 \succ \cdots \succ c_{6m}$ and are balanced group-separable with respect to a tree whose frontier is $c_1 \succ c_2 \succ \cdots \succ c_{6m}$ (formally, for this, we need to have a number of candidates equal to a power of two, which is easy to ensure). □

An Experiment. Using our algorithms from Theorems 3 and 4, we checked for each of the frequency matrices from our 8x80 dataset whether it is realizable by a single-peaked or a caterpillar/balanced group-separable election (for each election we tried all societal axes and all caterpillar trees). For all three domains we found that for each election in the dataset, its frequency matrix can be realized by an election from the domain only if the election itself belongs to this domain.[6] This indicates that frequency matrices (and also position matrices) of elections from restricted domains have specific features that are not likely to be produced by elections sampled from other models.

5 Condorcet Winners

Our final set of results regards Condorcet winners in elections that realize a given position matrix.

First, we consider the problem of deciding if a given position matrix can be realized by an election where a certain candidate is a Condorcet winner. In general, the complexity of this problem remains open, but if we restrict our attention to elections that only contain votes from a given set we obtain a hardness result (even if the input matrix can always be realized using votes from the given set).

Theorem 6. *Given a set $\mathcal{D}$ of votes, listed explicitly, a position matrix X (which can be realized by an election containing only votes from $\mathcal{D}$)[7], and a candidate c, it is* NP*-hard to decide if there is an election realizing X, in which c is a Condorcet winner and all votes come from $\mathcal{D}$.*

Making partial progress on the general problem, we provide a necessary condition for the existence of an election realizing a given position matrix in which a given candidate c is a Condorcet winner. Roughly speaking, for each $i \in [m]$, our condition looks for a set S of candidates (different from c) that frequently appear in the first i positions. If occurrences of candidates from S on the first i positions are "sufficiently frequent" compared to how often candidate c appears in the first $i-1$ positions, and both S and i are "small enough," then c cannot be a Condorcet winner in any election realizing the matrix.

Theorem 7. *For each position matrix X and each $c \in [m]$, if there is an election $\mathcal{E}$ realizing X, where c is a Condorcet winner, then for every $i \in [m]$ and $S \subseteq [m]$, it holds that $\sum_{j\in S}\sum_{k=1}^{i} X_{k,j} \le |S|\cdot\lfloor\frac{n-1}{2}\rfloor + \sum_{k=1}^{i-1}\big(X_{k,c}\cdot\min(|S|, i-k)\big)$. The condition can be checked in polynomial time.*

Experiment 1. We tested our condition on the elections from the 8x80 dataset. We checked for each election and each candidate whether the condition is satisfied, but there is no election realizing the matrix in which the candidate is a Condorcet winner (using an ILP formulation of the problem). It turns out that this situation is very rare: among all 480 matrices in the 8x80 dataset (i.e., among the position matrices of the elections from the dataset), there were only 6 in which there was one candidate for which our condition gave the wrong answer (there were none with more than one such candidate). Thus, our condition appears to be quite an effective way to detect potential Condorcet winners.

Experiment 2. We conclude with an experiment where for each position matrix from our 8x80 dataset we count how many different candidates are a Condorcet winner in at least one election realizing the matrix. The results are in Fig. 3b.

First, we observe that while 94 elections from our 8x80 dataset do not have a Condorcet winner, only two of them have a position matrix that cannot be realized by an election with a Condorcet winner. Second, examining Fig. 3b, we see that for most matrices there are multiple different possible Condorcet winners, with the average number of Condorcet winners being 2.6 and 120 matrices having four or more possible Condorcet winners. The number of possible Condorcet winners is correlated with the position of the matrix on the map. Generally speaking, it seems that the closer a matrix is to UN, the more possible Condorcet winners we have. However, in the close proximity of UN there is a slight drop in the number of possible Condorcet winners. Overall, these results confirm that elections realizing a given position matrix can be very different from each other (in terms of pairwise comparisons of candidates).

6 Conclusions

We have analyzed various properties of elections that realize given position or frequency matrices. Among others, (i) we have shown algorithms for deciding if such elections can be implemented using votes from particular structured domains, and (ii) we have found that for a given matrix, such elections can be very diverse. The latter result is witnessed by the fact that two elections realizing a matrix may have large isomorphic swap distance and may have different Condorcet winners. Hence, while maps of elections (based on position matrices) certainly are very convenient tools for visualizing some experimental results (including ours), for others their value might be limited. It would be interesting to find such experiments and establish their common features.

[6]Elections from our dataset that are part of a restricted domain are almost exclusively sampled from models that are guaranteed to produce such elections.

[7]Verifying this condition is not part of the problem as, by Theorem 5, such a test is NP-hard. It is simply a feature of our reduction.

Acknowledgments

NB was supported by the DFG project ComSoc-MPMS (NI 369/22). This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 101002854).

References

Allen, T. E.; Goldsmith, J.; Justice, H. E.; Mattei, N.; and Raines, K. 2017. Uniform random generation and dominance testing for CP-nets. *Journal of Artificial Intelligence Research*, 59: 771–813.

Bezáková, I.; Štefankovič, D.; Vazirani, V.; and Vigoda, E. 2008. Accelerating Simulated Annealing for the Permanent and Combinatorial Counting Problems. *SIAM Journal on Computing*, 37(5): 1429–1454.

Black, D. 1958. *The Theory of Committees and Elections*. Cambridge University Press.

Boehmer, N.; Bredereck, R.; Faliszewski, P.; and Niedermeier, R. 2021a. Winner Robustness via Swap- and Shift-Bribery: Parameterized Counting Complexity and Experiments. In *Proceedings of IJCAI-2021*, 52–58.

Boehmer, N.; Bredereck, R.; Faliszewski, P.; Niedermeier, R.; and Szufa, S. 2021b. Putting a Compass on the Map of Elections. In *Proceedings of IJCAI-2021*, 59–65.

Boehmer, N.; Faliszewski, P.; Niedermeier, R.; Szufa, S.; and Wąs, T. 2022. Understanding Distance Measures Among Elections. In *Proceedings of IJCAI-2022*, 102–108.

Boehmer, N.; and Schaar, N. 2022. Collecting, Classifying, Analyzing, and Using Real-World Elections. Technical Report arXiv:2204.03589 [cs.GT], arXiv.org.

Brandt, F.; Conitzer, V.; Endriss, U.; Lang, J.; and Procaccia, A., eds. 2016. *Handbook of Computational Social Choice*. Cambridge University Press.

Cai, J.-Y.; Guo, H.; and Williams, T. 2016. The complexity of counting edge colorings and a dichotomy for some higher domain Holant problems. *Research in the Mathematical Sciences*, 3: 18.

Conitzer, V. 2009. Eliciting Single-Peaked Preferences Using Comparison Queries. *Journal of Artificial Intelligence Research*, 35: 161–191.

Elkind, E.; Lackner, M.; and Peters, D. 2022. Preference Restrictions in Computational Social Choice: A Survey. Technical Report arXiv:2205.09092 [cs.GT], arXiv.org.

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

Fishburn, P. 1977. Condorcet Social Choice Functions. *SIAM Journal on Applied Mathematics*, 33(3): 469–489.

Hong, L.; and Miklós, I. 2021. A Markov Chain on the Solution Space of Edge-Colorings of Bipartite Graphs. Technical Report arXiv:2103.11990 [cs.GT], arXiv.org.

Inada, K. 1964. A Note on the Simple Majority Decision Rule. *Econometrica*, 32(32): 525–531.

Inada, K. 1969. The Simple Majority Decision Rule. *Econometrica*, 37(3): 490–506.

Jacobson, M.; and Matthews, P. 1996. Generating Uniformly Distributed Random Latin Squares. *Journal of Combinatorial Designs*, 4(6): 405–437.

Jerrum, M.; Sinclair, A.; and Vigoda, E. 2004. A Polynomial-Time Approximation Algorithm for the Permanent of a Matrix with Nonnegative Entries. *Journal of the ACM*, 51(4): 671–697.

Karpov, A. 2019. On the number of group-separable preference profiles. *Group Decision and Negotiation*, 28(3): 501–517.

Leep, D.; and Myerson, G. 1999. Marriage, Magic, and Solitaire. *The American Mathematical Monthly*, 106(5): 419–429.

McGarvey, D. 1953. A Theorem on the Construction of Voting Paradoxes. *Econometrica*, 21(4): 608–610.

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

Ryser, H. J. 1963. *Combinatorial Mathematics*, volume 14 of *The Carus Mathematical Monographs*. MAA Press: An Imprint of the American Mathematical Society.

Spaanjard, O.; and Weng, P. 2016. Single-peakedness Based on the Net Preference Matrix: Characterization and Algorithms. In *Proceedings of COMSOC-2016*.

Szufa, S.; Faliszewski, P.; Skowron, P.; Slinko, A.; and Talmon, N. 2020. Drawing a Map of Elections in the Space of Statistical Cultures. In *Proceedings of AAMAS-2020*, 1341–1349.

Tideman, T.; and Plassmann, F. 2012. Modeling the Outcomes of Vote-Casting in Actual Elections. In Felsenthal, D.; and Machover, M., eds., *Electoral Systems: Paradoxes, Assumptions, and Procedures*, chapter 9, 217–251. Springer.

Valiant, L. 1979. The Complexity of Computing the Permanent. *Theoretical Computer Science*, 8(2): 189–201.

Yang, Y.; and Guo, J. 2016. Exact algorithms for weighted and unweighted Borda manipulation problems. *Theoretical Computer Science*, 622: 79–89.

Zwicker, W. 2015. Introduction to the Theory of Voting. In Brandt, F.; Conitzer, V.; Endriss, U.; Lang, J.; and Procaccia, A. D., eds., *Handbook of Computational Social Choice*, chapter 2. Cambridge University Press.