ECAI 2023
K. Gal et al. (Eds.)

doi:10.3233/FAIA230326

Algorithmic Recognition of 2-Euclidean Preferences

Bruno Escoffier[a,b], **Olivier Spanjaard**[a;*] **and Magdaléna Tydrichová**[a]

[a]Sorbonne Université, CNRS, LIP6, Paris, France
[b]Institut Universitaire de France, Paris, France
ORCiD ID: Bruno Escoffier https://orcid.org/0000-0002-6477-8706,
Olivier Spanjaard https://orcid.org/0000-0002-9948-090X,
Magdaléna Tydrichová https://orcid.org/0000-0002-0329-0264

Abstract. A set of voters' preferences on a set of candidates is 2-Euclidean if candidates and voters can be mapped to the plane so that the preferences of each voter decrease with the Euclidean distance between her position and the positions of candidates. Based on geometric properties, we propose a recognition algorithm, that returns either "yes" (together with a planar positioning of candidates and voters) if the preferences are 2-Euclidean, or "no" if it is able to find a concise certificate that they are not, or "unknown" if a time limit is reached. Our algorithm outperforms a quadratically constrained programming solver achieving the same task, both in running times and the percentage of instances it is able to recognize. In the numerical tests conducted on the PrefLib library of preferences, 91.5% (resp. 4.5%) of the available sets of complete strict orders are proven not to be (resp. to be) 2-Euclidean, and the status of only 4.5% of them could not be decided. Furthermore, for instances involving 5 (resp. 6, 7) candidates, we were able to find planar representations that are compatible with 87.4% (resp. 58.1%, 60.1%) of voters' preferences.

1 Introduction

This paper deals with the recognition of *Euclidean preferences* on the plane, a domain restriction studied in computational social choice but also in many other fields, such as economics, psychology or political science. The idea of d-Euclidean preferences is to view candidates and voters as points in $\mathbb{R}^d$. The nearer a voter is to a candidate, the more this candidate is preferred by the voter. The distance is usually measured by the Euclidean l_2 norm, but the l_1 and l_∞ norms have also been considered [16, 29, 10, 20]. Given a specific domain restriction and a set of preferences (also called *preference profile* hereafter), a recognition algorithm aims at deciding whether the preferences belong or not to the domain restriction, and if possible also provide a concise certificate of membership or non-membership. Recognition algorithms have been proposed for various domain restrictions in social choice, among which single-peaked preferences on an axis [2] or on a tree [33] or on a circle [30], single-crossing preferences [15], intermediate preferences on median graphs [12], etc. For more details on preference restrictions in computational social choice, the reader can for instance refer to the recent survey by Elkind et al. [18].

In spite of many works aiming at understanding the structure of d-Euclidean preferences (see Section 2), the recognition problem appears to be very challenging for $d \geq 2$, and remains widely open from an algorithmic perspective:

- From a theoretical viewpoint, Peters [29] showed that the recognition problem (under ℓ_2) is NP-hard for any fixed dimension $d \geq 2$, and that some Euclidean preference profiles require exponentially many bits in order to specify any Euclidean embedding. He also proved the conjecture of Chen and Grottke [9] that, for any $d \geq 2$, the set of d-Euclidean preference profiles does not admit a good characterisation by forbidden substructures unless $\exists\mathbb{R} \subseteq$ coNP.
- From a practical viewpoint, Peters [29] pointed out that using an ETR-solver to recognize Euclidean profiles for $d \in \{2, 3\}$ is inefficient in practice, and to the best of our knowledge, no efficient algorithm is known even for small-size instances.

We propose here a new approach to recognize Euclidean preferences on the plane ($d = 2$). The principle of the algorithm is simple: we fix the candidates' positions randomly in the plane, we determine the set of votes that are compatible with these positions, and then we check if the input profile is included in this set of votes. If yes, the input is 2-Euclidean. We repeat this test a certain number of times (changing the random positions) to detect 2-Euclidean profiles. We complement this with a test aiming at detecting when an input profile is *not* 2-Euclidean. If the preferences are neither detected as 2-Euclidean nor as not 2-Euclidean after a given time limit, then the algorithm returns that their status is unknown. Put another way, the algorithm we propose belongs to the class of incomplete methods, i.e., there is no guarantee that it will eventually conclude [24]. Nevertheless, in the numerical tests carried out (see Section 5), the status of only 4.5% of real-world preference data could not be determined.

Although the naive implementation of this idea is not very efficient, we propose several theoretical and algorithmic improvements that make it way more operational. We made some experiments both on real-world and synthetic datasets involving up to 8 candidates and found that 91.5% of them are not 2-Euclidean. To go further and counterbalance this informative but rather negative result, we also investigated the percentage of preferences that can be explained by a 2-Euclidean representation. Interestingly enough, the results we obtain are of the same order of magnitude as the empirical findings of Sui et al. [32] regarding multi-dimensional single-peakedness, who showed that in real-world election datasets, one can find 2-dimensional axes on which 47–65% voters are single-peaked. Besides, empirical studies conducted with the implemented algorithm allow us to provide some new insights on some questions about Euclidean preferences on the plane, such as the number of inclusion-wise maximal profiles for a given number of candidates.

[*] Corresponding Author. Email: olivier.spanjaard@lip6.fr.

The paper is organized as follows. The related work is presented in Section 2. Then, Section 3 is devoted to some preliminaries, before the description of the algorithm itself in Section 4. Finally, experimental results are provided and discussed in Section 5. Supplementary material is available on Zenodo (10.5281/zenodo.8157233)

2 Related work

The topic of Euclidean preferences has been studied both in economics and in computational social choice.

The domain of Euclidean preferences on the line (i.e., $d=1$), also known as *1-Euclidean preferences*, were introduced by C. Coombs under the name of "unfolding model" [14, 13]. Several polynomial-time recognition algorithms for 1-Euclidean preferences have been proposed, first in 1994 [15], and then in 2010 [25] and 2014 [17]. All of these algorithms use the observation that any 1-Euclidean preference profile is necessarily single-peaked [4] and single-crossing [31]: while Doignon and Falmagne [15] combine both conditions, Knoblauch [25] starts by finding a single-peaked axis and Elkind and Faliszewski [17] establishes a single-crossing order. In all cases, linear programming is used to find the positions of voters and candidates on the real line. It was shown by Chen et al. [11] that the domain cannot be characterized by finitely many forbidden minors, contrary to, for example, the single-peaked domain [1]. Chen and Grottke [9] characterized the sub-domain of "small" 1-Euclidean profiles: in particular, they showed that any profile over at most 5 candidates is 1-Euclidean if and only if it is both single-peaked and single-crossing.

The domain of Euclidean preferences was also studied for $d \geq 2$, starting from the works of Bennett and Hays [3, 21] (with the l_2 norm). In addition to setting the geometrical framework, some structural questions were discussed - among others, the maximal size (number of distinct votes) of a Euclidean profile is given, as a function of the number m of candidates and the dimension d. Furthermore, given a preference profile, the authors studied the minimal dimension d such that the profile is d-Euclidean. They give some techniques to obtain bounds on d. Some decades later, Bogomolnaia and Laslier [5] showed that $d \geq \min\{m, n-1\}$ is necessary to ensure that any profile of n votes on m candidates is d-Euclidean (i.e., Euclidean in dimension d). They also showed that any profile of at most 2 voters or 3 candidates is 2-Euclidean. These results were deepened by Bulteau and Chen [7]: they proved that any profile of 3 voters on at most 7 candidates is 2-Euclidean. Finally, Kamiya et al. [23], and later Escoffier et al. [20], gave a characterization of 2-Euclidean profiles on 4 candidates. Note that there are also some works focusing on metric preferences using l_1 and l_∞ norms [9, 20], providing in particular results on the size of Euclidean profiles under these norms, geometrical properties of representations under these norms, and differences between ℓ_1, ℓ_2 and ℓ_∞ Euclidean profiles.

We would like to stress that the recognition problem considered in the above works and in the present paper differs from the *non-metric multidimensional unfolding problem* (see e.g., [27]), where the aim is to determine a positioning of voters and candidates in the plane that minimizes a measure of error w.r.t. the input preference profile (optimizing goodness of fit [26]). The main problem encountered in multidimensional unfolding is that of *degenerate* solutions, that is, placing all the candidates in the same position yields a very small violation of the constraints, or placing all the candidates on a circle and all the voters clustered around the circle center. Although multiple works have proposed alternative methods to overcome this problem (see e.g., [6, 8]), they are often not able to distinguish between Euclidean and non-Euclidean profiles, as emphasized by Peters [29].

3 Notations and preliminaries

We consider a set $V=\{v_1, v_2, \ldots, v_n\}$ of n voters who express their preferences over a set $C = \{c_1, c_2, \ldots, c_m\}$ of m candidates. The *preference* of each voter v is a strict total order $>_v$ over C : we say that voter v prefers candidate c_i to candidate c_j if $c_i >_v c_j$. For conciseness, we will often write the preference $c_i >_v c_j >_v \ldots >_v c_k$ as $(c_i, c_j, \ldots, c_k)$. We call *a preference profile* the set of all voters' preferences $P = \{>_{v_1}, >_{v_2}, \ldots, >_{v_n}\}$. We denote by $\|\cdot\| : \mathbb{R}^2 \to \mathbb{R}$ the classical 2-Euclidean norm defined by $\|x\| \mapsto \sqrt{x_1^2 + x_2^2}$ for $x=(x_1, x_2)$.

Definition 1. *(2-Euclidean preferences) A preference profile P of a set V of* n *voters over a set C of* m *candidates is 2-Euclidean if there exists a mapping $f : V \cup C \to \mathbb{R}^2$ such that for each $v \in V$ and each couple of candidates $c_i, c_j \in C$:*

$$c_i >_v c_j \Leftrightarrow \|f(v) - f(c_i)\| < \|f(v) - f(c_j)\|.$$

If there is no ambiguity, we will just say *Euclidean* profile instead of *2-Euclidean* profile. Note that in the sequel we will assume, w.l.o.g., that no couple of voters have the same preference (if so, we can simply remove one of them).

The mapping f is called *a Euclidean representation of P*. Obviously, such a mapping is not necessarily unique. Let us interpret the definition in a slightly different way : imagine that the positions $f(c)$ are fixed for each candidate $c \in C$. Our goal is now to extend f on $C \cup V$ (i.e., define the positions of the voters $v \in V$) in such a way that f fulfills the condition in Definition 1. For each voter v, we define the set D_v^f of possible positions $f(v)$:

$$D_v^f = \{p \in \mathbb{R}^2 : \forall c_i, c_j \in C, c_i >_v c_j \Rightarrow \|p - f(c_i)\| < \|p - f(c_j)\|\}.$$

Then, mapping f is a Euclidean representation of P if and only if $D_v^f \neq \emptyset$ for each $v \in V$. We note that, following this idea, to build a Euclidean representation of P we only need to define the positions of the candidates in such a way that each D_v^f is non-empty. The set of points equidistant to c_i and c_j, called bisector, is denoted by $B^f(c_i, c_j)$, or simply $B(c_i, c_j)$ if no confusion is possible. The notations are illustrated in Figure 1.

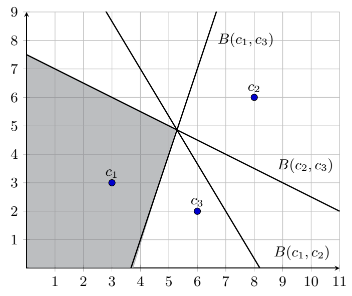

Figure 1: A Euclidean representation of the profile consisting of the 6 possible preferences on 3 candidates c_1, c_2, c_3. The grey area corresponds to D_v^f for $c_1 >_v c_3 >_v c_2$.

Definition 1 can be now reformulated as follows : The preference profile P is *2-Euclidean* if there exists a mapping $f : C \to \mathbb{R}^2$ such that the set D_v^f induced by f is non-empty for each $v \in V$.

Following this idea, for a given position of candidates let us define the set of all ranking (votes) that are compatible with these positions.

Definition 2. *Let C be a set of candidates, and $f : C \to \mathbb{R}^2$ be a mapping. We define the profile P_f associated to f as the set of all rankings v such that D_v^f is non-empty.*

We have the following property, used in our algorithm.

Property 1. *A profile P is 2-Euclidean if and only if there exists a mapping $f : C \to \mathbb{R}^2$ such that $P \subseteq P_f$.*

Furthermore, we will use the notation $[P]$ to refer to any profile obtained by a permutation of the names of the candidates in P. By extension, we will write $P \subseteq [P_f]$ (resp. $P = [P_f]$) to state that P is included in (resp. is equal to) a profile obtained by renaming the candidates of P_f. For instance,

$$\left\{ \begin{array}{c} c_2 \succ c_3 \succ c_1, \\ c_1 \succ c_3 \succ c_2 \end{array} \right\} = \left[\left\{ \begin{array}{c} c_1 \succ c_2 \succ c_3, \\ c_3 \succ c_2 \succ c_1 \end{array} \right\} \right]$$

because the profile on the left hand side is obtained from the profile on the right hand side by renaming c_1 (resp. c_2, c_3) in c_2 (resp. c_3, c_1). Property 1 remains clearly true if we replace $P \subseteq P_f$ by $P \subseteq [P_f]$.

Finally, let us define the notion of *maximal* Euclidean profile (we recall that the votes are pairwise distinct):

Definition 3. *A Euclidean profile P is maximal is for any preference $>_{v'} \notin P$, the profile $P' = P \cup \{>_{v'}\}$ is not Euclidean.*

Let us call a representation function f *degenerate* if either three points $f(c_i)$, $f(c_j)$ and $f(c_k)$ are aligned, or at least 4 bisectors intersect in the same point. By slightly moving the positions of candidates, it is easy to see that for any Euclidean profile there exists a non-degenerate representation of it. Note that if the representation function f is non-degenerate, then the profile P_f is maximal.[1]

4 Recognition of 2-Euclidean preferences

Assume that we have a profile P in input, and we want to decide whether it is 2-Euclidean or not. We propose here an algorithm based on the following principles. First, as some necessary conditions for a profile to be Euclidean have been identified in the literature, the algorithm checks if these conditions are fulfilled. If it is not the case, the profile is not Euclidean and NO is returned. Otherwise, the algorithm tries to "guess" the positions $f(c_1), f(c_2), \dots, f(c_m)$ of candidates. It then builds the (maximal) Euclidean profile P_f associated to the representation f, and finally checks if $P \subseteq [P_f]$. If this is the case, the profile is Euclidean and YES is returned. Otherwise, we reiterate this process of guessing positions of candidates. Finally, if none of the tests succeeded, then the status is undefined and UNKNOWN is returned. The pseudocode is given in Algorithm 1.

Algorithm 1 is_euclid(P)

```
Input : a preference profile $P$
Output: NO if $P$ is not Euclidean, YES if $P$ is Euclidean, UNKNOWN if not decided
if there is a NO-certificate for $P$ return NO
while timeout not reached do
    Generate a random Euclidean representation $f$ of candidates
    Build the profile $P_f$ associated to $f$
    if $P \subseteq [P_f]$ return YES
end while
return UNKNOWN
```

We now detail how each step of this algorithm is performed, explaining the main ideas used to make it as efficient as possible.

[1] Indeed, it can be shown that then $|P_f| = \mathrm{ub}(m)$, where $\mathrm{ub}(m) = m(3m-10)(m-1)(m+1)/24 + m(m-1)+1$ is the maximal number of pairwise distinct votes in a 2-Euclidean preference profile on m candidates (this can be easily derived from a result by Bennett and Hayes [3], see e.g. [20]).

4.1 NO-Certificates

As explained above, the first step of the algorithm is to detect profiles that are *not* Euclidean. To do so, we use in our algorithm two necessary conditions (also called NO-certificates in which follows, as they guarantee that the profile is not Euclidean) known in the literature, based on the size of the profile and on some forbidden substructures. We will discuss possible improvements of this step of the algorithm in the conclusion.

- **Cardinality condition:** Using the maximum number $\mathrm{ub}(m)$ of pairwise distinct votes in a Euclidean profile on m candidates (see footnote in Section 3), the algorithm simply outputs NO if $n > \mathrm{ub}(m)$. Note that this is actually tested for any restriction of the profile to a subset of candidates. More formally, given a profile P over a set C of m candidates, and given $S \subseteq C$, we denote by $P_{|S}$ the restriction of P to S where, for each $v \in V$, we define the preference $>_{v_{|S}}$ of voter v as follows: $\forall c_i, c_j \in S, c_i >_{v_{|S}} c_j$ iff $c_i >_v c_j$. In other words, $P_{|S}$ is a copy of P in which we have kept only the candidates of S (and removed the possible duplicate preferences). Any subset S of candidates such that $|P_{|S}| > \mathrm{ub}(|S|)$ is a NO-certificate for P.
- **Condition on subprofiles on 4 candidates:** Kamiya et al. [23] showed that for 4 candidates, there are only 3 maximal Euclidean profiles P_1, P_2, P_3 (up to a permutation of candidates). Consequently, a profile P on 4 candidates is Euclidean if and only if there exists $i \in \{1, 2, 3\}$ for which $P \subseteq [P_i]$. We use this characterization to derive a NO-certificate, as follows. We say that P' is a *k-restriction* of P if there exists $S \subseteq C$ of cardinal k such that $P' = P_{|S}$. We will note by $\mathcal{P}_k$ the set of all k-restrictions of a given profile P: $\mathcal{P}_k = \{P_{|S} : S \subseteq C, |S| = k\}$.
  Obviously, if P is Euclidean, for each k, any k-restriction of P is also Euclidean. We use the characterization of Euclidean profiles on 4 candidates to identify non-Euclidean profiles P on m candidates: we generate all $\binom{m}{4}$ elements of $\mathcal{P}_4$ and check if they are subprofiles of $[P_1]$, $[P_2]$ or $[P_3]$, one of the 3 maximal Euclidean profiles on 4 candidates. Any 4-restriction of P that is not Euclidean is a NO-certificate for P.

4.2 The random generation of the representation f

The most straightforward idea consists in picking up the positions $f(c_1), f(c_2), \dots, , f(c_m)$ according to a uniform distribution in the square $[0, M] \times [0, M]$ (for some constant M). However, this turns out to be inefficient in practice because it does not take into account the input profile P, which yields a low chance that the positions are correctly guessed. This led us to adapt the random generation process by observing that, if a candidate c_i is ranked last by at least one voter v, then for any Euclidean representation f, point $f(c_i)$ must be a vertex of the convex hull of the set $\{f(c_1), f(c_2), \dots, f(c_m)\}$. This is shown in Lemma 1 (proof in Supplementary Material A.1). We therefore generate positions of candidates by imposing that the number of vertices of the convex hull of the set of positions of candidates is at least the number of candidates ranked last by at least one voter in P.

Lemma 1. *Let $C = \{c_1, \dots, c_m\}$ be a set of candidates and $f : C \to \mathbb{R}^2$ an injective mapping of the candidates in the plane. Given $c \in C$ and $\{i_0, i_1, \dots, i_k\} \subseteq \{1, \dots, m\}$ such that $f(c_{i_0})$ is a convex combination of $f(c_{i_1}), f(c_{i_2}), \dots, f(c_{i_k})$. For each* possible *position $f(v)$ of a voter v in the plane (i.e., inducing a strict order $>_v$), there exists $i_v \in \{i_1, i_2, \dots, i_k\}$ such that $c_{i_0} >_v c_{i_v}$. In particular, c_{i_0} is never ranked last in the profile P_f associated to f.*

Hence, to generate a random Euclidean representation, we take into account the number of vertices that the convex hull of the set of points $\{f(c_1), f(c_2), \dots, f(c_m)\}$ should have. We call this number the *size* of the convex hull in the following. More precisely, we proceed as follows:

1. We go through the input profile P and we determine the set $C_L \subseteq C$ of the candidates ranked last at least once in P.
2. We pick randomly, according to a given probability distribution π, an integer $k \in \{L, \dots, m\}$, where $L = |C_L|$. In fact, as there are L candidates ranked last at least once in P, the convex hull of any representation of P must contain at least L vertices (and possibly more if P is not maximal).
3. Using Valtr's algorithm (see [34]), we generate uniformly at random a polygon with k vertices in a square $[0, M] \times [0, M]$ (for some constant M). We assume that the set of positions of vertices corresponds to $\{f(c_1), f(c_2), \dots, f(c_k)\}$. We choose then (uniformly at random) $m - k$ points inside the polygon in order to fix the remaining positions $f(c_{k+1}), \dots, f(c_m)$. We recall that, following Lemma 1, only the candidates $c_1, \dots, c_k$ can be ranked last in the associated profile P_f.

Let us now specify the probability distribution π used in step 2 above. We propose here three different distributions (the efficiency of which will be compared in the experimental study):

- **Uniform distribution:** We pick the size of the convex hull between L and m using the uniform distribution π_U. Formally, for each $k \in \{L, \dots, m\}$, we have $\pi_U(k) = 1/(m - |C_L| + 1)$.
- **Vertices-based distribution:** The idea is to fit the probability distribution of the size of the convex hull that we would get by picking m points uniformly at random points in the plane (this distribution of the sizes has no reason to be uniformly distributed). Denoting by $\pi_m(k)$ the probability of having a convex hull of size exactly k for m randomly drawn points, we define the conditional probability distribution π_m^L as follows, as we want to consider only instances with a convex hull of size at least L:
$$\pi_m^L(k) = \begin{cases} 0 & \text{if } k < L, \\ \frac{\pi_m(k)}{\sum_{i=L}^{m} \pi_m(i)} & \text{if } k \geq L. \end{cases}.$$
To evaluate the values $\pi_m(k)$, we used a Monte Carlo simulation: we generated N random instances (with $N = 10^6$). Denoting by n_k the number of instances for which the size of the convex hull was equal to k, we set $\pi_m(k) = \frac{n_k}{N}$.
- **Profile-based distribution:** The idea is to fit the probability distribution of the number of candidates ranked last in the set of maximal Euclidean profiles on m candidates, distribution that we estimate here also using a Monte Carlo simulation (see Supplementary material A.2 for a precise definition and computation).

4.3 The profile P_f associated to a mapping f

The previous method generates a random representation function $f : C \to \mathbb{R}^2$. We now describe how, given the points $f(c_1), f(c_2), \dots, f(c_m)$, we determine the (unique) maximal Euclidean profile P_f associated to f. As any preference area borders at least one intersection point of bisectors, we examine the different intersection points to determine the set of all preference areas induced by f. If we assume w.l.o.g. that f is non-degenerate[2], it amounts to consider all triples and pairs of candidates, because four or more bisectors do not intersect in a non-degenerate representation. For the sake of brevity, we only detail how we proceed for triples, the idea being similar for pairs.

For each triple of candidates c_i, c_j, c_k, we compute the point I_{ijk} which is the circumcenter of the triangle $\{f(c_i), f(c_j), f(c_k)\}$. As f is non-degenerate (i.e., the points $f(c_i), f(c_j)$ and $f(c_k)$ are not aligned), this point exists and is unique. We define a fictitious voter v_{ijk} such that $f(v_{ijk}) = I_{ijk}$. By circumcenter definition, v_{ijk} is indifferent between candidates c_i, c_j and c_k because $\|f(v_{ijk}) - f(c_i)\| = \|f(v_{ijk}) - f(c_j)\| = \|f(v_{ijk}) - f(c_k)\|$. Let us denote by d (resp. d_c) this common distance (resp. $\|f(v_{ijk}) - f(c)\|$). The preference of v_{ijk} is of the form:

$$R_1 >_{v_{ijk}} \{c_i, c_j, c_k\} >_{v_{ijk}} R_2,$$

where R_1 (resp. R_2) is a strict order on candidates $c \in C$ such that $d_c < d$ (resp. $d_c > d$).

As said above, I_{ijk} is the only point at equal distance from c_i, c_j and c_k. It is easy to see that there exists $\varepsilon > 0$ such that moving v_{ijk} within the open ball $\text{Ball}(I_{ijk}, \varepsilon)$ will only impact the order of c_i, c_j and c_k in $>_{v_{ijk}}$, without changing the order of the other candidates (see Figure 2). More formally, for any strict order R_{ijk} on the set $\{i, j, k\}$, there is a point $p \in \text{Ball}(I_{ijk}, \varepsilon)$ such that the preference order $>_{v_p}$ of a voter v_p positioned in p is of the form $R_1 >_{v_p} R_{ijk} >_{v_p} R_2$. Each point I_{ijk} thus allows us to determine

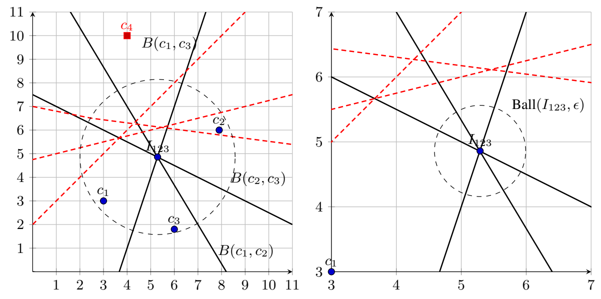

Figure 2: Left part: there are 6 preference areas around the intersection point I_{ijk}. Right part: there exists a neighbourhood of I_{ijk} not crossed by any other perpendicular bisector.

6 preferences of the profile P_f, one per each of the 6 possible strict orders R_{ijk}.

As each preference area D_v can be adjacent to more than one intersection point, there will be some duplicates among the found preferences. The procedure is therefore completed by deleting duplicate preferences in the resulting profile.

4.4 Testing if the profile P is a subprofile of $[P_f]$

Finally, we need to check if $P \subseteq [P_f]$. This operation reveals to be very time-consuming (thus making the algorithm non-operational) if it is not optimized. The following procedure allows us to greatly alleviate the computational burden. Let us denote by $>_{v_0}$ a preference arbitrarily chosen in P. Assume that P is a subprofile of $[P_f]$. Then $>_{v_0}$ necessarily corresponds to some preference $>_v$ of P_f, up to a permutation σ of the candidates in P_f. If $c_{i_1} >_{v_0} c_{i_2} >_{v_0} \dots >_{v_0} c_{i_m}$ and $c_{j_1} >_v c_{j_2} >_v \dots >_v c_{j_m}$, then the permutation is defined by $\sigma(c_{j_k}) = c_{i_k}$ for $k \in \{1, \dots, m\}$. By applying the permutation σ to all the preferences of P_f, we have then $P \subseteq P_f^\sigma$, where P_f^σ denotes the profile obtained from P_f by permuting the candidates according

[2] In practice, we can get a degenerate representation while using random generation. However, it is easy to detect, so we can simply reject it. As this phenomenon almost never occurs, it has no impact on the performance of the algorithm in practice.

to σ. Denoting by σ_v the permutation obtained for a preference v in P_f (for the same choice of v_0 in P), testing whether $P \subseteq [P_f]$ thus amounts to testing if there exists $v \in P_f$ such that $P \subseteq P_f^{\sigma_v}$.

In practice, we actually proceed symmetrically, i.e., we test if there exists $v \in P$ such that $P^{\sigma_v} \subseteq P_f$ for an arbitrarily chosen preference $v_0 \in P_f$ (testing whether $[P] \subseteq P_f$ is equivalent to testing whether $P \subseteq [P_f]$). This indeed allows us to take advantage of precomputing all profiles of $[P]$, which avoids repeated computation of profile permutations. We store these profiles in a lookup table where they can be found quickly. Instead of performing $O(m^4)$ profile permutations for *each* profile P_f (because there are $O(m^4)$ votes v in P_f [3]), $m!$ permutations of P are computed only *once* in the precomputation phase. For the small values of m we are working with (profiles involving up to 9 candidates), it represents a significant saving in computation time. Indeed, by storing the permuted profiles of P in a lookup table, we can consider a thousand times more profiles P_f per a fixed period of time than without this table.

5 Experimental study

The recognition algorithm has been implemented in C++ (the code is available on 10.5281/zenodo.8157233). To analyse how it performs in practice, several numerical tests were carried out on an Intel Xeon X5677 (3.46 GHz base, 3.73 GHz turbo). Besides the computation time, we paid a special attention to the *recognition rate* of the algorithm, defined as the fraction of instances for which it was able to conclude (yes or no) within a given timeout, fixed at one hour in the results we present. The section is organized as follows. We first present our results on real-world data from the PrefLib library [28] in Section 5.1. In Section 5.2, we give the results on synthetic data: we start by explaining how the data were generated, and then we study the recognition rate and execution time in function of the number of voters and candidates. While the recognition rate turned out to be high in most cases both on the real-world and synthetic datasets, we also noted some weak points for specific values of m and n, where many instances remained undecided. To evaluate whether this was mainly due to the no-test or the yes-test, we made also some experiments on randomly generated 2-Euclidean profiles, to study the recognition rate on these instances. Moreover, we make a brief comparison with the Gurobi Quadratic Constraint Optimizer, which is outperformed by our algorithm both on the running time and the recognition rate. Finally, in Section 5.3, we discuss several observations made from our experiments about the number of maximal profiles over 5 and 6 candidates.

5.1 *Experimental study on real-world data*

PrefLib is a reference library that contains several types of preference data. For our experiments, we focused only on the complete strict order datasets (SOC data files). There are 7741 such files.

5.1.1 *Recognition rates*

About 4% of the instances were detected as Euclidean and 91.5% as non-Euclidean, while 4.5% remained undecided after the timeout of one hour. More detailed information about the recognition rates on the different datasets as well as some characteristics of instances are summarized in Supplementary material A.3. Let us look closer to the profiles that were detected as 2-Euclidean. Actually, most of these profiles are "trivially" Euclidean: [7] proved that any profile containing at most 3 candidates, or at most 2 voters, or at most 3 voters and 7 candidates, is 2-Euclidean (and in our tests, we directly answered yes on such profiles). In addition, there are few non-trivial 2-Euclidean profiles among the YES-instances: 10 among 13 (non-trivial) profiles on 5 candidates, 3 among 18 profiles on 6 candidates, 5 among 22 profiles on 7 candidates and 1 among 27 profiles on 8 candidates were detected as 2-Euclidean.

5.1.2 *Real-world data subprofiles*

According to the previous paragraph, we see that many (non-trivial) profiles on 5, 6, 7, or 8 candidates were non-Euclidean (or remained undecided). A natural question is to study up to what degree these preferences fit the 2-Euclidean structure. This has been studied for other preference structures such as one-dimensional single-peakedness (see for instance [19]) and multi-dimensional single-peakedness [32]. One of the most popular structural approximation is the so-called *voter deletion (VD)*, which focuses on the minimum number of voters to remove from a given profile so that the remaining subprofile fits the preference structure. Alternatively, it looks at the maximum-size subprofile (in terms of voters) fitting the preference structure. We adopt this approach here, and try to evaluate to what extent the Preflib profiles are near to being 2-Euclidean.

Using a heuristic approach to find 2-Euclidean subprofiles, we found that the average proportion of the profile that can be explained by a 2-Euclidean representation was at least 87 % for profiles on 5 candidates (i.e., on average, given a Preflib profile P on 5 candidates with N voters, we found a 2-Euclidean subprofile of P of size $0.87N$), at least 59 % on 6 candidates, at least 60 % on 7 candidates and at least 42 % on 8 candidates. This is much larger than what is usually observed for 1-dimension, even for the less restrictive single-peakedness condition: Sui *et al.* [32] reported that the best axis explains only 2.87% (resp. 0.38%) of the profile for the 2002 Irish general election in Dublin West (resp. North). In contrast, this is of the same order of magnitude as the results the same authors obtained for two-dimensional single-peakedness, namely 65.7% (resp. 47.3%) for Dublin West (resp. North).

Our heuristic follows a simple greedy strategy: for every profile in Preflib involving between 5 and 8 candidates, we have considered its subprofile composed of the k pairwise distinct preferences with the most occurrences (several voters may have the same preference), and run our algorithm on it to determine if it is 2-Euclidean or not. Then, we determined the maximum value k_e for which P was detected as 2-Euclidean. Denoting by K_e the number of voters whose preference corresponds to one of these k_e preferences, the ratio K_e/N gives the proportion of the profile which is 2-Euclidean, where N is the total number of voters in the profile. The detailed results can be found in Supplementary material A.4, Table 5.

5.2 *Experimental study on synthetic data*

To generate a random profile of *n* preferences over *m* candidates, we draw n rankings uniformly at random from the $m!$ possible rankings (impartial culture assumption). In all the tests, the timeout was again set at one hour (note that it was also the time bound set in [29] for the experiments with `nlsat` [22]). For each couple of values (n, m), the results are averaged over 10000 instances (preference profiles).

5.2.1 *Probability distribution used for the convex-hull size*

As discussed in the Section 4.2, several probability distributions have been considered to pick up randomly m two-dimensional points representing the positions of candidates, namely the *uniform distribution* (on the size of the convex hull, imposed to be greater or equal to the number L of candidates ranked last by at least one voter), the *vertices-based* distribution and the *profile-based* distribution.

The profile-based distribution outperformed the two others for 5 candidates. For a larger number of candidates, as it was computationally hard to estimate this distribution with a sufficient precision, we compared the performances of the algorithm with the uniform distribution on the one side and the vertices-based distribution of the other side. It turns out that, in general, the time needed to recognize a profile (i.e., detect whether it is Euclidean or not) is shorter using the uniform distribution. Nevertheless, this has hardly any impact on the recognition rate for profiles with up to 6 candidates because the recognition time remains below the timeout set to one hour. The difference in the recognition rates obtained for the two distributions grows for profiles with more candidates, as the recognition time then approaches the timeout.

5.2.2 *Phase transition*

We now give the recognition rates (proportion of positive, negative and undecided instances) for profiles with 5, 6, 7 and 8 candidates, depending on the number of voters. The results are given in Figure 3. We observe a phase transition: below some threshold (that depends on m) almost all the inputs are Euclidean, whereas above some other threshold almost all the inputs are not, and there is a transition phase in between. We note that the phase transition is done very quickly: it occurs between 5 and 7 voters with 6 candidates, between 5 and 6-7 voters with 7 candidates and between 4 and 6 voters for profiles with 8 candidates.

Concerning the recognition rates, quite unsurprisingly, we observe that the largest proportions of undecided instances occur in the phase transition. Figure 3 gives the curves of the recognition rate in function of the number of voters, for profiles involving 6 to 8 candidates. Clearly, the proportion of undecided instances in the phase transition increases with the number of candidates (see the peaks of the curves of the proportions of undecided instances in the four plots).

5.2.3 *Running times*

First note that testing the NO-certificate is performed very quickly (typically less than 1 second for instances with up to 8 candidates and 10 voters). Thus, for the couples (m, n) that are beyond the phase transition in Figure 3, as almost all the instances are recognized as non-Euclidean, the median (i.e., 50th percentile) running time of the algorithm is very small (typically less than 1s).

In any case, the NO-certificate part of the algorithm does not significantly impact the global running time. Thus, we found more interesting to focus on the (median) running time of the YES-part of the algorithm. To do this, we measure the running time of the algorithm on the yes-instances (i.e., recognized as Euclidean). The results are given in Table 1. We note that the running time is very far from the timeout of 3600 seconds. The dash mark means that we could not get meaningful information: indeed, they correspond to cases where the proportion of Euclidean profiles is very small (after the phase transition), and we could not produce enough Euclidean instances to get a relevant value of the median (note that, as said before, in these cases the NO-certificates allows anyway to reach a very small median running time for the algorithm).

0

n m	4	5	6	7	8	9	10
5	0	0	0.003	0.015	0.082	0.68	0.81
6	0	0.2	0.3	24	85	392	443
7	3.543	8.6	19.9	193.3	-	-	-
8	6.2	31.275	-	-	-	-	-

Table 1: The median running time [s] for random instances where the answer is YES.

5.2.4 *Performances on Euclidean input profiles*

The problem of the random generation of input profiles with the impartial culture assumption is that we do not know if the generated profile is Euclidean or not. Hence, we cannot distinguish the non-Euclidean profiles not detected by the NO-certificate from the Euclidean profiles not recognized by the algorithm. In consequence, we do not know if we should focus more on improving the YES-part of the algorithm (finding a representation) or on finding out a more efficient NO-certificate to reduce the proportion of undecided instances. For this reason, we did another set of tests, with only Euclidean profiles as input, to have some ideas on the performance of the YES-part of the algorithm.

Regarding the recognition rate, all Euclidean profiles of at most 25 voters over 5 candidates have been recognized within the fixed timeout (one hour). For 6 candidates, all the instances with at most 8 voters were also recognized, and then the recognition rate decreased very slowly with the number of voters for instances involving between 9 and 25 voters, while remaining greater than 97%. The recognition rates for profiles with 7, 8 and 9 candidates are given in Table 2 (upper table). For 9 candidates, the recognition rate decreases from 7 voters, due to the fact that we often reach the timeout. Globally, we note that the recognition rates are pretty good (including the “grey areas” in the phase transitions), which makes us believe that the algorithm performs quite well for recognizing Euclidean profiles with up to 9 candidates, and that one should focus more on improving the NO-certificates to reduce the proportion of undecided profiles.

The medians of running times are summarized in Table 2 (lower table). The table stops at 10 voters, but we could go further. Indeed, for 5 candidates, whatever the number of voters, the median time did not exceed 10 seconds. For 6 candidates, the profiles with up to 25 voters could be recognized in the median time of 60 seconds, and for 7 candidates, the profiles with up to 16 candidates were recognized in the median time of 300 seconds. Comparing the recognition times of Table 1 to those of Table 2 (lower table), we note that the second ones are better. This is not surprising as the profiles are not generated the same way. The way they are generated here, with random points in the plane, is closer to the way the algorithm recognizes them.

5.2.5 *Comparison with the Gurobi QCP optimizer*

We have implemented a Gurobi Quadratic Constraint Program to recognize Euclidean profiles[3] (see Supplementary material A.5).

At first, we have performed a set of experiments with random profiles as input. However, *no* profile was detected as non-Euclidean within the timeout of one hour by the solver (the solver was only able to identify some Euclidean instances). So the solver is not able to identify non-Euclidean instances, in contrast with our algorithm.

Then, we evaluated the efficiency of the solver on Euclidean instances. The results are summarised in the Table 3. We give here only

[3] To the best of our knowledge, there is no algorithm available in the literature that would guarantee to return an exact solution if it exists.

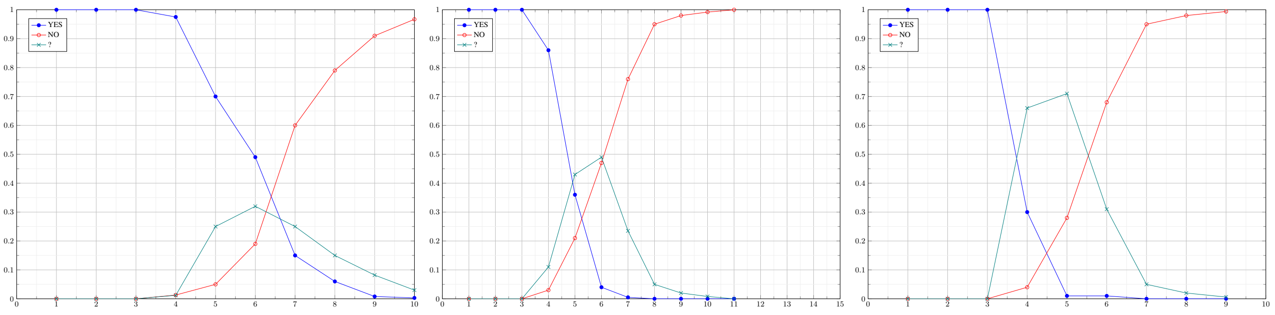

Figure 3: Recognition rates for random profiles with (from left to right) 6, 7 and 8, w.r.t. the number of voters.

m \ n	4	5	6	7	8	9	10
7	1	1	1	1	0.99	0.99	0.97
8	1	1	0.98	0.95	0.89	0.81	
9	1	0.96	0.81	0.63	0.21		

m \ n	4	5	6	7	8	9	10
5	0	0	0	0	0.0006	0.0006	0.001
6	0	0.004	0.01	0.03	0.05	0.09	0.13
7	0.04	0.26	0.94	2.76	7	16.75	29.6
8	3.5	7.5	38.1	108	415	1417	
9	43.3	254	592	691	1824		

Table 2: The recognition rate (upper table) and median running time [s] (lower table) for Euclidean input profiles.

the results on profiles involving 5 to 7 voters and 5 or 6 candidates: in fact, the solver recognized almost no profile within the timeout of 1 hour with more voters or candidates. We observe that our algorithm significantly outperforms the QCP-model[4].

	Our algorithm		Gurobi QCPsolver	
	5 candidates			
voters	RR	median [s]	RR	median [s]
5	1	0	0.53	14
6	1	0	0.49	66
7	1	0	0.34	118
	6 candidates			
voters	RR	median [s]	RR	median [s]
5	1	0.004	0.22	186
6	1	0.01	0.2	234
7	1	0.03	0.06	926

Table 3: Recognition rates (RR) and median running times of our algorithm and the Gurobi QCP model, for Euclidean input profiles.

5.3 *Maximal Euclidean profiles for 5 and 6 candidates*

As explained before, it is known that there are 3 maximal Euclidean profiles on 4 candidates (up to a permutation of candidates). This gives a simple characterization of Euclidean profiles on 4 candidates. As far as we know, nothing is known on the number of maximal Euclidean profiles with more than 4 candidates. For $m = 5$ or 6, we use our algorithm to build a set of maximal Euclidean profiles, thus providing a lower bound on this number. We simply repeat the following procedure: we first generate m random points in the plane (using the uniform distribution on the convex hull size). If the positions are non-degenerated, we build the (maximal) Euclidean profile associated to these positions. We add this profile to our set, after having tested that it is not already contained in it (up to a permutation of candidates).

We could find 543 maximal profiles on 5 candidates[5], and about 230 000 maximal profiles on 6 candidates! These lower bounds already indicate that the number of maximal profiles seem to grow incredibly fast with the number of candidates.

For 5 candidates, we conjecture that 543 is the exact number, as all these 543 profiles were found within 3 hours and the timeout was set up to one week (see Figure 4 in Supplementary material A.6 giving the convergence speed). Note that if the conjecture is true, then we would get a characterization of the Euclidean profiles on 5 candidates. This would in turn give a potentially powerful NO-certificate using subprofiles on 5 candidates. As another point, our experiments indicate that, not surprisingly, some profiles appear much more frequently than others: while 519 solutions were discovered in 93 792 iterations, 710 192 iterations were needed to get 536 solutions, and we made 7 129 863 iterations to get all 543 solutions. We recall that if the distribution over k solutions were uniform, the expected number of iterations to obtain all solutions would be $k(\log k)$.

For 6 candidates, we even think that the lower bound of 230 000 is actually very far from the correct number. Indeed, we found these profiles in about 250 000 iterations only, clearly without reaching convergence (we stopped here as with such a high number of profiles the test of uniqueness becomes time-consuming).

6 Conclusion

In this work, we provided an algorithm for recognizing 2-dimensional Euclidean profiles, and evaluated its performance on both real-world and synthetic datasets. A first question that arises from our work concerns the phase transition we observed in the experiments. It would be very interesting to find some mathematical arguments proving bounds on this phenomenon. This would allow to see how this transition evolves with the number of candidates.

A second question concerns the characterization of Euclidean profiles on 5 candidates. We were able to generate a set of 543 maximal Euclidean profiles, but we do not know if this list is exhaustive or not. Finding such a characterization would improve the NO-test of the algorithm (hence in particular the recognition rate), which seems to be, from our experiments, the key point to address. More generally, we can think of adding forbidden structures in the NO-certificates.

Finally, it would be of course a natural extension of our work to provide an algorithm able to recognize d-Euclidean preferences for $d = 3$ or bigger.

[4] Note that regarding PrefLib profiles, the QCP-model basically does not provide any answer, because most of the instances are non-Euclidean or too large to be solved within 1 hour (i.e., with at least 6 candidates or 6 votes).

[5] The list of these profiles is available on 10.5281/zenodo.8157233.

References

[1] Miguel A Ballester and Guillaume Haeringer, ‘A characterization of the single-peaked domain’, *Social Choice and Welfare*, **36**(2), 305–322, (February 2011).

[2] John Bartholdi III and Michael A Trick, ‘Stable matching with preferences derived from a psychological model’, *Operations Research Letters*, **5**(4), 165–169, (1986).

[3] Joseph F. Bennett and William L. Hays, ‘Multidimensional unfolding: Determining the dimensionality of ranked preference data’, *Psychometrika*, **25**(1), 27–43, (Mar 1960).

[4] Duncan Black, ‘On the rationale of group decision-making’, *The Journal of Political Economy*, **56**(1), 23–34, (1948).

[5] Anna Bogomolnaia and Jean-Francois Laslier, ‘Euclidean preferences’, *HAL, Working Papers*, (01 2004).

[6] Ingwer Borg and Patrick Groenen, *Modern Multidimensional Scaling: Theory and Applications*, Springer Science & Business Media, 2005.

[7] Laurent Bulteau and Jiehua Chen, ‘2-dimensional Euclidean preferences’, *CoRR*, **abs/2205.14687**, (2022).

[8] Frank Busing, Patrick Groenen, and Willem Heiser, ‘Avoiding degeneracy in multidimensional unfolding by penalizing on the coefficient of variation’, *Psychometrika*, **70**, 587–587, (09 2005).

[9] Jiehua Chen and Sven Grottke, ‘Small one-dimensional Euclidean preference profiles’, *Social Choice and Welfare*, 1–28, (2021).

[10] Jiehua Chen, Martin Nöllenburg, Sofia Simola, Anaïs Villedieu, and Markus Wallinger, ‘Multidimensional Manhattan preferences’, in *LATIN 2022*, volume 13568 of *Lecture Notes in Computer Science*, pp. 273–289. Springer, (2022).

[11] Jiehua Chen, Kirk Pruhs, and Gerhard J. Woeginger, ‘The one-dimensional euclidean domain: finitely many obstructions are not enough’, *Soc. Choice Welf.*, **48**(2), 409–432, (2017).

[12] Adam Clearwater, Clemens Puppe, and Arkadii Slinko, ‘Generalizing the single-crossing property on lines and trees to intermediate preferences on median graphs’, in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, (2015).

[13] C.H. Coombs, *A Theory of Data*, Wiley, 1964.

[14] Clyde H. Coombs, ‘Psychological scaling without a unit of measurement’, *Psychological Review*, **57**(3), 145–158, (1950).

[15] J.P. Doignon and J.C. Falmagne, ‘A polynomial time algorithm for unidimensional unfolding representations’, *Journal of Algorithms*, **16**(2), 218–233, (1994).

[16] Jon X. Eguia, ‘Foundations of spatial preferences’, *Journal of Mathematical Economics*, **47**(2), 200 – 205, (2011).

[17] Edith Elkind and Piotr Faliszewski, ‘Recognizing 1-Euclidean preferences: An alternative approach’, in *SAGT 2014*, volume 8768 of *Lecture Notes in Computer Science*, pp. 146–157. Springer, (2014).

[18] Edith Elkind, Martin Lackner, and Dominik Peters, ‘Preference restrictions in computational social choice: A survey’, *CoRR*, **abs/2205.09092**, (2022).

[19] Gábor Erdélyi, Martin Lackner, and Andreas Pfandler, ‘Computational aspects of nearly single-peaked electorates’, *J. of Art. Intelligence Research*, **58**, 297–337, (2017).

[20] Bruno Escoffier, Olivier Spanjaard, and Magdaléna Tydrichová, ‘Euclidean preferences in the plane under ℓ_1, ℓ_2 and ℓ_∞ norms’, *CoRR*, **abs/2202.03185**, (2022).

[21] William L. Hays and Joseph F. Bennett, ‘Multidimensional unfolding: Determining configuration from complete rank order preference data’, *Psychometrika*, **26**(2), 221–238, (Jun 1961).

[22] Dejan Jovanović and Leonardo de Moura, ‘Solving non-linear arithmetic’, in *Automated Reasoning*, eds., Bernhard Gramlich, Dale Miller, and Uli Sattler, pp. 339–354, Berlin, Heidelberg, (2012). Springer Berlin Heidelberg.

[23] Hidehiko Kamiya, Akimichi Takemura, and Hiroaki Terao, ‘Ranking patterns of unfolding models of codimension one’, *Advances in Applied Mathematics*, **47**(2), 379–400, (2011).

[24] Henry A Kautz, Ashish Sabharwal, and Bart Selman, ‘Incomplete algorithms.’, *Handbook of satisfiability*, **185**, 185–203, (2009).

[25] Vicki Knoblauch, ‘Recognizing one-dimensional Euclidean preference profiles’, *Journal of Mathematical Economics*, **46**, 1–5, (01 2010).

[26] J. B. Kruskal, ‘Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis’, *Psychometrika*, **29**(1), 1–27, (Mar 1964).

[27] Patrick Mair, Patrick JF Groenen, and Jan de Leeuw, ‘More on multidimensional scaling and unfolding in r: smacof version 2’, *Journal of Statistical Software*, **102**, 1–47, (2022).

[28] Nicholas Mattei and Toby Walsh, ‘Preflib: A library of preference data HTTP://PREFLIB.ORG’, in *Proceedings of the 3rd International Conference on Algorithmic Decision Theory (ADT 2013)*, Lecture Notes in Artificial Intelligence. Springer, (2013).

[29] Dominik Peters, ‘Recognising multidimensional Euclidean preferences’, in *Thirty-First AAAI Conference on Artificial Intelligence*, (2017).

[30] Dominik Peters and Martin Lackner, ‘Preferences single-peaked on a circle’, *Journal of Artificial Intelligence Research*, **68**, 463–502, (2020).

[31] Paul Rothstein, ‘Representative voter theorems’, *Public Choice*, **72**(2), 193–212, (1991).

[32] Xin Sui, Alex Francois-Nienaber, and Craig Boutilier, ‘Multi-dimensional single-peaked consistency and its approximations’, in *Twenty-Third International Joint Conference on Artificial Intelligence*, (2013).

[33] Michael A Trick, ‘Recognizing single-peaked preferences on a tree’, *Mathematical Social Sciences*, **17**(3), 329–334, (1989).

[34] Pavel Valtr, ‘Probability that n random points are in convex position’, *Discrete & Computational Geometry*, **13**(3), 637–643, (Jun 1995).