On Exact Computation of Tukey Depth Central Regions

Vít Fojtíka, Petra Laketa, Pavlo Mozharovskyi, and Stanislav Nagy

ABSTRACT
The Tukey (or halfspace) depth extends nonparametric methods toward multivariate data. The multivariate analogues of the quantities are the central regions of the Tukey depth, defined as sets of points in the $d$-dimensional space whose Tukey depth exceeds given thresholds $k$. We address the problem of fast and exact computation of those central regions. First, we analyze an efficient Algorithm (A) from Liu, Mosler, and Mozharovskyi, and prove that it yields exact results in dimension $d = 2$, or for a low threshold $k$ in arbitrary dimension. We provide examples where Algorithm (A) fails to recover the exact Tukey depth region for $d > 2$, and propose a modification that is guaranteed to be exact. We express the problem of computing the exact central region in its dual formulation, and use that viewpoint to demonstrate that further substantial improvements to our algorithm are unlikely. An efficient C++ implementation of our exact algorithm is freely available in the R package TukeyRegion.

1. Introduction: Tukey Depth and its Central Regions
The Tukey depth (or halfspace depth, or simply depth) is a prominent method of nonparametric analysis of multivariate data. Proposed by Tukey (1975), it is firmly established in nonparametric and robust statistics since the 1990s (Donoho and Gasko 1992). For a point $x \in \mathbb{R}^d$ and a dataset $X = \{x_1, \ldots, x_n\} \subseteq \mathbb{R}^d$, the Tukey depth $hD(x)$ of $x$ with respect to $X$ is defined as the minimum number of data points in any halfspace that contains $x$ on its boundary

$$hD(x; X) = \min_{u \in S^{d-1}} \# \{i \in \{1, \ldots, n\} : \langle x, u \rangle \leq \langle x_i, u \rangle\}.$$

Here, $S^{d-1}$ is the unit sphere in $\mathbb{R}^d$, and $u \in S^{d-1}$ is the inner normal of the halfspace $\{y \in \mathbb{R}^d : \langle x, u \rangle \leq \langle y, u \rangle\}. The depth assesses the degree of centrality of $x$ with respect to the geometry of the data cloud $X$. The higher the depth, the more “centrally positioned” $x$ is within $X$. While immensely successful in applications (Liu, Parelus, and Singh 1999; Rousseeuw and Ruts 1999; Zuo and Serfling 2000), the exact and fast computation of the depth for $d > 2$ has been resolved only relatively recently (Dyckerhoff and Mozharovskyi 2016).

Perhaps even more important than the depth of a single point $x$ are the central regions of the depth of $X$ at levels $k \geq 1$, defined as the upper level sets

$$hD_k(X) = \left\{ x \in \mathbb{R}^d : hD(x; X) \geq k \right\}.$$ 

The central regions form a system of nested compact convex polytopes. For $k = 1$ we obtain the convex hull of $X$. For $k = 2$ we get the intersection of halfspaces that contain all points from $X$ except for a single point, see Figure 1. For $k \geq 1$, this construction is sometimes called the $k$-hull of $X$ (Cole, Sharir, and Yap 1987). The smallest non-empty set $hD_1(X)$ is a generalization of the median set to $\mathbb{R}^d$-valued data, and is called the Tukey (or halfspace) median of $X$. In case when a unique point representing the median set is required, the barycenter of the median set is frequently singled out. The central regions of $X$ describe the shape of the dataset. Interestingly, they encode the complete information present in $X$, as there exist methods for reconstructing the data points from the central regions only (Struyf and Rousseeuw 1999; Laketa and Nagy 2021). The central regions are vital in many applications—they are used in (i) data visualization and exploratory data analysis (Liu, Parelus, and Singh 1999; Nagy and Dvořák 2021), (ii) construction of multivariate confidence regions (Yeh and Singh 1997), (iii) anomaly detection (Mozharovskyi 2022), (iv) construction of multivariate boxplots (Rousseeuw, Ruts, and Tukey 1999), (v) the construction of multivariate depth contours (Rousseeuw, Ruts, and Tukey 1999), to give a few examples.

If the dataset $X$ is in general position, each polytope $hD_{k+1}(X)$ lies in the interior of the previous region $hD_k(X)$, for $k \geq 1$ (Struyf and Rousseeuw 1999, Lemma 6). Throughout this article, we assume that the points of $X$ are in general position.
A motivating example: A dataset $X$ of $n = 8$ points $A–H$ in $\mathbb{R}^2$, the relevant lines forming boundaries of halfplanes from $\mathcal{H}(k)$ (thick lines), and the central regions (shaded regions) for $k = 1$ (top left), $k = 2$ (top right), $k = 3$ (bottom left), and $k = 4$ (bottom right). For $k = 1$ and $k = 3$ all ridges (that is, data points for $d = 2$) lie in a single orbit, and RidgeSearch is exact with any single initial ridge in $Q$. For $k = 2$ there are two orbits—starting from the ridge $A$, the only two relevant halfplanes from $\mathcal{H}(2)$ that contain $A$ in their boundary are those given by $[A, C]$ and $[A, G]$. The search initialized at $A$ in (S2) recovers its orbit $\{A, C, E, G\}$. A second orbit in $\mathcal{H}(2)$ is given by $\{B, D, F, H\}$. Using Algorithm (A) we initialize at a ridge (say) $A$ in (A1), then include in $Q$ also ridges $C$ and $G$ in (A2), and finally take into $Q$ also ridges $B$ and $H$ in (A3). Thus, Algorithm (A) gives an exact result. For $k = 4$ we have four orbits in $\mathcal{H}(4)$, given by $\{A, E\}$, $\{B, F\}$, $\{C, G\}$, and $\{D, H\}$, respectively. The median set $hD_k(X)$ is the single point (yellow diamond) in the center of the figure.

That assumption is standard in the depth literature. It greatly facilitates both the analysis and computation. If $X$ is sampled from a distribution with a density, it is in general position almost surely.

We are concerned with the exact computation of the regions $hD_k(X)$. Recently, an efficient algorithm was proposed in the literature (Liu, Mosler, and Mozharovskyi 2019, Algorithm 2). We call that program Algorithm (A) for brevity. Algorithm (A) is based on a more general computational scheme termed RidgeSearch in Section 2 below. It cleverly combines the ideas of projecting the data into two-dimensional subspaces, and a consecutive breadth-first search strategy along those hyperplanes determined by $d$ data points that may form a piece of the boundary of $hD_k(X)$. RidgeSearch starts with an initial set of data-determined hyperplanes, and spreads to all hyperplanes possibly supporting $hD_k(X)$ that are, in a sense, neighboring them. The output of the algorithm should be the complete set of hyperplanes that determine the boundary of $hD_k(X)$.

Algorithm (A) is fast, and possible to be also used for data of dimension $d > 2$. Despite not being proved theoretically, ample empirical evidence presented in Liu, Mosler, and Mozharovskyi (2019) suggested that the algorithm may give the exact Tukey depth regions $hD_k(X)$ for any $d \geq 1$ and $k \geq 1$.

We begin in Section 2 by analyzing the exactness of Algorithm (A) from a theoretical perspective. In Sections 2.1 and 2.2 we prove that Algorithm (A) does indeed give exact results in dimension $d = 2$ for any $k \geq 1$, and in any dimension $d > 2$ for $k = 1, 2$. In Section 2.3 we proceed with a surprising negative result. We provide a dataset of $n = 12$ points in $\mathbb{R}^3$ in general position where Algorithm (A) fails to recover the Tukey depth central region, meaning that Algorithm (A) is not exact in general. We argue that this failure occurs due to the initial set of hyperplanes in Algorithm (A) in RidgeSearch being too small.

Based on our observations, in Section 3 we modify Algorithm (A), and use a different initialization of RidgeSearch.
to devise a new Algorithm (B). We prove that Algorithm (B) recovers the exact central regions for any dataset in general position, for any dimension \( d \geq 1 \) and any \( k \geq 1 \). In Section 3.2 we compare the computational complexity of Algorithms (A) and (B) and show that they are comparable. An extensive simulation study presented in Section 4 and the supplementary material highlights that despite Algorithm (B) is more complex than Algorithm (A), in the task of computing multiple central regions, the two procedures are on par in terms of speed. In particular, Algorithm (B) is well suited for the exact computation of the complete collection of central regions of \( X \), including the Tukey median set.

In the concluding Section 5 we recast our results in view of the so-called dual graph of \( B \), useful for visualization and diagnostics for the central regions. Using dual graphs, we demonstrate that none of the several appealing simplifications of our Algorithm (B) can guarantee exactness. It therefore appears unlikely that a procedure substantially simpler than our Algorithm (B) (for simultaneous computation of all (lower-level) central regions), or the simple Algorithm (C) (the naïve version of RidgeSearch from Section 3.1 in Liu, Mosler, and Mozharovskyi 2019, detailed also in Section 2, for computing single central regions) would be able to recover the exact Tukey depth regions. The extensive technical proofs of our main results are gathered in the Appendix.

Our proofs employ notions from convex geometry, and rely heavily on the polarity theory for convex polytopes. After defining the essential notations in Section 1.1, we therefore begin our exposition by a brief overview of the necessary theory on polar polytopes in Section 1.2.

### 1.1. Notations

The boundary of a set \( A \subset \mathbb{R}^d \) is denoted by \( \partial A \), and its interior by \( \text{int}(A) = A \setminus (\partial A) \). We write \( \text{conv}(A) \) for the convex hull of \( A \), and \( \overline{A} \) for its affine hull\(^4\). In most situations this notation will be applied to a finite set \( A = \{a_1, \ldots, a_m\} \), where we write also \( \text{conv}(a_1, a_2) \) for the convex hull of these points, and \( \langle a_1, \ldots, a_m \rangle \) for their affine hull. For example, for \( a_1 \neq a_2 \in \mathbb{R}^d \), \( \text{conv}(a_1, a_2) \) and \( \langle a_1, a_2 \rangle \) stand for the line segment and the infinite line delimited by \( a_1 \) and \( a_2 \), respectively.

Denote by \( X \) a set of \( n > d + 1 \) data points in \( \mathbb{R}^d \) in general position. A ridge is any subset of \( d - 1 \) points from \( X \). We say that a hyperplane is observational if it is determined by the affine hull of \( d - 1 \) points from \( X \), meaning that it is the unique hyperplane that contains all those \( d \) points. A closed halfspace whose boundary is an observational hyperplane is called an observational halfspace. An observational halfspace \( H \) is relevant (at level \( k \)) if the complementary open halfspace \( \mathbb{R}^d \setminus H \) contains exactly \( k - 1 \) points from \( X \); we write \( \mathcal{H}(k) \) for the set of all relevant halfspaces at level \( k \). A relevant hyperplane (at level \( k \)) is the boundary hyperplane \( \partial H \) of a relevant halfspace \( H \in \mathcal{H}(k) \). We also say that a relevant halfspace \( H \in \mathcal{H}(k) \) (or its boundary \( \partial H \)) cuts off \( k - 1 \) points from \( X \).

Two halfspaces \( H, H' \in \mathcal{H}(k) \) are (mutually) reachable (in \( \mathcal{H}(k) \)) if (i) they are neighboring, meaning that their boundaries contain the same ridge, or (ii) there exists \( H'' \in \mathcal{H}(k) \) that is reachable in \( \mathcal{H}(k) \) from both \( H \) and \( H' \). Note that this definition is given recursively — \( H \) and \( H' \) are reachable if and only if there exists a finite sequence of halfspaces \( \{H_j\}_{j=1}^\infty \subseteq \mathcal{H}(k) \) such that \( H_1 = H, H_j = H' \), and for each \( j = 1, \ldots, J - 1 \) the boundaries of \( H_j \) and \( H_{j+1} \) share a ridge. Starting from a given ridge \( I \subset X \), the search strategy through ridges employed in our algorithms finds all relevant halfspaces \( H' \in \mathcal{H}(k) \) that are reachable from (any) halfspace \( H \in \mathcal{H}(k) \) such that \( I \subset \partial H \). The collection of all these halfspaces reachable from \( H \in \mathcal{H}(k) \) (or equivalently reachable from the ridge \( I \)) will be called an orbit of \( H \) (or \( I \)) in \( \mathcal{H}(k) \). Mathematically speaking, the orbit of \( H \in \mathcal{H}(k) \) is the transitive closure of the binary relation of halfspaces in \( \mathcal{H}(k) \) being neighboring to \( H \). The orbits in \( \mathcal{H}(k) \) partition \( \mathcal{H}(k) \) into equivalence classes.

### The Rationale of the Algorithms

Since \( hD_k(X) \) can be defined as the intersection of all the elements of \( \mathcal{H}(k) \) (Rousseeuw and Ruts 1999, Proposition 6), the problem of finding the central region of \( X \) at level \( k \) reduces to the task of identifying all relevant halfspaces \( \mathcal{H}(k) \). In the sequel, we are therefore concerned with algorithms for finding all halfspaces from \( \mathcal{H}(k) \), or equivalently, all \( d \)-tuples of points from \( X \) whose affine hulls cut off exactly \( k - 1 \) data points from \( X \).

### 1.2. Preliminaries: Polar Polytopes

We use duality considerations from convex geometry (Schneider 2014, sec. 2.4). First, we recall basic definitions and facts about polar polytopes. A polytope \( P \subset \mathbb{R}^d \) is the convex hull of a finite number of points in \( \mathbb{R}^d \). In this work we deal only with full-dimensional polytopes, that is polytopes whose interior is non-empty. A face of \( P \) is a convex subset \( F \subset P \) that satisfies that \( x, y \in P \) and \( (x + y)/2 \in F \) implies \( x, y \in F \). The single point faces \( F \) of \( P \) are called vertices of \( P \), the one-dimensional faces are the edges of \( P \). A \((d - 1)\)-dimensional face of \( P \) is a facet of \( P \). For a polytope \( P \subset \mathbb{R}^d \) that contains the origin in its interior, the polar polytope of \( P \) is defined as (Schneider 2014, sec. 2.1)

\[
P^* = \left\{ x \in \mathbb{R}^d : \langle x, y \rangle \leq 1 \text{ for all } y \in P \right\}.
\]

Denote by \( F_1, \ldots, F_m \) all the facets of \( P \). The conjugate face of \( F_j \), \( j = 1, \ldots, m \), is

\[
\hat{F}_j = \left\{ x \in P^* : \langle x, y \rangle = 1 \text{ for all } y \in F_j \right\}.
\]

By Schneider (2014, formula (2.28)) we know that each \( \hat{F}_j \) is a point in \( \mathbb{R}^d \), and Schneider (2014, Lemma 2.4.5) gives that \( P^* = \text{conv}(\hat{F}_1, \ldots, \hat{F}_m) \). Thus, the vertices \( \hat{F}_j \) of \( P^* \) correspond to the outer normals of the facets \( F_j, j = 1, \ldots, m \), in the sense that we can write

\[
P = \bigcap_{j=1}^m \left\{ x \in \mathbb{R}^d : \langle x, \hat{F}_j \rangle \leq 1 \right\}.
\]

---

\( ^4 \)Convex hull of \( A \) is defined as the intersection of all convex sets that contain \( A \); its affine hull is the intersection of all translations of vector subspaces (that is, affine subspaces of \( \mathbb{R}^d \)) that contain \( A \).
Main algorithm RidgeSearch: The search procedure through ridges in the boundaries of $H \in \mathcal{H}(k)$ from [LMM].

\begin{itemize}
  \item [(S1)] Initialization (generalization of Steps 1 and 2 from [LMM]):
    \begin{itemize}
      \item A queue $Q$ of ridges of $X$ is initialized, using an algorithm-specific rule.
      \item A set of found relevant halfspaces $\mathcal{H}_k$ is initialized to be empty.
    \end{itemize}

  \item [(S2)] The main loop runs through the queue of ridges $Q$ (Step 3 from [LMM]):
    \begin{itemize}
      \item For each $I \in Q$ do (Step 4 from [LMM])
        \begin{itemize}
          \item Find all relevant halfspaces reachable from $I$ in $\mathcal{H}(k)$, that is the orbit of $I$ in $\mathcal{H}(k)$. This is performed by a search strategy where first, all halfspaces in $\mathcal{H}(k)$ containing $I$ are found, and then this process is iterated for all ridges of those halfspaces. The search finishes when the complete orbit $O$ of the ridge $I$ in $\mathcal{H}(k)$ is found.
          \item Append the whole orbit $O$ of the ridge $I$ to the set $\mathcal{H}_k$.
        \end{itemize}
    \end{itemize}

  \item [(S3)] In the final Steps 5–8 from [LMM], the intersection of all halfspaces from $\mathcal{H}_k$ is found. This is the output of the algorithm, being an estimate of the central region $hD_k(X)$. It is hoped that in $\mathcal{H}_k$ we recovered all orbits, or equivalently the whole set $\mathcal{H}(k)$. In that case $hD_k(X) = \bigcap \mathcal{H}_k$.
\end{itemize}

A pair of vertices $\hat{F}_i$ and $\hat{F}_k$ of $P^d$ is joined by an edge on the boundary of $P^d$ if and only if the facets $F_i$ and $F_k$ share a $(d - 2)$-dimensional face $F$ of $P$ (Schneider 2014, Theorem 2.4.9 and formula (2.28)), and in particular $F$ is then the convex hull of some $d - 1$ vertices of $P$. In what follows, this theory will be applied in the situation when the vertices of $P$ are a subset of the dataset $X$, and both $F_i$ and $F_k$ determine boundaries of halfspaces from $\mathcal{H}(k)$. In that case, in terms of reachability introduced in Section 1.1, we see that $\hat{F}_i$ and $\hat{F}_k$ are joined by an edge if and only if the affine hulls of $F_i$ and $F_k$ are neighboring in $\mathcal{H}(k)$.

\section{Theoretical Analysis of Algorithm (A)}

For a dataset $X \subset \mathbb{R}^d$ and a level $k \geq 1$, Algorithm (A) for finding the central region $hD_k(X)$ is based on a general procedure, which we call RidgeSearch, given in the pseudo-code above. This scheme encompasses a whole family of algorithms in the spirit of both Algorithms 1 and 2 from Liu, Mosler, and Mozharovskyi (2019) (in the sequel, we refer to that paper by [LMM]). Steps 1–8 indicated in the pseudo-code above refer to the description of Algorithm 2 from [LMM].

The crucial part of RidgeSearch is the selection of the initial set of ridges $Q$ at stage (S1). This stage is the one where the particular instances of the procedure RidgeSearch differ.

Trivially, if all possible $\binom{d - 1}{d - 1}$ ridges of the dataset $X$ are included in $Q$, RidgeSearch yields an exact solution. This was observed already in [LMM], where the last algorithm was presented as Algorithm 1, and is also called the combinatorial algorithm. For brevity, we call this exact program Algorithm (C) (for “combinatorial”). It is currently the only relatively fast procedure for the computation of the central regions with an exactness guarantee. Nevertheless, as argued already in [LMM], the initial selection of all $\binom{d - 1}{d - 1}$ ridges in $Q$ makes Algorithm (C) slow for many setups.

In contrast, in the original fast Algorithm (A) from [LMM], the initial set of ridges $Q$ at stage (S1) is chosen according to the following heuristic (Step 2):

\begin{itemize}
  \item [(A1)] A single ridge $I$ of $d - 1$ points on the boundary of the convex hull of $X$ is found.
  \item [(A2)] All $^3$ relevant halfspaces $H \in \mathcal{H}(k)$ that contain $I$ in their boundary hyperplanes are obtained, and all ridges determined by points of $X$ in those hyperplanes are placed into $Q$.
  \item [(A3)] Finally, all ridges formed by $d - 2$ points of $I$ and a single point cut off by any halfspace $H \in \mathcal{H}(k)$ from step (A2) are added to $Q$.
\end{itemize}

Algorithm (A) therefore involves steps (S1)–(S3) of RidgeSearch, with the initialization (A1)–(A3) at step (S1). For a more detailed description of Algorithm (A) we refer to [LMM]. Here we provide only a small motivating example for $d = 2$ that is summarized in Figure 1. In this example we see that for any $k \geq 1$, Algorithm (A) gives an exact central region. We will show below that this is not a coincidence, and for $d = 2$ Algorithm (A) is always exact. Another example of Algorithm (A) will be given for $d = 3$ in Section 2.3.

In [LMM, Section 4.1], an extensive simulation study was performed to demonstrate that in tens of thousands of simulated runs, Algorithm (A) always recovered the exact central region. We validate some of those positive results from a theoretical perspective. Afterwards, we construct an example showing that Algorithm (A) may fail for $d > 2$.

\subsection{Algorithm (A) is Exact for $d = 2$}

Assume for a moment that $d = 2$ and $1 \leq k < n/2$.\footnote{At this step we slightly simplify Algorithm 2 from [LMM]. In the original version, only two relevant halfspaces $H \in \mathcal{H}(k)$ of this type are found in Step 2(d) [LMM, p. 686]. Of course, our inclusion of (possibly) more than two relevant halfspaces in (A2) makes Algorithm (A) to search through more ridges. Thus, if Algorithm 2 from [LMM] is exact, then so must be our Algorithm (A). This difference is of no importance for our exposition, and does not alter any of our conclusions.} Recall that for $x \neq y \in \mathbb{R}^2$ we write $[x, y]$ for the unique line passing through $x$ and $y$. The extreme cases $k \geq n/2$ are not interesting, because clearly $hD_k(X) = \emptyset$ if $k > n/2$ (see e.g., Liu, Luo, and Zuo 2020, Theorem 1). Furthermore, for $n$ even and $k = n/2$, if the set $hD_{n/2}(X)$ is non-empty, then $X$ is a halfspace
Theorem 1. Let \( H \in \mathcal{H}(k) \) be a relevant halfplane cutting off points \( U \subseteq X \) from \( X \). Then for all orbits \( O \) of halfplanes from \( \mathcal{H}(k) \), either \( H \in O \), or there exists \( H' \in O \) and \( x_i \in U \) such that \( L' = \partial H' \) passes through \( x_i \). That is, every orbit in \( \mathbb{R}^2 \) contains \( H \), or a halfplane whose boundary passes through a point cut off by \( H \).

The detailed proof of Theorem 1 is given in Section A.1 in the Appendix. As a direct consequence, we obtain our first main result.

Theorem 2. Algorithm (A) in dimension \( d = 2 \) finds all relevant hyperplanes at level \( 1 \leq k < n/2 \) in any dataset \( X \) in general position of size \( n \). In other words, with the set of initial ridges chosen using the heuristic (A1)–(A3), we have \( \mathcal{H}_k = \mathcal{H}(k) \). In particular, Algorithm (A) is exact for \( d = 2 \).

Proof. Algorithm (A) starts with at least one relevant hyperplane (line) \( L \) in (A3) and all relevant hyperplanes that pass through points that are cut off by \( L \) in (A3). Then, it generates the orbits of these hyperplanes. By Theorem 1, this includes all orbits in \( \mathcal{H}(k) \).

As a consequence of the proof of Theorem 2 we can reduce the initial set of ridges \( Q \) in Algorithm (A) to only \( k \) points (ridges) for \( d = 2 \), without losing exactness. For any relevant halfplane \( H \in \mathcal{H}(k) \), take the \( k - 1 \) points cut off by \( H \) and one of the two points on the boundary of \( H \) as the initial set of ridges. Then, Theorem 1 shows that the procedure \texttt{RidgeSearch} finds all relevant halfspaces and hence gives the exact central region \( hD_k(X) \).

2.2. Algorithm (A) is Exact for \( k = 1, 2 \) for any \( d \)

Denote by \( C \) the convex hull of \( X \). In our first lemma we deal with the simple case of \( k = 1 \). In that situation \( hD_1(X) = C \), and all relevant halfspaces in \( \mathcal{H}(1) \) form a single orbit as we saw also in Figure 1. In particular, Algorithm (A) is exact for \( k = 1 \) with any single initial ridge in the queue \( Q \).

Lemma 1. Any two halfspaces in \( \mathcal{H}(1) \) determined by facets of \( C \) are mutually reachable in \( \mathcal{H}(1) \).

The proof of Lemma 1 is given in Section A.2 in the Appendix. We now turn our attention to the more interesting case \( k = 2 \). In a series of auxiliary lemmas stated and proved in Section A.3 in the Appendix, we obtain a proof of exactness of Algorithm (A) for \( k = 2 \).

Theorem 3. Algorithm (A) finds all relevant halfspaces for \( k = 1, 2 \) and any dimension \( d = 1, 2, \ldots \). Consequently, Algorithm (A) gives an exact solution for \( k = 1, 2 \).

Again as a consequence of our proof of Theorem 3 we are able to reduce the initial set of ridges \( Q \) in Algorithm (A) for \( k = 2 \) to \( d \), still keeping the algorithm exact. Indeed, take a single facet of \( C \) and consider any collection of \( d \) halfspaces from \( \mathcal{H}(2) \), each of these halfspaces cutting off one of the \( d \) vertices of \( C \). The set of all ridges in the boundaries of these halfspaces is placed in the initial queue \( Q \). Then the proof of Theorem 3 guarantees that for \( k = 2 \) the procedure \texttt{RidgeSearch} gives the exact central region \( hD_2(X) \).

2.3. Algorithm (A) is not Exact in General

The output of Algorithm (A) is a collection of relevant halfspaces \( \mathcal{H}_k \subseteq \mathcal{H}(k) \) at level \( k \). Their intersection is therefore always a superset of the corresponding central region \( hD_k(X) \). In the following example we demonstrate that for \( k > 2 \) and \( d > 2 \), Algorithm (A) does not always recover the central region exactly.

Example. Consider a dataset \( X \) of \( n = 12 \) points in \( \mathbb{R}^3 \) in general position. Each of these points is labeled by a color: red, blue or green. We start with red points positioned in the vertices of a regular tetrahedron

\[
\begin{align*}
  r_1 &= \frac{1}{\sqrt{2}} (e_1 - \frac{1}{2}, 1, 1, 1) \quad \text{for } i = 1, 2, 3, \quad \text{and} \\
  r_4 &= \frac{1}{\sqrt{8}} (1, 1, 1, 1),
\end{align*}
\]

for \( e_1 = (1, 0, 0), e_2 = (0, 1, 0) \) and \( e_3 = (0, 0, 1) \). The blue and the green points are respectively placed at \( b_i = -0.3 \, r_i \) and \( g_i = 0.15 \, r_i \) for \( i = 1, \ldots, 4 \). To satisfy the condition of the points being in general position, we rotate a bit the vertices of the blue and the green tetrahedrons, each in a slightly different way. The data is constructed so that the convex hull of \( X \) is formed only by the four red points, and so that the green vertices lie outside the blue tetrahedron. An example of such a configuration of points is in Figure 2 and in the supplementary Mathematica notebook, where our whole construction is visualized in an interactive display.

Consider the depth level \( k = 3 \). Algorithm (A) begins in step (A1) with a ridge \( E \), that is, an edge of the convex hull of \( X \), which must consist of two red points. The algorithm then proceeds in step (A2) by finding two relevant planes \( P_1 \) and \( P_2 \) that contain \( E \). Because of how the points of \( X \) are positioned, each of these planes is determined by the two red points incident to \( E \), and an additional single green point, see also the left hand panel of Figure 2. Each of these planes cuts off one red and one blue point from \( X \). In the initial step we add into the queue \( Q \) all the ridges obtained by substituting one of the points from the initial ridge \( E \) with one point cut off by either \( P_1 \) or \( P_2 \) from \( X \) in step (A3). This gives us the initial set of ridges \( Q \) whose incident points are of the following colors: R-R, R-B and R-G, where R,G and B represent red, green and blue.

Note that any plane determined by three blue points is relevant for \( k = 3 \) — it cuts off exactly two points from \( X \), one
red and one green. At the same time, no other plane determined by two blue points and one point of another color is relevant for \( k = 3 \). This is very easy to see in the interactive Mathematica visualization provided in the supplementary material. Therefore, from our initial set of ridges \( Q \) it is impossible to obtain any ridge colored as B-B, meaning that Algorithm (A) fails to find any relevant plane determined by three blue points. For that reason, the resulting depth region for \( k = 3 \) obtained by Algorithm (A) contains all the green points \( g_i, i = 1, \ldots, 4 \), which certainly have Tukey depth only \( hD(g_i; X) = 2 < k \). The last claim is seen by considering a halfspace passing through a green point whose boundary plane is parallel to the adjacent blue facet. That halfspace contains only a single red point in its interior. As we intended to show, Algorithm (A) fails to recover the exact central region \( hD_3(X) \).

3. Algorithm (B): Exact Computation of Central Regions

3.1. Description of the Algorithm

We now turn to the problem of exact computation of the central regions by means of the ridge-wise search strategy \textbf{RidgeSearch}. Our intention is to find an initial set of ridges \( Q \) guaranteeing the exactness of the procedure. As demonstrated in our example from Section 2.3 and corroborated in Section 5, it turns out that in the task of computing \( hD_k(X) \) directly, it is unlikely that an initial set of much less than all \( \binom{n}{d-1} \) ridges (as for Algorithm (C)) suffices for an exact result.

We approach the problem in a different way, and argue that \textbf{RidgeSearch} is feasible to be run recursively. We show that given the set of all relevant halfspaces \( \mathcal{H}(k-1) \), to obtain all relevant halfspaces \( \mathcal{H}(k) \) using \textbf{RidgeSearch} it is enough to initialize the queue \( Q \) in \((S_1)\) in the following way (see also third bullet point at the end of Section 2 in [LMM]):

\((B_1)\) All ridges of points in the boundaries of relevant halfspaces from \( \mathcal{H}(k-1) \) are placed into \( Q \).

The complete Algorithm (B) involves running steps \((S_1)-(S_3)\) with the queue \( Q \) in step \((S_1)\) chosen using \((B_1)\); for a summary of our three procedures see Table 1.

The initialization \((B_1)\) typically results in a larger set of initial ridges \( Q \) than what is considered in \((A_1)-(A_3)\). In Section 5 and the examples in the supplementary material we however argue that this appears to be needed to guarantee exactness. On the other hand, the set of ridges from \((B_1)\) has usually much less elements than \( \binom{n}{d-1} \) needed for Algorithm (C). It turns out that numerically, running our algorithm with initialization \((B_1)\) several times to compute all central regions at levels \( k = 1, \ldots, K \) for \( K \geq 1 \) given is surprisingly not slower than running the fast (and not exact) Algorithm (A) to compute \( hD_k(X) \) for all \( k = 1, \ldots, K \). All this will be demonstrated in numerical studies in Section 4.

The following theorem is the main ingredient of our exact Algorithm (B) for the computation of the central regions. We describe, given the set of all relevant halfspaces \( \mathcal{H}(k) \) at level \( k \geq 2 \), a way to find all relevant halfspaces \( \mathcal{H}(k+1) \) at level \( k+1 \). We argue that any \( H \in \mathcal{H}(k+1) \) can be reached from a halfspace \( H' \in \mathcal{H}(k+1) \) that shares a common ridge with a relevant halfspace \( \tilde{H} \) from \( \mathcal{H}(k) \). Schematically, we obtain

\[
H \in \mathcal{H}(k+1) \iff \text{reachable} \iff H' \in \mathcal{H}(k+1) \iff \text{neighboring} \iff \tilde{H} \in \mathcal{H}(k).
\]

A detailed proof of Theorem 4 is found in Section A.4 in the Appendix.

**Theorem 4.** Let \( H \in \mathcal{H}(k+1) \). Then there exists (i) a halfspace \( H' \in \mathcal{H}(k+1) \) that is reachable from \( H \) and (ii) a halfspace \( \tilde{H} \in \mathcal{H}(k) \) that shares a ridge in the boundary with \( H' \). In particular, Algorithm (B) defined by \textbf{RidgeSearch} with the initialization \((B_1)\) is exact.
3.2. Computational Complexity

The computational complexity of RidgeSearch amounts to: (C1) the main loop over all ridges of relevant halfspaces from \( H_k \) in Step (S2), (C2) the cost of propagation to neighboring ridges, and finally (C3) computing the intersection of the found relevant halfspaces \( H_k \) in Step (S3). Since Step (C3) is common and unavoidable for all procedures based on RidgeSearch, we follow [LMM] and compare only the complexities of recovering \( H_k \) in Steps (C1) and (C2).

The complexity of the first task (C1) is determined by the maximum number of ridges in \( H(k) \). That is strongly related to the famous \( k \)-set problem from discrete geometry (Matoušek 2002, chap. 11), and the problem of determining the maximum number \( a_{k,d}(n) \) of (ridges in) the \((k-1)\)-facets formed by \( n \) data points in \( \mathbb{R}^d \), see Wagner (2003, sec. 2). In the terminology from Wagner (2003), a \((k-1)\)-facet of \( X \) is precisely what we call a relevant hyperplane at level \( k \). As \( k, n \to \infty \), the best known simple bounds are \( a_{k,2}(n) = \Theta(n^{k/2}) \), \( a_{k,3}(n) = \Theta(n^{k/2}) \), and \( a_{k,d}(n) = \Theta(n^{d/2}k^{d/2}) \) in general dimension \( d \geq 2 \) (Wagner 2003); for \( k \) proportional to \( n \) we have the trivial bound \( a_{n,d}(n) = \Theta\left(\binom{n}{d-1}\right) = \Theta(n^d) \). For each found ridge in \( H_k \), one propagates to neighboring ridges (in Step (C2)) with complexity \( \Theta(n \log(n)) \) [LMM, Section 3.1].

Algorithm (A) traverses through all ridges of \((k-1)\)-facets, which costs \( \Theta(a_{k,d}(n) \cdot n \log(n)) \). In Algorithm (B), we use all \( \ell \)-facets with \( \ell \leq k-1 \). The latter set of all \( \ell \)-facets is of size \( \Theta(n^{\ell/2}k^{\ell/2}) \) (Clarkson and Shor 1989, Corollary 3.3), which is of the same order as the best bound on \( a_{k,d}(n) \) with general \( d \) above. Our best bounds on complexities of Algorithm (A) are thus \( \Theta(n^2 \log(n)) \) for \( d = 2 \), \( \Theta(n^2 \log(n) \min(k^{1/2}, n)) \) for \( d = 3 \), and \( \Theta(n^{d/2}k^{d/2}) \log(n)) \) for \( d > 3 \). For Algorithm (B) we get the same result for \( d = 2 \) and \( d = 3 \), and for \( d \geq 3 \) the slightly different \( \Theta(n^2 \log(n) \min(k^{1/2}, n)) \). It is important to note that our bounds are not tight for two reasons: 1. the nontrivial results on \( a_{k,d}(n) \) above are for the maximum number of \((k-1)\)-facets; the maximum number of ridges in \((k-1)\)-facets might be smaller (different \((k-1)\)-facets can share ridges), and 2. it is still unknown whether the bounds on \( a_{k,d}(n) \) given above are tight. As a rough upper bound for both Algorithms (A) and (B) we can also always use \( \Theta(n^2 \log(n)) \), which is exactly the complexity of Algorithm (C) [LMM, Section 3.1]. For \( d = 2 \), this yields the same complexity as \( \text{isodepth} \) discussed in Section 4.1, namely \( \Theta(n^2 \log(n)) \) (Ruts and Rousseeuw 1996, p. 158).

### 4. Empirical Comparison: Numerical Studies

#### 4.1. Algorithm (A) and isodepth for \( d = 2 \)

Having an exactness guarantee for Algorithm (A) in dimension \( d = 2 \), in our first simulation exercise we compare this procedure with the exact implementation of the algorithm called \( \text{isodepth} \) for the computation of central regions for \( d = 2 \) from the R package \( \text{depth} \) (Genest, Mássé, and Plante 2019). The latter procedure was originally designed in Ruts and Rousseeuw (1996), and later revised in Rousseeuw and Ruts (1998) and Rousseeuw, Ruts, and Tukey (1999). It is based on the idea of a circular sequence applied to the angles between pairs of data points and a constant direction. This method is applicable only in dimension \( d = 2 \). The efficient Fortran implementation of \( \text{isodepth} \) available in the package \( \text{depth} \) is matched to our C++ implementation of Algorithm (A) in the package \( \text{TukeyRegion} \). For different values of sample sizes \( n \in \{100, 250, 500, 1000, 2500\} \) we generate random samples from the standard bivariate normal distribution and compute their central regions. The regions considered are first at single levels \( k \in \{\lfloor n/10 \rfloor, \lfloor n/5 \rfloor, \lfloor n/3 \rfloor\} \) and then also at all levels \( k \) at once. In the latter case, we simulate also the task of finding the Tukey median, the smallest nonempty central region. We ran 100 independent replications of this setup. The comparison was conducted on a machine having processor Intel(R) Core(TM) i7-4980HQ (2.8 GHz) with 16 GB of physical memory and macOS Monterey (Version 12.4) operating system. The resulting execution times (in seconds) are presented in Table 2. Since Algorithm (A) is a particular case that implements a general framework for any dimension \( d \), it is slightly outperformed by \( \text{isodepth} \), designed only for \( d = 2 \), for smaller sample sizes \( n \). On the other hand, Algorithm (A) gains the upper hand over \( \text{isodepth} \) already for \( n \geq 250 \) and with growing sample size becomes even more advantageous.

#### 4.2. Algorithm (B) and its Competitors for \( d > 2 \)

We have implemented the exact Algorithm (B) in the new version 0.1.6.3 of the R package \( \text{TukeyRegion} \) (Barber and Mozharovskyi 2023). We compare three algorithms for the computation of the central regions, each based on RidgeSearch: (i) the nonexact fast Algorithm (A) with the initialization \( \{A_1\} - \{A_3\} \); (ii) our new exact Algorithm (B); and (iii) the

| \( k \) | \( n \) | \( \text{TukeyRegion (A)} \) | \( \text{isodepth} \) |
|---|---|---|---|
| \( n = 100 \) | \( \lfloor n/10 \rfloor \) | 0.00219 (0.00010) | 0.00154 (0.000302) |
| \( n = 250 \) | \( \lfloor n/10 \rfloor \) | 0.00491 (0.000935) | 0.00945 (0.000607) |
| \( n = 500 \) | \( \lfloor n/10 \rfloor \) | 0.0133 (0.00115) | 0.0445 (0.00276) |
| \( n = 1000 \) | \( \lfloor n/10 \rfloor \) | 0.0437 (0.00234) | 0.237 (0.0116) |
| \( n = 2500 \) | \( \lfloor n/10 \rfloor \) | 0.271 (0.0225) | 2.83 (0.0552) |
| \( n = 100 \) | \( \lfloor n/5 \rfloor \) | 0.00277 (0.000566) | 0.00205 (0.000261) |
| \( n = 250 \) | \( \lfloor n/5 \rfloor \) | 0.00294 (0.000490) | 0.00249 (0.000437) |
| \( n = 500 \) | \( \lfloor n/5 \rfloor \) | 0.00647 (0.00451) | 0.0129 (0.00118) |
| \( n = 1000 \) | \( \lfloor n/5 \rfloor \) | 0.0191 (0.000900) | 0.0610 (0.00485) |
| \( n = 2500 \) | \( \lfloor n/5 \rfloor \) | 0.0240 (0.00124) | 0.0817 (0.00088) |
| \( n = 100 \) | \( \lfloor n/3 \rfloor \) | 0.0873 (0.00496) | 0.416 (0.0335) |
| \( n = 250 \) | \( \lfloor n/3 \rfloor \) | 0.355 (0.236) | 257 (66.6) |
| \( n = 500 \) | \( \lfloor n/3 \rfloor \) | 0.614 (0.0195) | 4.18 (0.253) |
| \( n = 1000 \) | \( \lfloor n/3 \rfloor \) | 3.52 (2.55) | > 1 hr |

Table 2. Means and standard deviations (in brackets) of execution times for algorithms (A) and isodepth when calculating depth contour sets for a bivariate normal sample (in seconds, over 100 random samples).
exact combinatorial Algorithm (C) based on plugging all \( \binom{n}{d-1} \) ridges of \( X \) into the initial queue \( Q \).

For several combinations of \( n \) and \( d \), we have drawn 100 independent samples \( X \) from three \( d \)-variate distributions (normal, multi-modal and uniform on simplex) of size \( n \). Since such samples \( X \) are in general position almost surely, the maximum Tukey depth of a full-dimensional central region is bounded from above by \( \lfloor (n-d+1)/2 \rfloor \), see Liu, Luo, and Zuo (2020, Theorem 1). For each \( k \in \{1, \ldots, \lfloor (n-d+1)/2 \rfloor \} \) we computed the first \( k \) central regions \( hD_k(X) \), \( \ell = 1, \ldots, k \), using Algorithms (A)–(C). We kept a record of (i) the number of found relevant halfspaces \( hD_k \) at level \( k \); (ii) the number of ridges visited by the algorithm; and (iii) the total execution time for computing the first \( k \) central regions of the data. The full study was run for the following combinations of \( (n, d) \): \((25, 3), (25, 4), (25, 5), (50, 3), (50, 4), (50, 5), (75, 3), (75, 4), (100, 3), (100, 4), (125, 3), (150, 3), (175, 3), (200, 3), (225, 3), \) and \((250, 3)\). With a smaller number of independent runs, we have tested the algorithms also with larger values of \( n \) up to \( n = 5000 \) for \( d = 3 \) and \( n = 1000 \) for \( d = 4 \); the results are quite analogous to those presented below. The reader is referred to Section S.5 of the supplementary material for a detailed description of the setting.

First, we evaluated the exactness of Algorithm (A). In the complete numerical study, we generated \( 16 \times 3 \times 100 = 4800 \) samples, and calculated a total of \( (11 + 11 + 10 + 24 + 23 + 36 + 36 + 49 + 48 + 61 + 74 + 86 + 99 + 111 + 124) \times 3 \times 100 = 247800 \) central regions. Out of these results, for 66 regions \((\approx 0.027 \% \text{ or } 1 \text{ out of } 3750 \text{ cases})\) Algorithm (A) failed to detect all relevant halfspaces (each time a single simplex of ridges containing \( d \) relevant halfspaces was missed). Resulting from a much more extensive simulation study, this result does not contradict the empirical evidence from [LMM]. For random samples from well-behaved distributions, Algorithm (A) is quite likely to give exact results; nevertheless, in general it is non-exact and the miscalculation event does occur. For detailed results the reader is referred to Section S.5.2 of the supplementary material.

Below, we discuss the detailed results of the numerical study for normal distribution when \( d = 4 \) and \( n = 100 \). The results are summarized in Figure 3, which consists of three parts: (i) In the top panel we see the boxplots of the proportion of ridges visited by Algorithm (B), compared to the total number of \( \binom{n}{d-1} \) ridges of \( X \) used by Algorithm (C). We see that this fraction is, especially for lower to moderate depth levels \( k \), relatively small. (ii) In the middle panel there are the boxplots of ratios of execution times of Algorithms (B) and (C). This figure resembles the one from the top panel. Even for the complete set of central regions (that is, all regions up to \( k = \lfloor (n-d+1)/2 \rfloor \) = 48), the total computation time for Algorithm (B) typically does not exceed 70 \% of the time used by Algorithm (A). For computation of lower regions the spared computation power is substantial. (iii) Finally, in the bottom panel we see an analogous display with a fraction of computation time, this time Algorithm (B) compared to the non-exact Algorithm (A). The first two boxplots \((k = 1, 2)\) may be disregarded as in that case Algorithm (A) is exact. In all other boxplots, we see that in addition to having a guarantee of exactness for Algorithm (B), our computation in Algorithm (B) does not increase the execution times of Algorithm (A). All these results are quite favorable. In the task of computing all the first \( k \) central regions of \( X \), the new Algorithm (B) is more efficient than the combinatorial Algorithm (C), and also not slower than the fast Algorithm (A). The final results for other distributions and combinations of \( n \) and \( d \) are quite similar, and given without additional commentary in the supplementary material. To get a rough idea about the raw computation times of the three algorithms, in the supplementary material we include tables with average execution times of the three considered algorithms, matching the simulation study presented in [LMM, Tables 2 and 3].

In our numerical study we computed all central regions at levels \( 1, \ldots, k \) at the same time. It is, however, important to mention that in contrast to the combinatorial Algorithm (C), Algorithm (B) is inherently recursive when initializing the queue \( Q \) in \((B_1)\). Therefore, Algorithm (B) is typically slower than the direct combinatorial Algorithm (C) if the computation of a single central region is of interest. We therefore conclude that for exact computation of Tukey depth central regions, Algorithm (B) is the fastest if all regions \( hD_k(X) \) are to be computed for \( k = 1, \ldots, K \), or if a region at a lower level \( k \) is to be found. Algorithm (C) is to be preferred if a single region \( hD_k(X) \) for a higher value \( k \) is searched for; in particular, if only the Tukey median set is to be found.

5. Dual graph: (Non-)Exactness of Algorithm (A) and Negative Results

We saw that Algorithm (A) does not always find the central region. On the other hand, the exact Algorithm (B) proposed in Section 3 is typically slower than Algorithm (A) if a single central region \( hD_k(X) \) is to be evaluated. In the present section we first explore the negative example of Section 2.3 in view of polarity considerations. We present a different vantage point on the search for relevant halfspaces based on the duality theory. It is shown that in the so-called dual graph of the dataset \( X \), the search for an exact algorithm manifests itself as a natural problem in the theory of graphs. This analysis serves us to show that multiple promising simplifications of our Algorithm (B) along the lines of Algorithm (A) fail to recover the exact Tukey depth central region.

5.1. Dual Graph: Definition

We have seen in Section 1.2 that any polytope \( P \) in \( \mathbb{R}^d \) whose interior contains the origin can be represented in its dual form as a polytope \( P^o \). The facets \( F_j \) of \( P \) correspond to the vertices \( \hat{F}_j \) of \( P^o \), and facets \( F_j \) and \( F_k \) of \( P \) are mutually neighboring if and only if the vertices \( \hat{F}_j \) and \( \hat{F}_k \) share an edge on \( P^o \). Instead of working with polytopes, we now generalize the polarity paradigm directly toward a dataset \( X \). To visualize our problem of finding all relevant halfspaces at a given level \( k = 1, 2, \ldots \) of \( X \), we introduce the dual graph of \( X \), and show how the search strategy employed in \textit{RidgeSearch} translates into a problem of graph connectivity in the dual space.

Any \( d \) distinct points \( a_1, \ldots, a_d \) from \( X \) uniquely determine a hyperplane \( [a_1, \ldots, a_d] \). We suppose that \( X \) is such that none of these \( \binom{n}{d} \) hyperplanes passes through the origin; in the other
Figure 3. Results of the numerical study, setup: normal distribution, $n = 100, d = 4$. For a detailed description of the results see Section 4.2.

The dataset $X$ is shifted. The polar to $[a_1, \ldots, a_d]$ is defined as

$$
[a_1, \ldots, a_d]^{\circ} = \{ x \in \mathbb{R}^d : \langle x, y \rangle = 1 \text{ for all } y \in [a_1, \ldots, a_d] \}.
$$

This definition is analogous to that of a conjugate face from (1). Indeed, for a face $F = \text{conv}(a_1, \ldots, a_d)$ we have

$$
\tilde{F} = [a_1, \ldots, a_d]^{\circ}.
$$

In particular, each polar to an observational hyperplane is a single point in the dual space. The polar from (3) is easy to express analytically. Writing $u \in S^{d-1}$ for a unit...
normal vector of $[a_1, \ldots, a_d]$, we have

$$[a_1, \ldots, a_d] = \left\{ \frac{u}{(a_1, u)} \right\}. \quad \text{(4)}$$

Note that by our assumption that $[a_1, \ldots, a_d]$ does not contain the origin, the single point set above is well defined as $(a_1, u)$ is the distance of the hyperplane $[a_1, \ldots, a_d]$ from the origin. Analogously as in Section 1.2, for two different observational hyperplanes $[a_1, \ldots, a_d]$ and $[b_1, \ldots, b_d]$ we join the pair of vertices $[a_1, \ldots, a_d]$ and $[b_1, \ldots, b_d]$ in the dual space by an edge if and only if the two hyperplanes are (mutually) neighboring, meaning that their defining sets of data points share a common subset of exactly $d - 1$ elements (a ridge)

$$\# \left( \langle a_1, \ldots, a_d \rangle \cap \{b_1, \ldots, b_d\} \right) = d - 1.$$  

This definition is an extension of the notion of neighboring facets of a polytope from Section 2.2 with $k = 2$. Finally, each vertex (4) in the dual space corresponding to a hyperplane $[a_1, \ldots, a_d]$ is assigned a weight $k$ being the smaller number of points from $X$ that are cut off by $[a_1, \ldots, a_d]$ plus one, that is

$$k = 1 + \min \left\{ \left\{ X \cap \{ x \in \mathbb{R}^d : \frac{\langle x, u \rangle}{(a_1, u)} > 1 \} \right\}, \left\{ X \cap \{ x \in \mathbb{R}^d : \frac{\langle x, u \rangle}{(a_1, u)} < 1 \} \right\} \right\}.$$  

This weight corresponds to the level $k$ at which $[a_1, \ldots, a_d]$ forms a boundary of some $H \in \mathcal{H}(k)$. Altogether, in the dual space we obtain a graph (i) whose $G$ vertices correspond to observational hyperplanes, and (ii) each such vertex is given a weight corresponding to the level $k$ at which it contributes to $hD_k(X)$. (iii) Two vertices are joined by an edge if and only if the corresponding hyperplanes share a ridge. We call this the dual graph of $X$.

For the following analysis of $G$ we involve tools from the theory of graphs. An induced subgraph $G'$ of $G$ is a graph formed by a subset $V'$ of the set of vertices $V$ of $G$, and all the edges of $G$ that join pairs of points from $V'$. An induced subgraph $G'$ of $G$ is called a clique if each pair of vertices of $G'$ is connected by an edge. It is a maximal clique if no other vertex of $G$ can be appended to $G'$ so that the resulting induced subgraph is a clique.

The dual graph $G$ describes all the relevant combinatorial structure of our arrangement of points $X$—the observational hyperplanes, the fact whether two such hyperplanes are neighboring, and also the fact whether they share a ridge. Indeed, to identify common ridges, note that two observational hyperplanes share a ridge $I$ if and only if their dual vertices are connected in $G$ by an edge. Any given ridge $I$ is shared by $n - (d - 1)$ hyperplanes, meaning that the corresponding $n - (d - 1)$ vertices of $G$ form a clique in $G$. Since no other observational hyperplane contains $I$, that clique is maximal. Consequentially, each ridge $I$ is equivalent with a maximal clique of $G$, and each such maximal clique has $n - d + 1$ vertices.

5.2. Algorithm (A²): The Counterexample Revisited

We illustrate the relevance of the dual graph by returning to our example from Section 2.3. The dual graph of the set of $n = 12$ points $X$ in $\mathbb{R}^3$ has $\binom{12}{3} = 220$ vertices, each connected with exactly $d \cdot (n - d) = 27$ other vertices. The weights of the vertices range from $k = 1$ for those corresponding to hyperplanes forming the boundary of conv $(X)$, to $k = \lfloor (n - d)/2 \rfloor + 1 = 5$. The latter bound follows from for example the ham sandwich theorem (Elton and Hill 2011; Matoušek 2003).

Its complexity makes direct visualization of $G$ cumbersome; the complete dual graph of $X$ can be found in the supplementary material. It turns out that it is more insightful to restrict to the induced monochrome (that is single-weight) subgraphs of $G$. Those are displayed in Figure 4, where we can see the subgraphs $G_1$, $G_2$, and $G_3$ of hyperplanes that cut off 0, 1, or 2 points from $X$, respectively. The vertices of the graph $G_3$ represent all the elements of $\mathcal{H}(k)$. We identify two connected components of $G_3$ corresponding to hyperplanes from $\mathcal{H}(3)$. The smaller connected component relates to the four faces of the blue tetrahedron from Figure 2. We have seen in Section 2.3 that these relevant hyperplanes are never reached using Algorithm (A). Now we argue that this difficulty is fundamental, and even a substantially expanded procedure RidgeSearch in the spirit of Algorithm (A) still fails to find all halfspaces from $\mathcal{H}(3)$. Consider the following variation of Algorithm (A) where, to search for all elements of $\mathcal{H}(k)$ (that is, all vertices of $G_k$), we make the following amendments:

(A²) In (A₁), only a single ridge $I$ of points on the convex hull of $X$ is considered. Instead, we take all ridges of all facets of conv $(X)$ in the initial step.

(A²) In Algorithm (A), only two hyperplanes relevant at level $k$ that share a given ridge $I$ are pushed into the queue $Q$ in Step 2(d) in [LM, Tables 2 and 3]. We push all hyperplanes relevant at level $k$ that share $I$ into the search queue $Q$, as we did in (A₂).

(A²) Finally, in (A₃), after $H \in \mathcal{H}(k)$ that contains a given ridge $I$ of the facet of conv $(X)$ is found, each point of $X$ that was cut off by $H$ from $X$ is combined with points of the ridge $I$ to form new ridges pushed to $Q$. In our setting, we expand the last set of ridges considerably. We add to $Q$ all the ridges of all hyperplanes that are relevant at all levels $\ell = 2, 3, \ldots, k$ and share at least a single ridge with any facet of conv $(X)$.

The steps (A²)–(A₃) can be summarized more succinctly in dual terms. To search for the central region at level $k$, all ridges of

- all vertices of $G_1$, and
- all vertices of $G_2, \ldots, G_k$ that share an edge with any vertex from $G_1$,

are added to $Q$ in the initial step (S₁) of RidgeSearch. Our expanded program then proceeds by (S₂) and (S₃). In plain words, for the graph $G$ it means that instead of through ridges in $Q$ we launch a search through relevant hyperplanes represented as the vertices of $G$ (and all ridges of those hyperplanes). Instead of a queue of ridges $Q$ we take a queue $V$ of vertices of $G$. After initializing $V$, in (S₃) all connected components of $G_k$ that share an edge with a vertex from the initial set $V$ are found, and their vertices are added to $V$. Our algorithm is exact if the resulting $V$ contains all vertices of $G_k$.

The expanded procedure based on (A²)–(A₃) and RidgeSearch is called Algorithm (A²) for brevity. The search for all relevant hyperplanes is easy to visualize in the dual graph $G$ of
Figure 4. Induced monochrome subgraphs $G_k$ (i.e., subgraphs of vertices of different weights) of the dual graph of $X$ from Section 2.3: $k = 1$ in the left hand panel, $k = 2$ in the middle panel, and $k = 3$ in the right hand panel.

Figure 5. The induced subgraph $G_3$ in the example from Section 2.3. In both graphs, the vertices highlighted as diamonds are those vertices of $G_3$ that are found in the initial step of RidgeSearch using Algorithm (A) (left hand panel) and its expanded version Algorithm (A$^2$) (right hand panel). Neither of these algorithms detects the tetrahedron of blue points from Figure 2, which corresponds to the smaller connected component of $G_3$ in these figures.

$X$. The results of applying both Algorithms (A) and (A$^2$) to the dataset $X$ from Section 2.3 for $k = 3$ are found in Figure 5. None of these algorithms recovers all relevant hyperplanes of $X$; they both miss the same blue tetrahedron from Figure 2.

Appendix A. Proofs of Theoretical Results

A.1. Proof of Theorem 1

For a relevant halfplane $H \in \mathcal{H}(k)$ with $L = [x_i, x_j] = \partial H$ we denote by $v_L = v_{[x_i, x_j]} \in S^1$ the outer unit normal vector of $H$. In other words, the vector $v_L = v_{[x_i, x_j]}$ is orthogonal to $L = [x_i, x_j]$ and heads toward the open halfplane that contains exactly $k-1$ points from $X$. For unit vectors $u_1, u_2 \in S^1$ define the closed spherical interval $[u_1, u_2]$ to be the shorter arc of unit vectors $S^1$ delimited by $u_1$ and $u_2$, including its endpoints. In the case $u_1 = -u_2$ any of the two half-circles between $u_1$ and $u_2$ can be chosen as $[u_1, u_2]$. The open spherical interval $(u_1, u_2)$ is defined analogously.

For the proof of Theorem 1 we need the following lemma.

**Lemma 2.** Let $L = \partial H_L$ for $H_L \in \mathcal{H}(k)$ and let $[x_i, x_j]$ be another relevant line at level $k$ such that $x_i, x_j \in H_L \setminus L$. That is, both points $x_i, x_j$ are in the interior of the halfplane $H_L$. Then there exists a relevant line $L'$ at level $k$ that passes through either $x_i$ or $x_j$ such that $v_{L'} \in (v_{[x_i, x_j]}, v_L)$. 
If $x_{i_k} \in [x_i, x_j]$, then $H$ is reachable from $G$, and thus $G$ is in the orbit of $H$. Otherwise, $x_{i_k} \in U$, meaning $[x_{i_{k-1}}, x_{i_k}]$ is a relevant line passing through $x_{i_k} \in U$ whose relevant halfplane $H_{[x_{i_{k-1}}, x_{i_k}]}$ lies in the same orbit as $G$, as we wanted to show.

### A.2. Proof of Lemma 1

Since $X$ is in general position, facets of $C$ are $(d - 1)$-simplices (Ziegler 1995, p. 8) and two facets are neighboring if and only if they share $d - 1$ vertices of $C$. Without loss of generality suppose that the origin is an interior point of $C$, and consider the polar $C^0$ of $C$ from in Section 1.2. We know that $C^0$ is also a convex polytope (Schneider 2014, Lemma 2.4.5), its vertices correspond to facets of $C$ and two vertices are connected by an edge if and only if the corresponding facets of $C$ are neighbors (Schneider 2014, sec. 2.4). All pairs of vertices of a polytope are connected by a sequence of edges (Ziegler 1995, sec. 3.5).

### A.3. Proof of Theorem 3

The proof of the main theorem is obtain by combining several auxiliary lemmas. Our first observation is that any two relevant hyperplanes that cut off the same single point of $X$ lie in the same orbit of $H(2)$.

**Lemma 3.** If two relevant halfspaces in $H(2)$ cut off the same point of $X$, they are mutually reachable in $H(2)$.

**Proof.** Let $x_1$ be the point cut off by both halfspaces and consider the convex hull $C_1$ of $X \setminus \{x_1\}$. Without loss of generality we may suppose that the origin lies in the interior of $C_1$. Each of the facets $F_1, \ldots, F_m$ of $C_1$ contains exactly $d$ points from $X \setminus \{x_1\}$. Using the dual construction (1) we find to each $F_j$ a vertex $\hat{F}_j$ of $C_1^0$. By duality considerations, for any pair $\hat{F}_j$ and $\hat{F}_k$ we have that the hyperplanes $[\hat{F}_j]$ and $[\hat{F}_k]$ are neighbors in $X \setminus \{x_1\}$ (that is, sharing $d - 1$ points of $X \setminus \{x_1\}$) if and only if $\hat{F}_j$ and $\hat{F}_k$ share an edge on the boundary of the polar polytope $C_1^0$.

The boundary $\partial H$ of each relevant halfspace $H \in H(2)$ that cuts off $x_1$ is determined by $d$ points from $X \setminus \{x_1\}$. Thus, it contains a facet of $C_1$, and this facet corresponds to a vertex of $C_1^0$ by (1). Denote by $U = \{\hat{F}_1, \ldots, \hat{F}_m\}$ the set of all vertices of $C_1^0$, and consider its subset $U'$ of those vertices that correspond to faces of $C_1$ determined by relevant hyperplanes that cut off $x_1$. Vertices $u \in U'$ are specific among those from $U$ by their property that we have $(x_1, u) > 1$. This is because for $u \in U'$ corresponding to a relevant halfspace $H$ that cuts off $x_1$ we can write by (2) that $\mathbb{R}^d \setminus H = \{x \in \mathbb{R}^d : (x, u) > 1\}$ and $x_1 \notin H$.

Consider the set $H_1 = \{x \in \mathbb{R}^d : (x_1, x) > 1\}$. In the dual space, this is an open halfspace with inner normal $x_1$ at the distance $x_1/\|x_1\|^2$ from the origin. This halfspace is well defined, as we assumed that in the primal space, the origin is contained in $C_1$ and $x_1 \notin C_1$, that is $\|x_1\| > 0$. Using this interpretation, we see that we can write $U' = U \cap H_1$. In particular, $U'$ can be obtained by intersecting $U$ with an open halfspace in the dual space.

We have reduced our problem to the question of whether any two vertices from $U' = U \cap H_1$ can be joined by a sequence of edges of the polytope $C_1^0$ joining two points from $U'$. That is known to be true by a result from the theory of graphs called the Balinski theorem (Balinski 1961, proof of the main Theorem), stated explicitly as the first result from Section 3 in Sallee (1967, p. 473, around formula (3.1)).
halspaces. This should not be confused with the relation of vertices of 
C being connected via a sequence of edges of C. Our definition is given 
alogously to the reachability of halspaces introduced in Section 1.1.

We say that two vertices x, x’ of C are (mutually) vertex-reachable 
if (i) they are vertex-neighborbing, meaning there exists a set V of d − 1 
vertices of C different from x and x’ such that both convex hulls 
conv(V ∪ {x}) and conv(V ∪ {x’}) are facets of C, or (ii) there exists a vertex x” of C that is 
vertex-reachable from both x and x’. In other words, vertices x ̸= x’ are 
vertex-reachable if and only if there exists a finite sequence of vertices 
{x_j} j=1 of C such that x_1 = x, x_j = x’, and for each j = 1, . . . , l − 1, the vertices x_j and x_{j+1} lie on two neighboring facets of C, but do not lie 
on the same facet of C. For example, in the top left panel of Figure 1, 
vertices A, C, E, and G are mutually vertex-reachable, but A is not 
vertex-reachable from B, D, F, or H.

**Lemma 4.** Let H_1, H_2 ∈ H(2) be two relevant halspaces that cut 
off points x_{i_1} and x_{i_2}, respectively, from X. If x_{j_1} and x_{j_2} are mutually 
vertex-reachable, then H_1 and H_2 are mutually reachable in H(2).

**Proof.** If x_{i_1} = x_{j_1}, then H_1 and H_2 are mutually reachable in H(2) 
by Lemma 3. Otherwise, it suffices to consider the case x_{i_1} and x_{j_2} 
vertex-neighborbing with x_{j_1} ̸= x_{i_2}, as the rest follows by induction. Let 
i_1, . . . , i_{l−1} be the points from X \ {x_{i_1}, x_{j_2}} such that both convex hulls 
F_1 = conv({x_{i_1}, x_{i_2}, . . . , x_{i_{l−1}}}) and F_2 = conv({x_{j_1}, x_{j_2}, . . . , x_{i_{l−1}}}) 
are facets of C. Then, in the convex hull C_1 of X \ {x_{i_1}} there exists 
a vertex x_{k_1} such that conv({x_{k_1}, x_{i_2}, . . . , x_{i_{l−1}}}) ̸= F_2 is a new facet of 
C_1. The corresponding halfspace H’_1 ∈ H(2) whose boundary is 
[x_{k_1}, x_{i_2}, . . . , x_{i_{l−1}}] that cuts off x_{i_1} is relevant and reachable from H_1 in 
H(2) by Lemma 3. Similarly there exists a relevant halfspace H’_2 ∈ H(2) 
with boundary [x_{k_2}, x_{j_2}, . . . , x_{i_{l−1}}], for some x_{k_2} ∈ X \ {x_{j_1}}, that 
cuts off x_{j_2} from X, and thus is reachable from H_2 in H(2). Since H’_1 
and H’_2 share the ridge [x_{i_1}, . . . , x_{i_{l−1}}] of X, they are neighboring, and 
thus mutually reachable in a single step in H(2). Since the relation of 
reachability of halspaces is transitive and symmetric, also H_1 and H_2 
are mutually reachable in H(2).

**Lemma 5.** Let F be a facet of C. Then, any vertex of C is vertex-reachable 
from at least one of the vertices of F.

**Proof.** Let x_{i_1}, . . . , x_{i_d} be the vertices of F. If F’ is a neighboring facet 
of F (they share d − 1 vertices), all vertices of F’ are trivially vertex-
reachable from some vertex of F. Therefore, by induction, if F” is a 
facet reachable from F in the sense of Lemma 1, then all vertices of F” 
are vertex-reachable from some vertex of F. By Lemma 1, all facets are 
reachable from F and so the claim holds for all vertices of C.

Putting together all our previous observations, we are now ready to 
prove Theorem 3.

**Main Proof of Theorem 3.**

For k = 1 the claim follows trivially from Lemma 1. We therefore 
consider only k = 2. The initial set of ridges Q of Algorithm (A) 
includes all ridges formed by vertices of

- a relevant halfspace in H(2) determined by [x_{i_1}, . . . , x_{i_d}] cutting off 
a point x_{j} such that x_{i_1}, . . . , x_{i_{d−1}} are vertices of the convex hull C 
(step (A_2)), and
- all halspaces in H(2) that pass through d − 1 points from 
[x_{j}, x_{i_1}, . . . , x_{i_{d−1}}] (step (A_3)).

This includes for each l ∈ [1, . . . , d − 1] a halfspace in H(2) that cuts 
off x_{i_l}.

The points x_{j_1}, x_{i_1}, . . . , x_{i_{d−1}} are all vertices of a facet of C. If H ∈ 
H(2) is a relevant halfspace cutting off x_{j_1} ∈ X, by Lemma 5 we know 
that x_{j_1} is vertex-reachable from one of the points x_{j_1}, x_{i_1}, . . . , x_{i_{d−1}}, say 
x_{j_p}. By Lemma 4, H is reachable from the initial relevant halfspace 
that cuts off x_{j_p}. Since Algorithm (A) finds all relevant halspaces reachable 
from the initial set in step (S_2), it must find all relevant halspaces from 
H(2).

**A.4. Proof of Theorem 4**

To prove that (B1) does indeed guarantee exactness, we introduce 
additional notation. For a subset A ⊂ X of k − 1 points we write H(A) 
for the collection of all the halspaces H ∈ H(k) that cut off A, that is, 
that satisfy X \ H = A. Technically, H(A) ⊂ H(k) depends also on k 
and this should be emphasized in our notation; we shall not do this 
because k is, in fact, implicitly present, as #A = k − 1. We start with a 
simple but useful lemma.

**Lemma 6.** Let S = \{a_1, . . . , a_{d−1}\} ⊂ R^d be in general position. Denote 
by p the projection of a_{d−1} into the hyperplane H = [S \{a_d−1\}]. Then 
there exists j ∈ [1, . . . , d] such that p and a_j lie on the same side of 
the (d−2)-dimensional hyperplane inside H determined by the d−1 points 
S \ {a_j, a_{d−1}}.

**Proof.** The set of d points S \ {a_{d−1}} lies in general position. Thus, it 
forms vertices of a (d − 1)-dimensional simplex inside the hyperplane 
H. Consider now only the (d − 1)-dimensional (affine) space H. Every 
vertex of the simplex lies inside a halfspace in H bounded by the 
(d−2)-dimensional affine subspace (that is, a hyperplane in H) containing 
the remaining vertices. These d halspaces cover H, and in particular, 
one of them contains p. We found a (d − 2)-dimensional hyperplane in H 
determined by d − 1 points from S \ {a_{d−1}} that bounds a halfspace containing 
p and a single point from S \ {a_{d−1}}, as desired. 

For the next lemma we need to consider also the metric structure of 
R^d. Writing ∥x∥ for the Euclidean norm of x ∈ R^d, we define the 
distance of a point x ∈ R^d from a hyperplane G ⊂ R^d by d(x; G) = 
min_y∈G ∥x − y∥.

**Lemma 7.** Let H ∈ H(k+1) be determined by points a_1, . . . , a_k ∈ X, 
and denote by A = X \ H the set of k points cut off from X by H. Let 
a ∈ A be any point of minimum distance to the hyperplane ∂H, that is, 
any a ∈ A that satisfies d(a; ∂H) = min_{x∈A} d(x; ∂H). Then there 
extists an observational halfspace H_1 ⊂ H that shares a ridge with H in 
its boundary and either (i) H_1 ∈ H(k), or (ii) H_1 cuts off from X the 
same k points as H does, that is, H_1 ∈ H(A) ∩ H(k+1), and at the 
same time d(a; ∂H_1) < d(a; ∂H).

**Proof.** We start by applying Lemma 6 with a_{d−1} = a. Let j ∈ 
[1, . . . , d] be the index from Lemma 6. Denote by G the unique 
halspace that satisfies the following three conditions: (i) the boundary 
of G is orthogonal to the boundary of H in the sense that the unit 
normals of G and H are orthogonal; (ii) G contains the d − 1 points 
M = \{a_1, . . . , a_{j−1}, a_{j+1}, . . . , a_d\} in its boundary; and (iii) a_j lies in 
the interior of G. Then by Lemma 6 we have that also a ∈ G, because the 
projection p of a into ∂H and a_j lie on the same side of the projection 
of G into ∂H. Denote by W a two-dimensional plane orthogonal to the 
(d−2)-dimensional affine hull of M. Write o ∈ W for the projection of 
M onto W and for any other point x ∈ R^d denote by x’ ∈ W its 
projection into W. Since M ⊂ ∂H is orthogonal to W, the projection 
of H into W is a halfplane H’ that does not contain a’ and its boundary 
is the line \[a, a’]∥. Likewise, because M ⊂ ∂G, also the projection 
of G is a halfplane G‘ in W that contains o on its boundary and has inner
normal $a'_j - o$. Because $a \in G$, we obtain that $a' \in G'$. For any point data point $b \in X$ denote by $\theta(b)$ the value of the smaller angle between the lines $[o, b']$ and $[o, a'_j]$ inside the plane $W$. This whole setup is visualized in Figure 7.

Denote $D_1 = G \setminus H$, and $D_2 = H \setminus G$, and set $D = D_1 \cup D_2$. Let $c \in D \cap X$ be the point that satisfies

$$\theta(c) = \min \{ \theta(b) : b \in D \cap X \}.$$ 

Such a point is necessarily unique, as $M$, which projects to $o \in W$, already contains $d - 1$ points from $X$, and if there were two different points $c_1, c_2$ from $X \setminus M$ with the same angle $\theta(c)$, there would necessarily exist a hyperplane in $\mathbb{R}^d$ passing through $M$ and both $c_1$ and $c_2$ in $\mathbb{R}^d$, which is impossible due to the assumption of $X$ being in general position.

Denote by $H_1$ the halfspace in $\mathbb{R}^d$ with boundary determined by $M \cup \{c\}$ that does not contain $a$ in its interior. Note that $H_1$ is actually a halfspace obtained by rotating $H$ around $M$ in $\mathbb{R}^d$ in the direction that “keeps” $a_1$ in $H_1$, until its boundary $\partial H_1$ hits the first new point from $X$. Also, $H_1$ obviously shares the ridge $M$ with $H$.

We distinguish two cases. (i) $c 

\in D_1$: In this case, we know that $c$ is one of the $k$ points in $A$. Consequently, $H_1$ cuts off exactly $k - 1$ points from $X$, that is, $H_1 \in \mathcal{H}(k)$, and $H_1 \in \mathcal{H}(A \setminus \{c\})$. (ii) $c \in D_2$: This situation implies that $H_1$ cuts off the same $k$ points as $H$ does, that is, $H_1 \in \mathcal{H}(A)$. What remains to be proved is that $d(a; \partial H) > d(a; \partial H_1)$. To see this, note that because the plane $W$ is parallel to normal vectors of both $\partial H$ and $\partial H_1$, we have that for a given point $x \in \mathbb{R}^d$ the distances $d(x; \partial H)$ and $d(x; \partial H_1)$ equal to the distances of the point of the plane $W$. Therefore, it is enough to prove that $d(a'; [o, a'_j]) > d(a'; [o, c'])$, which follows directly from $\theta(a') > \theta(c)$ and from the fact that $a, c \in D$.

Everything is now ready for the main proof of Theorem 4.

**Main Proof of Theorem 4.**

Denote by $A = X \setminus H$ the set of $k$ points cut off from $X$ by $H$. We apply Lemma 7 to the halfspace $H$ and obtain a halfspace $H_1$. If $H_1$ cuts off $k - 1$ points from $X$, then we set $H' = H$ and $\overline{H} = H_2$. Otherwise, we know that $H_1 \in \mathcal{H}(k + 1) \cap \mathcal{H}(A)$ and for $a \in A$ from the statement of Lemma 7 we have $\min_{x \in A} d(x; \partial H) = d(a; \partial H) > d(a; \partial H_1) \geq \min_{x \in A} d(x; \partial H_1)$. We apply again Lemma 7 to the halfspace $H_1 \in \mathcal{H}(k + 1)$ and obtain another halfspace $H_2$. Note that $H_2$ is reachable from $H$ by our construction. Again, if $H_2$ cuts off $k - 1$ points, then we set $H' = H_1$ and $\overline{H} = H_2$, otherwise, we continue and apply Lemma 7 iteratively. We obtain a sequence $H_1, H_2, H_3, \ldots$. Because in each step we obtain a halfspace with a strictly smaller distance from the closest point of the set $A$, it is not possible for any two halfspaces from the sequence to coincide. Because there are only finitely many observational halfspaces, there must exist an integer $m \geq 2$ such that $H_m \in \mathcal{H}(k)$. Taking $m$ to be the lowest such integer, we set $H' = H_{m-1}$ and $\overline{H} = H_m$. Note that $H_{m-1}$ is a relevant halfspace that cuts off $k$ points from $X$ and also is reachable from $H$. The proof is concluded.

**Supplementary Materials**

- An updated R package **TukeyRegion**, version 0.1.6.3 where the novel exact Algorithm (B) is implemented.
- A pdf file with an additional Algorithm (A') motivated by an extension of Algorithm (A) with $k = 2$. Using the dual graph, we present a dataset where also this possible simplification of Algorithms (B) and (C) fails to recover the central region. Further, we propose to use the dual graph for heuristic assessment of the quality of approximation using non-exact algorithms like Algorithms (A), (A') or (A''). This file also contains very detailed results of the complete simulation study.
- A Mathematica notebook with functions for computing the dual graph of X, containing also interactive visualizations of all the examples provided in this article.
- Complete R source codes for the simulation studies performed in Section 4 and the supplementary material.

**Disclosure Statement**

The authors report there are no competing interests to declare.

**Funding**

This work was partially supported by a mobility grant 8J21FR013 of the Czech Ministry of Education, Youth and Sports and the Programme.
Barrande (Campus France) mobility grant 46745VD of the French Ministry of Europe and Foreign Affairs and French Ministry of Higher Education and Research. P. Laketa was supported by the OP RDE project “International mobility of research, technical and administrative staff at the Charles University,” grant CZ.02.2.69/0.0/0.0/18_053/0016976. The work of S. Nagy was supported by Czech Science Foundation (EXPRO project n. 19-28231X). The work of P. Mozharovskyi was supported by the Young Researcher Grant of the French National Agency for Research (ANR JCJC 2021) in category Artificial Intelligence (registered under the number ANR-21-CE23-0029-01).

**ORCID**

Stanislav Nagy [http://orcid.org/0000-0002-8610-4227](http://orcid.org/0000-0002-8610-4227)

**References**

Balinski, M. L. (1961), "On the Graph Structure of Convex Polyhedra in n-space," *Pacific Journal of Mathematics*, 11, 431–434. [710]

Barber, C., and Mozharovskyi, P. (2023), *Tukey Region: Tukey Region and Median*, R package version 0.1.6.3. [705]

Clarkson, K. L., and Shor, P. W. (1989), "Applications of Random Sampling in Computational Geometry, II," *Discrete & Computational Geometry*, 4, 387–421. [705]

Cole, R., Sharir, M., and Yap, C.-K. (1987), "On k-hulls and Related Problems," *SIAM Journal on Computing*, 16, 61–77. [699]

Donoho, D. L., and Gasko, M. (1992), "Breakdown Properties of Location Estimates based on Halfspace Depth and Projected Outlyingness," *The Annals of Statistics*, 20, 1803–1827. [699]

Dyckerhoff, R., and Mozharovskyi, P. (2016), "Exact Computation of the Halfspace Depth," *Computational Statistics & Data Analysis*, 98, 19–30. [699]

Elton, J. H., and Hill, T. P. (2011), "A Stronger Conclusion to the Classical Ham Sandwich Theorem," *European Journal of Combinatorics*, 32, 657–661. [708]

Genest, M., Massé, J.-C., and Plante, J.-F. (2019), *Depth: Nonparametric Depth Functions for Multivariate Analysis*, R package version 2.1-1.1. [705]

Hubert, M., Rousseauw, P., and Segaert, P. (2017), "Multivariate and Functional Classification Using Depth and Distance," *Advances in Data Analysis and Classification*, 11, 445–466. [699]

Laketa, P., and Nagy, S. (2021), "Reconstruction of Atomic Measures from their Halfspace Depth," *Journal of Multivariate Analysis*, 183, 104727. [699]

Liu, R. Y., Parelius, J. M., and Singh, K. (1999), *Multivariate Analysis by Data Depth: Descriptive Statistics, Graphics and Inference*, *The Annals of Statistics*, 27, 783–858. [699]

Liu, X., Luo, S., and Zuo, Y. (2020), "Some Results on the Computing of Tukey’s Halfspace Median," *Statistical Papers*, 61, 303–316. [702,703,706]

Liu, X., Mosler, K., and Mozharovskyi, P. (2019), "Fast Computation of Tukey Trimmed Regions and Median in Dimension p > 2," *Journal of Computational and Graphical Statistics*, 28, 682–697. [700,701,702]

Matousek, J. (2002), *Lectures on Discrete Geometry*, volume 212 of Graduate Texts in Mathematics, New York: Springer-Verlag. [705]

——— (2003), *Using the Borsuk-Ulam Theorem: Lectures on Topological Methods in Combinatorics and Geometry*, Universitext, Berlin: Springer-Verlag. [708]

Mozharovskyi, P. (2022), "Anomaly Detection Using Data Depth: Multivariate Case," arXiv preprint arXiv:2210.02851. [699]

Nagy, S., and Dvorák, J. (2021), "Illumination Depth," *Journal of Computational and Graphical Statistics*, 30, 78–90. [699]

Pokorný, D., Laketa, P., and Nagy, S. (2023), "Another Look at Halfspace Depth: Flag Halfspaces with Applications," *Journal of Nonparametric Statistics*, to appear. [703]

Rousseeuw, P. J., and Ruts, I. (1998), "Constructing the Bivariate Tukey Median," *Statistica Sinica*, 8, 827–839. [705]

——— (1999), "The Depth Function of a Population Distribution," *Metrika*, 49, 213–244. [699,701]

Rousseeuw, P. J., Ruts, I., and Tukey, J. W. (1999), "The Bagplot: A Bivariate Boxplot," *The American Statistician*, 53, 382–387. [699,705]

Ruts, I., and Rousseeuw, P. J. (1996), "Computing Depth Contours of Bivariate Point Clouds," *Computational Statistics & Data Analysis*, 23, 153–168. [705]

Sallee, G. T. (1967), "Incidence Graphs of Convex Polytopes," *Journal of Combinatorial Theory*, 2, 466–506. [710]

Schneider, R. (2014), *Convex Bodies: the Brunn-Minkowski Theory*, volume 151 of Encyclopedia of Mathematics and its Applications, expanded ed., Cambridge: Cambridge University Press [701,702,710]

Struyf, A., and Rousseeuw, P. J. (1999), "Halfspace Depth and Regression Depth Characterize the Empirical Distribution," *Journal of Multivariate Analysis*, 69, 135–153. [699]

Tukey, J. W. (1975), "Mathematics and the Picturing of Data," in *Proceedings of the International Congress of Mathematicians* (Vancouver, B. C., 1974), Vol. 2, pp. 523–531. Canadian Mathematical Congress, Montreal, Que. [699]

Wagner, U. (2003), "On k-sets and Applications," Ph.D. thesis. ETH Zürich. [705]

Yeh, A. B., and Singh, K. (1997), "Balanced Confidence Regions based on Tukey’s Depth and the Bootstrap," *Journal of the Royal Statistical Society, Series B*, 59, 639–652. [699]

Ziegler, G. M. (1995), *Lectures on Polytopes*, volume 152 of Graduate Texts in Mathematics, New York: Springer-Verlag. [705]

Zuo, Y., and Serfling, R. (2000a), "General Notions of Statistical Depth Function," *Annals of Statistics*, 28, 461–482. [699]

——— (2000b), "On the Performance of Some Robust Nonparametric Location Measures Relative to a General Notion of Multivariate Symmetry," *Journal of Statistical Planning and Inference*, 84, 55–79. [703]