Computing Enclosing Depth*

Bernd Gärtner[1], Fatime Rasiti[2], and Patrick Schnider[3]

1 **Department of Computer Science, ETH Zürich**
`gaertner@inf.ethz.ch`
2 **Department of Mathematics, ETH Zürich**
`frasiti@student.ethz.ch`
3 **Department of Computer Science, ETH Zürich**
`patrick.schnider@inf.ethz.ch`

Abstract

Enclosing depth is a recently introduced depth measure which gives a lower bound to many depth measures studied in the literature. So far, enclosing depth has only been studied from a combinatorial perspective. In this work, we give the first algorithms to compute the enclosing depth of a query point with respect to a data point set in any dimension. In the plane we are able to optimize the algorithm to get a runtime of $O(n \log n)$. In constant dimension, our algorithms still run in polynomial time.

Related Version arXiv:2402.12371

1 Introduction

Medians play an important role in statistics. In contrast to the mean value of some given data, the median depends only on the order of the data points and not on their exact positions. Hence, it is robust against outliers. As data sets are multidimensional in many cases, we are interested in an extension of the term 'median' to higher dimensions. Since there is no clear order of the data points, there are various generalizations of the median to higher dimensions [4, 12, 13]. In order to define the median of some data, the notion of depth of a query point has been introduced. A median is then a query point with the highest depth. Many depth measures only depend on the relative positions of the data points, just like the median, making them again robust against outliers.

After the first depth measure was introduced by Tukey [22] (and is therefore known as Tukey depth), Donoho and Gasko [9] established the idea of a multidimensional median as a deepest point relative to the data points. Various depth measures with different properties have since been introduced, such as simplicial depth [11] and convex hull peeling depth [4].

Depth measures are an important tool in Computer Science for example in geometric matching, pattern matching, clustering [8, 10, 19] and shape fitting applications [2]. Since depth measures give a way to compute medians of data points they also find applications in Statistics such as data visualization [22] and regression analysis [16, 21].

▶ **Definition 1.1** (Depth measure). Let $d \in \mathbb{N}$ and $(\mathbb{R}^d)^S$ be the family of all finite point sets in $\mathbb{R}^d$. A depth measure is a function $D : (\mathbb{R}^d)^S \times \mathbb{R}^d \to \mathbb{R}_{\geq 0}$, $(S, q) \mapsto D(S, q)$. In particular, the function D assigns to a given finite point set S and a query point q a value, which describes how deep the query point q lies within the data set S.

Assume we are given a data set S. Consider all hyperplanes spanned by the points of S. This arrangement A of hyperplanes divides $\mathbb{R}^d$ into connected components of $\mathbb{R}^d \backslash A$. We call

* This work is based on the Master thesis of the second author.

40th European Workshop on Computational Geometry, Ioannina, Greece, March 13–15, 2024.
This is an extended abstract of a presentation given at EuroCG'24. It has been made public for the benefit of the community and should be considered a preprint rather than a formally reviewed paper. Thus, this work is expected to appear eventually in more final form at a conference with formal proceedings and/or in a journal.

these connected components cells. A depth measure where all points in a cell have the same depth is called *combinatorial*.

Aloupis et al. [4] used the fact that simplicial depth is a combinatorial depth measure to compute the simplicial median for $d = 2$ in $O(n^4)$ time. Sachini [15] modified this algorithm to compute the simplicial depth for the whole plane in $O(n^4)$ time. For the case $d = 3$ there are various algorithms that compute the simplicial depth of a single query point in $O(n^2)$ [7]. Cheng and Ouyang [7] discussed an extension of this algorithm to compute simplicial depth in $\mathbb{R}^4$ in $O(n^4)$ time. Afshani et al. [1] later introduced methods to compute simplicial depth in $O(n^d \log n)$ time for $d > 4$. This bound was improved by Pilz et al. [14] to $O(n^{d-1})$.

Another well studied combinatorial depth measure is Tukey depth, also known as halfspace depth. For the case $d = 2$, Aloupis et al. [3] gave a worst case lower bound of $\Omega(n \log n)$ for computing the Tukey depth of an arbitrary query point q with respect to a given point set S of size n. In fact, the Tukey depth of a query point relative to a point set of size n can be computed in $O(n \log n)$ time [17]. There are different approaches to compute the Tukey depth in different dimensions [6, 5, 18]. The algorithm of Rousseeuw et al. [18] to compute the Tukey depth of a query point in $\mathbb{R}^d$ for $d > 2$ has a run time of $O(n^{d-1} \log n)$.

Studying more general families of combinatorial depth measures, Schnider introduced the notion of *enclosing depth*, which turns out to be a natural lower bounds for many combinatorial depth measures [20]. In this work, we will focus on enclosing depth and provide algorithms to compute it.

▶ **Definition 1.2** (k-enclosing). Let S be a point set of size $(d+1)k$ in $\mathbb{R}^d$ and q a query point. If S can be partitioned into $d+1$ pairwise disjoint subsets $S_1, ..., S_{d+1}$, each of size k, such that for any transversal $p_1 \in S_1, ..., p_{d+1} \in S_{d+1}$ the point q lies in the convex hull of $p_1, ..., p_{d+1}$, then we say that $S_1, ..., S_{d+1}$ k-encloses the point q. (Figure 1)

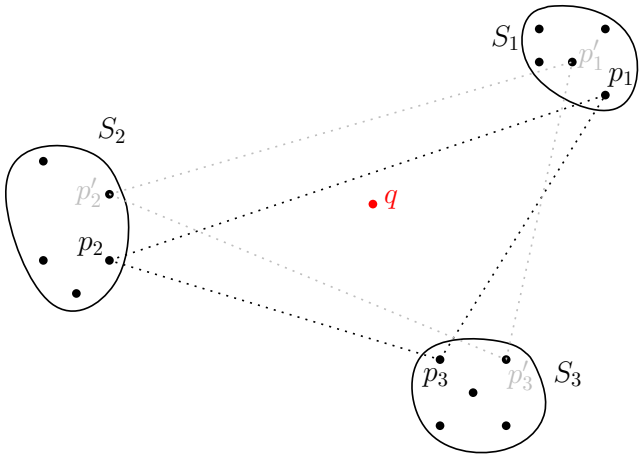

Figure 1 The set $S = S_1 \cup S_2 \cup S_3$ 5-encloses the query point q in $\mathbb{R}^2$.

▶ **Definition 1.3** (Enclosing Depth). Let S be a finite point set in $\mathbb{R}^d$ and q be a query point. The enclosing depth of q with respect to S is the maximum k such that there exist subsets $S_1, ..., S_{d+1}$ of S which k-enclose q. We denote it by $ED(S, q)$. (Figure 2)

In this work, we present algorithms to compute the enclosing depth of a query point q with respect to a data set S in $\mathbb{R}^2$ (Section 2) as well as in general dimension (Section 3).

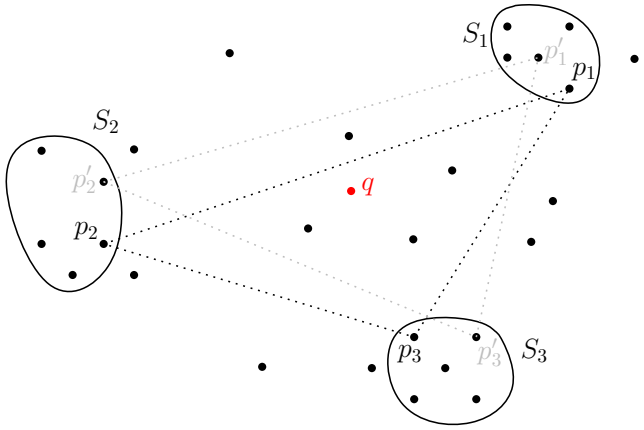

Figure 2 The enclosing depth of the query point q is at least 5.

2 The planar case

Before describing our algorithm, we introduce some combinatorial lemmas that will be helpful in proving the correctness of our algorithm. For these lemmas, we will assume that the query point q is the origin and that the data point set S lies on the unit circle. Combinatorially, this is not a restriction, as the following lemma shows.

▶ **Lemma 2.1.** *Let $S = \{s_1, \dots, s_n\} \subset \mathbb{R}^2$ be a data point set and $q \in \mathbb{R}^2$ a query point such that $S \cup \{q\}$ is in general position. Denote by $S' = \{s'_1, \dots, s'_n\}$ the point set defined by centrally projecting each point in S to a circle of unit radius with center q. Then q is in the convex hull of s_i, s_j, s_k if and only if it is in the convex hull of s'_i, s'_j, s'_k.*

Proof. Assume that q is not in the convex hull of a, b, c. Then there is a line ℓ through q having all of a, b, c on the same side. This is invariant under central projection from q. ◀

Let now $S = \{s_1, \dots, s_n\}$ on the unit circle be ordered in counter-clockwise direction. By an *interval* $[s_a, s_b]$ we denote all the points in S that lie between s_a and s_b, that is,

$$[s_a, s_b] = \begin{cases} \{s \in S \mid s_a \le s \le s_b\} & s_a \le s_b \\ \{s \in S \mid s \ge s_a \text{ or } s \le s_b\} & s_a > s_b \end{cases}$$

In the following, we write indices modulo n, that is, $s_i = s_{i-n}$ for $i \ge n$. We show that in order to find k-enclosing sets we can restrict our attention to intervals and their endpoints.

▶ **Lemma 2.2.** *Let $a_1, a_2, b_1, b_2, c_1, c_2 \in S$ such that the intervals $[a_1, a_2]$, $[b_1, b_2]$ and $[c_1, c_2]$ are pairwise disjoint and for every choice of $a \in \{a_1, a_2\}$, $b \in \{b_1, b_2\}$, $c \in \{c_1, c_2\}$ the origin lies in the convex hull of a, b, c. Then for every choice $a \in [a_1, a_2]$, $b \in [b_1, b_2]$, $c \in [c_1, c_2]$ the origin lies in the convex hull of a, b, c.*

Proof. Assume for the sake of contradiction that the convex hull of a, b, c does not contain the origin and let ℓ be a line through the origin that has all of a, b, c on the same side ℓ^+. As the intervals are pairwise disjoint, one of a_1 or a_2 must also be in ℓ^+. Thus the convex hull of a_i, b, c does not contain the origin for some $i \in \{1, 2\}$. Repeating this argument two more times, we find that one of the 8 triangles spanned by the endpoints does not contain the origin, which is a contradiction to the assumptions of the lemma. ◀

We want to restrict the considered intervals even further. To this end, we define for each point $s \in S$ its *opposite neighbors* $s^{(r)}$ and $s^{(\ell)}$ as the last point in S before $-s$ and the first point in S after $-s$, respectively, see Figure 3 for an illustration.

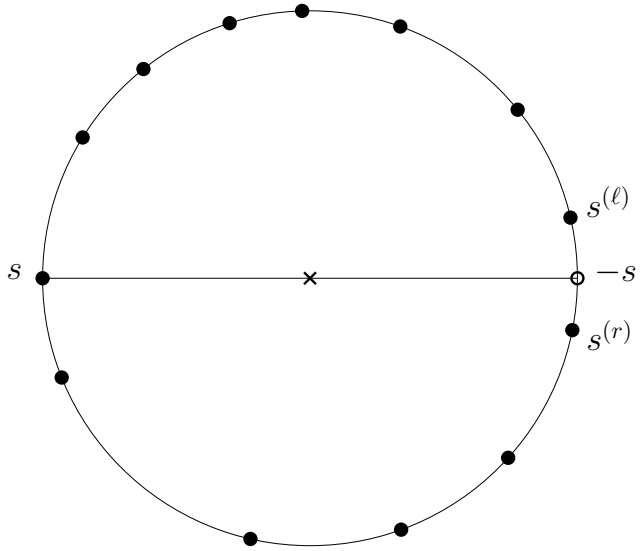

Figure 3 The opposite neighbors $s^{(r)}$ and $s^{(\ell)}$ of a point $s \in S$.

▶ **Lemma 2.3.** *Let $S_1 = [s_i, s_{i+k}]$, S_2 and S_3 be subsets of S that $(k+1)$-enclose the origin. Denote $s_j = s_i^{(r)}$ and $s_m = s_{i+k}^{(\ell)}$. Then S_1, $S_2' := [s_{j-k}, s_j]$ and $S_3' := [s_m, s_{m+k}]$ also $(k+1)$-enclose the origin.*

Proof. See Figure 4 for an illustration of the proof. We first note that (up to relabeling) we must have that $S_2 \subset [s_{i+k+1}, s_j]$ and $S_3 \subset [s_m, s_{i-1}]$. Indeed, both lines through s_i and the origin, as well as through s_{i+k} and the origin must separate S_2 and S_3, as otherwise there would be a choice $a \in S_1$, $b \in S_2$, $c \in S_3$ whose convex hull does not contain the origin. Thus, we can write $S_2 = [s_b, s_{b+k}]$ for $i + k + 1 \leq b \leq j - k$ and similarly $S_2 = [s_{c-k}, s_c]$ for $m + k \leq c \leq i - 1$. In particular both s_i, s_b, s_c and s_{i+k}, s_b, s_c contain the origin.

We claim that s_i, s_b, s_m also contains the origin. Assume it does not. Then either the line ℓ_i through s_i and the origin or the line ℓ_b through s_b and the origin must separate s_c and s_m. By construction, ℓ_i has both s_c and s_m on the same side, so it must be ℓ_b. But this would imply that $m > c$, which is a contradiction. A symmetric argument also shows that s_{i+1}, s_b, s_m contains the origin. Analogously it can be shown that s_i, s_j, s_c as well as s_{i+1}, s_j, s_c contains the origin. Finally, by construction s_i, s_j, s_m as well as s_{i+k}, s_j, s_m contains the origin. The statement now follows from Lemma 2.2 and the fact that $b \leq j - k$ and $m + k \leq c$. ◀

Description of the algorithm: We are given a set S of n data points in the plane and a query point q. In a pre-processing step we first sort the points radially around q, giving a counter-clockwise order $s_1, \ldots, s_n$ on S and then for each $s \in S$ we compute $s^{(r)}$ and $s^{(\ell)}$ using binary search. For the main part of the algorithm we run the following subroutine, which for a given integer k checks whether there are three sets that $(k+1)$-enclose q: for each $s_i \in S$, with $s_j = s_i^{(r)}$ and $s_m = s_{i+k}^{(\ell)}$ the subroutine checks whether all 8 triangles a, b, c for $a \in [s_i, s_{i+k}]$, $b \in [s_{j-k}, s_j]$, $c \in [s_m, s_{m+k}]$ contain q and the intervals are pairwise disjoint, returning TRUE if this holds for some s_i, and FALSE otherwise. By doing a binary

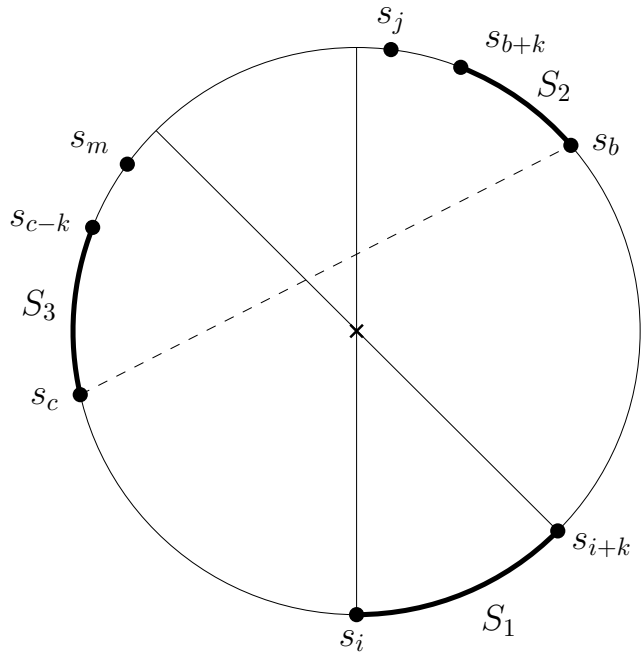

Figure 4 An illustration of the proof of Lemma 4.

search over the values of $k \in \{0, \dots, n\}$ we find the largest value k for which the subroutine returns true and return $(k+1)$.

▶ **Theorem 2.4.** *The above algorithm computes the enclosing depth of q with respect to $S \subset \mathbb{R}^2$ in time $O(n \log n)$.*

Proof. We first show the correctness of the algorithm. It follows from Lemma 2.2 that if the subroutine returns TRUE then the considered intervals are indeed $(k+1)$-enclosing. On the other hand, if there are $(k+1)$-enclosing sets, by Lemma 2.3 the subroutine will find them.

As for the runtime, we can sort in time $O(n \log n)$. After this, we perform $2n$ binary searches, each taking $O(\log n)$ time, thus the total runtime of the pre-processing step is $O(n \log n)$. For the runtime of the subroutine, we notice that for each choice of s_i the required checks can be done in time $O(1)$, so the runtime of the subroutine is $O(n)$. As we call it for $O(\log n)$ many values of k we get the desired runtime. ◀

3 Higher dimensions

Our algorithm in general dimension is based on the following observation:

▶ **Lemma 3.1** (Lemma 20 in [20]). *Let $S_1, \dots, S_{d+1} \subset S \subset \mathbb{R}^d$ be point sets which enclose a point q, where $S \cup \{q\}$ is in general position. Then there are $d+1$ closed halfspaces $H_1^-, \dots, H_{d+1}^-$ such that each H_i^- contains q on its boundary, $H_i^- \cap S = S_i$ for each i and $H_1^- \cup \dots \cup H_{d+1}^- = \mathbb{R}^d$.*

Denoting by H_i^+ the complement of H_i^- we show in the full version that we get $S_i \subset \bigcap_{j \neq i} H_j^+$ and $\bigcap_i H_i^+ = \{q\}$. As we show in the full version, given enclosing sets $S_1, \dots, S_{d+1}$ we can rotate the halfspaces H_i^+ to get closed halfspaces H_i' whose boundaries contain q and $d-1$ points of S and for which $H_i' \cap S = H_i^+ \cap S$ and $\bigcap_i H_i' = \{q\}$. On the other

hand, given halfspaces $H'_1, \ldots, H'_{d+1}$, each boundary containing q and $d-1$ points of S with $\bigcap_i H'_i = \{q\}$, defining $S_i := \bigcap_{j \neq i}(H'_j \cap S)$ we show in the full version that for every transversal $p_1 \in S_1, ..., p_{d+1} \in S_{d+1}$ the point q lies in the convex hull of $p_1, ..., p_{d+1}$. Combining these facts, we get the following strengthening of Lemma 3.1.

▶ **Lemma 3.2.** *Let $S_1, \ldots, S_{d+1} \subset S \subset \mathbb{R}^d$ and $q \in \mathbb{R}^d$ such that $S \cup \{q\}$ is in general position. Then $S_1, \ldots, S_{d+1}$ enclose q if and only if there are $d+1$ halfspaces $H'_1, \ldots H'_{d+1}$ whose boundaries contain q and $d-1$ points of S, and for which $S_i \subset \bigcap_{j \neq i} H_j^+$ and $\bigcap_i H_i^+ = \{q\}$.*

Description of the algorithm: For each choice of $d-1$ points in S, consider the two halfspaces defined by the hyperplane through q and these $d-1$ points. This defines a set $\mathcal{H}$ of $2\binom{n}{d-1}$ halfspaces. For any $d+1$ halfspaces $H_1, \ldots, H_{d+1} \in \mathcal{H}$ first check if $\bigcap_i H_i = \{q\}$. If not, continue with the next choice, otherwise count for each i the number of points of S in $\bigcap_{j \neq i} H_j$ and denote by k the smallest of the $d+1$ numbers. In the end, return the largest such k encountered over all choices of $d+1$ halfspaces in $\mathcal{H}$.

▶ **Theorem 3.3.** *The above algorithm computes the enclosing depth of q with respect to $S \subset \mathbb{R}^d$ in time $O(n^{d^2})$.*

Proof. The correctness follows from Lemma 3.2. As for the runtime, we have $|\mathcal{H}| \in O(n^{d-1})$ out of which we choose sets of $d+1$ elements, so there are $O(n^{(d-1)(d+1)})$ sets of halfspaces that we consider. For each set the check and the counting can be done in time $O(n)$ so the total runtime is $O(n^{(d-1)(d+1)+1}) = O(n^{d^2})$. ◀

4 Conclusion

We have given two algorithms to compute the enclosing depth of a query point q with respect to a data point set S, one for the plane and one in general dimension. The planar algorithm matches the runtimes of computing many other depth measures. For some measures, such as Tukey depth, matching lower bounds for the computation time have been shown [3]. It would be interesting to adapt these lower bounds to enclosing depth.

In higher dimension, many depth measures can be computed in time $O(n^{d-1})$, which is significantly faster than the runtime of our algorithm. We believe that enclosing depth can be computed more efficiently as well, but some additional ideas are likely required for this.

Finally, there are other natural algorithmic problems for depth measures even in the plane, such as computing the depth of the entire plane or finding a deepest query point. Using our algorithm and the fact that the arrangement spanned by the lines through all pairs of data points has $O(n^4)$ cells we can solve both those problems in $O(n^5 \log n)$, but we again believe that this is not optimal.

References

1 Peyman Afshani, Donald R. Sheehy, and Yannik Stein. Approximating the simplicial depth in high dimensions. *In Proceedings of 32nd European workshop on Computational Geometry*, pages 39–42, 2001.

2 Dror Aiger, Haim Kaplan, and Micha Sharir. Duality-based approximation algorithms for depth queries and maximum depth. *arXiv preprint*, page arXiv:2006.12318, 2020.

3 Greg Aloupis, Carmen Cortés, Francisco Gómez, Michael Soss, and Godfried Toussaint. Lower bounds for computing statistical depth. *Computational Statistics & Data Analysis*, 40(2):223–229, 2002.

4 Greg Aloupis, Stefan Langerman, Michael Soss, and Godfried Toussaint. Algorithms for bivariate medians and a fermat-torricelli problem for lines. *Computational Geometry*, 2003.

5 David Bremner, Dan Chen, John Iacono, Stefan Langerman, and Pat Morin. Output-sensitive algorithms for tukey depth and related problems. *Statistics and Computing*, 18(3): 259–266, 2008.

6 Dan Chen, Pat Morin, and Uli Wagner. Absolute approximation of tukey depth: Theory and experiments. *Computational Geometry*, 46(5): 566–573, 2013.

7 Andrew Y. Cheng and Ming Ouyang. On algorithms for simplicial depth. *In Proceedings of the 13th Canadian Conference of Computation Geometry (CCCG)*, pages 53–56, 2001.

8 Andreas Christmann, Paul Fischer, and Thorsten Joachims. Classification based on the support vector machine, regression depth, and discriminant analysis. page 225–230, 2002.

9 David L. Donoho and Miriam Gasko. Breakdown properties of location estimates based on halfspace depth and projected outlyingness. *The Annals of Statistics*, 20(4): 1803-1827 ,1992.

10 Rebecka Jörnsten. Clustering and classification based on the l1 data depth. *Journal of Multivariate Analysis*, 90(1): 67-89, 2004.

11 Regina Y. Liu. On a notion of data depth based on random simplices. *The Annals of Statistics*, 18(1): 405–414, 1990.

12 Regina Y. Liu, Jesse M. Parelius, and Kesar Singh. Multivariate analysis by data depth: descriptive statistics, graphics and inference. *Ann. Statist.*, 27(3): 783–858, 06.1999.

13 Karl Mosler. Springer Berlin Heidelberg, Berlin, Heidelberg. *Depth Statistics.*, pages 17–34, 06.1999.

14 Alexander Pilz, Emo Welzl, and Manuel Wettstein. From crossing-free graphs on wheel sets to embracing simplices and polytopes with few vertices. *Discrete & Computational Geometry*, 64(3): 1067–1097, 2020.

15 Sachini Kanchana Rajapakse. Computing Batched Depth Queries and the Depth of a Set of Points. *Faculty of Graduate Studies of The University of Manitoba Winnipeg*, pages 20–26, 2022.

16 Peter J. Rousseeuw and Mia Hubert. Regression depth. *Journal of the American Statistical Association*, 94(446): 388-402, 1999.

17 Peter J. Rousseeuw and Ida Ruts. Bivariate location depth. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 45(4): 516–526, 1996.

18 Peter J. Rousseeuw and Anja Struyf. Computing location depth and regression depth in higher dimensions. *Statistics and Computing*, 8(3): 193–203, 1998.

19 Ida Ruts and Peter J. Rousseeuw. Computing depth contours of bivariate point clouds. *Computational Statistics and Data Analysis*, 23(1): 153–168, 1996.

20 Patrick Schnider. Enclosing depth and other depth measures. *Combinatorica*, pages 1–23, 2023.

21 Patrick Schnider and Pablo Soberón. Combinatorial depth measures for hyperplane arrangements. In *39th International Symposium on Computational Geometry (SoCG 2023)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2023.

22 John W. Tukey. Mathematics and the picturing of data. 1975.