Computing Shapley Values in the Plane

Sergio Cabello\textsuperscript{1,2} · Timothy M. Chan\textsuperscript{3}

Received: 3 May 2020 / Revised: 29 September 2021 / Accepted: 11 October 2021 / Published online: 22 February 2022
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

Abstract
We consider the problem of computing Shapley values for points in the plane, where each point is interpreted as a player, and the value of a coalition is defined by the area or the perimeter of usual geometric objects, such as the convex hull or the minimum axis-parallel bounding box. For sets of \( n \) points in the plane, we show how to compute in roughly \( O(n^{3/2}) \) time the Shapley values for the area of the minimum axis-parallel bounding box and the area of the union of the rectangles spanned by the origin and the input points. When the points form an increasing or decreasing chain, the running time can be improved to near-linear. In all these cases, we use linearity of the Shapley values and algebraic methods. We also show that Shapley values for the area and the perimeter of the convex hull can be computed in \( O(n^2) \) time, while for the minimum enclosing disk it takes \( O(n^3) \) time. These problems are closely related to the model of stochastic point sets considered in computational geometry, but here we have to consider random insertion orders of the points instead of a probabilistic existence of points.

Keywords Shapley values · Stochastic computational geometry · Convex hull · Minimum enclosing disk · Bounding box · Arrangements · Convolutions · Airport problem

Editor in Charge: Kenneth Clarkson

S. Cabello: supported by the Slovenian Research Agency, program P1-0297 and projects J1-8130, J1-8155, J1-9109, J1-1693. T. M. Chan: supported in part by NSF Grant CCF-1814026.

Sergio Cabello
sergio.cabello@fmf.uni-lj.si

Timothy M. Chan
tmc@illinois.edu

1 Department of Mathematics, Institute of Mathematics, Physics and Mechanics, Ljubljana, Slovenia
2 Department of Mathematics, FMF, University of Ljubljana, Ljubljana, Slovenia
3 Department of Computer Science, University of Illinois at Urbana-Champaign, Champaign, IL, USA
Fig. 1 Different costs associated to a point set that are considered in this paper. The cross represents the origin. In all cases we focus on the area

1 Introduction

One can associate several meaningful values to a set $P$ of points in the plane, like for example the area of the convex hull or the area of the axis-parallel bounding box. How can we split this value among the points of $P$? Shapley values are a standard tool in cooperative games to “fairly” split common cost between different players. Our objective in this paper is to present algorithms to compute the Shapley values for points in the plane when the cost of each subset is defined by geometric means.

Coalitional games in the plane Formally, a coalitional game is a pair $(P, v)$, where $P$ is the set of players and $v : 2^P \to \mathbb{R}$ is the characteristic function, which must satisfy $v(\emptyset) = 0$. Depending on the problem at hand, the characteristic function can be seen as a cost or a payoff associated to each subset of players (usually called a coalition). Coalitional games are a very common model for cooperative games with transferable utility.

In our setting, the players will be points in the plane. Thus $P \subset \mathbb{R}^2$. Such scenario arises naturally in the context of game theory through modeling: each point represents an agent, and each coordinate of the point represents an attribute of the agent.

We will consider characteristic functions given by the area of shapes that “enclose” the points. The shapes that we consider are succinctly described in Fig. 1. More precisely, we consider the following coalitional games; other simpler auxiliary games are considered later.

**AreaConvexHull game**: The characteristic function is $v_{\text{ch}}(Q) = \text{area}(\text{CH}(Q))$ for each nonempty $Q \subset P$, where $\text{CH}(Q)$ denotes the convex hull of $Q$.

**AreaEnclosingDisk game**: The characteristic function is $v_{\text{ed}}(Q)$ defined by $\text{area}(\text{med}(Q))$ for each nonempty $Q \subset P$, where $\text{med}(Q)$ is a minimum enclosing disk for $Q$.

**AreaAnchoredRectangles game**: The characteristic function is $v_{\text{ar}}(Q) = \text{area}\left(\bigcup_{p \in Q} R_p\right)$ for each nonempty $Q \subset P$, where $R_p$ is the axis-parallel rectangle with one corner at $p$ and another corner at the origin.

**AreaBoundingBox game**: The characteristic function is $v_{\text{bb}}(Q) = \text{area}(\text{bb}(Q))$ for each nonempty $Q \subset P$, where $\text{bb}(Q)$ is the smallest axis-parallel bounding box of $Q$.

**AreaAnchoredBoundingBox game**: The characteristic function is $v_{\text{abb}}(Q)$, defined by $\text{area}(\text{bb}(Q \cup \{o\}))$ for each nonempty $Q \subset P$, where $o$ is the origin.

Most of our discussion focuses on the area of the shapes. One can consider the variants where the perimeter of the shapes is used. The name of the problem is obtained
by replacing AREA$\star$ by PERIMETER$\star$. For most shapes, handling the perimeter is either substantially easier or can be solved by similar methods, and it will be discussed along the way.

Note that in all the problems we consider monotone characteristic functions: whenever $Q \subset Q' \subset P$ we have $v(Q) \leq v(Q')$.

As examples where such games appear naturally one could consider distributing the cost of covering a region of users with mobile Internet or splitting the cost of maintaining a fence around a region.

**Shapley values** Shapley values are probably the most popular solution concept for coalitional games. The objective is to split the value $v(P)$ between the different players of a coalitional game $(P, v)$ in a meaningful way. It is difficult to overestimate the relevance of Shapley values. See the book edited by Roth [32] or the survey by Winter [36] for a general discussion showing their relevance. There are also different axiomatic characterizations of the concept, meaning that Shapley values can be shown to be the only map satisfying certain natural conditions. Shapley values can be interpreted as a cost allocation, a split of the payoff, or, after normalization, as a power index. The Shapley–Shubik power index arises from considering voting games, a particular type of coalitional game. We refer to some textbooks in Game Theory ([15, Chap. IV.3], [25, Sect. 9.4], [27, Sect. 14.4]) for a comprehensive treatment.

In a nutshell, the Shapley value of a player $p$ in a coalitional game is the expected increase in the value of the characteristic function when inserting the player $p$, if the players are inserted in an order chosen uniformly at random. We next make this definition precise.

Consider a coalitional game $(P, v)$. We denote by $n$ the number of players and by $[n]$ the set of integers $\{1, \ldots, n\}$. A permutation of $P$ is a bijective map $\pi: P \rightarrow [n]$. Let $\Pi(P)$ be the set of permutations of $P$. Each permutation $\pi \in \Pi(P)$ defines an ordering in $P$, where $\pi(p)$ is the position of $p$ in that order. We will heavily use this interpretation of permutations as defining an order in $P$. For each element $p \in P$ and each permutation $\pi \in \Pi(P)$, let $P(\pi, p)$ be the elements of $P$ before $p$ in the order defined by $\pi$, including $p$. Thus $P(\pi, p) = \{q \in P \mid \pi(q) \leq \pi(p)\}$. We can visualize $P(\pi, p)$ as adding the elements of $P$ one by one, following the order defined by $\pi$, until we insert $p$. The increment in $v(\cdot)$ when adding player $p$ is

$$\Delta(v, \pi, p) = v(P(\pi, p)) - v(P(\pi, p) \setminus \{p\}).$$

The **Shapley value** of player $p \in P$ in the game $(P, v)$ is

$$\phi(p, v) = \frac{1}{n!} \sum_{\pi \in \Pi(P)} \Delta(v, \pi, p) = \mathbb{E}_{\pi}[\Delta(v, \pi, p)],$$

where $\pi$ is picked uniformly at random from $\Pi(P)$. It is not difficult to see that the Shapley values indeed split the value $v(P)$ among the players, that is, $\sum_{p \in P} \phi(p, v) = v(P)$. Since several permutations $\pi$ define the same subset $P(\pi, p)$, the Shapley value
of \( p \) is often rewritten as

\[
\phi(p, v) = \sum_{S \subseteq P \setminus \{p\}} \frac{|S|!(n - |S| - 1)!}{n!} \left( v(S \cup \{p\}) - v(S) \right).
\]

Computing Shapley values from these formulas is computationally infeasible because we have to consider either all the permutations or all the subsets of the players. In fact, there are several natural instances where computing Shapley values is difficult.

**Overview of our contribution** We show that the Shapley values for the **AreaConvexHull** and **AreaEnclosingDisk** games can be computed in \( O(n^2) \) and \( O(n^3) \) time, respectively. The same time bounds hold for the perimeter. These problems resemble the models recently considered in stochastic computational geometry; see for example [1,2,16,18,19,29,37]. However, there are some key differences in the models. In the most basic model in stochastic computational geometry, sometimes called unipoint model, we have a point set and, for each point, a known probability of being actually present, independently for each point. Then we want to analyze a certain functional, like the expected area of the minimum enclosing disk, the expected area of the smallest bounding box, or the probability that some point is contained in the convex hull.

In our scenario, we have to consider random insertion orders of the points and analyze the expected increase in the value of the characteristic function after the insertion of a fixed point \( p \). Thus, we have to consider subsets of points constructed according to a different random process. In particular, whether other points precede \( p \) or not in the random order are not independent events. In our analysis, we condition on properties of the shape before adding the new point \( p \). In the case of the minimum enclosing disk we use that each minimum enclosing disk is determined by at most three points, while in the case of the convex hull we condition separately on each single edge of the convex hull before the insertion of \( p \). A straightforward application of this principle gives polynomial-time algorithms. To improve the running time we carry out this idea rewriting the Shapley values in a different way and grouping permutations with a similar behavior. Finally, we use arrangements of lines and planes to speed up the computation by an additional linear factor.

For the **AreaAnchoredRectangles**, **AreaBoundingBox**, and **AreaAnchoredBoundingBox** games we show that Shapley values can be computed in \( O(n^{3/2} \log n) \) time. In the special case where the points form a chain (increasing or decreasing \( y \)-coordinate for increasing \( x \)-coordinate), the Shapley values of those games can be computed in \( O(n \log^2 n) \) time. We refer to these games as axis-parallel games.

It is relative easy to compute the Shapley values for these axis-parallel games in quadratic time using arrangements of rectangles and the linearity of Shapley values. We will discuss this as an intermediary step towards our solution. However, it is not obvious how to get subquadratic time. Besides using the linearity of Shapley values, a key ingredient in our algorithms is using convolutions to evaluate at multiple points some special rational functions that keep track of the ratio of permutations with a certain property. The use of algebraic methods in computational geometry is not very common, and there are few results [3,5,20,24] using such techniques in geometric...
problems. (In contrast, there is a quickly growing body of works in discrete geometry using real algebra.)

Our $O(n^{3/2} \log n)$ algorithm bears some similarities with other existing algorithms with near $n^{3/2}$ time complexity in the computational geometry literature, for problems like Klee’s measure problem [28]. As in these previous algorithms, we employ an orthogonal subdivision where each region is empty of input points inside, and is “influenced” on average by $O(\sqrt{n})$ of the points outside. What is new is our combination of such a geometric partitioning scheme with the aforementioned algebraic techniques.

At the basement of our algorithms, we need to count the number of permutations with certain properties. For this we use some simple combinatorial counting. In summary, our results combine fundamental concepts from several different areas and motivated by classical concepts of game theory, we introduce new problems related to stochastic computational geometry and provide efficient algorithms for them.

**Related work** The book by Chalkiadakis et al. [8] and the chapter by Deng and Fang [9] give a summary of computational aspects of coalitional games. The book by Nisan et al. [26] provides a general overview of the interactions between game theory and algorithms.

In the classical AIRPORT problem considered by Littlechild and Owen [21], we have a set $P$ of points with positive coordinate on the real line, and the cost of a subset $Q$ of the points is given by $\max Q$. It models the portion of the runway that has to be used by each airplane, and Shapley values provide a way to split the cost of the runway among the airplanes. As pointed out before, the points represent agents, in this case airplanes. Several other airport problems are discussed in the survey by Thomson [33]. Using inclusion–exclusion, the airport problem is equivalent to the problem of allocating the length of the smallest interval that contains a set of points on the line. The problems considered in this paper are natural generalizations of the concept of interval when going from one to two dimensions.

Another very common solution concept for a coalitional game $(P, v)$ is the **core**. In the case when the characteristic function $v(\cdot)$ denotes a cost, the core is defined as

$$\left\{ (x_p)_{p \in P} \in \mathbb{R}^P \mid \sum_{p \in P} x_p = v(P) \text{ and } \forall S \subseteq P : \sum_{p \in S} x_p \geq v(S) \right\}.$$

When $v(\cdot)$ is a gain for the coalition, the inequality in the definition is reversed. In both cases it models the fact that a proper subset of the players does not have an incentive to go on their own. The size of the core is considered a proxy to the stability of the game and, in particular, it is of interest whether the core is nonempty. There are other solution concepts for coalitional games; we refer the reader to the aforementioned general references.

Puerto et al. [30] study the Minimum Radius Location Game in general metric spaces. When specialized to the Euclidean plane, this is equivalent to using the perimeter of the minimum enclosing disk centered at one of the points. In the definition of the game there is also a special point playing the role of root; we refer the reader...
to their work for a precise definition. The paper also considers the $L_1$-metric, which
is proportional to the perimeter of the minimum enclosing axis-parallel square (after
applying a rotation). However, their work focuses on understanding the core of the
game, and does not discuss the computation of Shapley values. In particular, they show
that the Minimum Radius Location Game in the Euclidean plane has nonempty core.
Puerto et al. [31] also discuss the Minimum Diameter Location Game, which can be
defined for arbitrary metric spaces, but then focus their discussion on graphs.

Faigle et al. [13] consider the TSP coalitional game in general metric spaces, spe-
cialize some results to the Euclidean plane, and provide approximate allocations of
the costs.

The computation of Shapley values has been considered for several games on
graphs. The aforementioned Airport problem can be considered a shortest spanning-
path game in a (graph-theoretic) path. Megiddo [23] extended this to trees, while Deng
and Papadimitriou [10] discuss a game on arbitrary graphs defined by induced sub-
graphs. They show that the Shapley values are easy to compute, while characterizing
the core is NP-complete.

In the Minimum Cost Spanning Tree Game, we are given a complete graph with
non-negative edge weights and a special vertex $r$, the root, the players are the vertices
of the graph, and the cost for each subset of the vertices is the weight of the minimum
tree spanning those vertices and the root $r$. Granot and Huberman [17] show that the
core is always non-empty and an element in the core can be computed efficiently.
Ando [4] shows that computing the Shapley values in such games is #P-hard, even
when the edge weights take values on $\{0, 1\}$, and provides some cases that can be
solved in polynomial time. Still for the Minimum Cost Spanning Tree Game, Faigle
et al. [14] show that it is NP-hard to decide whether a given element belongs to the
core.

There is a very large body of follow up works for graphs, but we could not trace
other works considering the computation of Shapley values for games defined through
planar objects, despite being very natural.

Assumptions We will assume general position in the following sense: no two points
have the same $x$ or $y$ coordinate, no three points are collinear, and no four points are
cocircular. In particular, the points are all different. The actual assumptions depend on
the game under consideration. It is simple to consider the general case, but it makes
the notation more tedious.

We assume a unit-cost real-RAM model of computation. In a model of computation
that accounts for bit complexity, time bounds may increase by polynomial factors (even
if the input numbers are integers, the outputs may be rationals with large numerators
and denominators).

Organization We start with preliminaries in Sect. 2, where we set the notation, present
basic properties of Shapley values, explain some basic consequences of the AIRPORT
game, discuss our needs of algebraic computations, and count permutations with some
properties. The section is slightly long because of our use of tools from different areas.

Then we analyze different games. The AREA Convex Hull and AREA Enclosing Disk
games are considered in Sects. 3 and 4, respectively. In Sect. 5 we discuss the
AreaAnchoredRectangles game, while in Sect. 6 we discuss the AreaBoundingBox and AreaAnchoredBoundingBox games. To understand Sect. 6 one has to go through Sect. 5 first. The remaining sections (and games) have no dependencies between them and can be read in any order. We conclude in Sect. 7 with some discussion.

2 Preliminaries

In this section we provide notation and background used through the paper. In most cases we will consider points in the Euclidean plane $\mathbb{R}^2$. The origin is denoted by $o$. For a point $p \in \mathbb{R}^2$, let $R_p$ be the axis-parallel rectangle with one corner at the origin $o$ and the opposite corner at $p$. As already mentioned in the introduction, for each $Q \subset \mathbb{R}^2$ we use $\text{bb}(Q)$ for the (minimum) axis-parallel bounding box of $Q$, $\text{med}(Q)$ for a minimum (actually, the minimum) enclosing disk for $Q$, and $\text{CH}(Q)$ for the convex hull of $Q$.

We try to use the word rectangle when one corner is defined by an input point, while the word box is used for more general cases. All rectangles and boxes in this paper are axis-parallel, and we drop the adjective axis-parallel when referring to them. An anchored box is a box with one corner at the origin.

For a point $p \in \mathbb{R}^2$ we denote by $x(p)$ and $y(p)$ its $x$- and $y$-coordinate, respectively. We use $x(Q)$ for $\{x(q) \mid q \in Q\}$, and similarly $y(Q)$. A set $P$ of points is a decreasing chain, if $x(p) < x(q)$ implies $y(p) > y(q)$ for all $p, q \in P$. A set $P$ of points is an increasing chain if $x(p) < x(q)$ implies $y(p) < y(q)$ for all $p, q \in P$.

2.1 Shapley Values

It is easy to see that Shapley values are linear in the characteristic functions. Indeed, since for any two characteristic functions $v_1, v_2 : 2^P \to \mathbb{R}$ and for each $\lambda_1, \lambda_2 \in \mathbb{R}$ we have

$$\Delta(\lambda_1 v_1 + \lambda_2 v_2, \pi, p) = \lambda_1 \cdot \Delta(v_1, \pi, p) + \lambda_2 \cdot \Delta(v_2, \pi, p),$$

we obtain

$$\phi(p, \lambda_1 v_1 + \lambda_2 v_2) = \lambda_1 \cdot \phi(p, v_1) + \lambda_2 \cdot \phi(p, v_2).$$

In fact, (a weakened version of) linearity is one of the properties considered when defining Shapley values axiomatically. It is easy to obtain the Shapley values when the characteristic function $v$ is constant over all nonempty subsets of $P$. In this case $\phi(p, v) = v(P)/n$.

We say that two games $(P, v)$ and $(P', v')$, where $P$ and $P'$ are point sets in Euclidean space, are isometrically equivalent if there is some isometry that transforms one into the other. That is, there is some isometry $\rho : \mathbb{R}^2 \to \mathbb{R}^2$ such that $P' = \rho(P)$ and $v' = v \circ \rho$. Finding Shapley values for $(P, v)$ or $(P', v')$ is equivalent because $\phi(\rho(p), v') = \phi(p, v)$ for each $p \in P$. 

Springer
2.2 Airport and Related Games

We will consider the following games, some of them 1-dimensional games.

**Airport game**: The set of players $P$ is on the positive side of the real line and the characteristic function is $v_{\text{air}}(Q) = \max Q$ for each nonempty $Q \subset P$.

**LengthInterval game**: The set of players $P$ is on the real line and the characteristic function is $v_{\text{li}}(Q) = \max Q - \min Q$ for each nonempty $Q \subset P$.

**AreaBand game**: The set of players $P$ is on the plane with positive $x$ coordinate and the characteristic function is $v_{\text{ab}}(Q) = (\max y(P) - \min y(P)) \cdot v_{\text{air}}(x(Q))$ for each nonempty $Q \subset P$.

**PerimeterBoundingBox game**: The set of players $P$ is on the plane and the characteristic function is $v_{\text{pbb}}(Q) = \text{per} \left( \text{bb}(Q) \right)$ for each nonempty $Q \subset P$.

**PerimeterAnchoredBoundingBox game**: The set of players $P$ is on the plane and the characteristic function is $v_{\text{pabb}}(Q) = \text{per} \left( \text{bb}(Q \cup \{o\}) \right)$ for each nonempty $Q \subset P$.

**Lemma 2.1** The Shapley values for the Airport, LengthInterval, AreaBand, PerimeterBoundingBox, and PerimeterAnchoredBoundingBox games can be computed in $O(n \log n)$ time.

**Proof** The computation of the Shapley values for the Airport game was done by Littlechild and Owen [21], as follows. If $P$ is given by $0 < x_1 < \ldots < x_n$, then

$$
\phi(x_i, v_{\text{air}}) = \begin{cases} 
\frac{x_1}{n} & \text{if } i = 1, \\
\phi(x_{i-1}, v_{\text{air}}) + \frac{x_i - x_{i-1}}{n - i + 1} & \text{for } i = 2, \ldots, n.
\end{cases}
$$

One arrives at this formula by noticing that the Airport game is a sum of $n$ coalitional games, one per interval of $[0, x_n] \setminus \{x_1, x_2, \ldots, x_{n-1}\}$, and that in a random permutation of the players, each of the $n - i + 1$ players $x_i, \ldots, x_n$ is equally likely to be the first one adding the length of the interval between $x_{i-1}$ and $x_i$ to $v_{\text{air}}(\cdot)$. We will be using the same line of thought in Lemma 5.1. This shows the result for the Airport game because it requires sorting.

The LengthInterval game can be reduced to the Airport problem using simple inclusion–exclusion on the line. Consider the values $\alpha = \min P$ and $\beta = \max P$ and define the characteristic functions

$$
v_1(Q) = \max Q - \alpha \quad \text{for } Q \neq \emptyset;
$$

$$
v_2(Q) = \beta - \min Q \quad \text{for } Q \neq \emptyset;
$$

$$
v_3(Q) = \alpha - \beta \quad \text{for } Q \neq \emptyset.
$$

They define games that are isometrically equivalent to Airport games or a constant game, and we have $v_k(Q) = v_1(Q) + v_2(Q) + v_3(Q)$ for all nonempty $Q \subset P$. The result follows from the linearity of the Shapley values.
For the AREA\textsc{Band} game we just notice that \( v_{\text{ab}}(Q) = v_{\text{air}}(x(Q)) \cdot (\max y(P) - \min y(P)) \) and use again the linearity of Shapley values. (The value \( \max y(P) - \min y(P) \) is constant.) For the PERIMETER\textsc{BoundingBox} game we note that
\[
v_{\text{pbb}}(Q) = 2 \cdot (\max x(Q) - \min x(Q)) + 2 \cdot (\max y(Q) - \min y(Q)),
\]
which means that we need to compute the Shapley values for two LENGTH\textsc{Interval} games. For the PERIMETER\textsc{AnchoredBoundingBox} we use that
\[
v_{\text{pabb}}(Q) = 2 \cdot (\max (x(Q), 0) - \min (x(Q), 0)) + 2 \cdot (\max (y(Q), 0) - \min (y(Q), 0)),
\]
which is a linear combination of a few AIRPORT games. \( \square \)

The perimeter for the anchored bounding box is the same as for the union of anchored rectangles. So this solves the perimeter games for all axis-parallel objects considered in the paper.

\textbf{Remark 2.2} We can now point out the convenience of using the real-RAM model of computation. In the AIRPORT game with \( x_i = i \) for each \( i \in [n] \), we have \( \phi(x_n, v_{\text{air}}) = \sum_{i \in [n]} 1/i \), the \( n \)-th harmonic number. Thus, even for simple games it would become a challenge to carry precise bounds on the number of bits.

\subsection{2.3 Algebraic Computations}

For the axis-parallel problems we will be using the fast Fourier transform to compute convolutions. Assume we are given values \( a_0, \ldots, a_n \) and \( b_0, \ldots, b_n \), and define \( a_i \) and \( b_i \) equal zero for all other indices. Using the fast Fourier transform we can compute the convolutions \( c_k = \sum_{i+j=k} a_i b_j \) for all integers \( k \) in \( O(n \log n) \) operations. (The value is nonzero for at most \( 2n+1 \) indices.) Our use of this is encoded in the following lemma, which provides multipoint evaluation for a special type of rational functions. Note that using the more generic approach of Aronov, Katz, and Moroz [5,24] for multipoint rational functions gives a slightly worse running time.

\textbf{Lemma 2.3} Let \( b_0, \ldots, b_n, \Delta \) be real numbers and consider the rational function
\[
\mathcal{F}(x) = \sum_{t=0}^{n} \frac{b_t}{\Delta + t + x}.
\]
Given an integer \( \ell > -\Delta \), possibly negative, and a positive integer \( m \), we can evaluate \( \mathcal{F}(x) \) at all the integer values \( x = \ell, \ell + 1, \ldots, \ell + m \) in \( O((n + m) \log(n + m)) \) time.

\textbf{Proof} Set \( a_i = 1/((\Delta + \ell + m - i)) \) for all \( i \in \{0, \ldots, m\} \). Note that the assumption \( \Delta + \ell > 0 \) implies that the denominator is always positive. All the other values of \( a_* \) and \( b_* \) are set to 0. Define the convolutions \( c_k = \sum_{i} a_i b_{k-j} \) for all \( k \in \{0, \ldots, m\} \).
and compute them using the fast Fourier transform in \( O((n + m) \log(n + m)) \) time. Then, for each integer \( x \) we have

\[
\mathcal{F}(x) = \sum_{t=0}^{n} \frac{b_t}{\Delta + t + x} = \sum_{t=0}^{n} b_t a_{m+\ell-t-x} = c_{m+\ell-x}.
\]

Thus, by computing the convolution of \( a_n \) and \( b_n \), we get the values \( \mathcal{F}(\ell), \ldots, \mathcal{F}(\ell + m) \).

\[ \Box \]

### 2.4 Permutations

We will have to count permutations with certain properties. The following lemmas encode this.

**Lemma 2.4** Let \( N \) be a set with \( n \) elements. Let \( \{x\}, A, \) and \( B \) be disjoint subsets of \( N \) and set \( \alpha = |A|, \beta = |B| \). There are \( \frac{n!\alpha!\beta!}{(\alpha + \beta + 1)!} \) permutations of \( N \) such that all the elements of \( B \) are before \( x \) and all the elements of \( A \) are after \( x \).

**Proof** Permute the \( \alpha \) elements of \( A \) arbitrarily and put them after \( x \). Similarly, permute the \( \beta \) elements of \( B \) arbitrarily and put them before \( x \). Finally, insert the remaining \( |N \setminus (A \cup B \cup \{x\})| = n - \alpha - \beta - 1 \) elements arbitrarily, one by one. Each permutation that we want to count is constructed exactly once with the procedure, and thus there are

\[
\alpha! \cdot \beta! \cdot ((\alpha + \beta + 2) \cdots n) = \frac{n!\alpha!\beta!}{(\alpha + \beta + 1)!}.
\]

An alternative way to count, which may be easier for some readers, is as follows. Select the positions of \( A \cup B \cup \{x\} \) in the final permutation. There are \( \binom{n}{\alpha+\beta+1} \) ways to do it. Now permute the elements of \( A \) among the last \( \alpha \) positions, the elements of \( B \) among the first \( \beta \) positions, and permute the elements of \( N \setminus (A \cup B \cup \{x\}) \) within the non-selected positions. In total we have

\[
\binom{n}{\alpha+\beta+1} \cdot \alpha! \cdot \beta! \cdot (n - (\alpha + \beta + 1))! = \frac{n!\alpha!\beta!}{(\alpha + \beta + 1)!},
\]

permutations.

\[ \Box \]

**Lemma 2.5** Let \( N \) be a set with \( n \) elements. Let \( A, B, \) and \( C \) be pairwise disjoint subsets of \( N \) with \( \alpha, \beta, \) and \( \gamma \) elements, respectively.

- Consider a fixed element \( a \in A \). There are

\[
n! \cdot \left( \frac{1}{\alpha + \beta} + \frac{1}{\alpha + \gamma} - \frac{1}{\alpha + \beta + \gamma} \right)
\]

permutations of \( N \) such that: (i) \( a \) is the first element of \( A \) and (ii) \( a \) is before all elements of \( B \) or before all elements of \( C \). (Thus, the whole \( B \) or the whole \( C \) is after \( a \).)

\[ \square \]
• Consider a fixed element \( b \in B \). There are
\[
n! \cdot \left( \frac{1}{\alpha + \beta} - \frac{1}{\alpha + \beta + \gamma} \right)
\]
permutations of \( N \) such that: (i) \( b \) is the first element of \( B \), (ii) \( b \) is before all elements of \( A \), and (iii) at least one element of \( C \) is before \( b \).

**Proof** We start showing the first item. Fix an element \( a \in A \). For a set \( X \), disjoint from \( A \), let \( \Pi_X \) be the set of permutations that have \( a \) before all elements of \((A \setminus \{a\}) \cup X\). We can count these permutations as follows. First, permute the elements of \((A \setminus \{a\}) \cup X\) and fix their order. Then insert \( a \) at the front. Finally, insert the elements of \( N \setminus (A \cup X) \) anywhere, one by one. All permutations of \( \Pi_X \) are produced with this procedure exactly once. Therefore, there are
\[
(\alpha + |X| - 1)! \cdot ((\alpha + |X| + 1) \cdot (\alpha + |X| + 2) \cdots n) = n! \cdot \frac{1}{\alpha + |X|}
\]
permutations in \( \Pi_X \). An alternative way to count \( |\Pi_X| \) that may be easier for some readers is to start selecting the positions of \( A \cup X \) in the final permutation. There are \( \binom{n}{\alpha + |X|} \) ways to do it. We place \( a \) in the first selected position, permute the elements of \((A \cup X) \setminus \{a\}\) among the remaining selected positions, and permute the rest of the elements, \( N \setminus (A \cup X) \), among the non-selected positions. In this way we get
\[
|\Pi_X| = \binom{n}{\alpha + |X|} \cdot (\alpha + |X| - 1)! \cdot (n - (\alpha + |X|))! = n! \cdot \frac{1}{\alpha + |X|}
\]
permutations. Using inclusion–exclusion we have
\[
|\Pi_B \cup \Pi_C| = |\Pi_B| + |\Pi_C| - |\Pi_B \cap \Pi_C| = |\Pi_B| + |\Pi_C| - |\Pi_B \cup \Pi_C|
\]
\[
= n! \cdot \left( \frac{1}{\alpha + \beta} + \frac{1}{\alpha + \gamma} - \frac{1}{\alpha + \beta + \gamma} \right).
\]
This finishes the proof of the first item.

Now we prove the second item. Fix an element \( b \in B \). We construct the desired permutations as follows. Select one element \( c \in C \) and place \( c \) before \( b \). Now insert all the elements of \( A \cup B \setminus \{b\} \) after \( b \) in arbitrary order. Then, insert all the elements of \( C \setminus \{c\} \) after \( c \), one by one. Finally, insert the remaining elements in any place, one by one. Each of permutations under consideration is created exactly once by this procedure. Therefore, there are
\[
\gamma \cdot (\alpha + \beta - 1)! \cdot ((\alpha + \beta + 1) \cdots (\alpha + \beta + \gamma - 1)) \cdot ((\alpha + \beta + \gamma + 1) \cdots n)
\]
permutations, which is exactly \( n! \gamma / (\alpha + \beta)(\alpha + \beta + \gamma) \). Then we use that
\[
\frac{1}{\alpha + \beta} - \frac{1}{\alpha + \beta + \gamma} = \frac{\gamma}{(\alpha + \beta)(\alpha + \beta + \gamma)}.
\]
\[ \square \]
3 Convex Hull

In this section we consider the area and the perimeter of the convex hull of the points. We will focus on the area, thus the characteristic function $v_{\text{ch}}$, and at the end we explain the small changes needed to handle the perimeter. Consider a fixed set $P$ of points in the plane. For simplicity we assume that no three points are collinear.

**Lemma 3.1** For each point $p$ of $P$ we can compute $\phi(p, v_{\text{ch}})$ in $O(n^2)$ time.

**Proof** For each $q, q' \in P$, $q \neq q'$, let $H(q, q')$ be the open halfplane containing all points to the left of the directed line from $q$ to $q'$. Define $\text{level}(q, q')$ to be the number of points in $P \cap H(q, q')$.

We can decompose the difference between $\text{CH}(P(\pi, p))$ and $\text{CH}(P(\pi, p) \setminus \{p\})$ into a set $T(\pi, p)$ of triangles (see Fig. 2 (left)), where

$$T(\pi, p) = \{ \Delta pqq' \mid p \in H(q, q'), q \text{ and } q' \text{ appear before } p \text{ in } \pi, \text{ and no points before } p \text{ in } \pi \text{ lie in } H(q, q') \}.$$ 

In other words, $\Delta pqq' \in T(\pi, p)$ if and only if $p \in H(q, q')$, and among the level($q, q'$) + 2 points in $H(q, q') \cup \{q, q'\}$, the two earliest points are $q$ and $q'$, and the third earliest point is $p$. See Fig. 2 (right). (Note that if $\text{CH}(P(\pi, p)) = \text{CH}(P(\pi, p) \setminus \{p\})$, then $T(\pi, p)$ is empty.) For fixed $p, q, q' \in P$ with $p \in H(q, q')$, Lemma 2.4 with $x = p$, $B = \{q, q'\}$, and $A = H(q, q') \cap P \setminus \{p\}$ says that the probability that $\Delta pqq' \in T(\pi, p)$ with respect to a random permutation $\pi$ is exactly

$$\rho(q, q') = \frac{(\text{level}(q, q') - 1)!2!}{(\text{level}(q, q') + 2)!} = \frac{2}{(\text{level}(q, q') + 2)(\text{level}(q, q') + 1)\text{ level}(q, q')}.$$
It follows that the Shapley value \( \phi(p, v_{\text{ch}}) \) of \( p \) is

\[
\mathbb{E}_\pi [\Delta(v_{\text{ch}}, \pi, p)] = \mathbb{E}_\pi \left[ \sum_{\Delta pqq' \in T(\pi, p)} \text{area}(\Delta pqq') \right] = \sum_{q, q' \in P \ (q \neq q')} \text{area}(\Delta pqq') \cdot \rho(q, q'). \tag{1}
\]

\( \phi(p, v_{\text{ch}}) \) for any given \( p \in P \) in \( O(n^2) \) time, if all \( \rho(q, q') \) values have been precomputed.

Each value \( \rho(q, q') \) can be computed from level \( (q, q') \) using \( O(1) \) arithmetic operations. Thus, precomputing \( \rho(q, q') \) requires precomputing all \( O(n^2) \) pairs \( q, q' \). In the dual, this corresponds to computing the levels of all \( O(n^2) \) vertices in an arrangement of \( n \) lines. The arrangement of \( n \) lines can be constructed in \( O(n^2) \) time [6, Chap. 8], and the levels of all vertices can be subsequently generated by traversing the arrangement in \( O(1) \) time per vertex. \( \square \)

Naively applying Lemma 3.1 to all points \( p \in P \) gives \( O(n^3) \) total time. We can speed up the algorithm by a factor of \( n \):

**Theorem 3.2** The Shapley values of the \textsc{AreaConvexHull} game for \( n \) points can be computed in \( O(n^2) \) time.

**Proof** Let \( p = (x, y) \in P \). Observe that for fixed \( q, q' \in P \), \( q \neq q' \), if \( p \in H(q, q') \), then area \( \triangle pqq' \) is a linear function in \( x \) and \( y \) and can thus be written as \( a(q, q')x + b(q, q')y + c(q, q') \). Let \( A(q, q') = a(q, q') \cdot \rho(q, q') \), \( B(q, q') = b(q, q') \cdot \rho(q, q') \), and \( C(q, q') = c(q, q') \cdot \rho(q, q') \). (As noted earlier, we can precompute all the \( \rho(q, q') \) values in \( O(n^2) \) time from the dual arrangement of lines.) By (1),

\[
\phi(p, v_{\text{ch}}) = \sum_{q, q' \in P \ (q \neq q')} (A(q, q')x + B(q, q')y + C(q, q'))
\]

\[
= A(p)x + B(p)y + C(p),
\]

where

\[
A(p) = \sum_{q, q' \in P \ (q \neq q')} A(q, q'), \quad B(p) = \sum_{q, q' \in P \ (q \neq q')} B(q, q'),
\]

\[
C(p) = \sum_{q, q' \in P \ (q \neq q')} C(q, q').
\]

We describe how to compute \( A(p), B(p), \) and \( C(p) \) for all \( p \in P \) in \( O(n^2) \) total time. Afterwards, we can compute \( \phi(p, v_{\text{ch}}) \) for all \( p \in P \) in \( O(n) \) additional time. The problem can be reduced to three instances of the following:
Given a set $P$ of $n$ points in the plane, and given $O(n^2)$ lines each through two points of $P$ and each assigned a weight, compute for all $p \in P$ the sum of the weights of all lines below $p$ (or similarly all lines above $p$).

In the dual, the problem becomes:

Given a set $L$ of $n$ lines in the plane, and given $O(n^2)$ vertices in the arrangement, each assigned a weight, compute for all lines $\ell \in L$ the sum $S(\ell)$ of the weights of all vertices below $\ell$.

To solve this problem, we could use known data structures for halfplane range searching, but a direct solution is simpler. First construct the arrangement in $O(n^2)$ time for all $\ell$.

We adapt the algorithm for area to handle the perimeter. Let $\pi$ be a random permutation of the vertices. Thus, for fixed $p$, $E(\pi, p)$ is $O(n^2)$ time, since these values correspond to prefix or suffix sums over the sequence of weights of the $O(n)$ vertices on the line $p$. The total time for all lines $p \in L$ is $O(n^2)$.

Afterwards, for each $\ell \in L$, we can compute $S(\ell)$ in $O(n)$ time by summing $S(\ell, \ell')$ over all $\ell' \in L \setminus \{\ell\}$ and dividing by 2 (since each vertex is counted twice). The total time for all $\ell \in L$ is $O(n^2)$.

Theorem 3.3 The Shapley values of the PERIMETERCONVEXHULL game for $n$ points can be computed in $O(n^2)$ time.

Proof We adapt the algorithm for area to handle the perimeter. Let $E(\pi, p)$ be the set of directed edges of $CH(P(\pi, p))$ that are not in $CH(P(\pi, p) \setminus \{p\})$, where edges are oriented in clockwise order. (If $CH(P(\pi, p) = CH(P(\pi, p) \setminus \{p\})$, then $E(\pi, p)$ is empty; otherwise, it has exactly two edges because of general position.) Then $(q, p) \in E(\pi, p)$ if and only if $q$ appears before $p$ in $\pi$ and no points before $p$ in $\pi$ lie in $H(q, p)$; in other words, among the level $(q, p) + 2$ points in $H(q, p) \cup \{p\}$, the earliest point is $q$ and the second earliest point is $p$. We can use Lemma 2.4 to bound the probability of this event. Thus, for fixed $p, q \in P$, the probability that $(q, p) \in E(\pi, p)$ (with respect to a random permutation $\pi$) is exactly

$$\rho'(q, p) = \frac{1}{(\text{level}(q, p) + 2)(\text{level}(q, p) + 1)}.$$ 

Similarly, the probability that $(p, q) \in E(\pi, p)$ is exactly $\rho'(p, q)$. It follows that the expected total length of the edges in $E(\pi, p)$ is

$$\phi^+(p) = \sum_{q \in P \setminus \{p\}} \|p - q\| \cdot (\rho'(q, p) + \rho'(p, q)).$$ 

On the other hand, a modification of the proof of Lemma 3.1 shows that the expected total length of the edges in $CH(P(\pi, p) \setminus \{p\})$ that are not in $CH(P(\pi, p))$ is

$$\phi^-(p) = \sum_{q, q' \in P \ (q \neq q') \text{ with } p \in H(q, q')} \|q - q'\| \cdot \rho(q, q').$$

\(\Box\) Springer
The Shapley value of each point \( p \in P \) is \( \phi(p, v_{\text{pch}}) = \phi^+(p) - \phi^-(p) \) because of linearity of expectation. We can compute \( \phi^+(p) \) naively in \( O(n) \) time for each \( p \in P \); the total time is \( O(n^2) \). We can compute \( \phi^-(p) \) for all \( p \in P \) as in the proof in Theorem 3.2, in \( O(n^2) \) total time (in fact, the algorithm is a little simpler, since linear functions are not required).

\[ \square \]

4 Minimum Enclosing Disk

In this section we consider the area and the perimeter of the smallest enclosing disk of the points. Recall that \( v_{\text{ed}} \) is the characteristic function. For simplicity, we assume general position in the following way: no four points are cocircular and circles through three input points do not have a diameter defined by two input points. We use \( P \) for the set of points. First we explain how to compute the Shapley values for the area of the minimum enclosing disk.

Lemma 4.1 For each point \( p \) of \( P \) we can compute \( \phi(p, v_{\text{ed}}) \) in \( O(n^3) \) time.

Proof For each subset of points \( Q \subset P \), let \( X(Q) \) be the subset of points of \( Q \) that lie on the boundary of \( \text{med}(Q) \). We now recall some well known properties; see for example [34] or [6, Sect. 4.7]. The disk \( \text{med}(Q) \) is unique and \( \text{med}(X(Q)) = \text{med}(Q) \). If a point \( p \) is outside \( \text{med}(Q) \), then \( p \) is on the boundary of \( \text{med}(Q \cup \{p\}) \), that is, \( p \in X(Q \cup \{p\}) \). Because of our assumption on general position, \( X(Q) \) has at most three points. When \( X(Q) \) has two points, then they define a diameter of \( \text{med}(Q) \). We have \( |X(Q)| \leq 1 \) only when \( |Q| \leq 1 \). See Fig. 3.

A subset \( B \subset P \) of size at most 3 such that \( X(B) = B \) is called a basis. For a basis \( B \), define \( \text{out}(B) = P \setminus \text{med}(B) \) and \( \text{level}(B) = |\text{out}(B)| \). Note that \( \text{level}(B) \) is the number of points of \( P \) strictly outside \( \text{med}(B) \).

For a basis \( B \) and a point \( p \notin \text{med}(B) \), we have \( X(P(\pi, p) \setminus \{p\}) = B \) if and only if all points of \( B \) appear before \( p \) in \( \pi \), and no points before \( p \) in \( \pi \) belong to \( \text{out}(B) \). In other words, among the \( \text{level}(B) + |B| \) points in \( \text{out}(B) \cup B \), the \( |B| \) earliest points are the points of \( B \) and the next earliest point is \( p \). See Fig. 4 (left). Lemma 2.4 implies that, for a fixed \( B \) and a fixed \( p \), the probability that \( X(P(\pi, p) \setminus \{p\}) = B \) (with

![Fig. 3](https://example.com/figure3.png) The minimum enclosing disk for two sets \( Q \) of points. Left: \( X(Q) \) has three elements. Right: \( X(Q) \) has two elements.
Let $I(\pi, p)$ be the indicator variable that is 1 if $p \notin \text{med}(P(\pi, p) \setminus \{p\})$, and 0 otherwise. It follows that the expected value of area $\text{med}(P(\pi, p) \setminus \{p\}) \cdot I(\pi, p)$ is

$$\phi^- (p) = \sum_{\text{basis } B \text{ with } p \notin \text{med}(B)} \text{area}(\text{med}(B)) \cdot \rho(B). \quad (2)$$

On the other hand, for a basis $B$ and a point $p \in B$, we have $X(P(\pi, p)) = B$ if and only if all points of $B \setminus \{p\}$ appear before $p$ in $\pi$, and no points before $p$ in $\pi$ lie in $\text{out}(B)$. In other words, among the level($B$) + |$B$| points in $\text{out}(B) \cup B$, the |$B$| − 1 earliest points are the points of $B \setminus \{p\}$ and the next earliest point is $p$. See Fig. 4 (right). We can bound the probability of this event using Lemma 2.4. Thus, for a fixed $B$ and $p \in B$, the probability that $X(P(\pi, p)) = B$ (with respect to a random permutation $\pi$) is exactly

$$\rho'(B) = \frac{(|B| - 1)!}{(\text{level}(B) + |B|)!}.$$ 

It follows that the expected value of area $\text{med}(P(\pi, p))) \cdot I(\pi, p)$ is

$$\phi^+(p) = \sum_{\text{basis } B \text{ with } p \in B} \text{area}(\text{med}(B)) \cdot \rho'(B). \quad (3)$$

By linearity of expectation, the Shapley value of $p$ is $\phi(v_{\text{med}}, p) = \phi^+(p) - \phi^-(p)$. From the formulas (2) and (3), we can compute $\phi^+(p)$ in $O(n^2)$ time and $\phi^-(p)$ in $O(n^3)$ time for any given $p \in P$ (since $|B| \leq 3$), if all $\rho(B)$ and $\rho'(B)$ values have been precomputed.

Fig. 4 Left: in order for the shown disk to be $\text{med}(P(\pi, p) \setminus \{p\})$, the points $q, q', q''$ must appear before $p$, which in turn must appear before all other points outside the disk. Right: in order for the shown disk to be $\text{med}(P(\pi, p))$, the points $q, q'$ must appear before $p$, which in turn must appear before all points outside the disk.
Precomputing $\rho(B)$ and $\rho'(B)$ requires precomputing $\text{level}(B)$ for all bases $B$. Precomputing $\text{level}(B)$ for bases $B$ of size 2 can be naively done in $O(n^2 \cdot n) = O(n^3)$ time, so it suffices to focus on bases $B$ of size 3. By a standard lifting transformation (mapping point $(a, b)$ to the plane $z = 2ax - 2by + a^2 + b^2 = 0$ in three dimensions), the problem reduces to computing the level of $O(n^3)$ vertices in an arrangement of $n$ planes in three dimensions. See [22, Sect. 5.7] for a discussion on the transformation. The arrangement of $n$ planes can be constructed in $O(n^3)$ time [11,12], and the levels of all vertices can be subsequently generated by traversing the arrangement in $O(1)$ time per vertex. Note that for the perimeter we just use the perimeter of $\text{med}(B)$ instead of the area in the computations.

Naively applying Lemma 4.1 to all points $p \in P$ gives $O(n^4)$ total time. We can speed up the algorithm by a factor of $n$:

**Theorem 4.2** The Shapley values of the $\text{AreaEnclosingDisk}$ and $\text{PerimeterEnclosingDisk}$ games for $n$ points can be computed in $O(n^3)$ time.

**Proof** We already know how to compute $\phi^+(p)$ in $O(n^2)$ time for each $p \in P$, and thus in $O(n^3)$ total time. It suffices to focus on computing $\phi^-(p)$. In the formula (2), terms for bases of size 2 can be handled in $O(n^2)$ time for each $p$; so it suffices to focus on bases of size 3. The problem can be formulated as follows:

Given a set $P$ of $n$ points in the plane, and given $O(n^3)$ disks each with three points of $P$ on the boundary and each assigned a weight, compute for all $p \in P$ the sum of the weights of all disks not containing $p$.

By the standard lifting transformation, the problem reduces to the following:

Given a set $H$ of $n$ planes in three dimensions, and given $O(n^3)$ vertices in the arrangement, each assigned a weight, compute for all $h \in H$ the sum $S(h)$ of the weights of all vertices below $h$.

To solve this problem, we could use known data structures for halfspace range searching, but an approach as in the proof of Theorem 3.2 is simpler. First construct the arrangement in $O(n^3)$ time. Given $h, h', h'' \in H$, define $S(h, h', h'')$ to be the sum of the weights of all vertices on the line $h' \cap h''$ that are below $h$. For a fixed pair of planes $h', h'' \in H$, we can precompute $S(h, h', h'')$ for all $h \in H \setminus \{h', h''\}$ in $O(n)$ time, since these values correspond to prefix or suffix sums over the sequence of weights of the $O(n)$ vertices on the line $h' \cap h''$. The total time for all pairs $h', h'' \in H$ is $O(n^3)$.

Afterwards, for each $h \in H$, we can compute $S(h)$ in $O(n^2)$ time by summing $S(h, h', h'')$ over all pairs $h', h'' \in H \setminus \{h\}$ and dividing by 3 (since each vertex is counted thrice). The total time for all $h \in H$ is $O(n^3)$. □

**5 Union of Anchored Rectangles**

In this section we consider the $\text{AreaAnchoredRectangles}$ game defined by the characteristic function $v_{ar}$. It is easy to see that one can focus on the special case
where all the points are in a quadrant; see Fig. 5. Our discussion will focus on the case where the points of $P$ are on the positive quadrant of the plane.

Consider a fixed set $P$ of points in the positive quadrant. In the notation we will drop the dependency on $P$. For simplicity, we assume general position: no two points have the same $x$- or $y$-coordinate. We first introduce some notation that will be used in this section and in Sect. 6.

5.1 Notation for Axis-Parallel Problems

For each point $q$ of the plane, we use the “cardinal directions” to define subsets of points in quadrants with apex at $q$:

$$
\begin{align*}
\text{NW}(q) &= \{ p \in P \mid x(p) \leq x(q), y(p) \geq y(q) \}, \\
\text{NE}(q) &= \{ p \in P \mid x(p) \geq x(q), y(p) \geq y(q) \}, \\
\text{SE}(q) &= \{ p \in P \mid x(p) \geq x(q), y(p) \leq y(q) \}.
\end{align*}
$$

We use lowercase to denote their cardinality: $\text{nw}(q) = |\text{NW}(q)|$, $\text{ne}(q) = |\text{NE}(q)|$, and $\text{se}(q) = |\text{SE}(q)|$. See Fig. 6 (left).

Let $x_1 < \ldots < x_n$ denote the $x$-coordinates of the points of $P$, and let $y_1 < \ldots < y_n$ be their $y$-coordinates. We also set $x_0 = 0$ and $y_0 = 0$. For each $i, j \in [n]$ we use $w_i = x_i - x_{i-1}$ (for width) and $h_j = y_j - y_{j-1}$ (for height).

Let $L$ be the set of horizontal and vertical lines that contain some point of $P$. We add to $L$ both axes. Thus, $L$ has $2n + 2$ lines. The lines in $L$ break the plane into 2-dimensional cells (rectangles), usually called the arrangement and denoted by

Fig. 6 Notation for axis-parallel problems. Left: the quadrants to define NW$(q)$, NE$(q)$ and SE$(q)$.

Right: cells of $A$.
\(A = A(L)\). More precisely, a (2-dimensional) cell \(c\) of \(A\) is a maximal connected component in the plane after the removal of the points on lines in \(L\). Formally, the (2-dimensional) cells are open sets whose closure is a rectangle, possibly unbounded in some direction. We are only interested in the bounded cells, and with a slight abuse of notation, we use \(A\) for the set of bounded cells. We denote by \(c_{i,j}\) the cell between the vertical lines \(x = x_{i-1}\) and \(x = x_i\) and the horizontal lines \(y = y_{j-1}\) and \(y = y_j\). Note that \(c_{i,j}\) is the interior of a rectangle with width \(w_i\) and height \(h_j\). See Fig. 6 (right).

Since \(\text{NE}(q)\) is constant over each 2-dimensional cell \(c\) of \(A\), we can define \(\text{NE}(c)\), for each cell \(c \in A\). The same holds for \(\text{NW}(c)\) and \(\text{SE}(c)\) and their cardinalities, \(\text{ne}(c), \text{nw}(c)\) and \(\text{se}(c)\). See Fig. 7.

A block \(B\) is a set of cells \(B = B(i_0, i_1, j_0, j_1) = \{c_{i,j} \mid i_0 \leq i \leq i_1, j_0 \leq j \leq j_1\}\) for some indices \(i_0, i_1, j_0, j_1\), with \(1 \leq i_0 \leq i_1 \leq n\) and \(1 \leq j_0 \leq j_1 \leq n\). The number of columns and rows in \(B\) is \(i_1 - i_0 + 1 + j_1 - j_0 + 1 = O(i_1 - i_0 + j_1 - j_0)\). A block \(B\) is empty if no point of \(P\) is on the boundary of four cells of \(B\). Equivalently, \(B\) is empty if no point of \(P\) is in the interior of the union of the closure of the cells in \(B\). See Fig. 8 for an example.

We will be using maximal rows and columns within a block \(B\) to compute some partial information. Thus, for each block \(B\) and each index \(i\), we define the vertical slab \(V(i, B) = \{c_{i,j} \mid 1 \leq j \leq n, c_{i,j} \in B\}\). Similarly, for each block \(B\) and each index \(j\), we define the horizontal slab \(H(j, B) = \{c_{i,j} \mid 1 \leq i \leq n, c_{i,j} \in B\}\). Such slabs are meaningful only for indices within the range that defines the block. We call them the slabs within \(B\).

### 5.2 Interpreting Shapley Values Geometrically

First, we reduce the problem of computing Shapley values to a neat geometric problem. See Fig. 7 for the relevant counters considered in the following result.
Lemma 5.1 If $P$ is in the positive quadrant, then for each $p \in P$ we have

$$\phi(p, v_{\text{ar}}) = \sum_{c \in A \atop c \subset R_p} \frac{\text{area}(c)}{\text{ne}(c)}.$$  

Proof Each cell $c$ of the arrangement $A$ defines a game for the set $P$ of players with characteristic function

$$v_c(Q) = \begin{cases} 
\text{area}(c) & \text{if } c \subset R_p \text{ for some } p \in Q, \\
0 & \text{otherwise.}
\end{cases}$$

To analyze the Shapley values of the game defined by $v_c$, note that, because of symmetry, each point of the set $\text{NE}(c) = \{p \in P \mid c \subset R_p\}$ has the same probability of being the first point from $\text{NE}(c)$ in a random permutation. Therefore

$$\phi(p, v_c) = \begin{cases} 
\frac{\text{area}(c)}{\text{ne}(c)} & \text{if } c \subset R_p, \\
0 & \text{otherwise.}
\end{cases}$$

On the other hand,

$$v_{\text{ar}}(Q) = \sum_{c \in A} v_c(Q) \quad \text{for all } Q \subset P,$$
and because of linearity of Shapley values we get

$$\phi(p, v_{\text{ar}}) = \sum_{c \in \mathcal{A}} \phi(p, v_c) = \sum_{c \in \mathcal{A}, c \subseteq R_p} \sum_{c \in R_p} \frac{\text{area}(c)}{\text{ne}(c)}. \quad \square$$

Using standard tools in computational geometry we can compute the values $\phi(p, v_{\text{ar}})$ for all $p \in P$ in near-quadratic time. Here is an overview of the approach. An explicit computation of $\mathcal{A}$ takes quadratic time in the worst case. Using standard data structures for orthogonal range searching, see [35] or [6, Chap. 5], we can then compute $\text{ne}(c)$ for each cell $c \in \mathcal{A}$. Finally, replacing each cell $c$ by a point $q_c \in c$ with weight $w_c = \text{area}(c)/\text{ne}(c)$, we can reduce the problem of computing $\phi(p, v_{\text{ar}})$ to the problem of computing $\sum_{q_c \in R_p} w_c$, which is again an orthogonal range query. An alternative is to use dynamic programming across the cells of $\mathcal{A}$ to compute first $\text{ne}(c)$ and then partial sums of the weights $w_c$.

Our objective in the following sections is to improve this result using the correlation between adjacent cells. It could seem at first glance that segment trees [6, Sect. 10.3] may be useful. We did not see how to work out the details of this. Segment trees would be useful if the weights would be proportional to $\text{ne}(c)$, but here they are inversely proportional to $\text{ne}(c)$.

### 5.3 Handling Empty Blocks

In the following we assume that we have preprocessed $P$ in $O(n \log n)$ time such that $\text{ne}(q)$ can be computed in $O(\log n)$ time for each point $q$ given at query time [35]. This is a standard range counting for orthogonal ranges and the preprocessing has to be done only once.

For each subset $C$ of cells of $\mathcal{A}$, we define

$$\sigma(C) = \sum_{c \in C} \frac{\text{area}(c)}{\text{ne}(c)}.$$ (We will only consider sets $C$ of cells with $\text{ne}(c) > 0$ for all $c \in C$.) Note that we want to compute $\sigma(\cdot)$ for the sets of cells contained in the rectangles $R_p$ for all $p \in P$; see Lemma 5.1. When a block $B$ is empty, then we can use multipoint evaluation to obtain the partial sums $\sigma(\cdot)$ for each vertical and horizontal slab within the block.

**Lemma 5.2** Let $B$ be an empty block with at most $k$ columns and at most $k$ rows. We can compute in $O(k \log n)$ time the values $\sigma(C)$ for all vertical and horizontal slabs $C$ within $B$.

**Proof** Assume that $B$ is the block $B(i_0, i_1, j_0, j_1)$. We only explain how to compute the values $\sigma(V(i, B))$ for all $i_0 \leq i \leq i_1$. The computation for the horizontal slabs $\sigma(H(j_0, B))$, \ldots, $\sigma(H(j_1, B))$ is similar.

We look into the first vertical slab $V(i_0, B)$ and make groups of cells depending on their value $\text{ne}(\cdot)$. More precisely, for each $\ell$ we define $J(\ell) = \{ j \mid j_0 \leq j \leq j_1,$
ne(c_{i_0,j}) = \ell I. Let \ell_0 and \ell_1 be the minimum and the maximum \ell such that J(\ell) \neq 0, respectively. We set up the following rational function with variable x:

\[ F(x) = \sum_{\ell=\ell_0}^{\ell_1} \frac{\sum_{j \in J(\ell)} h_j}{\ell + x}. \]

Setting \( t = \ell - \ell_0 \), \( b_t = \sum_{j \in J(\ell_0+t)} h_j \), and \( \Delta = \ell_0 \), we have

\[ F(x) = \sum_{t=0}^{\ell_1-\ell_0} \frac{b_t}{\Delta + t + x}. \]

Thus, this is a rational function of the shape considered in Lemma 2.3 with \( \ell_1 - \ell_0 \leq j_1 - j_0 + 1 \leq k \) terms. The coefficients can be computed in \( O(k \log n) \) time because we only need the values \( h_j \) and \( ne(c_{i_0,j}) \) for each \( j \). These latter values \( ne(\cdot) \) are obtained from range counting queries. Note that

\[
w_{i_0} \cdot F(0) = w_{i_0} \cdot \sum_{\ell=\ell_0}^{\ell_1} \frac{\sum_{j \in J(\ell)} h_j}{\ell} = \sum_{j=j_0}^{j_1} \frac{w_{i_0} h_j}{ne(c_{i_0,j})} = \sum_{j=j_0}^{j_1} \frac{\text{area}(c_{i_0,j})}{\text{area}(c_{i_0,j})} = \sigma(V(i_0, B)).
\]

A similar statement holds for all the other vertical slabs within \( B \). We make the statement precise in the following.

Consider two consecutive vertical slabs \( V(i, B) \) and \( V(i+1, B) \) within the block \( B \). Because the block \( B \) is empty, the difference \( ne(c_{i+1,j}) - ne(c_{i,j}) \) is independent of \( j \). See Fig. 9. It follows that, for each index \( i \) with \( i_0 \leq i \leq i_1 \), there is an integer \( \delta_i \) such that \( ne(c_{i,j}) = ne(c_{i_0,j}) + \delta_i \) for all \( j \) with \( j_0 \leq j \leq j_1 \). Moreover, for each \( i \) with \( i_0 \leq i \leq i_1 \) and each \( \ell \) with \( \ell_0 \leq \ell \leq \ell_1 \), the value of \( ne(c_{i,j}) \) is constant over all \( j \in J(\ell) \). Therefore, for each \( j \in J(\ell) \) we have \( ne(c_{i,j}) = \ell + \delta_i \). Each value \( \delta_i \) can be obtained using that \( \delta_i = ne(c_{i_0,j}) - ne(c_{i_0,j_0}) \). This means that the values \( \delta_{i_0}, \ldots, \delta_{i_1} \) belong to \( \{-k, \ldots, +k\} \) and can be obtained in \( O(k \log n) \) time.

Now we note that, for each index \( i \) with \( i_0 \leq i \leq i_1 \), we have

\[
w_i \cdot F(\delta_i) = w_i \cdot \sum_{\ell=\ell_0}^{\ell_1} \frac{\sum_{j \in J(\ell)} h_j}{\ell + \delta_i} = \sum_{j=j_0}^{j_1} \frac{w_i h_j}{ne(c_{i,j})} = \sum_{j=j_0}^{j_1} \frac{\text{area}(c_{i,j})}{\text{area}(c_{i,j})} = \sigma(V(i, B)).
\]
Fig. 9 Changes in the values of \( \text{ne}(\cdot) \) when passing from a vertical slab to the next one. The rightmost transition shows the need to deal with empty blocks for our argument.

We use Lemma 2.3 to evaluate the \( i_1 - i_0 + 1 \leq k \) values \( (\delta_i) \), where \( i_0 \leq i \leq i_1 \), in \( O(k \log k) = O(k \log n) \) time. After this, we get each value \( \sigma(V(i, B)) = w_i \cdot \mathcal{F}(\delta_i) \) in constant time. \( \square \)

5.4 Chains

In this section we consider the case where the points are a chain. As discussed before, it is enough to consider that \( P \) is in the positive quadrant. After sorting, we can assume without loss of generality that the points of \( P \) are indexed so that \( 0 < x(p_1) < \ldots < x(p_n) \). We start with the easier case: increasing chains. The problem is actually an AIRPORT game in disguise.

Lemma 5.3 If \( P \) is an increasing chain in the positive quadrant, then we can compute the Shapley values of the AREAANCHOREDRECTANGLES game in \( O(n \log n) \).

Proof Set the point \( p_0 \) to be the origin \( o \). Note that, for each \( i \in [n] \) and each cell \( c \in \mathcal{A} \) contained \( R_{p_i} \setminus R_{p_{i-1}} \) we have \( \text{ne}(c) = n - i + 1 \). For each \( i \in [n] \), define the value

\[
z_i = \frac{\text{area}(R_{p_i}) - \text{area}(R_{p_{i-1}})}{n - i + 1}.
\]
Thus, $z_i$ is the area of the region $R_{p_i} \setminus R_{p_{i-1}}$ divided by $\text{ne}(c)$, for some $c \subset R_{p_i} \setminus R_{p_{i-1}}$. See Fig. 10. Because of Lemma 5.1, we have for each $i \in [n]$,

$$\phi(p_i, v_{\text{sr}}) = \sum_{c \in A, c \subset R_{p_i}} \frac{\text{area}(c)}{\text{ne}(c)} = \sum_{j \leq i} z_j.$$ 

Therefore $\phi(p_1, v_{\text{sr}}) = z_1$ and, for each $i \in [n] \setminus \{1\}$, we have $\phi(p_i, v_{\text{sr}}) = z_i + \phi(p_{i-1}, v_{\text{sr}})$. The result follows. (Actually this game is just the 1-dimensional AIRPORT game if each point $p_i$ is represented on the real line with the point with $x$-coordinate $\text{area}(R_{p_i})$.)

It remains the more interesting case, when the chain is decreasing. See Fig. 11 for an example. It is straightforward to see that, if $i + j > n + 1$, then $\text{ne}(c_{i,j}) = 0$, and if $i + j \leq n + 1$, we have $\text{ne}(c_{i,j}) = n + 2 - i - j$. So in this case we do not really need data structures to obtain $\text{ne}(c_{i,j})$ efficiently. (The proof of Lemma 5.2 can be slightly simplified for this case because $J(\ell)$ contains a single element due to the special structure of the values $\text{ne}(c)$. See Fig. 12.)

We use a divide-and-conquer paradigm considering certain empty blocks defined by two indices $\ell$ and $r$, where $\ell < r$. Since the indexing of rows is not the most convenient in this case, it is better to introduce the notation $B_{\ell,r}$ for the block $B(\ell + 1, m, n - r + 2, n - m + 1)$, where $m = m(\ell, r) = \lfloor (\ell + r)/2 \rfloor$. Initially we will have $\ell = 0$ and $r = n + 1$, which means that we start with the block $B_{0,n+1} = B(1, m, 1, m)$. See Fig. 13 for the base case, and Fig. 14 for a generic case. For each block $B_{\ell,r}$, we will compute $\sigma(C)$ for each slab $C$ within $B_{\ell,r}$. The blocks can be used to split the problem into two smaller subchains, and the interaction between them in encoded by the slabs within the block. This approach leads to the following result.
Fig. 11 Decreasing chain. Left: Each cell $c_{i,j}$ is marked with the multiplicative weight $1/\text{ne}(c)$ for its area. Right: the cells whose contribution we have to add for the point $p_4$

Fig. 12 Values $1/\text{ne}(c)$ in a decreasing chain

Lemma 5.4 If $P$ is a decreasing chain with $n$ points in the positive quadrant, then we can compute the Shapley values of the $\text{AREAANCHOREDRECTANGLES}$ game in $O(n \log^2 n)$ time.

Proof We assume for simplicity that $n + 1$ is a power of 2. Otherwise we replace each appearance of an index larger than $n + 1$ by $n + 1$. Let $\mathcal{I}$ be the set of pairs $(\ell, r)$ defined by $\ell = \alpha 2^\beta$ and $r = (\alpha + 1) 2^\beta \leq n + 1$, where $\alpha$ and $\beta$ are non-negative integers. For each $(\ell, r) \in \mathcal{I}$, we compute the values $\sigma(V(i, B_{\ell,r}))$ for all relevant indices $i$ and the values $\sigma(H(j, B_{\ell,r}))$ for all relevant $j$ using Lemma 5.2. This takes $O((r - \ell) \log n)$ time. The pairs $(\ell, r)$ of $\mathcal{I}$ can also be interpreted as intervals. Since intervals of $\mathcal{I}$ with the same length are disjoint and there are $O(\log n)$ different possible lengths of
For each $(\ell, r) \in \mathcal{I}$ we compute some additional prefix sums of columns and of rows, as follows. Assume that $B_{\ell, r} = B(i_j, j_0, j_1)$. For each $i$ with $i_0 \leq i \leq i_1$, we define $V_{\leq}(i, B_{\ell, r}) = \bigcup_{i_0 \leq i_0' \leq i} V(i_0', B_{\ell, r})$. For each $j$ with $j_0 \leq j \leq j_1$, we define $H_{\leq}(j, B_{\ell, r}) = \bigcup_{j_0 \leq j_0' < j} H(j_0', B_{\ell, r})$. Using that

$$\sigma(V_{\leq}(i, B_{\ell, r})) = \sigma(V_{\leq}(i - 1, B_{\ell, r})) + \sigma(V(i, B_{\ell, r})) \quad \text{for all } i \text{ with } i_0 < i \leq i_1,$$

and a similar relation for $\sigma(H_{\leq}(j, B_{\ell, r}))$, we can compute the values $\sigma(V_{\leq}(i, B_{\ell, r}))$ for all $i$ and the values $\sigma(H_{\leq}(j, B_{\ell, r}))$ for all $j$ in $O(\ell - r)$ time. In total, we spend an additional $O(n \log n)$ time over all pairs $(\ell, r)$ of $\mathcal{I}$.

Consider now a point $p_a$ of $P$. We can express the rectangle $R_{p_a}$ as the union of $O(\log n)$ rectangles for which we have computed the relevant partial sums. See Fig. 15 for the intuition. More precisely, let $\mathcal{I}(a)$ be the pairs $(\ell, r)$ of $\mathcal{I}$ with $\ell < a < r$. For
Fig. 15 Expressing $R_{pa}$ as the union of $O(\log n)$ rectangles of the form $X(a, B_{\ell,r})$. In this example, $3n/16 < a < n/4$.

each pair $(\ell, r)$ of $I(a)$, we set

$$X(a, B_{\ell,r}) = \begin{cases} V_{\leq}(a, B_{\ell,r}) & \text{if } a \leq m(\ell,r), \\ H_{\leq}(n-a+1, B_{\ell,r}) & \text{if } a > m(\ell,r). \end{cases}$$

Then $R_{pa}$ is the (the closure of the) disjoint union of the cells in $X(a, B_{\ell,r})$, where $(\ell,r)$ iterates over $I(a)$. It follows that

$$\sum_{(\ell,r)\in I(a)} \sigma(X(a, B_{\ell,r})) = \sigma(\{c \in A \mid c \subset R_{pa}\}) = \sum_{c\in A \atop c \subset R_{pa}} \frac{\text{area}(c)}{\text{ne}(c)} = \phi(p_a, v_{ar}),$$

where in the last equality we have used Lemma 5.1. For each point $p_a$ of $P$, we compute the set $I(a)$ and then the sum $\sum_{(\ell,r)\in I(a)} \sigma(X(a, B_{\ell,r})) = \phi(p_a, v_{ar})$ in $O(\log n)$ time. Over all points of $P$ we spend $O(n \log n)$ time in this last computation.

When the point set is a chain over different quadrants, we can reduce it to a few problems over the positive quadrant and obtain the following.

**Theorem 5.5** If $P$ is a chain with $n$ points, then we can compute the Shapley values of the AreaAnchoredRectangles game in $O(n \log^2 n)$ time.

**Proof** If the point set $P$ is a chain, then we get a chain in each quadrant. As pointed out before (recall Fig. 5), we can treat each quadrant independently. Using reflections, we can transform the instance in any quadrant to the positive quadrant. (Note this may change the increasing/decreasing character of the chains. For example, an increasing chain in the northwest quadrant gets reflected into a decreasing chain in the positive quadrant. When the positive quadrant is empty, we can apply Lemma 4.1 to compute the Shapley value of $R_{pa}$ in $O(n \log^2 n)$ time.)
quadrant.) Because of Lemmas 5.3 and 5.4, we can compute the Shapley values of the \textsc{AreaAnchoredRectangles} game for each quadrant in $O(n \log^2 n)$ time. The result follows.

5.5 General Point Sets

We consider now the general case; thus the points do not form a chain. See Fig. 7 for an example. Like before, we restrict the discussion to the case where $P$ is in the positive quadrant. In this scenario we consider horizontal bands. A horizontal band $B$ is the block between two horizontal lines. Thus, $B = \{c_{i,j} \mid j_0 \leq j \leq j_1\}$ for some indices $1 \leq j_0 \leq j_1 \leq n$. See Fig. 16. We keep using the notation introduced for blocks. Thus, for each $i \in [n]$, let $V(i, B)$ be the vertical slab with the cells $c_{i,j} \in B$. Let $P_B$ be the points of $P$ that are the top-right corner of some cell of $B$. We use $k_B = |P_B|$. Because of our assumption on general position, $k_B = j_1 - j_0 + 1$ and thus $k_B$ is precisely the number of horizontal slabs in the band $B$. Furthermore, for each point $p \in P_B$ we define the rectangle $R(p, B)$ as the cells of $B$ to the left and bottom of $p$. Formally $R(p, B) = \{c \in B \mid c \subset R_p\}$.

**Lemma 5.6** For a band $B$ with $k_B$ rows we can compute in $O(((k_B)^2 + n) \log n)$ time $\sigma(V(i, B))$, for all $i \in [n]$, and $\sigma(R(p, B))$, for all $p \in P_B$.

**Proof** Assume that $B$ is defined by the row indices $j_0 \leq j_1$. Let $q_1, \ldots, q_{k_B}$ be the points of $P_B$ sorted by increasing $x$ coordinate. Take also $q_0$ as a point on the $y$-axis and $q_{k_B+1}$ as a point on the right boundary. For each $t$ denote by $m_t$ the number of vertical slabs between the vertical lines through $q_{t-1}$ and $q_t$. See Fig. 16.

We divide the band $B$ into blocks $B_1, B_2, \ldots$ using the vertical lines through the points of $P_B$. We get $k_B + 1$ blocks (or $k_B$ if the rightmost point of $P$ belongs to $P_B$), and each of them is empty by construction. Since the block $B_t$ has $m_t$ vertical slabs and $k_B$ horizontal slabs we can compute $\sigma(V(\cdot, B_t))$ and $\sigma(H(\cdot, B_t))$ for all the slabs within $B_t$ in $O((k_B + m_t) \log n)$ time using Lemma 5.2. Using that the $O(k_B)$
blocks $B_1, B_2, \ldots$ are pairwise disjoint, which means that $\sum_i m_i = n$, we conclude that in $O((k_B^2 + n) \log n)$ time we can compute all the values $\sigma(V(\cdot, B_i))$ and $\sigma(H(\cdot, B_i))$ for all slabs within all blocks $B_i$.

Since for each vertical slab $V(i, B)$ of $B$ there is one block $B_t(i)$ that covers it, we then have $\sigma(V(i, B)) = \sigma(V(i, B_t(i)))$ for such index $t(i)$. This means that we have already computed the values $\sigma(V(i, B))$ for all $i$.

Now we explain the computation of $\sigma(R(p, B))$ for all $p \in P_B$. Note that within each block $B_i$ we have computed $\sigma(H(j, B_i))$ for all $j$. With this information we can compute the values $\sigma(R(p, B))$. Namely, for each block $B_i$ and each $j$ with $j_0 \leq j \leq j_1$, we define

$$H_{\leq}(j, B_i)) = \bigcup_{j_0 \leq j' \leq j} H(j', B_{t'})$$

$$S_{\leq}(j, B_i)) = \bigcup_{j_0 \leq j' \leq j} \bigcup_{1 \leq t' \leq i} H(j', B_{t'})$$

Using that

\begin{align*}
\forall t \text{ with } 1 \leq t \leq k_B + 1: & \quad \sigma(H_{\leq}(j_0, B_i)) = \sigma(H(j_0, B_i)), \\
\forall j, t \text{ with } j_0 < j \leq j_1, 1 \leq t \leq k_B + 1: & \quad \sigma(H_{\leq}(j, B_i)) = \sigma(H_{\leq}(j - 1, B_i)) + \sigma(H(j, B_i)), \\
\forall j \text{ with } j_0 \leq j \leq j_1: & \quad \sigma(S_{\leq}(j, B_i)) = \sigma(H_{\leq}(j, B_i)), \\
\forall j, t \text{ with } j_0 \leq j \leq j_1, 1 < t \leq k_B + 1: & \quad \sigma(S_{\leq}(j, B_i)) = \sigma(S_{\leq}(j, B_{t-1})) + \sigma(H_{\leq}(j, B_i)),
\end{align*}

we compute all the values $\sigma(S_{\leq}(j, B_i))$ in $O((k_B^2))$ time. Since $R(p, B) = S_{\leq}(j(t), B_{t(p)})$ for some indices $j(p)$ and $t(p)$ that we can easily obtain, the lemma follows. \hfill $\square$

**Theorem 5.7** The Shapley values of the \textsc{AreaAnchoredRectangles} game for $n$ points can be computed in $O(n^{3/2} \log n)$ time.

**Proof** As discussed earlier, it is enough to consider the case when $P$ is in the positive quadrant because instances in the other quadrants can be transformed to this one.

We split the set of cells into $k = \lceil \sqrt{n} \rceil$ bands, each with at most $k$ horizontal slabs; one of the bands can have fewer slabs. For each band $B$ we use Lemma 5.6. This takes $O(k(k^2 + n) \log n) = O(n^{3/2} \log n)$ time. For each band $B$ and $i \in [n]$, we further consider $V_{\leq}(i, B) = \bigcup_{i' \leq i} V(i, B)$ and compute the values $\sigma(V_{\leq}(i, B)) = \sum_{i' \leq i} \sigma(V(i', B))$ using prefix sums. This takes $O(n^{3/2})$ additional time. The cells inside a rectangle $R_p$ now correspond to the disjoint union

$$R(p, B_p) \cup \bigcup_{B' \text{ below } B_p} V_{\leq}(i_p, B').$$
where $B_p$ is the band that contains $p$ (perhaps on its top boundary) and the index $i_p$ is such that the cell $c_{i_p, j}$ has $p$ on its top right corner. See Fig. 17. Because of Lemma 5.1 it follows that

$$
\phi(p, v_{ar}) = \sum_{c \in A, c \subseteq R_{p}} \text{area}(c) \cdot \text{ne}(c) = \sigma(R(p, B_p)) + \sum_{B' \text{ below } B_p} \sigma(V_{\leq}(i_p, B')).
$$

Since the relevant values are already computed, and there are $O(\sqrt{n})$ of them, namely one per band, we obtain $\phi(p, v_{ar})$ in $O(\sqrt{n})$ time per point $p \in P$. \hfill \Box

## 6 Area of the Bounding Box

In this section we are interested in the \textsc{AreaBoundingBox} game defined by the characteristic function $v_{bb}$. The structure of this section is similar to the structure of Sect. 5. In particular, we keep assuming that all points have different coordinates and we keep using the notation introduced in Sect. 5.1.

First we note that it is enough to consider the \textsc{AreaAnchoredBoundingBox} problem and assume that the points are in one quadrant. Recall that in this problem the origin $o$ has to be included in the bounding box. Thus, it uses the characteristic function $v_{abb}(Q) = \text{area}(\text{bb}(Q \cup \{o\}))$.

**Lemma 6.1** If we can compute the Shapley values of the \textsc{AreaAnchoredBoundingBox} problem for $n$ points in the first quadrant in time $T(n)$, then we can compute the Shapley values of the \textsc{AreaBoundingBox} game in $T(n) + O(n \log n)$ time. Furthermore, if we can compute the Shapley values of the \textsc{AreaAnchoredBoundingBox} problem for any $n$ points in a quadrant that form any chain in $T_c(n)$ time, then we can compute the Shapley values of the \textsc{AreaBoundingBox} game for $n$ points that form a chain in $T_c(n) + O(n \log n)$ time.
The AreaBoundingBox game is a sum of other games, where the non-trivial games are AreaAnchoredBoundingBox games with points in one quadrant.

**Proof** We use inclusion–exclusion, as indicated in Fig. 18. There are characteristic functions $v_1, \ldots, v_9$ such that, for each $Q \subseteq P$, we have $v_{bb}(Q) = \sum_{i=1}^9 v_i(Q)$. Moreover, each characteristic function $v_i$ is either a constant value game (first term in Fig. 18), isometrically equivalent to an AreaBand game (last four terms in Fig. 18), or isometrically equivalent to an AreaAnchoredBoundingBox problem with all points in one quadrant (the remaining four terms in Fig. 18). For the game with constant value and for band games we can compute Shapley values in $O(n \log n)$ time (Lemma 2.1). Finally, note that if the points form a chain, they also form a chain in each of the cases, possibly exchanging the increasing/decreasing character through the isometric equivalence.

**6.1 Interpreting Shapley Values Geometrically**

First, we reduce the problem of computing Shapley values of the AreaAnchoredBoundingBox to a purely geometric problem. The situation here is slightly more complicated than in Sect. 5 because a cell $c$ of $A$ is inside $bb(Q \cup \{o\})$ if and only if $Q$ contains some point in $\text{NE}(c)$ or it contains some point in $\text{NW}(c)$ and in $\text{SE}(c)$. For a cell $c$ of $A$ we define the following values:

$$
\psi_{\text{NE}}(c) = \frac{1}{\text{ne}(c) + \text{nw}(c)} + \frac{1}{\text{ne}(c) + \text{se}(c)} - \frac{1}{\text{ne}(c) + \text{nw}(c) + \text{se}(c)},
$$

$$
\psi_{\text{NW}}(c) = \frac{1}{\text{ne}(c) + \text{nw}(c)} - \frac{1}{\text{ne}(c) + \text{nw}(c) + \text{se}(c)},
$$

$$
\psi_{\text{SE}}(c) = \frac{1}{\text{ne}(c) + \text{se}(c)} - \frac{1}{\text{ne}(c) + \text{nw}(c) + \text{se}(c)}.
$$

The following lemma is summarized in Fig. 19.

**Lemma 6.2** If $P$ is in the positive quadrant, then for each $p \in P$ the Shapley value $\phi(p, v_{abb})$ is

$$
\sum_{c \in A, p \in \text{NE}(c)} \text{area}(c) \cdot \psi_{\text{NE}}(c) + \sum_{c \in A, p \in \text{NW}(c)} \text{area}(c) \cdot \psi_{\text{NW}}(c) + \sum_{c \in A, p \in \text{SE}(c)} \text{area}(c) \cdot \psi_{\text{SE}}(c).
$$
Proof Each cell $c$ of the arrangement $A$ defines a game for the set of players $P$ with characteristic function

$$v_c(Q) = \begin{cases} \text{area}(c) & \text{if } c \subseteq \text{bb}(Q \cup \{o\}), \\ 0 & \text{otherwise}. \end{cases}$$

First note $\Delta(v_c, \pi, p)$ can only take the values 0 and $\text{area}(c)$. To analyze the Shapley values of the game defined by $v_c$ we use Lemma 2.5.

Consider a point $p \in \text{NE}(c)$. For a permutation $\pi \in \Pi(P)$, we have $\Delta(v_c, \pi, p) = \text{area}(c)$ if and only if $p$ is the first point of $\text{NE}(c)$ in the permutation $\pi$ and all the points of $\text{NW}(c)$ or all the points of $\text{SE}(c)$ are after $p$ in the permutation $\pi$. According to the first item of Lemma 2.5, with $N = P$, $a = p$, $A = \text{NE}(c)$, $B = \text{NW}(c)$, and $C = \text{SE}(c)$, there are precisely

$$n! \cdot \left( \frac{1}{\text{ne}(c) + \text{nw}(c)} + \frac{1}{\text{ne}(c) + \text{se}(c)} - \frac{1}{\text{ne}(c) + \text{nw}(c) + \text{se}(c)} \right) = n! \cdot \psi_{\text{NE}}(c)$$

permutations that fulfill this criteria. This means that, for each $p \in \text{NE}(c)$ we have

$$\phi(p, v_c) = \text{area}(c) \cdot \psi_{\text{NE}}(c).$$

Consider now a point $p \in \text{NW}(c)$. For a permutation $\pi \in \Pi(P)$, we have $\Delta(v_c, \pi, p) = \text{area}(c)$ if and only if $p$ is the first point of $\text{NW}(c)$ in the permutation $\pi$, all the points of $\text{NE}(c)$ are after $p$ in the permutation $\pi$, and at least one point of $\text{SE}(c)$ is before $p$ in the permutation $\pi$. According to the second item of Lemma 2.5, with $N = P$, $b = p$, $B = \text{NW}(c)$, $A = \text{NE}(c)$, and $C = \text{SE}(c)$, there are precisely

$$n! \cdot \left( \frac{1}{\text{ne}(c) + \text{nw}(c)} - \frac{1}{\text{ne}(c) + \text{nw}(c) + \text{se}(c)} \right) = n! \cdot \psi_{\text{NW}}(c)$$
permutations that fulfill this criteria. This means that, for each \( p \in \text{NW}(c) \) we have
\[
\phi(p, v_c) = \text{area}(c) \cdot \psi_{\text{NW}}(c).
\]

The case for a point \( p \in \text{SE}(c) \) is symmetric to the case \( p \in \text{NW}(c) \), where the roles of \( \text{se}(c) \) and \( \text{nw}(c) \) are exchanged. Therefore we conclude
\[
\phi(p, v_c) = \text{area}(c) \cdot \begin{cases} 
\psi_{\text{NE}}(c) & \text{if } p \in \text{NE}(c), \\
\psi_{\text{NW}}(c) & \text{if } p \in \text{NW}(c), \\
\psi_{\text{SE}}(c) & \text{if } p \in \text{SE}(c).
\end{cases}
\]
As a sanity check it is good to check that
\[
\text{ne}(c) \cdot \psi_{\text{NE}}(c) + \text{nw}(c) \cdot \psi_{\text{NW}}(c) + \text{se}(c) \cdot \psi_{\text{SE}}(c) = 1.
\]
This is the case. Noting that we have
\[
v_{\text{abb}}(Q) = \sum_{c \in A} v_c(Q) \quad \text{for all } Q \subset P,
\]
the result follows from the linearity of Shapley values. \( \square \)

Using the same technique as in the discussion after the proof of Lemma 5.1, we could compute \( \phi(p, v_{\text{abb}}) \) for all \( p \in P \) in near-quadratic time. Indeed, for computing the sums \( \sum_{c \in C, p \in \text{NE}(c)} \text{area}(c) \cdot \psi_{\text{NE}}(c) \) for all \( p \in P \), we can represent each cell \( c \) by a point with weight \( \text{area}(c) \cdot \psi_{\text{NE}}(c) \) and then use orthogonal range queries to compute the sums. Handling NW and SE is similar. Our objective is to carry out these computations faster.

### 6.2 Handling Empty Blocks

For each subset of cells \( C \) of \( A \), we define
\[
\sigma_{\text{NE}}(C) = \sum_{c \in C} \text{area}(c) \cdot \psi_{\text{NE}}(c), \quad \sigma_{\text{NW}}(C) = \sum_{c \in C} \text{area}(c) \cdot \psi_{\text{NW}}(c),
\]
\[
\sigma_{\text{SE}}(C) = \sum_{c \in C} \text{area}(c) \cdot \psi_{\text{SE}}(c).
\]
Like in the case of anchored boxes, our objective here is to compute these values for several vertical and horizontal slabs.

In the following we assume that we have preprocessed \( P \) in \( O(n \log n) \) time such that \( \text{ne}(q), \text{nw}(q) \) and \( \text{se}(q) \) can be computed in \( O(\log n) \) time for each point \( q \) given at query time [35]. We use again multipoint evaluation to compute the values \( \sigma_{\text{Abb}}(\cdot) \) for each vertical and horizontal slab of an empty block and for each \( \ast \in \{ \text{NE}, \text{NW}, \text{SE} \} \). However, the treatment has to be a bit more careful now because we have to deal with different rational functions.
Lemma 6.3 Let $B$ be an empty block with $k$ columns and rows. We can compute in $O(k \log n)$ time the values $\sigma_\ast(C)$ for all slabs $C$ within $B$ and each $\ast \in \{\text{NE}, \text{NW}, \text{SE}\}$.

Proof The proof is very similar to the proof of Lemma 5.2, but some adaptation has to be made. Assume that $B$ is the block $B(i_0, i_1, j_0, j_1)$. We explain how to compute the values $\sigma_{\text{NE}}(V(i, B))$ for all $i_0 \leq i \leq i_1$. The technique to compute $\sigma_{\text{NE}}(H(j_0, B)), \ldots, \sigma_{\text{NE}}(H(j_1, B))$ and for NW and SE instead of NE is similar. For each $\ell$ we define

$$J(\ell) = \{ j \mid j_0 \leq j \leq j_1, \ \text{ne}(c_{i_0, j}) + \text{se}(c_{i_0, j}) = \ell \},$$

$$J'(\ell) = \{ j \mid j_0 \leq j \leq j_1, \ \text{ne}(c_{i_0, j}) + \text{nw}(c_{i_0, j}) + \text{se}(c_{i_0, j}) = \ell \}.$$

Let $\ell_0$ and $\ell_1$ be the minimum and the maximum $\ell$ such that $J(\ell) \neq 0$, respectively. Similarly, let $\ell'_0$ and $\ell'_1$ be the minimum and the maximum $\ell$ such that $J'(\ell) \neq 0$.

We set up the following fractional constant and rational functions with variable $x$:

$$\mathcal{F}_1 = \sum_{j=j_0}^{j_1} \frac{h_j}{\text{ne}(c_{i_0, j}) + \text{nw}(c_{i_0, j})}, \quad \mathcal{F}_2(x) = \sum_{\ell=\ell_0}^{\ell_1} \frac{\sum_{j \in J(\ell)} h_j}{\ell + x},$$

$$\mathcal{F}_3(x) = \sum_{\ell=\ell'_0}^{\ell'_1} \frac{\sum_{j \in J'(\ell)} h_j}{\ell + x}.$$

Each of them is the sum of at most $k$ fractions. As discussed in the proof of Lemma 5.2, $\mathcal{F}_2(x)$ and $\mathcal{F}_3(x)$ can be rewritten so that we can use Lemma 2.3 to evaluate them at consecutive integral points. The coefficients can be computed in $O(k \log n)$ time.

Consider two consecutive vertical slabs $V(i, B)$ and $V(i+1, B)$ within the block $B$. Figure 20 shows the changes in the counters. Because the block $B$ is empty we have the following properties:
• The sum \( \text{ne}(c_{i,j}) + \text{nw}(c_{i,j}) \) is independent of \( i \). Thus we have

\[
\text{ne}(c_{i,j}) + \text{nw}(c_{i,j}) = \text{ne}(c_{i_0,j}) + \text{nw}(c_{i_0,j})
\]

for all relevant \( i \) and \( j \).

• The difference

\[
(\text{ne}(c_{i+1,j}) + \text{se}(c_{i+1,j})) - (\text{ne}(c_{i,j}) + \text{se}(c_{i,j}))
\]

is always \(-1\). Therefore we have

\[
\text{ne}(c_{i,j}) + \text{se}(c_{i,j}) = \text{ne}(c_{i_0,j}) + \text{se}(c_{i_0,j}) - (i - i_0)
\]

for all relevant \( i \) and \( j \). In particular, for each \( j \in J(\ell) \) we have \( \text{ne}(c_{i,j}) + \text{se}(c_{i,j}) = \ell - (i - i_0) \).

• The difference

\[
(\text{ne}(c_{i+1,j}) + \text{nw}(c_{i+1,j}) + \text{se}(c_{i+1,j})) - (\text{ne}(c_{i,j}) + \text{se}(c_{i,j}) + \text{se}(c_{i,j}))
\]

depends on whether the point \( p \in P \) with \( x(p) = x_i \) is above or below \( B \). Thus, this difference is independent of the row \( j \). This means that there exist, for each index \( i \), a value \( \delta_i \) such that

\[
\text{ne}(c_{i,j}) + \text{nw}(c_{i,j}) + \text{se}(c_{i,j}) = \text{ne}(c_{i_0,j}) + \text{nw}(c_{i_0,j}) + \text{se}(c_{i_0,j}) + \delta_i
\]

for all relevant \( j \). In particular, for each \( j \in J'(\ell) \) we have \( \text{ne}(c_{i,j}) + \text{nw}(c_{i,j}) + \text{se}(c_{i,j}) = \ell + \delta_i \). Each single value \( \delta_i \) can be computed as \( \delta_i = \text{se}(c_{i,j}) - \text{se}(c_{i_0,j}) \) in \( O(\log n) \) time using range counting queries. Note that \( |\delta_i - \delta_{i+1}| \leq 1 \).

Note that for each relevant \( i \) we have

\[
w_i \cdot (\mathcal{F}_1 + \mathcal{F}_2(-i + i_0) + \mathcal{F}_3(\delta_i))
\]

\[
= w_i \cdot \left( \sum_{j=j_0}^{j_1} \frac{h_j}{\text{ne}(c_{i_0,j}) + \text{nw}(c_{i_0,j})} + \sum_{\ell = \ell_0}^{\ell_1} \sum_{j \in J(\ell)} \frac{h_j}{\ell - i + i_0} + \sum_{\ell = \ell_0'}^{\ell_1'} \sum_{j \in J'(\ell)} \frac{h_j}{\ell + \delta_i} \right)
\]

\[
= \sum_{j=j_0}^{j_1} \left( \frac{w_i h_j}{\text{ne}(c_{i,j}) + \text{nw}(c_{i,j})} + \frac{w_i h_j}{\text{ne}(c_{i,j}) + \text{se}(c_{i,j})} - \frac{w_i h_j}{\text{ne}(c_{i,j}) + \text{nw}(c_{i,j}) + \text{se}(c_{i,j})} \right)
\]

\[
= \sum_{j=j_0}^{j_1} \text{area}(c_{i,j}) \cdot \psi_{\text{NE}}(c_{i,j}) = \sigma_{\text{NE}}(V(i, B)).
\]

Thus, we need to evaluate \( \mathcal{F}_2(x) \) at the values \( x = 0, -1, \ldots, i_0 - i_1 \) and we have to evaluate \( \mathcal{F}_3(x) \) at the integral values \( \delta_i \) for \( i = i_0, \ldots, i_1 \). Note that \( |\delta_i - \delta_{i+1}| \leq 1 \).
According to Lemma 2.3, this can be done in $O(k \log n)$ time. After this, we get each value $\sigma_{\text{NE}}(V(i, B))$ in constant time. \hfill \Box

### 6.3 Algorithms for Bounding Box

The same techniques that were used in Sects. 5.4 and 5.5 work in this case. We compute partial sums $\sigma_a(\cdot)$ for several vertical and horizontal slabs within empty blocks. The only difference is that we have to cover the whole bounding box $\text{bb}(P \cup \{o\})$ with empty blocks, instead of covering just the portion that was dominated by some point. In the case of an increasing chain in the positive quadrant, the anchored bounding box and the union of anchored rectangles is actually the same object. In the case of a decreasing chain, we have to work above and below the chain to cover the whole bounding box. See Fig. 21. This is twice as much work, so it does not affect the asymptotic running time. For arbitrary point sets, we again use the bands with roughly $\sqrt{n}$ horizontal slabs, and break each slab into empty boxes, as it was done in Lemma 5.6. For the partial sums that we consider (like $\sigma(H_\leq(\cdot))$ and $\sigma(V_\leq(\cdot))$) we also construct the symmetric versions $\sigma_a(H_\geq(\cdot))$ and $\sigma_a(V_\geq(\cdot))$. With this information we can recover the sums that we need in each of the three quadrants with an apex in $p \in P$; recall Fig. 19. In the case of a decreasing chain, we get an extra case that is handled by noting that

$$
\eta_a = \sum_{c \in A} \text{area}(c) \cdot \psi_{\text{SE}}(c)
$$

is

$$
\eta_a = \eta_{a-1} + \sum_{j=a+1}^{n} \text{area}(c_{a,j}) \cdot \psi_{\text{SE}}(c_{a,j}) + \sum_{i=1}^{a-1} \text{area}(c_{i,n-a+1}) \cdot \psi_{\text{SE}}(c_{i,n-a+1}).
$$

The last two terms are a vertical and a horizontal band around the cell $c_{a,n-a+1}$. The sums for $\psi_{\text{NW}}$ are handled similarly. We omit the details and summarize.

**Theorem 6.4** The Shapley values of the AREAANCHOREDBOUNDINGBOX game for $n$ points can be computed in $O(n^{3/2} \log n)$ time. If the points form a chain, then we need $O(n \log^2 n)$ time.

Because of Lemma 6.1 we also obtain the following.

**Theorem 6.5** The Shapley values of the AREABOUNDINGBOX game for $n$ points can be computed in $O(n^{3/2} \log n)$ time. If the points form a chain, then we need $O(n \log^2 n)$ time.

### 7 Conclusions

We have discussed the efficient computation of Shapley values, a classical topic in game theory, for coalitional games defined for points in the plane and characteristic
functions given by the area or the perimeter of geometric objects. For axis-parallel problems we used algebraic methods to get faster algorithms. For non-axis-parallel problems we provided efficient algorithms based on decomposing the sum into parts and grouping permutations that contribute equally to a part.

In game theory, quite often one considers coalitional games where some point, say the origin, has to be included in the solutions that are considered. For example, we could use the minimum enclosing disk that contains also the origin. We do this for the anchored versions of the games. This setting is meaningful in several scenarios, for example, when we split the costs to connect to a fixed landmark. Our results also hold in this setting through an easy adaptation.

The problems we consider here are a new type of stochastic problems in computational geometry. The relation to other problems in stochastic computational geometry is unclear. For example, computing the expected length of the minimum spanning tree (MST) in the plane for a stochastic point set is #P-hard [18]. Does this imply the same for the coalitional game based on the length of the Euclidean MST? Is there a two-way relation between Shapley values and the stochastic version for geometric problems? In particular, is there a FPRAS for computing the Shapley values of the game defined using the Euclidean MST? For the stochastic model this is shown by Kamousi et al. [18], while for non-geometric settings Ando [4] has shown #P-hardness of computing Shapley values of the minimum-cost spanning tree game. Note that the length of the Euclidean spanning tree is not monotone: adding points may reduce its length. Thus, some Shapley values could be potentially non-positive, which, at least intuitively, makes it harder to get approximations.

Finally, it would be worth to understand whether there is some relation between Shapley values in geometric settings and the concept of depth in point sets, in particular for the Tukey depth. For stochastic points, this relation has been explored and exploited by Agarwal et al. [1].

Our algorithms generalize to higher dimensions, increasing the degree of the polynomial describing the running time. Are there substantial improvements possible for
higher dimensions? In dimension $d$, computing the volume of the bounding box can be done in $O(dn)$ time, while the obvious algorithms to compute the Shapley values for the corresponding game require $n^{\Theta(d)}$ time. Is this linear dependency on $d$ in the degree of the polynomial actually needed, under some standard assumption in complexity theory?

**Acknowledgements** The authors are very grateful to Sariel Har-Peled for fruitful discussions. In particular, the possible link to Tukey depth was pointed out by him.

**References**

1. Agarwal, P.K., Har-Peled, S., Suri, S., Yildiz, H., Zhang, W.: Convex hulls under uncertainty. Algorithmica 79(2), 340–367 (2017)
2. Agarwal, P.K., Kumar, N., Sintos, S., Suri, S.: Range-max queries on uncertain data. J. Comput. Syst. Sci. 94, 118–134 (2018)
3. Ajwani, D., Ray, S., Seidel, R., Triwary, H.R.: On computing the centroid of the vertices of an arrangement and related problems. In: 10th International Workshop on Algorithms and Data Structures (Halifax 2007). Lecture Notes in Comput. Sci., vol. 4619, pp. 519–528. Springer, Berlin (2007)
4. Ando, K.: Computation of the Shapley value of minimum cost spanning tree games: #P-hardness and polynomial cases. Jpn. J. Ind. Appl. Math. 29(3), 385–400 (2012)
5. Aronov, B., Katz, M.J.: Batched point location in SINR diagrams via algebraic tools. ACM Trans. Algorithms 14(4), #41 (2018)
6. de Berg, M., Cheong, O., van Kreveld, M., Overmars, M.: Computational Geometry. Algorithms and Applications. Springer, Berlin (2008)
7. Cabello, S., Chan, T.M.: Computing Shapley values in the plane. In: 35th International Symposium on Computational Geometry (Portland 2019). Leibniz International Proceedings in Informatics, vol. 129, #20. Leibniz-Zent. Inform., Wadern (2019)
8. Chalkiadakis, G., Elkind, E., Wooldridge, M.: Computational Aspects of Cooperative Game Theory. Synthesis Lectures on Artificial Intelligence and Machine Learning, vol. 16. Morgan & Claypool, Williston (2012)
9. Deng, X., Fang, Q.: Algorithmic cooperative game theory. In: Pareto Optimality, Game Theory and Equilibria. Springer Optimization and Its Applications, vol. 17, pp. 159–185. Springer, New York (2008)
10. Deng, X., Papadimitriou, Ch.H.: On the complexity of cooperative solution concepts. Math. Oper. Res. 19(2), 257–266 (1994)
11. Edelsbrunner, H., O’Rourke, J., Seidel, R.: Constructing arrangements of lines and hyperplanes with applications. SIAM J. Comput. 15(2), 341–363 (1986)
12. Edelsbrunner, H., Seidel, R., Sharir, M.: On the zone theorem for hyperplane arrangements. SIAM J. Comput. 22(2), 418–429 (1993)
13. Faigle, U., Fekete, S.P., Hochstädtler, W., Kern, W.: On approximately fair cost allocation in Euclidean TSP games. OR Spektrum 20(1), 29–37 (1998)
14. Faigle, U., Kern, W., Fekete, S.P., Hochstädtler, W.: On the complexity of testing membership in the core of min-cost spanning tree games. Int. J. Game Theory 26(3), 361–366 (1997)
15. Ferguson, T.S.: A Course in Game Theory. World Scientific, Hackensack (2010)
16. Fink, M., Hershberger, J., Kumar, N., Suri, S.: Hyperplane separability and convexity of probabilistic point sets. J. Comput. Geom. 8(2), 32–57 (2017)
17. Granot, D., Huberman, G.: Minimum cost spanning tree games. Math. Program. 21(1), 1–18 (1981)
18. Kamouss, P., Chan, T.M., Suri, S.: Stochastic minimum spanning trees in Euclidean spaces. In: 27th Annual Symposium on Computational Geometry (Paris 2011), pp. 65–74. ACM, New York (2011)
19. Kamouss, P., Chan, T.M., Suri, S.: Closest pair and the post office problem for stochastic points. Comput. Geom. 47(2B), 214–223 (2014)
20. Langerman, S.: The complexity of halfspace area queries. Discrete Comput. Geom. 30(4), 639–648 (2003)
21. Littlechild, S.C., Owen, G.: A simple expression for the Shapely value in a special case. Manag. Sci. 20(3), 370–372 (1973)
22. Matoušek, J.: Lectures on Discrete Geometry. Graduate Texts in Mathematics, vol. 212. Springer, New York (2002)
23. Megiddo, N.: Computational complexity of the game theory approach to cost allocation for a tree. Math. Oper. Res. 3(3), 189–196 (1978)
24. Moroz, G., Aronov, B.: Computing the distance between piecewise-linear bivariate functions. ACM Trans. Algorithms 12(1), #3 (2016)
25. Myerson, R.B.: Game Theory: Analysis of Conflict. Harvard University Press, Cambridge (1997)
26. Nisan, N., Roughgarden, T., Tardos, É., Vazirani, V.V. (eds.): Algorithmic Game Theory. Cambridge University Press, Cambridge (2007)
27. Osborne, M.J., Rubinstein, A.: A Course in Game Theory. MIT Press, Cambridge (1994)
28. Overmars, M.H., Yap, C.-K.: New upper bounds in Klee’s measure problem. SIAM J. Comput. 20(6), 1034–1045 (1991)
29. Pérez-Lantero, P.: Area and perimeter of the convex hull of stochastic points. Comput. J. 59(8), 1144–1154 (2016)
30. Puerto, J., Tamir, A., Perea, F.: A cooperative location game based on the 1-center location problem. Eur. J. Oper. Res. 214(2), 317–330 (2011)
31. Puerto, J., Tamir, A., Perea, F.: Cooperative location games based on the minimum diameter spanning Steiner subgraph problem. Discrete Appl. Math. 160(7–8), 970–979 (2012)
32. Roth, A.E. (ed.): The Shapley Value: Essays in Honor of Lloyd S. Shapley. Cambridge University Press, Cambridge (1988)
33. Thomson, W.: Cost allocation and airport problems (2014). https://www.iser.osaka-u.ac.jp/collabo/20140524/Airport_Problems.pdf
34. Welzl, E.: Smallest enclosing disks (balls and ellipsoids). In: New Results and New Trends in Computer Science (Graz 1991). Lecture Notes in Comput. Sci., vol. 555, pp. 359–370. Springer, Berlin (1991)
35. Willard, D.E.: New data structures for orthogonal range queries. SIAM J. Comput. 14(1), 232–253 (1985)
36. Winter, E.: The Shapley value. In: Handbook of Game Theory with Economic Applications, vol. 3, pp. 2025–2054. North-Holland, Amsterdam (2002)
37. Xue, J., Li, Y., Janardan, R.: On the separability of stochastic geometric objects, with applications. Comput. Geom. 74, 1–20 (2018)

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.