Computing Shapley Values in the Plane

Sergio Cabello
Department of Mathematics, FMF, University of Ljubljana, Slovenia
Department of Mathematics, IMFM, Ljubljana, Slovenia
https://www.fmf.uni-lj.si/~cabello/
sergio.cabello@fmf.uni-lj.si

Timothy M. Chan
Department of Computer Science, University of Illinois at Urbana-Champaign, USA
http://tmc.web.engr.illinois.edu/
tmc@illinois.edu

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 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 of the convex hull or the minimum enclosing disk can be computed in $O(n^2)$ and $O(n^3)$ time, respectively. 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.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Shapley values, stochastic computational geometry, convex hull, minimum enclosing disk, bounding box, arrangements, convolutions, airport problem

Digital Object Identifier 10.4230/LIPIcs.SoCG.2019.20

Related Version Full version available at http://arxiv.org/abs/1804.03894.

Funding *Sergio Cabello*: Supported by the Slovenian Research Agency (P1-0297, J1-8130, J1-8155). *Timothy M. Chan*: Supported in part by NSF Grant CCF-1814026.

Acknowledgements The authors are very grateful to Sariel Har-Peled for fruitful discussions.

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 $v\colon 2^P \to \mathbb{R}$ 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.

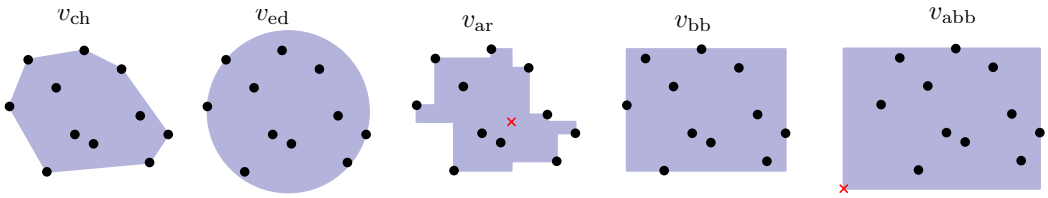

Figure 1 Different costs associated to a point set that are considered in this paper. The cross represents the origin. In all cases we focus on the area.

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 Figure 1. More precisely, we consider the following coalitional games.

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

AreaEnclosingDisk game: The characteristic function is $v_{\text{ed}}(Q) = \text{area}\,(\text{med}(Q))$ for each nonempty $Q \subset P$, where $\text{med}(Q)$ is a disk of smallest radius that contains Q.

AreaAnchoredRect 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.

In the full version [5] we also consider variants of these problems where the perimeter of the shapes is used.

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 [26] or the survey by Winter [29] 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 ([11, Chapter IV.3], [19, Section 9.4], [21, Section 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 $\pi\colon P \to [n]$. 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 π, 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 π, until we insert p. The increment in $v(\cdot)$ when adding player p is

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

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

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

where π 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 π define the same subset $P(\pi, p)$, the Shapley value of p is often rewritten as

$$\phi(p, v) = \sum_{S \subset P \setminus \{p\}} \frac{|S|!(n - |S| - 1)!}{n!} \Big(v(S \cup \{p\}) - v(S)\Big).$$

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. These problems resemble the models recently considered in *stochastic* computational geometry; see for example [1, 2, 12, 13, 14, 23, 30]. 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, for example, the expected area of the minimum enclosing disk.

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. Conditioning on properties of the shape before adding the new point p, one can get polynomial-time algorithms. We improve the running time by 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 AreaAnchoredRect, 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, 4, 15, 18] using such techniques in geometric problems.

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 [22]. 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.

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, Elkind, and Wooldridge [6] and the chapter by Deng and Fang [8] give a summary of computational aspects of coalitional games. The book by Nisan et al. [20] provides a general overview of the interactions between game theory and algorithms.

In the classical Airport problem considered by Littlechild and Owen [16], 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 [27]. 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*, defined as

$$\Big\{(x_p)_{p\in P} \in \mathbb{R}^P \mid \forall S \subseteq P : \sum_{p\in S} x_p \geq v(S)\Big\}.$$

Sometimes the condition $\sum_{p\in P} x_p = v(P)$ is also added to the definition. 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, Tamir and Perea [24] 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. 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, the focus of their work is on understanding the core of the game, and do 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, Tamir and Perea [25] 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. [10] consider the TSP coalitional game in general metric spaces, specialize 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 [17] extended this to trees, while Deng and Papadimitriou [9] discuss a game on arbitrary graphs defined by induced subgraphs. They show that the Shapley values are easy to compute, while characterizing the core is NP-complete.

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

Because of space constraints, we limit our presentation to a selection of our results and an overview of the ideas. In this version we do not include our algorithm for computing the Shapley values for the AREAENCLOSINGDISK game in $O(n^3)$ time. Also, our results about considering the perimeter are not described in this version; they can be found in the full version [5].

We start with preliminaries in Section 2, where we set the notation, present basic properties of Shapley values, and discuss our needs of algebraic computations. The AREACONVEXHULL game is considered in Section 3. In Section 4 we discuss the main ideas for the AREAANCHOREDRECT game and provide some discussion concerning the AREABOUNDINGBOX and AREAANCHOREDBOUNDINGBOX games. We conclude in Section 5 with some discussion.

2 Preliminaries

All point sets will be 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 $\mathrm{bb}(Q)$ for the (minimum) axis-parallel bounding box of Q, $\mathrm{med}(Q)$ for a smallest (actually, *the* smallest) disk that contains Q, and $CH(Q)$ 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 $p, q \in P$.

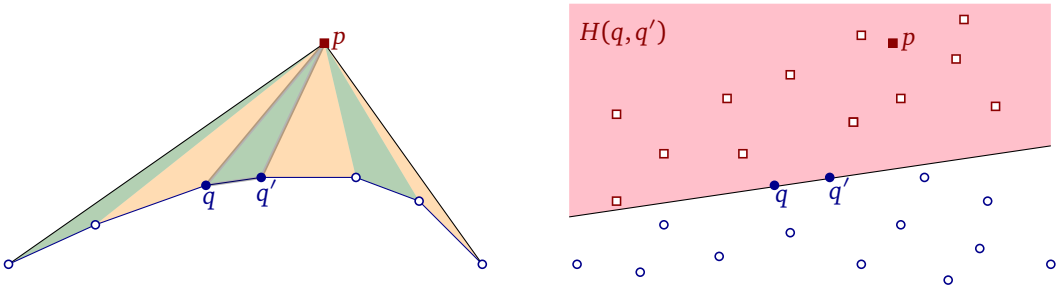

Figure 2 Left: triangulating the difference between $CH(P(\pi,p))$ and $CH(P(\pi,p)\setminus\{p\})$. Right: in order for $\triangle pqq'$ to be in $T(\pi,p)$, q and q' must appear before p, which in turn must appear before all other points in the halfplane $H(q,q')$.

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

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

The Shapley values when the characteristic function v is constant over all nonempty subsets of P is given $\phi(p,v) = v(P)/n$.

Algebraic computations. For axis-parallel problems we will use the lemma below, which provides multipoint evaluation for a special type of rational functions. The lemma follows from a simple application of convolution, via fast Fourier transform; see the full paper for the proof. (A more general approach of Aronov, Katz and Moroz [4, 18] for arbitrary rational functions gives a slightly worse running time.)

▶ **Lemma 1.** *Let $b_0, \ldots, b_n, \Delta$ be real numbers and consider the rational function*

$$R(x) \;=\; \sum_{t=0}^{n} \frac{b_t}{\Delta + t + x}.$$

Given an integer $\ell > -\Delta$, possibly negative, and a positive integer m, we can evaluate $R(x)$ at all the integer values $x = \ell, \ell+1, \ldots, \ell+m$ in $O((n+m)\log(n+m))$ time.

3 Convex hull

In this section we consider the area of the convex hull of the points. Consider a fixed set P of points in the plane. For simplicity we assume that no three points are collinear.

▶ **Lemma 2.** *For each point p of P we can compute $\phi(p, v_{\mathrm{ch}})$ 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 $\mathrm{level}(q,q')$ to be the number of points in $P \cap H(q,q')$.

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

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

In other words, $\triangle pqq' \in T(\pi, p)$ if and only if $p \in H(q, q')$, and among the $\text{level}(q, q') + 2$ points in $H(q, q') \cup \{q, q'\}$, the two earliest points are q and q', and the third earliest point is p. See Figure 2 (right). (Note that if $CH(P(\pi, p)) = CH(P(\pi, p) \setminus \{p\})$, then $T(\pi, p)$ is empty.) For fixed $p, q, q' \in P$ with $p \in H(q, q')$, simple counting implies that the probability that $\triangle pqq' \in T(\pi, p)$ with respect to a random permutation π 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 of p is

$$\phi(p, v_{\text{ch}}) \;=\; \sum_{\substack{q, q' \in P\ (q \neq q') \\ \text{with } p \in H(q, q')}} \text{area}(\triangle pqq') \cdot \rho(q, q'). \tag{1}$$

From the formula, we can immediately compute $\phi(p, v_{\text{ch}})$ 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 $\text{level}(q, q')$ using $O(1)$ arithmetic operations. Thus, precomputing $\rho(q, q')$ requires precomputing $\text{level}(q, q')$ for 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 [7, Chapter 8], and the levels of all vertices can be subsequently generated by traversing the arrangement in $O(1)$ time per vertex. ◀

Naively applying Lemma 2 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.** *The Shapley values of the* 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 $\text{area}(\triangle pqq')$ 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),

$$\phi(p, v_{\text{ch}}) \;=\; \sum_{\substack{q, q' \in P\ (q \neq q') \\ \text{with } p \in H(q, q')}} (A(q, q')x + B(q, q')y + C(q, q')) \;=\; \mathcal{A}(p)x + \mathcal{B}(p)y + \mathcal{C}(p),$$

where

$$\mathcal{A}(p) \;=\; \sum_{\substack{q, q' \in P\ (q \neq q') \\ \text{with } p \in H(q, q')}} A(q, q'), \quad \mathcal{B}(p) \;=\; \sum_{\substack{q, q' \in P\ (q \neq q') \\ \text{with } p \in H(q, q')}} B(q, q'), \quad \mathcal{C}(p) \;=\; \sum_{\substack{q, q' \in P\ (q \neq q') \\ \text{with } p \in H(q, q')}} C(q, q').$$

We describe how to compute $\mathcal{A}(p)$, $\mathcal{B}(p)$, and $\mathcal{C}(p)$ 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 3 instances of the following:

> Given a set P of n points in the plane, and given $O(n^2)$ lines each through 2 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).

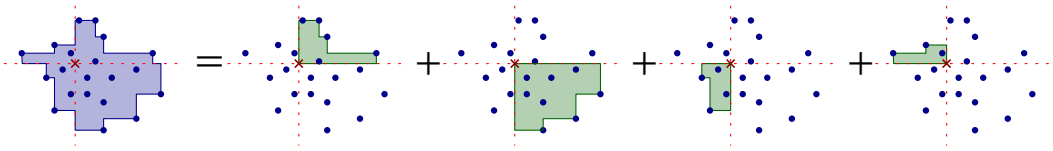

Figure 3 It is enough to consider the cases where all points are in a quadrant.

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 ℓ.

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. Given $\ell, \ell' \in L$, define $S(\ell, \ell')$ to be the sum of the weights of all vertices on the line ℓ' that are below ℓ. For a fixed $\ell' \in L$, we can precompute $S(\ell, \ell')$ for all $\ell \in L \setminus \{\ell'\}$ 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 ℓ'. The total time for all lines $\ell' \in L$ 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 $O(n^2)$. ◀

4 Axis-parallel problems

Most of this section is dedicated to the AreaAnchoredRect game defined by the characteristic function v_{ar}. Towards the end we discuss the additional challenges to handle the AreaBoundingBox and AreaAnchoredBoundingBox games. Unless stated explicitly, the game under consideration is the AreaAnchoredRect.

For the AreaAnchoredRect game, it is easy to see that one can focus on the special case where all the points are in a quadrant; see Figure 3.

4.1 Notation for axis-parallel problems

Consider a fixed set P of points in the positive quadrant. In the notation we 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.

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

$$\begin{aligned} \mathrm{NW}(q) &= \{p \in P \mid x(p) \le x(q),\ y(p) \ge y(q)\}, \\ \mathrm{NE}(q) &= \{p \in P \mid x(p) \ge x(q),\ y(p) \ge y(q)\}, \\ \mathrm{SE}(q) &= \{p \in P \mid x(p) \ge x(q),\ y(p) \le y(q)\}. \end{aligned}$$

We use lowercase to denote their cardinality: $\mathrm{nw}(q) = |\mathrm{NW}(q)|$, $\mathrm{ne}(q) = |\mathrm{NE}(q)|$ and $\mathrm{se}(q) = |\mathrm{SE}(q)|$. See Figure 4, 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*).

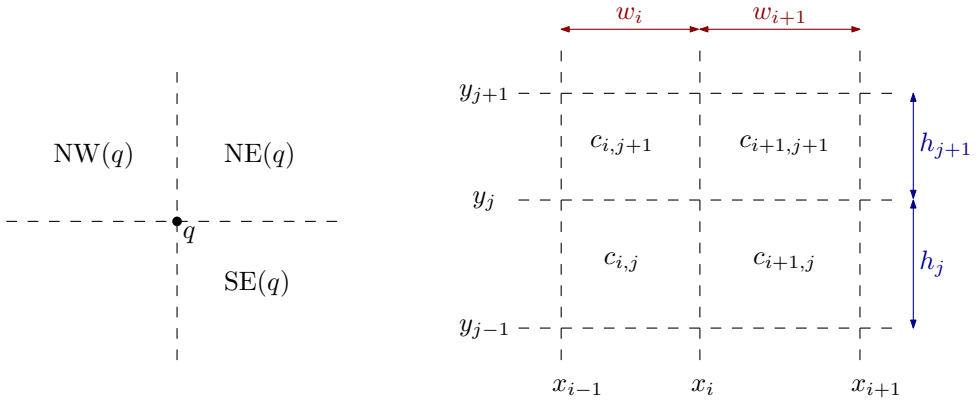

Figure 4 Left: the quadrants to define NW(q), NE(q) and SE(q). Right: cells of $\mathcal{A}$.

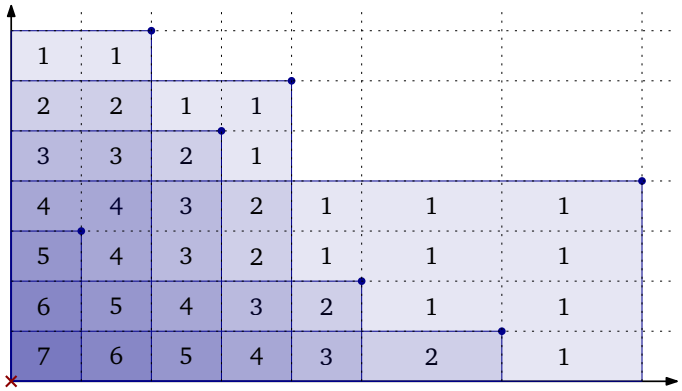

Figure 5 The non-zero counters ne(c) for the bounded cells c of $\mathcal{A}$. The intensity of the color correlates with the counter ne(c).

Let L be the set of horizontal and vertical lines that contain some point of P. We add to L both axes. The lines in L define a partition of the plane into cells, usually called the arrangement and denoted by $\mathcal{A} = \mathcal{A}(L)$. The (2-dimensional) cells of $\mathcal{A}$ 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 $\mathcal{A}$ 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 Figure 4, right.

Since NE(q) is constant over each 2-dimensional cell c of $\mathcal{A}$, we can define NE(c), for each cell $c \in \mathcal{A}$. The same holds for NW(c), SE(c), ne(c), nw(c) and se(q). See Figure 5.

A **block** is a set of cells $B = B(i_0, i_1, j_0, j_1) = \{c_{i,j} \mid i_0 \le i \le i_1,\ j_0 \le j \le j_1\}$ for some indices i_0, i_1, j_0, j_1, with $1 \le i_0 \le i_1 \le n$ and $1 \le j_0 \le j_1 \le n$. 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 at least three 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 Figure 6 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** $V(i, B) = \{c_{i,j} \mid 1 \le j \le n,\ c_{i,j} \in B\}$. Similarly, for each block B and each index j, we define the **horizontal slab** $H(j, B) = \{c_{i,j} \mid 1 \le i \le n,\ c_{i,j} \in B\}$. Such slabs are meaningful only for

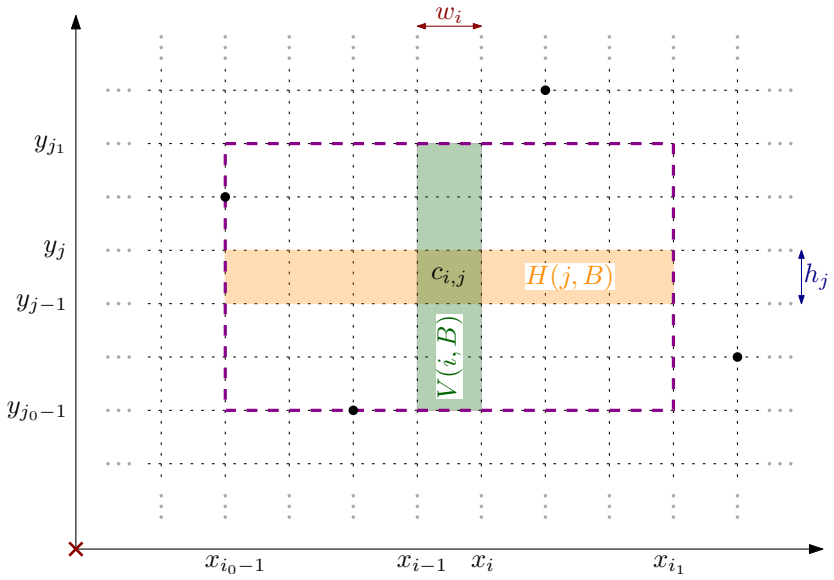

Figure 6 An *empty* block $B = B(i_0, i_1, j_0, j_1)$ with a vertical and a horizontal slab shaded.

indices within the range that defines the slab. We call them the **slabs within** B.

4.2 Interpreting Shapley values geometrically

First, we reduce the problem of computing Shapley values to a neat geometric problem. The following result can be shown by decomposing the game into several games, one per cell of $\mathcal{A}$, and using the linearity of Shapley values.

▶ **Lemma 4.** *If P is in the positive quadrant, then for each $p \in P$ we have*

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

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 $p \in P$.

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, as follows. An explicit computation of $\mathcal{A}$ takes quadratic time, and we can use standard data structures for orthogonal range searching ([28], [7, Chapter 5]) to 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 $w_c = \text{area}(c)/\,\text{ne}(c)$, computing $\phi(p, v_{\text{ar}})$ reduces to 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 $\text{ne}(c)$ and partial sums of the weights w_c.

Our objective in the following sections is to improve this result using the correlation between adjacent cells. While at first glance it seems that segment trees [7, Section 10.3] may be useful, the weights are inversely proportional to $\text{ne}(c)$, which gives problems

4.3 Handling empty blocks

In the following we assume that we have preprocessed P in $O(n \log n)$ time such that $\mathrm{ne}(q)$ can be computed in $O(\log n)$ time for each point q given at query time [28]. This is a standard range counting for orthogonal ranges.

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 of the block.

▶ **Lemma 5.** *Let B be an empty block with k columns and rows. We can compute in $O(k \log n)$ time the values $\sigma(C)$ for all 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 $i_0 \le i \le i_1$. 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 $\mathrm{ne}(\cdot)$. More precisely, for each ℓ we define $J(\ell) = \{j \mid j_0 \le j \le j_1,\ \mathrm{ne}(c_{i_0,j}) = \ell\}$. Let ℓ_0 and ℓ_1 be the minimum and the maximum ℓ such that $J(\ell) \neq 0$, respectively.

Recall that h_j is the height of the cell $c_{i,j}$ for all $i_0 \le i \le i_1$. We set up the following rational function with variable x:

$$R(x) \;=\; \sum_{\ell=\ell_0}^{\ell_1} \frac{\sum_{j \in J(\ell)} h_j}{\ell + x}.$$

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

$$R(x) \;=\; \sum_{t=0}^{\ell_1 - \ell_0} \frac{b_t}{\Delta + t + x}.$$

Thus, this is a rational function of the shape considered in Lemma 1 with $\ell_1 - \ell_0 \le j_1 - j_0 + 1 \le k$ terms. The coefficients can be computed in $O(k \log n)$ time because we only need the values h_j and $\mathrm{ne}(c_{i_0,j})$ for each j.

Note that

$$w_{i_0} \cdot R(0) \;=\; w_{i_0} \cdot \sum_{\ell=\ell_0}^{\ell_1} \sum_{j \in J(\ell)} \frac{h_j}{\ell} \;=\; \sum_{j=j_0}^{j_1} \frac{w_{i_0} h_j}{\mathrm{ne}(c_{i_0,j})} \;=\; \sum_{j=j_0}^{j_1} \frac{\mathrm{area}(c_{i_0,j})}{\mathrm{ne}(c_{i_0,j})} \;=\; \sigma(V(i_0, B)).$$

A similar statement holds for all the other vertical slabs within B, as follows.

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 $\mathrm{ne}(c_{i+1,j}) - \mathrm{ne}(c_{i,j})$ is independent of j. See Figure 7. It follows that, for each index i with $i_0 \le i \le i_1$, there is an integer δ_i such that $\mathrm{ne}(c_{i,j}) = \mathrm{ne}(c_{i_0,j}) + \delta_i$ for all j with $j_0 \le j \le j_1$. Moreover, for each i with $i_0 \le i \le i_1$ and each ℓ with $\ell_0 \le \ell \le \ell_1$, the value of $\mathrm{ne}(c_{i,j})$ is constant over all $j \in J(\ell)$. Therefore, for each $j \in J(\ell)$ we have $\mathrm{ne}(c_{i,j}) = \ell + \delta_i$.

Each value δ_i can be obtained using that $\delta_i = \mathrm{ne}(c_{i,j_0}) - \mathrm{ne}(c_{i_0,j_0})$. This means that the values $\delta_{i_0}, \ldots, \delta_{i_1}$ can be obtained in $O(k \log n)$ time.

Now we note that, for each index i with $i_0 \le i \le i_1$, we have

$$w_i \cdot R(\delta_i) \;=\; w_i \cdot \sum_{\ell=\ell_0}^{\ell_1} \sum_{j \in J(\ell)} \frac{h_j}{\ell + \delta_i} \;=\; \sum_{j=j_0}^{j_1} \frac{w_i h_j}{\mathrm{ne}(c_{i,j})} \;=\; \sum_{j=j_0}^{j_1} \frac{\mathrm{area}(c_{i,j})}{\mathrm{ne}(c_{i,j})} \;=\; \sigma(V(i, B)).$$

We use Lemma 1 to evaluate the $i_1 - i_0 + 1 \le k$ values $R(\delta_i)$, where $i_0 \le i \le i_1$, in $O(k \log k) = O(k \log n)$ time. After this, we get each value $\sigma(V(i, B)) = w_i \cdot R(\delta_i)$ in constant time. ◀

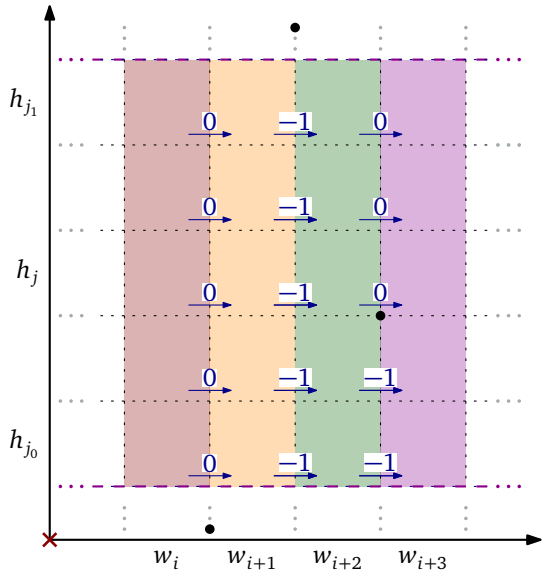

Figure 7 Changes in the values of ne(·) when passing from a vertical slab to the next one. The rightmost transition shows the need to deal with empty blocks for our argument.

4.4 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 $0 < x(p_1) < \cdots < x(p_n)$.

For increasing chains, the problem is actually an AIRPORT game in disguise and Shapley values can be computed using prefix sums. See Figure 8.

▶ **Lemma 6.** *If P is an increasing chain in the positive quadrant, then we can compute the Shapley values of the* AREAANCHOREDRECT *game in $O(n \log n)$.*

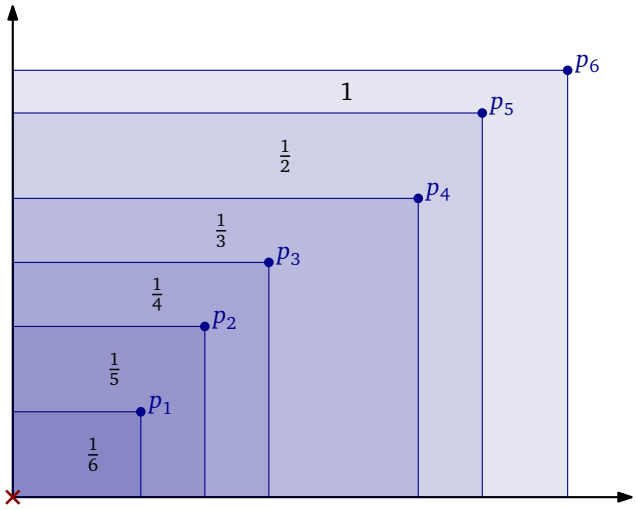

Figure 8 Increasing chain. Each region is marked with the multiplicative weight for its area.

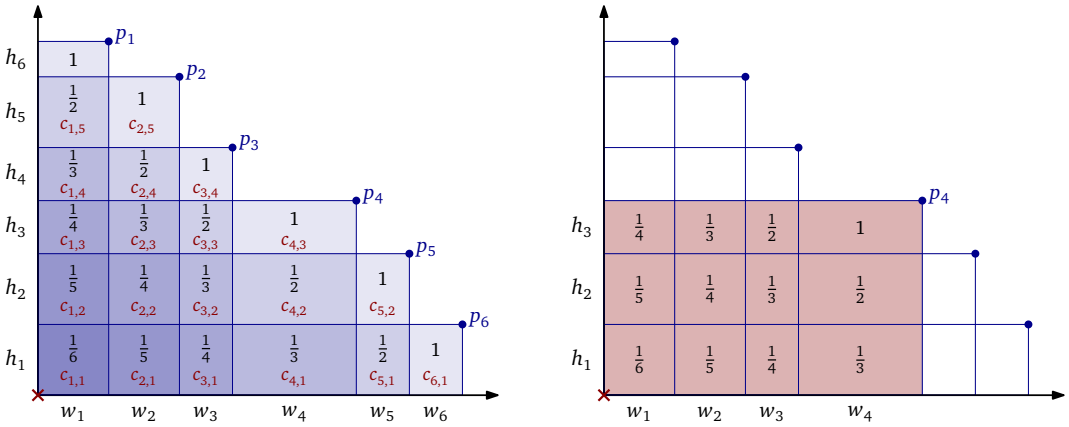

Figure 9 Decreasing chain. Left: Each cell $c_{i,j}$ is marked with the multiplicative weight $\frac{1}{\mathrm{ne}(c)}$ for its area. Right: the cells whose contribution we have to add for the point p_4.

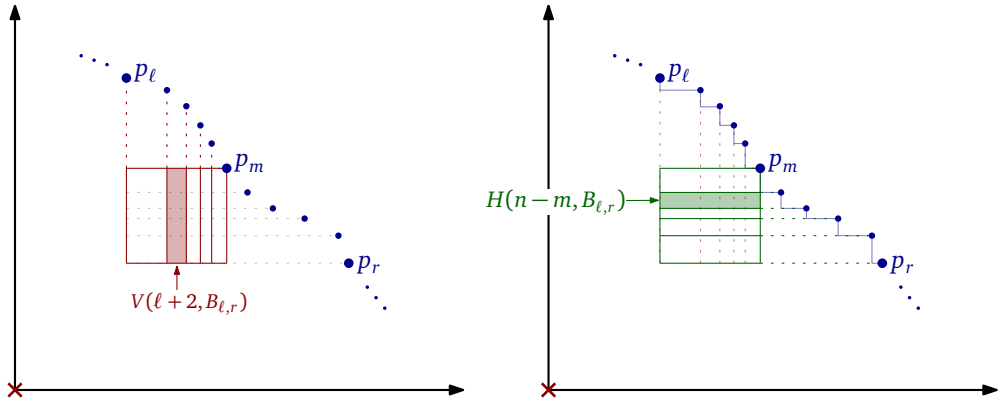

Figure 10 Vertical and horizontal slabs for the divide-and-conquer.

It remains the more interesting case, when the chain is decreasing. See Figure 9 for an example. In this case, the values $\mathrm{ne}(c)$ have a special structure, we do not need data structures to obtain $\mathrm{ne}(c_{i,j})$, and the proof of Lemma 5 can be simplified.

We use a divide-and-conquer paradigm considering certain empty blocks defined by two indices ℓ 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 Figure 10 for a generic case.

The blocks $B_{\ell,r}$ that we consider are in bijection with the intervals (ℓ, r) that correspond to nodes in a binary search tree with values $\{0, \ldots, n+1\}$ at the leaves. For each block $B_{\ell,r}$ considered during the recursion, we compute $\sigma(C)$ for each slab C within $B_{\ell,r}$ using Lemma 5. Furthermore, within each block we compute the sums of $\sigma(\,)$ for prefixes of horizontal and vertical slabs within the block. This means that, for a block $B_{\ell,r}$, we compute

$$\sigma(V(\ell+1, B_{\ell,r})),\ \sigma(V(\ell+1, B_{\ell,r}) \cup V(\ell+2, B_{\ell,r})),\ \ldots\ , \sigma(B_{\ell,r}),$$

and a similar prefix sums for horizontal slabs.

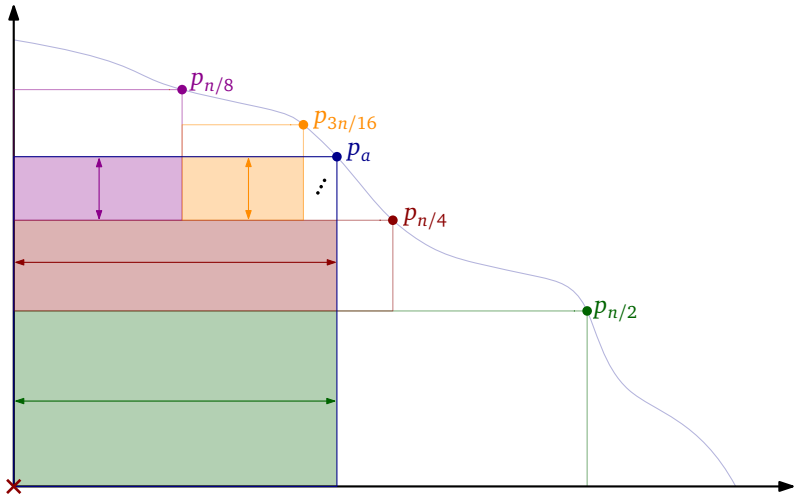

Figure 11 Expressing R_{p_a} as the union of $O(\log n)$ prefixes of slabs within blocks.

For each point $p_a \in P$, the rectangle R_{p_a} is the disjoint union of $O(\log n)$ prefixes of slabs within blocks considered in the algorithm. Whether we use a column prefix or a row prefix of a block $B(\ell, r)$ depends on the relative order of the index a of the point and the median index $m = m(\ell, r)$. See Figure 11. Thus, $\sigma(R_{p_a})$ can be computed as the sum of $O(\log n)$ values that are already computed. This approach leads to the following result.

▶ **Lemma 7.** *If P is a decreasing chain with n points in the positive quadrant, then we can compute the Shapley values of the* AreaAnchoredRect *game in $O(n \log^2 n)$ time.*

When the point set is a chain over different quadrants, we can use symmetry to reduce it to a few problems over the positive quadrant, possibly changing the increasing/decreasing character of chain. From Lemmas 6 and 7 we then obtain the following.

▶ **Theorem 8.** *If P is a chain with n points, then we can compute the Shapley values of the* AreaAnchoredRect *game in $O(n \log^2 n)$ time.*

4.5 General point sets

We consider now the general case, where the points do not form a chain, as in Figure 5. 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, $B = \{c_{i,j} \mid j_0 \le j \le j_1\}$ for some indices $1 \le j_0 \le j_1 \le n$. See Figure 12. 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 $R(p, B) = \{c \in B \mid c \subset R_p\}$.

Each band can be decomposed further into $O(k_B)$ empty blocks, and for each such block we can use Lemma 5 to compute $\sigma(C)$ for each slab C within each block. We can then compute prefix sums of the values $\sigma(\,)$ for the horizontal and vertical slabs within each block, and express $\sigma(R(p, B))$, for each single $p \in P_B$, as the sum of $O(k_B)$ prefixes of rows, at most one per block. This leads to the following.

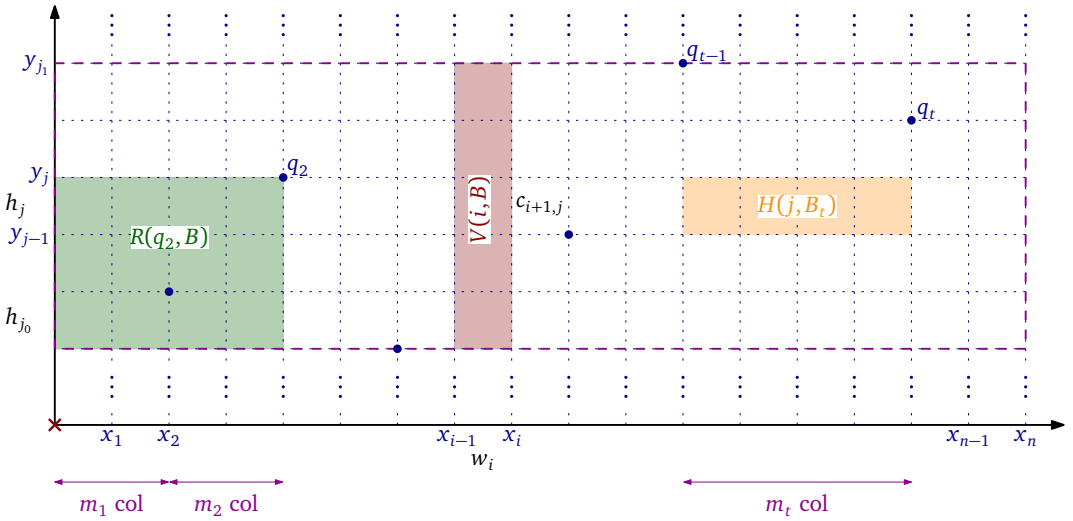

Figure 12 Notation for a band B.

▶ **Lemma 9.** *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 $p \in P_B$.*

Finally, we can express $\sigma(R_p)$ as the sum of at most one prefix sum of vertical slabs per band, and $\sigma(R(p,B))$ for the band B that contains p. Using $O(\sqrt{n})$ bands, each with $k_B = O(\sqrt{n})$, this means that we can compute each $\sigma(R_p)$ as the sum of $O(\sqrt{n})$ values. Again, the relevant values can be computed with some bookkeeping as prefix sums.

▶ **Theorem 10.** *The Shapley values of the* AreaAnchoredRect *game for n points can be computed in $O(n^{3/2}\log n)$ time.*

4.6 Area of the bounding box

Here we would like to present some of the additional challenges when adapting the algorithm for the AreaAnchoredRect game to the AreaBoundingBox game, defined by the characteristic function v_{bb}. A reduction using inclusion-exclusion shows that it suffices to consider the anchored version, AreaAnchoredBoundingBox, when the points are in the positive quadrant.

The geometric interpretation of the Shapley values for the AreaAnchoredBoundingBox game becomes slightly more complicated because a cell c of $\mathcal{A}$ is inside $\mathrm{bb}(Q \cup \{o\})$ if and only if Q contains some point in $\mathrm{NE}(c)$ or it contains some point in $\mathrm{NW}(c)$ and in $\mathrm{SE}(c)$. To obtain a precise description, we define the following values for each cell c of $\mathcal{A}$:

$$\psi_{\mathrm{NE}}(c) = \frac{1}{\mathrm{ne}(c)+\mathrm{nw}(c)} + \frac{1}{\mathrm{ne}(c)+\mathrm{se}(c)} - \frac{1}{\mathrm{ne}(c)+\mathrm{nw}(c)+\mathrm{se}(c)},$$
$$\psi_{\mathrm{NW}}(c) = \frac{1}{\mathrm{ne}(c)+\mathrm{nw}(c)} - \frac{1}{\mathrm{ne}(c)+\mathrm{nw}(c)+\mathrm{se}(c)},$$
$$\psi_{\mathrm{SE}}(c) = \frac{1}{\mathrm{ne}(c)+\mathrm{se}(c)} - \frac{1}{\mathrm{ne}(c)+\mathrm{nw}(c)+\mathrm{se}(c)}.$$

The next lemma provides a geometric description of the Shapley values; see Figure 13.

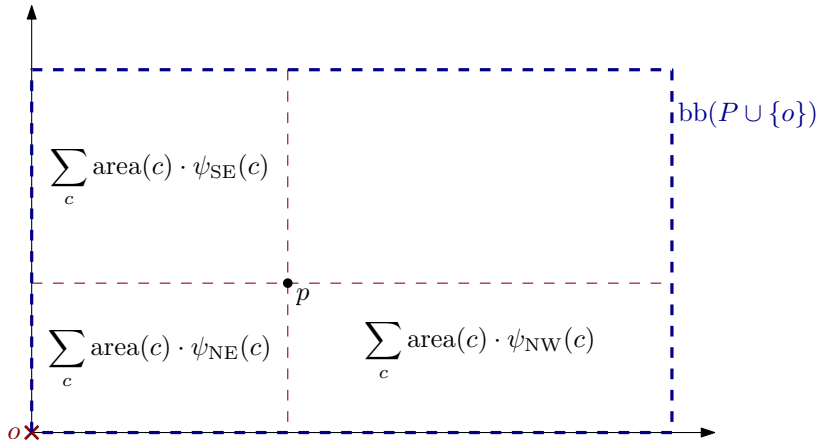

Figure 13 The formula in Lemma 11.

▶ **Lemma 11.** *If P is in the positive quadrant, then for each $p \in P$ the Shapley value $\phi(p, v_{\mathrm{abb}})$ is*

$$\sum_{c \in \mathcal{A},\ p \in \mathrm{NE}(c)} \mathrm{area}(c) \cdot \psi_{\mathrm{NE}}(c) + \sum_{c \in \mathcal{A},\ p \in \mathrm{NW}(c)} \mathrm{area}(c) \cdot \psi_{\mathrm{NW}}(c) + \sum_{c \in \mathcal{A},\ p \in \mathrm{SE}(c)} \mathrm{area}(c) \cdot \psi_{\mathrm{SE}}(c).$$

Following the paradigm used for the AreaAnchoredRect game, we compute

$$\begin{aligned} \sigma_{\mathrm{NE}}(C) &= \sum_{c \in C} \mathrm{area}(c) \cdot \psi_{\mathrm{NE}}(c), \\ \sigma_{\mathrm{NW}}(C) &= \sum_{c \in C} \mathrm{area}(c) \cdot \psi_{\mathrm{NW}}(c), \\ \sigma_{\mathrm{SE}}(C) &= \sum_{c \in C} \mathrm{area}(c) \cdot \psi_{\mathrm{SE}}(c) \end{aligned}$$

for multiple horizontal and vertical slabs C inside empty blocks and inside bands. For this we use again the coherence between adjacent slabs of an empty block. More precisely, for two columns within an empty block, the difference between the counters ne( ), nw( ) and se( ) is independent of the row. See Figure 14. This is the key idea to obtain an analogue of Lemma 5 for the anchored bounding box.

The rest of the approach used for the AreaAnchoredRect game essentially works for the AreaAnchoredBoundingBox game, with a few minor modifications. We refer to the full version [5] for additional information. We summarize the final result.

▶ **Theorem 12.** *The Shapley values of the* AreaAnchoredBoundingBox *and* AreaBoundingBox *games 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.*

5 Conclusions

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. This setting is meaningful, for example, when we split the costs to connect to a fixed landmark. For example, we could use the minimum enclosing disk that contains also the origin. We do this for the anchored versions of the games. Our results also hold in this setting through an easy adaptation.

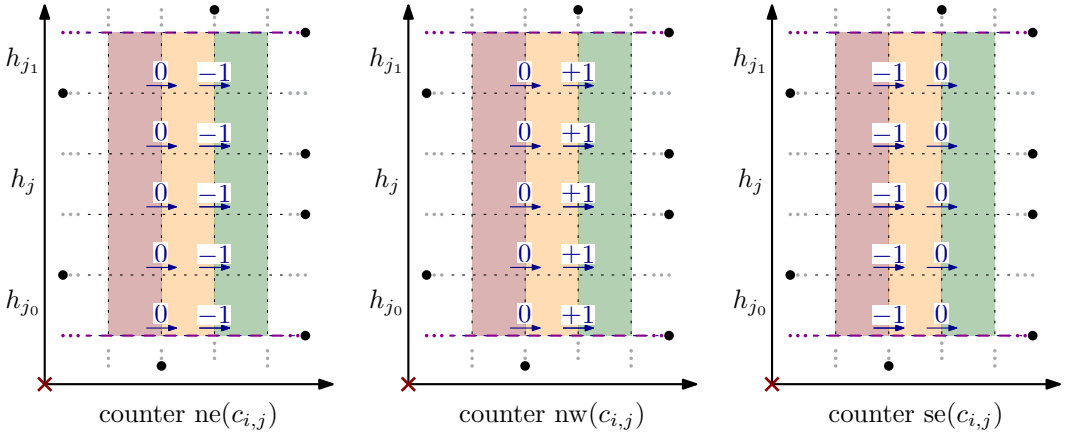

Figure 14 Changes in the counter ne(·) (left), nw(·) (center) and se(·) (right) depending on the position of the points of P.

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 [13]. Does this imply the same for the coalitional game based on the length of the Euclidean MST? Is there also a FPRAS for computing the Shapley values of the game defined using the Euclidean MST? Since the length of the Euclidean spanning tree is not monotone, a priori some Shapley values could be potentially 0, which, at least intuitively, makes it harder to get approximations.

In the coalitional games that we consider, the Shapley values that we compute can be interpreted as the relevance of each point within the point set for different geometric concepts. It would be worthwhile to understand whether there is some relation between Shapley values in geometric settings and the concept of depth in point sets, like the Tukey depth. For stochastic points, this relation has been explored and exploited by Agarwal et al. [1].

References

1 Pankaj K. Agarwal, Sariel Har-Peled, Subhash Suri, Hakan Yildiz, and Wuzhou Zhang. Convex Hulls Under Uncertainty. *Algorithmica*, 79(2):340–367, 2017. doi:10.1007/s00453-016-0195-y.

2 Pankaj K. Agarwal, Nirman Kumar, Stavros Sintos, and Subhash Suri. Range-max queries on uncertain data. *J. Comput. Syst. Sci.*, 94:118–134, 2018. doi:10.1016/j.jcss.2017.09.006.

3 Deepak Ajwani, Saurabh Ray, Raimund Seidel, and Hans Raj Tiwary. On Computing the Centroid of the Vertices of an Arrangement and Related Problems. In Frank K. H. A. Dehne, Jörg-Rüdiger Sack, and Norbert Zeh, editors, *10th International Workshop Algorithms and Data Structures, WADS 2007*, volume 4619 of *Lecture Notes in Computer Science*, pages 519–528. Springer, 2007. doi:10.1007/978-3-540-73951-7_45.

4 Boris Aronov and Matthew J. Katz. Batched Point Location in SINR Diagrams via Algebraic Tools. *ACM Trans. Algorithms*, 14(4):41:1–41:29, 2018. doi:10.1145/3209678.

5 Sergio Cabello and Timothy M. Chan. Computing Shapley values in the plane. *CoRR*, abs/1804.03894, 2018. arXiv:1804.03894.

6 Georgios Chalkiadakis, Edith Elkind, and Michael Wooldridge. *Computational Aspects of Cooperative Game Theory*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2011. doi:10.2200/S00355ED1V01Y201107AIM016.

7 Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational Geometry: Algorithms and Applications*. Springer, 3rd edition, 2008.

8 Xiaotie Deng and Qizhi Fang. Algorithmic Cooperative Game Theory. In Altannar Chinchuluun, Panos M. Pardalos, Athanasios Migdalas, and Leonidas Pitsoulis, editors, *Pareto Optimality, Game Theory And Equilibria*, pages 159–185. Springer New York, 2008. doi:10.1007/978-0-387-77247-9_7.

9 Xiaotie Deng and Christos H. Papadimitriou. On the Complexity of Cooperative Solution Concepts. *Mathematics of Operations Research*, 19(2):257–266, 1994. doi:10.1287/moor.19.2.257.

10 Ulrich Faigle, Sándor P. Fekete, Winfried Hochstättler, and Walter Kern. On approximately fair cost allocation in Euclidean TSP games. *Operations-Research-Spektrum*, 20(1):29–37, 1998. doi:10.1007/BF01545526.

11 Thomas S. Ferguson. Game Theory, 2nd edition, 2014. Electronic text available at https://www.math.ucla.edu/~tom/Game_Theory/Contents.html.

12 Martin Fink, John Hershberger, Nirman Kumar, and Subhash Suri. Hyperplane separability and convexity of probabilistic point sets. *JoCG*, 8(2):32–57, 2017. URL: http://jocg.org/index.php/jocg/article/view/321.

13 Pegah Kamousi, Timothy M. Chan, and Subhash Suri. Stochastic minimum spanning trees in Euclidean spaces. In Ferran Hurtado and Marc J. van Kreveld, editors, *Proceedings of the 27th ACM Symposium on Computational Geometry, SoCG'11*, pages 65–74. ACM, 2011. doi:10.1145/1998196.1998206.

14 Pegah Kamousi, Timothy M. Chan, and Subhash Suri. Closest pair and the post office problem for stochastic points. *Comput. Geom.*, 47(2):214–223, 2014. doi:10.1016/j.comgeo.2012.10.010.

15 Stefan Langerman. On the Complexity of Halfspace Area Queries. *Discrete & Computational Geometry*, 30(4):639–648, 2003. doi:10.1007/s00454-003-2856-2.

16 Stephen C. Littlechild and Guillermo Owen. A Simple Expression for the Shapely Value in A Special Case. *Management Science*, 20(3):370–372, 1973. doi:10.1287/mnsc.20.3.370.

17 Nimrod Megiddo. Computational Complexity of the Game Theory Approach to Cost Allocation for a Tree. *Mathematics of Operations Research*, 3(3):189–196, 1978. doi:10.1287/moor.3.3.189.

18 Guillaume Moroz and Boris Aronov. Computing the Distance between Piecewise-Linear Bivariate Functions. *ACM Trans. Algorithms*, 12(1):3:1–3:13, 2016. doi:10.1145/2847257.

19 Roger B. Myerson. *Game theory - Analysis of Conflict*. Harvard University Press, 1997.

20 Noam Nisan, Tim Roughgarden, Éva Tardos, and Vijay V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, 2007.

21 Martin J. Osborne and Ariel Rubinstein. *A Course in Game Theory*. The MIT Press, 1994.

22 Mark H. Overmars and Chee-Keng Yap. New Upper Bounds in Klee's Measure Problem. *SIAM J. Comput.*, 20(6):1034–1045, 1991. doi:10.1137/0220065.

23 Pablo Pérez-Lantero. Area and Perimeter of the Convex Hull of Stochastic Points. *Comput. J.*, 59(8):1144–1154, 2016. doi:10.1093/comjnl/bxv124.

24 Justo Puerto, Arie Tamir, and Federico Perea. A cooperative location game based on the 1-center location problem. *European Journal of Operational Research*, 214(2):317–330, 2011. doi:10.1016/j.ejor.2011.04.020.

25 Justo Puerto, Arie Tamir, and Federico Perea. Cooperative location games based on the minimum diameter spanning Steiner subgraph problem. *Discrete Applied Mathematics*, 160(7-8):970–979, 2012. doi:10.1016/j.dam.2011.07.020.

26 Alvin E. Roth, editor. *The Shapley Value: Essays in Honor of Lloyd S. Shapley*. Cambridge University Press, 1988.

27 William Thomson. Cost allocation and airport problems, 2013. Rochester Center for Economic Research Working Paper. Version of 2014 available at http://www.iser.osaka-u.ac.jp/collabo/20140524/Airport_Problems.pdf.

28 Dan E. Willard. New Data Structures for Orthogonal Range Queries. *SIAM J. Comput.*, 14:232–253, 1985. doi:10.1137/0214019.

29 Eyal Winter. The Shapley value. In R.J. Aumann and S. Hart, editors, *Handbook of Game Theory with Economic Applications*, volume 3, chapter 53, pages 2025–2054. Elsevier, 1 edition, 2002. doi:10.1016/S1574-0005(02)03016-3.

30 Jie Xue, Yuan Li, and Ravi Janardan. On the separability of stochastic geometric objects, with applications. *Comput. Geom.*, 74:1–20, 2018. doi:10.1016/j.comgeo.2018.06.001.