

Computing Shapley values in the plane

Sergio Cabello*

Timothy M. Chan[†]

November 30, 2018

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 and the perimeter 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.

Keywords: Shapley values; stochastic computational geometry; convex hull; minimum enclosing disk; bounding box; arrangements; convolutions; airport problem

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.

Formally, a *coalitional game* is a pair (N, v) , where N is the set of players and $v: 2^N \rightarrow \mathbb{R}$ 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.

In our setting, the players will be points in the plane, and we will use $P \subset \mathbb{R}^2$ for the set of players. 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 defined through the area, such as the area of the convex hull. As mentioned before, such area could be interpreted as a cost or a payoff associated to each subset of the points P , depending on the problem.

Shapley values are probably the most popular solution concept for coalitional games, which in turn are a very common model for cooperative games with transferable utility. The objective

*Department of Mathematics, IMFM, and Department of Mathematics, FMF, University of Ljubljana, Slovenia. Supported by the Slovenian Research Agency, program P1-0297 and projects J1-8130, J1-8155. Email address: sergio.cabello@fmf.uni-lj.si.

[†]Department of Computer Science, University of Illinois at Urbana-Champaign, USA. Email address: tmc@illinois.edu.

is to split the value $v(N)$ between the different players in a meaningful way. It is difficult to overestimate the relevance of Shapley values. See the book edited by Roth [25] 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. In our context, the Shapley values that we compute can be interpreted as the relevance of each point within the point set for different geometric concepts.

In a nutshell, the Shapley value of a player p is the expected increase in the value of the characteristic function when inserting the player p , if the players are inserted in a uniformly random order. We provide a formal definition of Shapley values later on. Since the definition is based on considering all the permutations of the players, this way to compute Shapley values is computationally unfeasible. In fact, there are several natural instances where computing Shapley values is difficult. Thus, there is an interest to find scenarios where computing the Shapley values is feasible. In this paper we provide some natural scenarios and discuss the efficient computation.

The problems we consider here can be seen as natural extensions to \mathbb{R}^2 of the classical airport problem considered by Littlechild and Owen [15]. In the airport problem, 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 [26].

The airport problem is closely related to the problem of allocating the length of the smallest interval that contains a set of points on the line. There are several natural generalizations of the intervals when we go from one to two dimensions, and in this paper we consider several of those natural generalizations.

Coalitional games in the plane. The shapes that we consider in this paper are succinctly described in Figure 1. We next provide a summary of the main coalitional games that we consider; other simpler auxiliary games are considered later. In all cases, the set of players is a finite subset $P \subset \mathbb{R}^2$.

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 .

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 $v_{\text{abb}}(Q) = \text{area}(\text{bb}(Q \cup \{o\}))$ for each nonempty $Q \subset P$, where o is the origin.

In our discussion we focus 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 . Handling the perimeter is either substantially easier or can be solved by similar methods, and it will be discussed along the way.

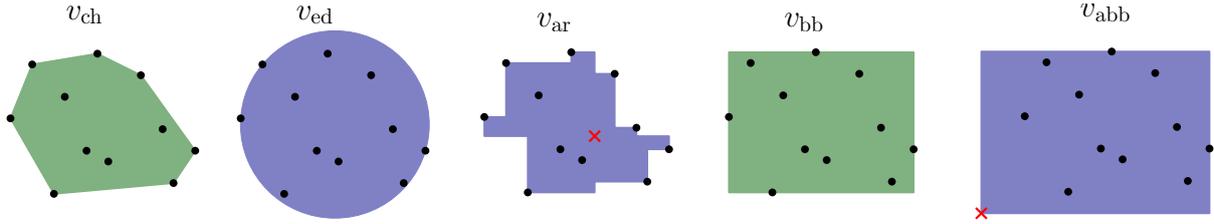


Figure 1: Different costs associated to a point set that are considered in this paper. The cross represents the origin. In all cases we focus on the area.

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

Overview of our contribution. We show that the Shapley values for the `AREA CONVEX HULL` and `AREA ENCLOSING DISK` 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, 22, 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, 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 3 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 but split it into two computations: the expected value after the insertion of p , if the insertion of p changed the shape; the expected value before the insertion of p , if the insertion of p changed the shape. Finally, we use arrangements of lines and planes to speed up the computation by an additional linear factor.

For the `AREA ANCHORED RECTANGLES`, `AREA BOUNDING BOX`, and `AREA ANCHORED BOUNDING BOX` 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 computer algebra in computational geometry is very recent, and there are very few results [3, 4, 18] using such techniques in geometric problems. (In contrast, there is a quickly growing body of works in *discrete* geometry using real algebra.)

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 chapter by Deng and Fang [6] gives 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.

Puerto, Tamir and Perea [23] study the the Minimum Radius Location Game in general metric spaces. When specialized to the Euclidean plane, this is equivalent to the PERIMETERENCLOSINGDISK. The paper also considers the L_1 -metric, which is in some way related to the axis-parallel problems we consider here. However, the focus of their research 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 [24] 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 and specialize some results to the Euclidean plane. They also analyze an approximate allocation of the costs.

The computation of Shapley values have 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 [7] discuss a game on arbitrary graphs defined by induced subgraphs. 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 Section 2, where we set the notation, define 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 long because of our use of tools from different areas.

Then we analyze different games. The AREACONVEXHULL and AREAENCLOSINGDISK games are considered in Section 3 and Section 4, respectively. In Section 5 we discuss the AREAANCHOREDRECTANGLES game, while in Section 6 we discuss the AREABOUNDINGBOX and AREAANCHOREDBOUNDINGBOX games. To understand Section 6 one has to go through Section 5 before. The remaining sections (and games) have no dependencies between them and can be read in any order. We conclude in Section 7 with some discussion.

2 Preliminaries

Here we provide notation and background used through the paper.

Geometry. 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 smallest (actually, *the* smallest) disk that contains 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.

We provide a condensed overview suitable for understanding our work. We refer to some textbooks in Game Theory ([11, Chapter IV.3], [19, Section 9.4], [21, Section 14.4]) for a comprehensive treatment that includes a discussion of the properties of Shapley values.

Consider a coalitional game (P, v) . For us, a player is usually denoted by $p \in P$. We denote by n the number of players and by $[n]$ the set of integers $\{1, \dots, n\}$.

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 π . 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

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

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

$$\phi(p, v) = \frac{1}{n!} \sum_{\pi \in \Pi(P)} \Delta(v, \pi, p).$$

It is straightforward to note that

$$\phi(p, v) = \mathbb{E}_{\pi} [\Delta(v, \pi, p)],$$

where π is picked uniformly at random from $\Pi(P)$. This means that Shapley values describe the expected increase in $v(\cdot)$ when we insert the player p in a random permutation.

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

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

We will use a similar principle in our algorithms: find groups of permutations that contribute the same to the sums. For non-axis-parallel games we will also use other equivalent expressions of the Shapley value that follow from rewriting the sum differently.

It is easy to see that Shapley values are linear in the characteristic functions. Indeed, since for any two characteristic functions $v_1, v_2: 2^P \rightarrow \mathbb{R}$ 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 $\rho: \mathbb{R}^2 \rightarrow \mathbb{R}^2$ 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 $p \in P$.

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 and the characteristic function is $v_{\text{ab}}(Q) = (\max(y(P)) - \min(y(P))) \cdot v_{\text{li}}(x(Q))$ for each nonempty $Q \subset P$.

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

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

Lemma 1. *The AIRPORT, LENGTHINTERVAL, AREABAND games, PERIMETERBOUNDINGBOX and PERIMETERANCHOREDBOUNDINGBOX games can be solved in $O(n \log n)$ time.*

Proof. The AIRPORT game was solved by Littlechild and Owen [15], as follows. If P is given by $0 < x_1 < \dots < x_n$, then

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

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

$$\begin{aligned} v_1(Q) &= \max(Q) - \alpha \text{ for } Q \neq \emptyset; \\ v_2(Q) &= \beta - \min(Q) \text{ for } Q \neq \emptyset; \\ v_3(Q) &= \alpha - \beta \text{ for } Q \neq \emptyset. \end{aligned}$$

They define games that are isometrically equivalent to AIRPORT games or a constant game, and we have $v_{\text{li}}(Q) = v_1(Q) + v_2(Q) + v_3(Q)$ for all nonempty $Q \subset P$. The result follows from the linearity of the Shapley values.

For the AREABAND game we just notice that $v_{\text{ab}}(Q) = v_{\text{li}}(Q) \cdot (\max(y(P)) - \min(y(P)))$ and use again the linearity of Shapley values. (The value $\max(y(P)) - \min(y(P))$ is constant.) For the PERIMETERBOUNDINGBOX game we note that

$$v_{\text{pbb}}(Q) = 2 \cdot (\max(x(P)) - \min(x(P))) + 2 \cdot (\max(y(P)) - \min(y(P))),$$

which means that we need to compute the Shapley values for two LENGTHINTERVAL games. For the PERIMETERANCHOREDBOUNDINGBOX we use that

$$v_{\text{pabb}}(Q) = 2 \cdot (\max(x(P), 0) - \min(x(P), 0)) + 2 \cdot (\max(y(P), 0) - \min(y(P), 0)),$$

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.

Remark. 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 $\phi(x_n, v_{\text{air}}) = \sum_{i \in [n]} \frac{1}{i}$, the n -th harmonic number. Thus, even for simple games it would become a challenge to carry precise bounds on the number of bits.

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, \dots, a_n and b_0, \dots, 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 integer 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 [4, 18] for multipoint rational functions gives a slightly worse running time.

Lemma 2. *Let b_0, \dots, b_n, Δ 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 $x = \ell, \ell + 1, \dots, \ell + m$ in $O((n + m) \log(n + m))$ time.

Proof. Set $a_i = 1/(\Delta + \ell + m - i)$ for all $i \in \{0, \dots, m\}$. 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 $k \in \{0, \dots, m\}$ and compute them using the fast Fourier transform in $O((n + m) \log(n + m))$ time. Then, for each integer x we have

$$R(x) = \sum_{t=0}^n \frac{b_t}{\Delta + t + x} = \sum_{t=0}^n b_t a_{m+\ell-t-x} = c_{m+\ell-x}.$$

Thus, from computing the convolution of a_* and b_* , we get the values $R(\ell), \dots, R(\ell + m)$ \square

2.4 Permutations.

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

Lemma 3. *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 $n! \cdot \frac{\alpha! \beta!}{(\alpha + \beta + 1)!}$ 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 α elements of A arbitrarily and put them after x . Similarly, permute the β 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

$$\alpha! \cdot \beta! \cdot ((\alpha + \beta + 2) \cdots n) = \frac{n! \alpha! \beta!}{(\alpha + \beta + 1)!}. \quad \square$$

Lemma 4. *Let N be a set with n elements. Let A , B and C be pairwise disjoint subsets of N with α , β and γ 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 behind a .)*
- *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 Π_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 Π_X are produced with this procedure exactly once. Therefore, there are

$$(\alpha + |X| - 1)! \cdot ((\alpha + |X| + 1) \cdot (\alpha + |X| + 2) \cdots n) = n! \frac{1}{\alpha + |X|}$$

permutations in Π_X . Using inclusion-exclusion we have

$$\begin{aligned} |\Pi_B \cup \Pi_C| &= |\Pi_B| + |\Pi_C| - |\Pi_B \cap \Pi_C| \\ &= |\Pi_B| + |\Pi_C| - |\Pi_{B \cup C}| \\ &= n! \cdot \left(\frac{1}{\alpha + \beta} + \frac{1}{\alpha + \gamma} - \frac{1}{\alpha + \beta + \gamma} \right). \end{aligned}$$

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 $n! \cdot \frac{\gamma}{(\alpha + \beta)(\alpha + \beta + \gamma)}$. Then we use that

$$\frac{1}{\alpha + \beta} - \frac{1}{\alpha + \beta + \gamma} = \frac{\gamma}{(\alpha + \beta)(\alpha + \beta + \gamma)}. \quad \square$$

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_{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.

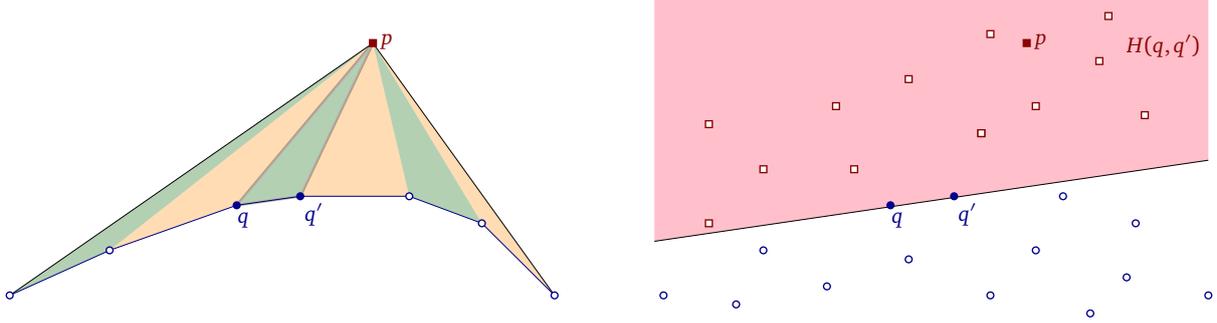


Figure 2: Left: triangulating the difference between $CH(P(\pi, p))$ and $CH(P(\pi, p) \setminus \{p\})$. Right: in order for $\Delta pqq'$ to be in $T(\pi, p)$, q and q' must appear before p , which in turn must appear before all other points in the halfplane $H(q, q')$.

Lemma 5. 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 $CH(P(\pi, p))$ and $CH(P(\pi, p) \setminus \{p\})$ into a set $T(\pi, p)$ of triangles (see Figure 2 (left)), where

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

In other words, $\Delta pqq' \in T(\pi, p)$ 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 $p \in H(q, q')$, Lemma 3 with $x = p$, $B = \{q, q'\}$ and $A = H(q, q') \cap P \setminus \{p\}$ tells that the probability that $\Delta pqq' \in T(\pi, p)$ 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

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

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 [5, Chapter 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 5 to all points $p \in P$ gives $O(n^3)$ total time. We can speed up the algorithm by a factor of n :

Theorem 6. The Shapley values of the AREA CONVEXHULL 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 $\text{area}(\triangle pq'q)$ 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).

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 $O(n^2)$. \square

Theorem 7. *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 π and no points before p in π lie in $H(q, p)$; in other words, among the $\text{level}(q, p) + 2$ points in $H(q, p) \cup \{p, q\}$, the earliest point is q and the second earliest point is p . We can use Lemma 3 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 π) is exactly

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

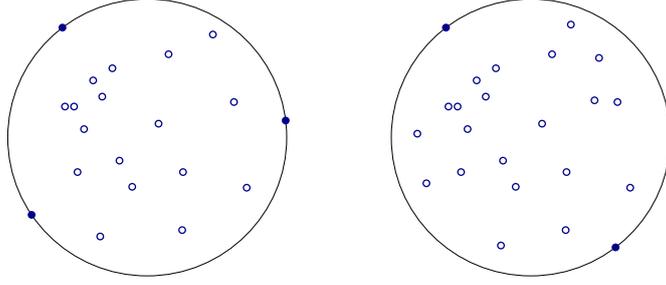


Figure 3: The minimum enclosing disk for two sets Q of points. Left: $X(Q)$ has three elements. Right: $X(Q)$ has two elements.

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 5 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

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

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 6, in $O(n^2)$ total time (in fact, the algorithm is a little simpler, since linear functions are not required). \square

4 Smallest enclosing disk

In this section we consider the area and the perimeter of the smallest enclosing disk of the points. Recall that v_{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 8. *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 [27] or [5, Section 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 Figure 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 π , and no points before p in π 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

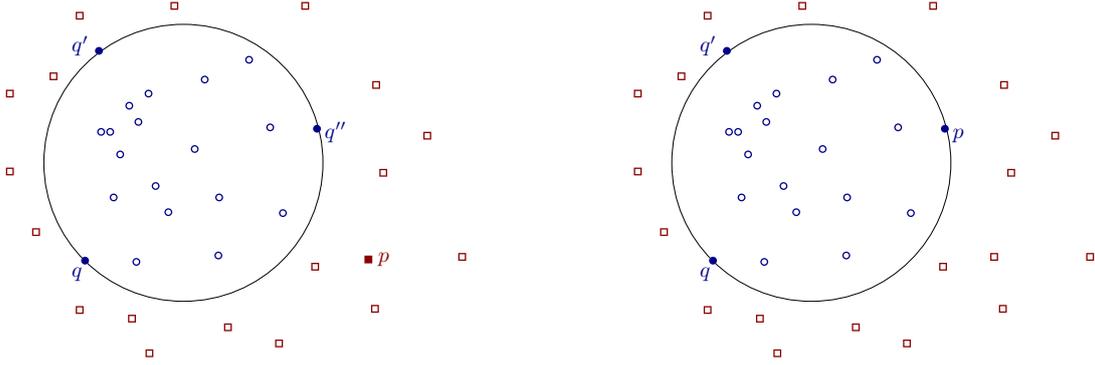


Figure 4: Left: in order for the shown disk to be $\text{med}(P(\pi, p) \setminus \{p\})$, the points q, q', q'' must appear before p , which in turn must appear before all other points outside the disk. Right: in order for the shown disk to be $\text{med}(P(\pi, p))$, the points q, q' must appear before p , which in turn must appear before all points outside the disk.

earliest point is p . See Figure 4 (left). Lemma 3 implies that, for a fixed B and a fixed p , the probability that $X(P(\pi, p) \setminus \{p\}) = B$ (with respect to a random permutation π) is exactly

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

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 $\text{area}(\text{med}(P(\pi, p) \setminus \{p\})) \cdot I(\pi, p)$ is

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

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 π , and no points before p in π lie in $\text{out}(B)$. In other words, among the $\text{level}(B) + |B|$ 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 Figure 4 (right). We can bound the probability of this event using Lemma 3. Thus, for a fixed B and $p \in B$, the probability that $X(P(\pi, p) \setminus \{p\}) = B$ (with respect to a random permutation π) is exactly

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

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

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

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

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 3 dimensions), the problem reduces to compute the *level* of $O(n^3)$ vertices in an arrangement of n planes in 3 dimensions. See [16, Section 5.7] for a discussion on the transformation. The arrangement of n planes can be constructed in $O(n^3)$ time [8, 9], and

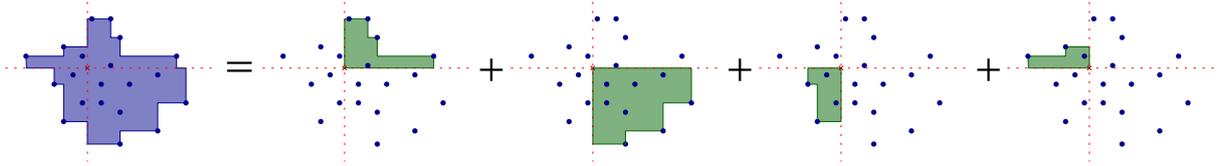


Figure 5: It is enough to consider the cases where all points are in a quadrant.

the levels of all vertices can be subsequently generated by traversing the arrangement in $O(1)$ time per vertex.

For the perimeter, we just use the perimeter of $\text{med}(B)$ instead of the area in the computations. \square

Naively applying Lemma 8 to all points $p \in P$ gives $O(n^4)$ total time. We can speed up the algorithm by a factor of n :

Theorem 9. *The Shapley values of the AREAENCLOSINGDISK and 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 3 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 3 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 6 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 $O(n^3)$. \square

5 Union of anchored rectangles

In this section we consider the 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 Figure 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 Section 6.

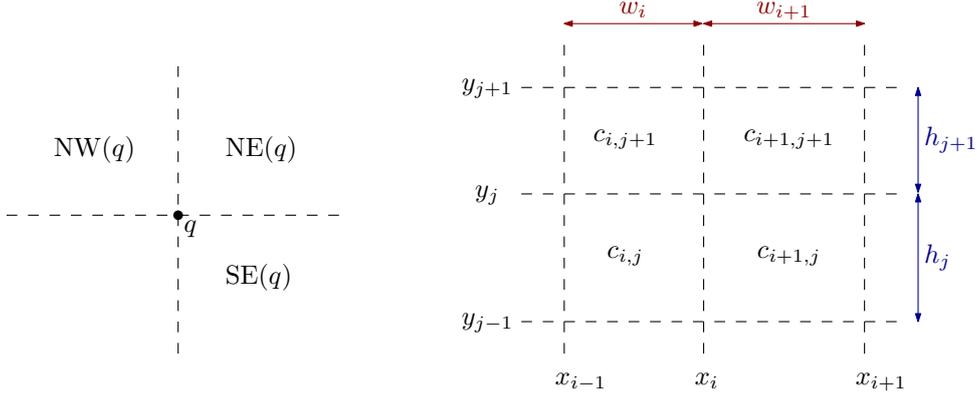


Figure 6: Notation for axis-parallel problems. Left: the quadrants to define $\text{NW}(q)$, $\text{NE}(q)$ and $\text{SE}(q)$. Right: cells of \mathcal{A} .

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 :

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

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 Figure 6, left.

Let $x_1 < \dots < x_n$ denote the x -coordinates of the points of P , and let $y_1 < \dots < y_n$ 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 axis. 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 $\mathcal{A} = \mathcal{A}(L)$. More precisely, a (2-dimensional) cell c of \mathcal{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 \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 6, right.

Since $\text{NE}(q)$ is constant over each 2-dimensional cell c of \mathcal{A} , we can define $\text{NE}(c)$, for each cell $c \in \mathcal{A}$. 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 Figure 7.

A **block** 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 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 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 slab. We call them the **slabs within B** .

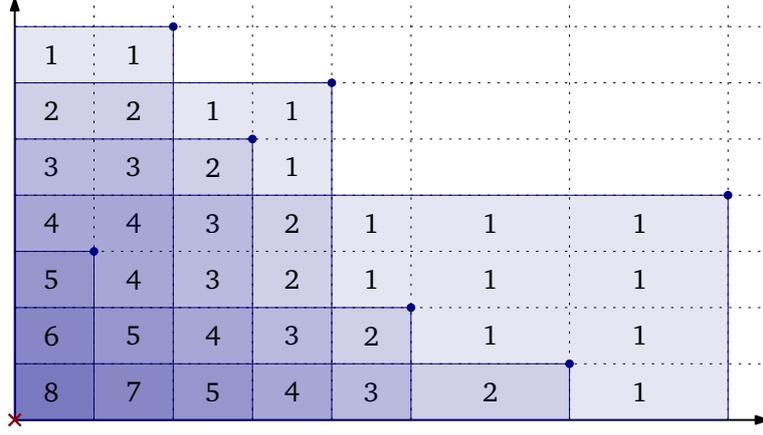


Figure 7: The non-zero counters $ne(c)$ for the bounded cells c of \mathcal{A} . The intensity of the color correlates with the counter $ne(c)$.

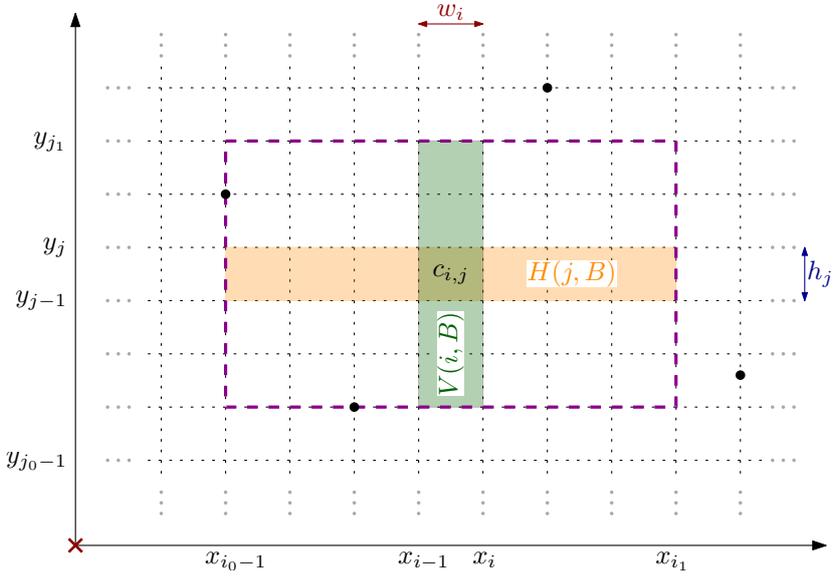


Figure 8: An *empty* block $B = B(i_0, i_1, j_0, j_1)$ with a vertical and a horizontal slab shaded.

5.2 Interpreting Shapley values geometrically

First, we reduce the problem of computing Shapley values to a purely geometric problem. See Figure 7 for the relevant counters considered in the following result.

Lemma 10. *If P is in the positive quadrant, then for each $p \in P$ we have*

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

Proof. Each cell c of the arrangement \mathcal{A} defines a game for the set of players P 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 $NE(c) = \{p \in P \mid c \subset R_p\}$ 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,

$$v_{\text{ar}}(Q) = \sum_{c \in \mathcal{A}} v_c(Q) \quad \text{for all } Q \subset P,$$

and because of linearity of Shapley values we get

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

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. 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 [28] or [5, Chapter 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 [5, Section 10.3] may be useful. We did not see how to work out the details of this; the problem is that the weights 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 [28]. This is a standard range counting for orthogonal ranges and the preprocessing has to be done only once.

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 11. *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 $i_0 \leq i \leq i_1$. The computation for the horizontal slabs $\sigma(H(j_0, B)), \dots, \sigma(H(j_1, B))$ 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 ℓ we define $J(\ell) = \{j \mid j_0 \leq j \leq j_1, \text{ne}(c_{i_0, j}) = \ell\}$. Let ℓ_0 and ℓ_1 be the minimum and the maximum ℓ such that $J(\ell) \neq \emptyset$, respectively.

We set up the following rational function with variable x :

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

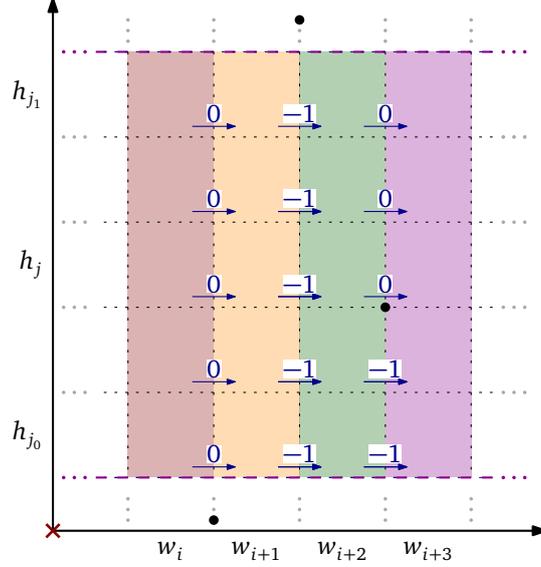


Figure 9: Changes in the values of $\text{ne}(\cdot)$ when passing from a vertical slab to the next one. The rightmost transition shows the need to deal with empty blocks for our argument.

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 2 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 $\text{ne}(c_{i_0, j})$ for each j . These latter values $\text{ne}(\cdot)$ are obtained from range counting queries.

Note that

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

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 $\text{ne}(c_{i+1, j}) - \text{ne}(c_{i, j})$ is independent of j . See Figure 9. It follows that, for each index i with $i_0 \leq i \leq i_1$, there is an integer δ_i such that $\text{ne}(c_{i, j}) = \text{ne}(c_{i_0, j}) + \delta_i$ 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 ℓ with $\ell_0 \leq \ell \leq \ell_1$, the value of $\text{ne}(c_{i, j})$ is constant over all $j \in J(\ell)$. Therefore, for each $j \in J(\ell)$ we have $\text{ne}(c_{i, j}) = \ell + \delta_i$.

Each value δ_i can be obtained using that $\delta_i = \text{ne}(c_{i, j_0}) - \text{ne}(c_{i_0, j_0})$. This means that the values $\delta_{i_0}, \dots, \delta_{i_1}$ 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

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

We use Lemma 2 to evaluate the $i_1 - i_0 + 1 \leq k$ values $R(\delta_i)$, where $i_0 \leq i \leq i_1$, in $O(k \log k) = O(k \log n)$ time. After this, we get each value $\sigma(V(i, B)) = w_i \cdot R(\delta_i)$ in constant time. \square

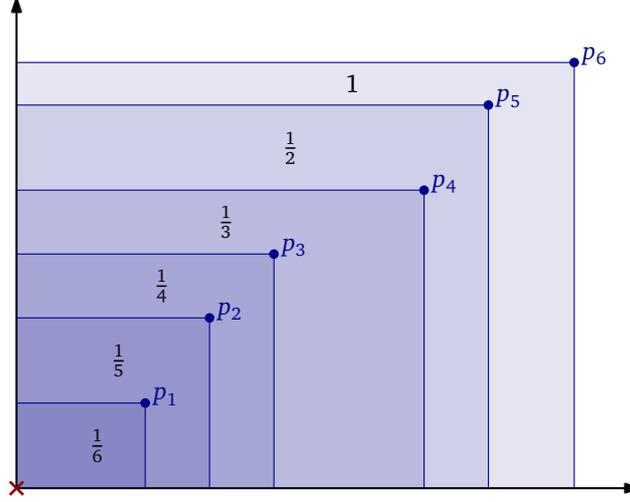


Figure 10: Increasing chain in the positive quadrant. Each region is marked with the multiplicative weight for its area.

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 $0 < x(p_1) < \dots < x(p_n)$.

We start with the easier case: increasing chains. The problem is actually an AIRPORT game in disguise.

Lemma 12. *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 Figure 10. Because of Lemma 10, we have for each $i \in [n]$

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

Therefore $\phi(p_1, v_{\text{ar}}) = z_1$ and, for each $i \in [n] \setminus \{1\}$, we have $\phi(p_i, v_{\text{ar}}) = z_i + \phi(p_{i-1}, v_{\text{ar}})$. 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 $\text{area}(R_{p_i})$.) \square

It remains the more interesting case, when the chain is decreasing. See Figure 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 11 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 Figure 12.)

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

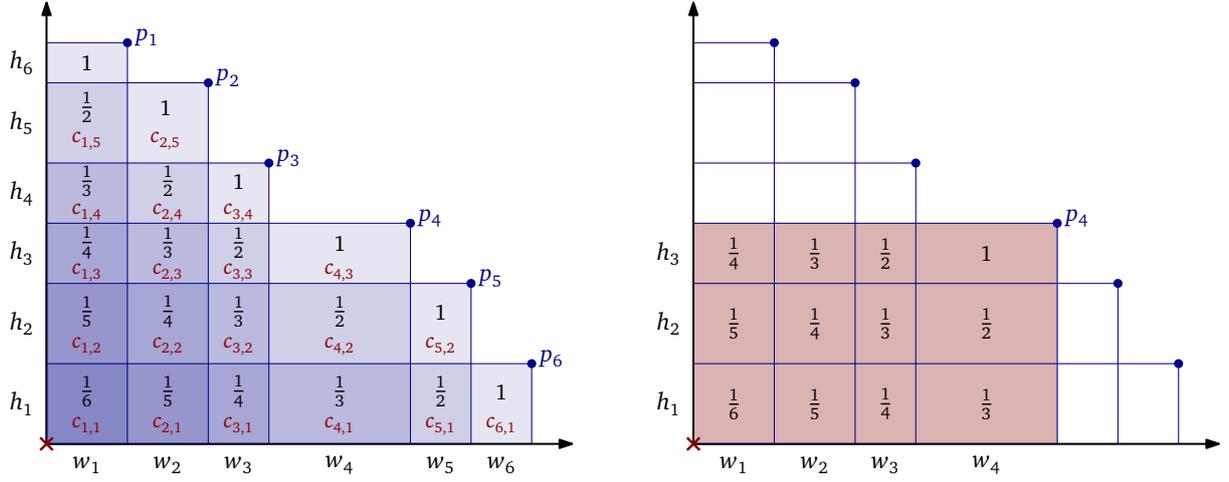


Figure 11: Decreasing chain. Left: Each cell $c_{i,j}$ is marked with the multiplicative weight $\frac{1}{ne(c)}$ for its area. Right: the cells whose contribution we have to add for the point p_4 .

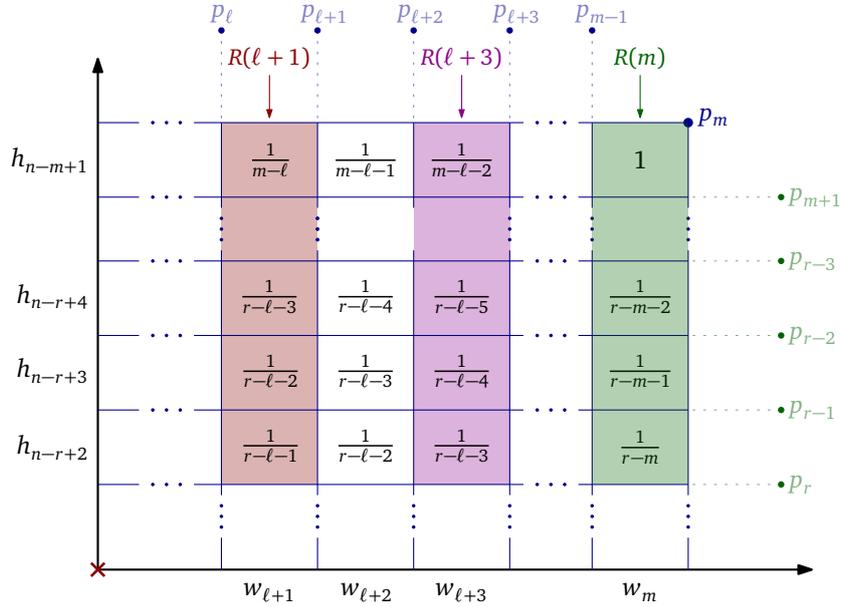


Figure 12: Values $1/ne(c)$ in a decreasing chain.

$m = m(\ell, r) = \lfloor (\ell + r)/2 \rfloor$. 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 13 for the base case, and Figure 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 is encoded by the slabs within the block. This approach leads to the following result.

Lemma 13. *If P is a decreasing chain with n points in the positive quadrant, then we can compute the Shapley values of the 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 (ℓ, r) defined by $\ell = \alpha 2^\beta$ and $r = (\alpha + 1)2^\beta \leq n + 1$, where α and β 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 11. This takes $O((r - \ell) \log n)$ time. The pairs (ℓ, r) of \mathcal{I} can also be interpreted as intervals. Since intervals of

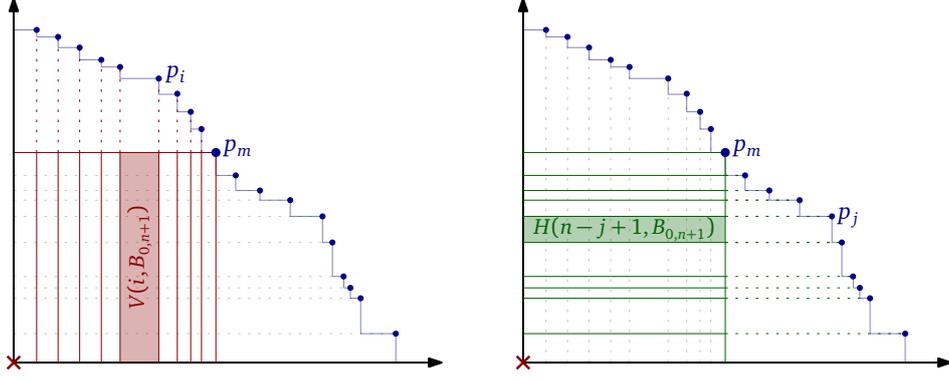


Figure 13: Vertical and horizontal slabs for the base case, where we consider the block $B_{0,n+1}$.

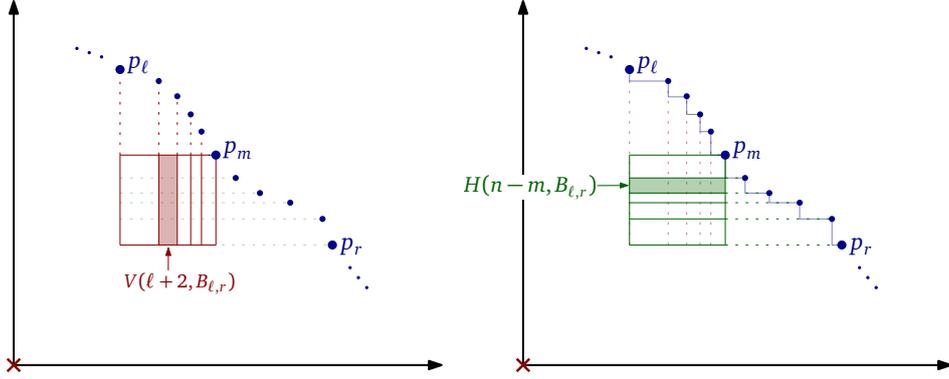


Figure 14: Vertical and horizontal slabs for the divide-and-conquer (general case).

\mathcal{I} with the same length are disjoint and there are $O(\log n)$ different possible lengths of intervals in \mathcal{I} , the computation over all intervals of \mathcal{I} and all indices takes $O(n \log^2 n)$ time.

For each $(\ell, r) \in \mathcal{I}$ we compute some additional prefix sums of columns and of rows, as follows. Assume that $B_{\ell,r} = B(i_0, i_1, j_0, j_1)$. For each i with $i_0 \leq i \leq i_1$, we define $V_{\leq}(i, B_{\ell,r}) = \bigcup_{i_0 \leq i' \leq i} V(i', B_{\ell,r})$. For each j with $j_0 \leq j \leq j_1$, we define $H_{\leq}(j, B_{\ell,r}) = \bigcup_{j_0 \leq j' < j} H(j', B_{\ell,r})$. 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 $\sigma(V_{\leq}(i, B_{\ell,r}))$ and $\sigma(H_{\leq}(j, B_{\ell,r}))$ for all i and j in $O(\ell - r)$ time. In total, we spend $O(n \log n)$ time over all pairs (ℓ, 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 Figure 15 for the intuition. More precisely, let $\mathcal{I}(a)$ be the pairs (ℓ, r) of \mathcal{I} with $\ell < a < r$. For each pair (ℓ, r) of $\mathcal{I}(a)$, 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_{p_a} is the (the closure of the) disjoint union of the cells in $X(a, B_{\ell,r})$, where (ℓ, r) iterates over $\mathcal{I}(a)$. It follows that

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

where in the last equality we have used Lemma 10.

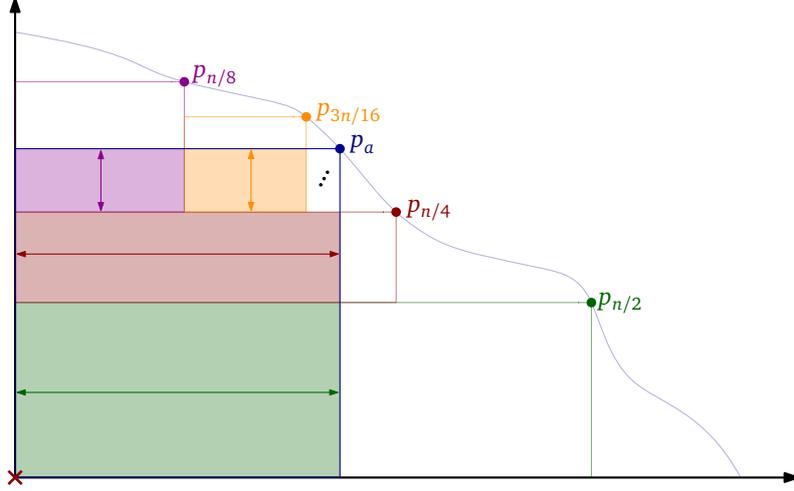


Figure 15: Expressing R_{p_a} as the union of $O(\log n)$ rectangles of the form $X(a, B_{\ell,r})$. In this example, $3n/16 < a < n/4$.

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

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 14. *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 Figure 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.) Because of Lemma 12 and Lemma 13, we can compute the Shapley values of the AREAANCHOREDRECTANGLES game for each quadrant in $O(n \log^2 n)$ time. The result follows. \square

5.5 General point sets

We consider now the general case; thus the points do not form a chain. See Figure 7 for an example. Like before, we restrict the discussion to the case where P is in the positive quadrant. In this case, $\text{ne}(c_{i,j})$ is not a simple expression of i, j anymore, As mentioned in Section 5.3, we preprocess P in $O(n \log n)$ time such that $\text{ne}(q)$ can be computed in $O(\log n)$ time [28].

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 Figure 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 15. *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 $p \in P_B$.*

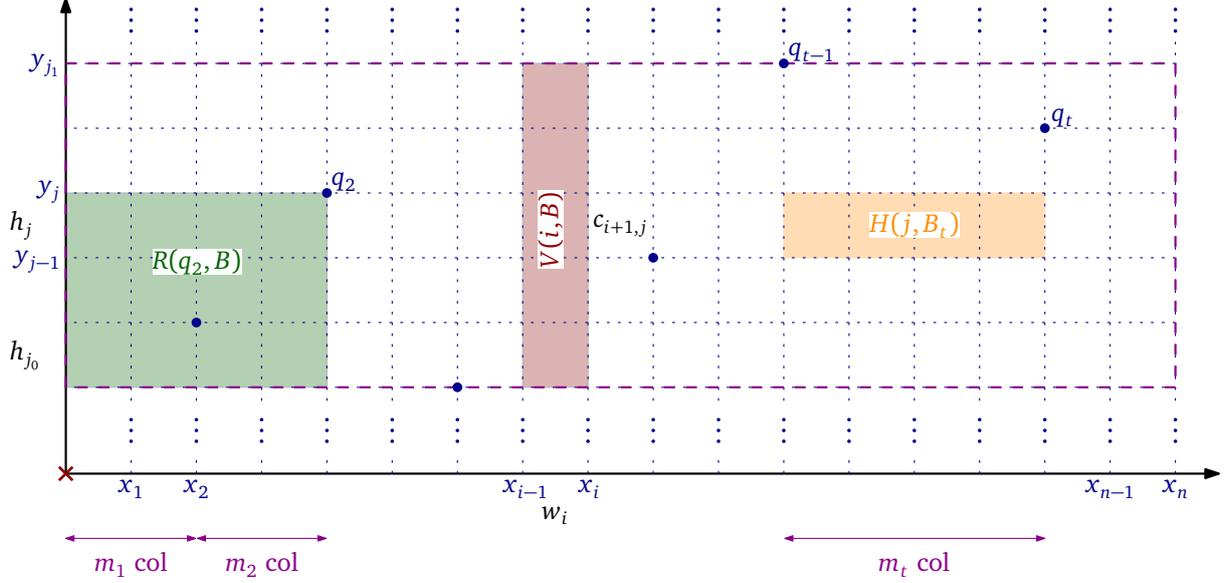


Figure 16: Notation for a band B .

Proof. Assume that B is defined by the row indices $j_0 \leq j_1$. Let q_1, \dots, 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 Figure 16.

We divide the band B into blocks B_1, B_2, \dots 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 11. Using that the $O(k_B)$ blocks B_1, B_2, \dots are pairwise disjoint, which means that $\sum_t m_t = n$, we conclude that in $O((k_B)^2 + n) \log n$ time we can compute all the values $\sigma(V(\cdot, B_t))$ and $\sigma(H(\cdot, B_t))$ for all slabs within all blocks B_t .

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_t we have computed $\sigma(H(j, B_t))$ for all j . With this information we can compute the values $\sigma(R(p, B))$. Namely, for each block B_t and each j with $j_0 \leq j \leq j_1$, we define

$$H_{\leq}(j, B_t) = \bigcup_{j_0 \leq j' \leq j} \bigcup_{1 \leq t' \leq t} H(j', B_{t'}).$$

Using that

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

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

Theorem 16. *The Shapley values of the 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 the other quadrants can be transformed to this one.

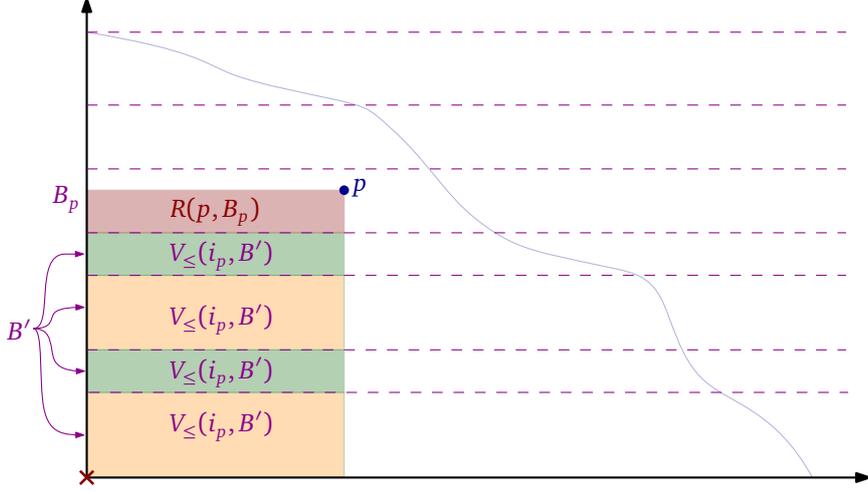


Figure 17: Expressing R_p as the union of other groups of cells.

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 15. This takes $O(k \cdot (k^2 + n) \log n) = O(n^{3/2} \log n)$ time. For each band B and $i \in [n]$, we further consider $V_{\le}(i, B) = \bigcup_{i' \leq i} V(i', B)$ and compute the values $\sigma(V_{\le}(i, B)) = \sum_{i' \leq i} \sigma(V(i', B))$ using prefix sums. This takes $O(n^{3/2})$ additional time.

The cells inside a rectangle R_p now correspond to the disjoint union

$$R(p, B_p) \cup \bigcup_{B' \text{ below } B_p} V_{\le}(i_p, B'),$$

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 Figure 17. Because of Lemma 10 it follows that

$$\phi(p, v_{\text{ar}}) = \sum_{c \in \mathcal{A}, c \subset R_{p_a}} \frac{\text{area}(c)}{\text{ne}(c)} = \sigma(R(p, B_p)) + \sum_{B' \text{ below } B_p} \sigma(V_{\le}(i_p, B')).$$

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

6 Area of the bounding box

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

First we note that it is enough to consider the 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 $v_{\text{abb}}(Q) = \text{area}(\text{bb}(Q \cup \{o\}))$.

Lemma 17. *If we can solve the AREAANCHOREDBOUNDINGBOX problem for n points in the first quadrant in time $T(n)$, then we can solve the AREABOUNDINGBOX game in $T(n) + O(n \log n)$ time. Furthermore, if we can solve the AREAANCHOREDBOUNDINGBOX problem for any n points in a quadrant that form any chain in $T_c(n)$ time, then we can solve the AREABOUNDINGBOX game for n points that form a chain in $T_c(n) + O(n \log n)$ time.*

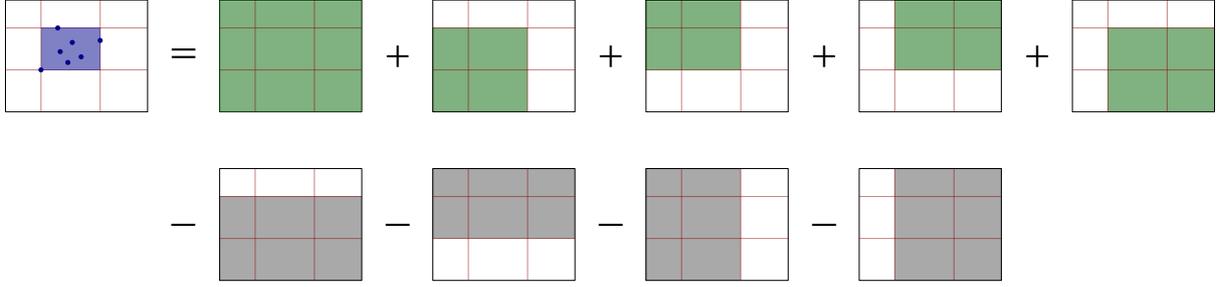


Figure 18: The AREABOUNDINGBOX game is a sum of other games, where the non-trivial games are AREAANCHOREDBOUNDINGBOX games with points in one quadrant.

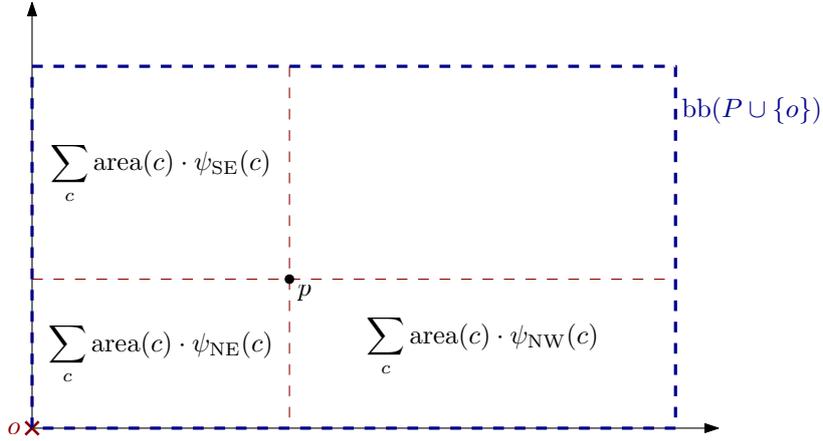


Figure 19: The formula in Lemma 18.

Proof. We use inclusion exclusion, as indicated in Figure 18. There are characteristic functions v_1, \dots, v_9 such that, for each $Q \subseteq P$, we have $v_{\text{bb}}(Q) = \sum_{i=1}^9 v_i(Q)$. Moreover, each characteristic function v_i is either a constant value game (first summand in Figure 18), isometrically equivalent to an AREABAND game (last four terms in Figure 18), or isometrically equivalent to an AREAANCHOREDBOUNDINGBOX problem with all points in one quadrant (the remaining four summands in Figure 18). 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. \square

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 Section 5 because a cell c of \mathcal{A} is inside $\text{bb}(Q \cup \{o\})$ 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 \mathcal{A} we define the following values

$$\begin{aligned} \psi_{\text{NE}}(c) &= \frac{1}{\text{ne}(c) + \text{nw}(c)} + \frac{1}{\text{ne}(c) + \text{se}(c)} - \frac{1}{\text{ne}(c) + \text{nw}(c) + \text{se}(c)}, \\ \psi_{\text{NW}}(c) &= \frac{1}{\text{ne}(c) + \text{nw}(c)} - \frac{1}{\text{ne}(c) + \text{nw}(c) + \text{se}(c)}, \\ \psi_{\text{SE}}(c) &= \frac{1}{\text{ne}(c) + \text{se}(c)} - \frac{1}{\text{ne}(c) + \text{nw}(c) + \text{se}(c)}. \end{aligned}$$

The following lemma is summarized in Figure 19.

Lemma 18. *If P is in the positive quadrant, then for each $p \in P$ the Shapley value $\phi(p, v_{\text{abb}})$ is*

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

Proof. Each cell c of the arrangement \mathcal{A} defines a game for the set of players P with characteristic function

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

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 4.

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 π and all the points of $\text{NW}(c)$ or all the points of $\text{SE}(c)$ are after p in the permutation π . According to the first item of Lemma 4, 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 π , all the points of $\text{NE}(c)$ are after p in the permutation π , and at least one point of $\text{SE}(c)$ is before p in the permutation π . According to the second item of Lemma 4, 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 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,

$$v_{\text{abb}}(Q) = \sum_{c \in \mathcal{A}} v_c(Q) \quad \text{for all } Q \subset P,$$

the result follows from the linearity of Shapley values. \square

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

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

Like in the case of anchored boxes, our objective here is to compute these values for several vertical and horizontal slabs.

6.2 Handling empty blocks

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 [28]. We use again multipoint evaluation to compute the values $\sigma_*(\cdot)$ for each vertical and horizontal slab of an empty block and for each $* \in \{\text{NE}, \text{NW}, \text{SE}\}$. However, the treatment has to be a bit more careful now because we have to deal with different rational functions.

Lemma 19. *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 and each $* \in \{\text{NE}, \text{NW}, \text{SE}\}$.*

Proof. The proof is very similar to the proof of Lemma 11, 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 $\sigma_{\text{NE}}(H(j_0, B)), \dots, \sigma_{\text{NE}}(H(j_1, B))$ and for NW and SE instead of NE is similar.

For each ℓ we define $J(\ell) = \{j \mid j_0 \leq j \leq j_1, \text{ne}(c_{i_0, j}) + \text{se}(c_{i_0, j}) = \ell\}$ and $J'(\ell) = \{j \mid j_0 \leq j \leq j_1, \text{ne}(c_{i_0, j}) + \text{nw}(c_{i_0, j}) + \text{se}(c_{i_0, j}) = \ell\}$. Let ℓ_0 and ℓ_1 be the minimum and the maximum ℓ such that $J(\ell) \neq \emptyset$, respectively. Similarly, let ℓ'_0 and ℓ'_1 be the minimum and the maximum ℓ such that $J'(\ell) \neq \emptyset$.

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

$$\begin{aligned}R_1 &= \sum_{j=j_0}^{j_1} \frac{h_j}{\text{ne}(c_{i_0, j}) + \text{nw}(c_{i_0, j})}, \\ R_2(x) &= \sum_{\ell=\ell_0}^{\ell_1} \frac{\sum_{j \in J(\ell)} h_j}{\ell + x}, \\ R_3(x) &= \sum_{\ell=\ell'_0}^{\ell'_1} \frac{\sum_{j \in J'(\ell)} h_j}{\ell + x}.\end{aligned}$$

Each of them is the sum of at most k fractions. As discussed in the proof of Lemma 11, $R_2(x)$ and $R_3(x)$ can be rewritten so that we can use Lemma 2 to evaluate them at consecutive 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 .

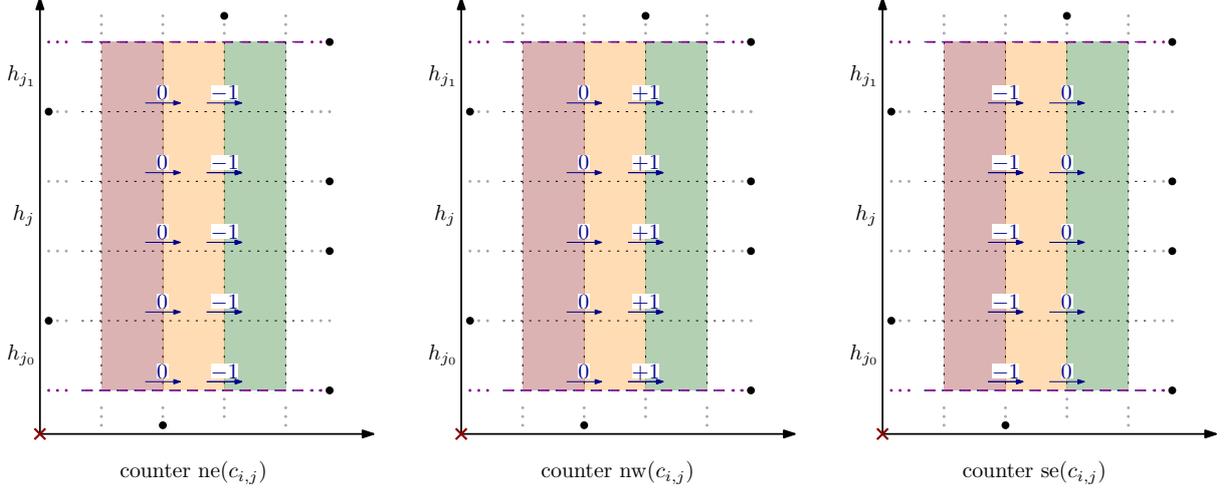


Figure 20: Changes in the counter $\text{ne}(\cdot)$ (left), $\text{nw}(\cdot)$ (center) and $\text{se}(\cdot)$ (right) depending on the position of the points of P .

- 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

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

depends on whether the point 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 δ_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 δ_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 for each relevant i we have

$$\begin{aligned} & w_i \cdot (R_1 + R_2(-i + i_0) - R_3(\delta_i)) \\ &= w_i \cdot \left(\sum_{j=j_0}^{j_1} \frac{h_j}{\text{ne}(c_{i_0,j}) + \text{nw}(c_{i_0,j})} + \sum_{\ell=\ell_0}^{\ell_1} \frac{\sum_{j \in J(\ell)} h_j}{\ell - i + i_0} + \sum_{\ell=\ell'_0}^{\ell'_1} \frac{\sum_{j \in J'(\ell)} h_j}{\ell + \delta_i} \right) \\ &= \sum_{j=j_0}^{j_1} \left(\frac{w_i h_j}{\text{ne}(c_{i,j}) + \text{nw}(c_{i,j})} + \frac{w_i h_j}{\text{ne}(c_{i,j}) + \text{se}(c_{i,j})} - \frac{w_i h_j}{\text{ne}(c_{i,j}) + \text{nw}(c_{i,j}) + \text{se}(c_{i,j})} \right) \\ &= \sum_{j=j_0}^{j_1} \text{area}(c_{i,j}) \cdot \psi_{\text{NE}}(c_{i,j}) \\ &= \sigma_{\text{NE}}(V(i, B)). \end{aligned}$$

Thus, we need to evaluate $R_2(x)$ at the values $x = 0, -1, \dots, i_0 - i_1$ and we have to evaluate $R_3(x)$ at the values δ_i for $i = i_0, \dots, i_1$. According to Lemma 2, 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. \square

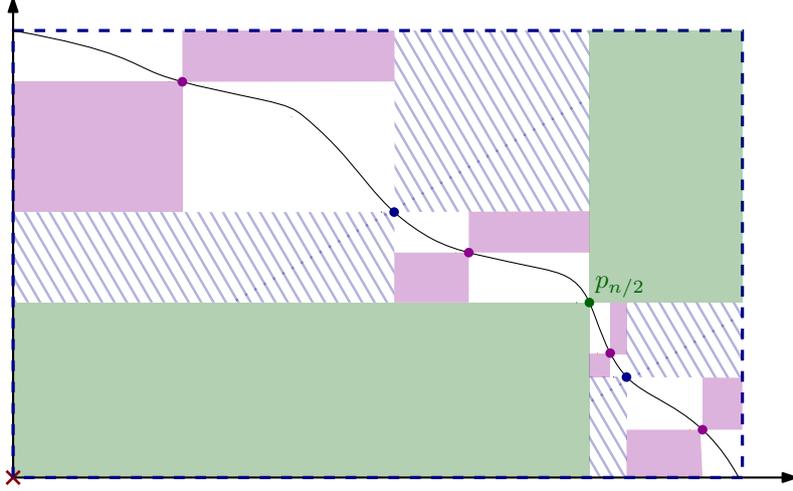


Figure 21: The empty blocks to be considered for the decreasing chain.

6.3 Algorithms for bounding box

The same techniques that were used in Sections 5.4 and 5.5 work in this case. We compute partial sums $\sigma_*(\cdot)$ for several vertical bands and horizontal bands. The only difference is that we have to cover the whole bounding box $\text{bb}(P \cup \{o\})$, instead of only 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 Figure 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 15. For the partial sums that we consider (like $\sigma(H_{\leq}(\cdot))$ and $\sigma(V_{\leq}(\cdot))$) we also construct the symmetric versions $\sigma_*(H_{\geq}(\cdot))$ and $\sigma_*(V_{\geq}(\cdot))$. 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 Figure 19. In the case of a decreasing chain, we get an extra case that is handled by noting that

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

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 ψ_{NW} are handled similarly. We omit the details and summarize.

Theorem 20. *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 17 we also obtain the following.

Theorem 21. *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 of

geometric objects. For axis-parallel problems we used computer algebra 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 something. 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 [13]. 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. [13]. 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 0, 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?

Acknowledgments.

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

References

- [1] P. K. Agarwal, S. Har-Peled, S. Suri, H. Yildiz, and W. Zhang. Convex hulls under uncertainty. *Algorithmica* 79(2):340–367, 2017, <https://doi.org/10.1007/s00453-016-0195-y>.
- [2] P. K. Agarwal, N. Kumar, S. Sintos, and S. Suri. Range-max queries on uncertain data. *J. Comput. Syst. Sci.* 94:118–134, 2018, <https://doi.org/10.1016/j.jcss.2017.09.006>.
- [3] D. Ajwani, S. Ray, R. Seidel, and H. R. Tiwary. On computing the centroid of the vertices of an arrangement and related problems. *10th International Workshop Algorithms and Data Structures, WADS 2007*, pp. 519–528. Springer, Lecture Notes in Computer Science 4619, 2007, https://doi.org/10.1007/978-3-540-73951-7_45.
- [4] B. Aronov and M. J. Katz. Batched point location in SINR diagrams via algebraic tools. *ACM Trans. Algorithms* 14(4):41:1–41:29, 2018, <http://doi.acm.org/10.1145/3209678>.
- [5] M. de Berg, O. Cheong, M. J. van Kreveld, and M. H. Overmars. *Computational geometry: algorithms and applications, 3rd Edition*. Springer, 2008.

- [6] X. Deng and Q. Fang. Algorithmic cooperative game theory. *Pareto Optimality, Game Theory And Equilibria*, pp. 159–185. Springer New York, 2008, https://doi.org/10.1007/978-0-387-77247-9_7.
- [7] X. Deng and C. H. Papadimitriou. On the complexity of cooperative solution concepts. *Mathematics of Operations Research* 19(2):257–266, 1994, <https://doi.org/10.1287/moor.19.2.257>.
- [8] H. Edelsbrunner, J. O’Rourke, and R. Seidel. Constructing arrangements of lines and hyperplanes with applications. *SIAM J. Comput.* 15(2):341–363, 1986, <https://doi.org/10.1137/0215024>.
- [9] H. Edelsbrunner, R. Seidel, and M. Sharir. On the zone theorem for hyperplane arrangements. *SIAM J. Comput.* 22(2):418–429, 1993, <https://doi.org/10.1137/0222031>.
- [10] U. Faigle, S. P. Fekete, W. Hochstättler, and W. Kern. On approximately fair cost allocation in Euclidean TSP games. *Operations-Research-Spektrum* 20(1):29–37, 1998, <https://doi.org/10.1007/BF01545526>.
- [11] T. S. Ferguson. Game theory, 2nd edition, 2014. Electronic text available at https://www.math.ucla.edu/~tom/Game_Theory/Contents.html.
- [12] M. Fink, J. Hershberger, N. Kumar, and S. Suri. Hyperplane separability and convexity of probabilistic point sets. *JoCG* 8(2):32–57, 2017, <http://jocg.org/index.php/jocg/article/view/321>.
- [13] P. Kamousi, T. M. Chan, and S. Suri. Stochastic minimum spanning trees in Euclidean spaces. *Proceedings of the 27th ACM Symposium on Computational Geometry, SoCG’11*, pp. 65–74. ACM, 2011, <http://doi.acm.org/10.1145/1998196.1998206>.
- [14] P. Kamousi, T. M. Chan, and S. Suri. Closest pair and the post office problem for stochastic points. *Comput. Geom.* 47(2):214–223, 2014, <https://doi.org/10.1016/j.comgeo.2012.10.010>.
- [15] S. Littlechild and G. Owen. A simple expression for the Shapely value in a special case. *Management Science* 20(3):370–372, 1973, <https://doi.org/10.1287/mnsc.20.3.370>.
- [16] J. Matoušek. *Lectures on Discrete Geometry*. Springer-Verlag, Berlin, Heidelberg, 2002.
- [17] N. Megiddo. Computational complexity of the game theory approach to cost allocation for a tree. *Mathematics of Operations Research* 3(3):189–196, 1978, <https://doi.org/10.1287/moor.3.3.189>.
- [18] G. Moroz and B. Aronov. Computing the distance between piecewise-linear bivariate functions. *ACM Trans. Algorithms* 12(1):3:1–3:13, 2016, doi:10.1145/2847257, <http://doi.acm.org/10.1145/2847257>.
- [19] R. B. Myerson. *Game theory - Analysis of Conflict*. Harvard University Press, 1997.
- [20] N. Nisan, T. Roughgarden, É. Tardos, and V. V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, 2007.
- [21] M. J. Osborne and A. Rubinstein. *A course in game theory*. The MIT Press, 1994.
- [22] P. Pérez-Lantero. Area and perimeter of the convex hull of stochastic points. *Comput. J.* 59(8):1144–1154, 2016, <https://doi.org/10.1093/comjnl/bxv124>.

- [23] J. Puerto, A. Tamir, and F. Perea. A cooperative location game based on the 1-center location problem. *European Journal of Operational Research* 214(2):317–330, 2011, <https://doi.org/10.1016/j.ejor.2011.04.020>.
- [24] J. Puerto, A. Tamir, and F. Perea. Cooperative location games based on the minimum diameter spanning Steiner subgraph problem. *Discrete Applied Mathematics* 160(7-8):970–979, 2012, <https://doi.org/10.1016/j.dam.2011.07.020>.
- [25] A. E. Roth, editor. *The Shapley Value: Essays in Honor of Lloyd S. Shapley*. Cambridge University Press, 1988.
- [26] W. 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.
- [27] E. Welzl. Smallest enclosing disks (balls and ellipsoids). *New Results and New Trends in Computer Science*, pp. 359–370. Springer, Lecture Notes in Computer Science 555, 1991.
- [28] D. E. Willard. New data structures for orthogonal range queries. *SIAM J. Comput.* 14:232–253, 1985, <https://doi.org/10.1137/0214019>.
- [29] E. Winter. The Shapley value. *Handbook of Game Theory with Economic Applications*, 1 edition, vol. 3, chapter 53, pp. 2025–2054. Elsevier, 2002, [https://doi.org/10.1016/S1574-0005\(02\)03016-3](https://doi.org/10.1016/S1574-0005(02)03016-3).
- [30] J. Xue, Y. Li, and R. Janardan. On the separability of stochastic geometric objects, with applications. *Comput. Geom.* 74:1–20, 2018, doi:10.1016/j.comgeo.2018.06.001, <https://doi.org/10.1016/j.comgeo.2018.06.001>.