Computing Shapley Values for Mean Width in 3-D

Shuhao Tan
Department of Computer Science, University of Maryland, College Park, MD, USA

Abstract

The Shapley value is a classical concept from game theory, which is used to evaluate the importance of a player in a cooperative setting. Assuming that players are inserted in a uniformly random order, the Shapley value of a player p is the expected increase in the value of the characteristic function when p is inserted. Cabello and Chan (SoCG 2019) recently showed how to adapt this to a geometric context on planar point sets. For example, when the characteristic function is the area of the convex hull, the Shapley value of a point is the average amount by which the convex-hull area increases when this point is added to the set. Shapley values can be viewed as an indication of the relative importance/impact of a point on the function of interest.

In this paper, we present an efficient algorithm for computing Shapley values in 3-dimensional space, where the function of interest is the mean width of the point set. Our algorithm runs in $O(n^3 \log^2 n)$ time and $O(n)$ space. This result can be generalized to any point set in d-dimensional space ($d \geq 3$) to compute the Shapley values for the mean volume of the convex hull projected onto a uniformly random $(d-2)$-subspace in $O(n^d \log^2 n)$ time and $O(n)$ space. These results are based on a new data structure for a dynamic variant of the convolution problem, which is of independent interest. Our data structure supports incremental modifications to n-element vectors (including cyclical rotation by one position). We show that n operations can be executed in $O(n \log^2 n)$ time and $O(n)$ space.

2012 ACM Subject Classification Theory of computation → Computational geometry; Theory of computation → Data structures design and analysis

Keywords and phrases Shapley value, mean width, dynamic convolution

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2021.67

Related Version *Full Version*: `https://arxiv.org/abs/2002.05252`

Acknowledgements The author is very grateful to David Mount for discussions and advice for the paper. The author is also thankful to Geng Lin for proofreading and helping improving the readability.

1 Introduction

Given a point set P in d-dimensional space, many different functions can be applied to extract information about the set's geometric structure. Often, this involves properties of the convex hull of the set, such as its surface area and mean width. In the context of approximation, a natural question involves the "impact" that any given point of P has on the measure of interest. One method for modeling the notion of impact arises from the concept of the Shapley value (defined below) in the context of cooperative games in game theory. In this paper, we will present an efficient algorithm to compute the Shapley values for points in 3-D where the payoff of a point set is defined as the mean width of its convex hull.

Formally, a *coalition game* consists of a set of players N and a characteristic function $v : 2^N \to \mathbb{R}$, where $v(\emptyset) = 0$. In our setting, we take the players to be a point set $N \subset \mathbb{R}^3$ and consider the characteristic function $v(Q) = w(\text{conv}(Q))$ for $Q \subset N$, where $\text{conv}(Q)$ denotes the convex hull of Q, and $w(\text{conv}(Q))$ denotes the mean width of $\text{conv}(Q)$. (Formal definitions will be given in Section 2.)

To define Shapley values, let π be a uniformly random permutation of N, and $P_N(\pi, i)$ be the set of players in N that appear before player i in the permutation π. The *Shapley value* of player $i \in N$ is defined to be the expected increase in v induced by the addition of player i, that is,

$$\phi(i) = \mathbb{E}_\pi[v(P_N(\pi, i) \cup \{i\}) - v(P_N(\pi, i))]. \tag{1}$$

Thus, the Shapley value represents the expected marginal contribution of i to the objective function over all permutations of N.

There are wide applications of Shapley values. A survey by Winter [27] and a book dedicated to this topic [24] provide insights on how the concept can be interpreted and applied in multiple ways, such as utility of players, allocation of resources of the grand coalition and measure of power in a voting system. Moreover, the values can be characterized axiomatically, making it the only natural quantity that satisfies certain properties. More details can be found in standard game theory textbooks ([14], [8, Chapter 5], [22, Section 9.4]).

In convex geometry and measure theory, *intrinsic volumes* are a key concept to characterize the "size" and "shape" of a convex body regardless of the translation and rotation in its underlying space. For example, Steiner's formula [12] relates intrinsic volumes to the volume of the Minkowski-sum of a convex body and a ball. In general, one can define *valuation* to be a class of measure-like maps on open sets in a topological space. A formal definition of this concept can be found in [4, 16]. Functions such as volume and surface area fall into this class. Hadwiger's Theorem [13, 17] asserts that every valuation that is continuous and invariant under rigid motion in $\mathbb{R}^d$ is a linear combination of intrinsic volumes. In d-dimensional space, the d-th, $(d-1)$-st and first intrinsic volumes are proportional to the usual Lebesgue measure, the surface area and the mean width, respectively [25, Chapter 4]. Cabello and Chan [7] presented efficient algorithms to compute Shapley values for area and perimeter for a point set in 2-D, which can be naturally extended to volume and surface area in 3-D. An algorithm that efficiently computes Shapley values for mean width in 3-D will then imply an algorithm that efficiently computes Shapley values for any continuous valuation in 3-D.

Related work

The problem of computing Shapley values for area functions on a point set was introduced by Cabello and Chan [7]. They provided algorithms to compute Shapley values for area of convex hull, area of minimum enclosing disk, area of anchored rectangle, area of bounding box and area of anchored bounding box for a point set in 2-D. They also gave algorithms for the hull perimeter by slightly modifying these algorithms. All the quantities they considered are defined in 2-D space. Although their algorithms naturally extend to higher dimension, there are interesting applications of Shapley values in higher dimension that have yet been explored. The mean width considered in this paper is one such example.

Cabello and Chan also drew connections between computing Shapley values and stochastic computational geometry models. There have been many studies on the behavior of the convex hull under unipoint model where each point has an existential probability, for example see [1, 10, 15, 19, 26, 28]. In particular, Xue et al. [28] discussed the expected diameter and width of the convex hull. Huang et al. [15] presented a way to construct ϵ-coreset for directional width under the model.

Mean width is often considered in the context of random polytopes in stochastic geometry. Müller [21] showed the asymptotic behavior of the mean width for a random polytope generated by sampling points on a convex body. Böröczyky [6] refined the result under the assumption that the convex body is smooth. Alonso-Gutiérrez and Prochno [3] considered

the case when the points are sampled inside an isotropic convex body. However, these results only consider the statistics when the number of points is large and the results are asymptotic. This paper views mean width on a computational perspective instead.

Our contributions

We show that the Shapley values for mean width of the convex hull for a point set in $\mathbb{R}^3$ can be computed in $O(n^3 \log^2 n)$ time and $O(n)$ space. Our approach is similar to the one used by Cabello and Chan for the case of the convex-hull area, in the sense that we look at the incremental formation of convex hull and consider the contribution of individual geometric objects. In our case, we break down the mean width and express that in terms of quantities related only to edges and then apply the linearity of Shapley values.

A major difference is that the expression for mean width contains angle at the edge, making the calculation for the probability term for Shapley values depend on the intersection of two halfspaces as opposed to only one halfspace. This prompts an expression that looks like a convolution. This convolution only changes slightly when evaluated from one point to another. The setup can be captured as an instance of dynamic convolution, where one can change the convolution kernel at any position, and query any single position in the vector involved in the convolution. Algebraic computations of this form were explored by Reif and Tate [23]. Frandsen et al. [11] gave a worst-case lower bound of $\Omega(\sqrt{n})$ time per operation for this problem. By exploiting the structure of sweeping by polar angle, we obtain a variant of dynamic convolution, where we have a pointer to the convolution kernel and another pointer to the convolution result. We are only allowed to query at the pointer, update the convolution kernel at the pointer and move the pointers by one position. We present an online data structure for this variant that has $O(n \log^2 n)$ overall time for n operations. There are some occurrences of algebraic computation in computational geometry, but many works [2, 5, 7, 18] do not have such dynamic setting and rely mostly on computing a single convolution or multi-point evaluation of polynomials. Only a few (see, e.g., [9]) employed a dynamic data structure. We believe our data structure is of independent interest for algorithms based on sweeping.

2 Preliminaries

Mean width. The mean width of a compact convex body X can be seen as the mean 1-volume of X projected on a uniformly random 1-subspace. More formally, let $X \subset \mathbb{R}^d$ be a compact convex body. For $1 \leq s \leq d$, the *mean s-projection* of X is defined as:

$$M_s(X) = \int_{Q^s} |X_u| f_s(u)\, du, \tag{2}$$

where Q^s is the set of s-subspace, X_u is the projection of X on u, $|\cdot|$ is the volume or the canonical measure in the underlying space, f_s is the probability density function for uniformly sampling s-subspace from Q^s.

In this manner, the mean width of a point set P can be defined as $M_1(\mathrm{conv}(P))$ where $\mathrm{conv}(P)$ is the convex hull of P. It turns out that the mean s-projection of a convex polytope $(1 \leq s \leq d-2)$ can be decomposed into values only related to its s-faces.

For a vertex of a polygon in 2-D, its interior angle can be naturally defined as the angle containing points within the polygon. And its exterior angle is the complement of the interior angle. This notion can be similarly generalized to higher dimensional faces in higher dimension. More formally, let $X \subset \mathbb{R}^d$ be a convex polytope. Let V be a s-face of X. Let $Q(V)$ be

the set of halfspaces such that V is on the hyperplanes defining the halfspace. The *exterior angle* of V is defined as $\psi(V) = \frac{|\{q \in Q(V) : X \subset q\}|}{|Q(V)|}$. Let $n(q)$ be the normal vector contained by the halfspace q. The *interior angle* of V is defined as $\chi(V) = \frac{|\{q \in Q(V) : \exists p \in V, \exists \epsilon > 0, p + \epsilon n(q) \in X|}{|Q(V)|}$. Moreover, when $s = d - 2$, and $V \neq X$, we have $\chi(V) + \psi(V) = 1/2$ for all s-face V of X.

▶ **Lemma 1** (Miles [20]). *Let $X \subset \mathbb{R}^d$ be a convex polytope, and $1 \leq s \leq d-2$. Let $V_s(X)$ be the set of s-face of X. We have:*

$$M_s(X) = C_{s,d} \sum_{V \in V_s(X)} |V| \psi(V). \tag{3}$$

where $C_{s,d}$ is some constant related to s and d, in particular, $C_{1,3} = 1/2$.

Random permutations. When considering random permutations, it is common to compute probabilities where constraints on the order of appearance of disjoint sets are imposed. The following lemma follows from simple counting where a proof can be found in [7].

▶ **Lemma 2.** *Let N be a set with n elements. Let $\{x\}$, A and B be disjoint sets. The probability that all of A appears before x and all of B appears after x in a uniformly random permutation π is $\frac{|A|!|B|!}{(|A|+|B|+1)!}$.*

Assumptions. We assume points in 3-D are of general positions where no three points are co-linear and no four points are co-planar. All points are assumed distinct. Throughout the paper, we often look at the projection of a 3-D point p on a plane, we will still use the same symbol p when looking on the plane for simplicity. We also assume a unit-cost real-RAM model of computation, where any arithmetic operation between two real numbers costs unit time, and a real number costs unit space to store.

3 Dynamic Convolution with Local Updates and Queries

Before presenting the algorithm to compute mean width, we first present a data structure that is central to the algorithm.

We consider a variant of dynamic convolution where only local updates and queries are permitted. More formally, let $g : \mathbb{Z} \to \mathbb{R}$ be a fixed function where we can evaluate $g(x)$ in constant time for any x. Let $f : \mathbb{Z} \to \mathbb{R}$ initially be a zero function where we can change its values later. Let $p, c \in \mathbb{Z}$ be two variables initially 0. We want to design a data structure that supports the following operations:

Query	Update(x)	IncP	DecP
Output $\sum_{i \in \mathbb{Z}} f(i+c)g(i)$	$f(p) \leftarrow f(p) + x$	$p \leftarrow p + 1$	$p \leftarrow p - 1$
RotateLeft	**RotateRight**		
$c \leftarrow c + 1$	$c \leftarrow c - 1$		

Let $f^{(i)}$ be the f after the i-th operation, so $f^{(0)} = 0$. And similarly for $p^{(i)}$, and $c^{(i)}$. We can make the following observations:

▶ **Lemma 3.** *For any $0 \leq i < j$, $\max \operatorname{supp}(f^{(j)} - f^{(i)}) - \min \operatorname{supp}(f^{(j)} - f^{(i)}) \leq j - i$, where $\operatorname{supp}(f)$ is the support of f.*

Proof. There are at most $j - i$ IncP or DecP operations between the i-th and the j-th operation, so $\max_{i \leq k \leq j} p^{(k)} - \min_{i \leq k \leq j} p^{(k)} \leq j - i$. And we have $\max \operatorname{supp}(f^{(j)} - f^{(i)}) \leq \max_{i \leq k \leq j} p^{(k)}$ and $\min \operatorname{supp}(f^{(j)} - f^{(i)}) \geq \min_{i \leq k \leq j} p^{(k)}$. ◀

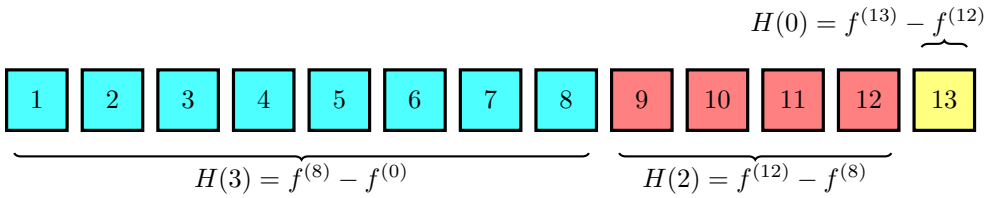

Figure 1 An example for $n = 13 = 2^3 + 2^2 + 2^0$. Each box represents an operation.

▶ **Corollary 4.** *For any* $0 \le i < j$, $|\operatorname{supp}(f^{(j)} - f^{(i)})| \le j - i$.

▶ **Lemma 5.** *For any* $0 \le i < j$, $\max_{i \le k \le j} c^{(k)} - \min_{i \le k \le j} c^{(k)} \le j - i$.

Proof. There are at most $j - i$ RotateLeft or RotateRight operations between the i-th and the j-th operation. ◀

▶ **Lemma 6.** *For any* $0 \le i < j$, $\sum_k f^{(j)}(k + c)g(k) = \sum_k f^{(i)}(k + c)g(k) + \sum_k (f^{(j)} - f^{(i)})(k + c)g(k)$.

From Lemma 6, we can see that it is possible to break the operations into chunks, and only consider the changes on f for each chunk. Lemma 3 ensures that the actual differences are proportional to the size of a chunk. So the idea is: after a chunk of size m is finished, we pre-compute all queries for the next m operations by considering all possible changes in c. We incrementally build different levels of chunks so that after finishing a chunk, we merge it with smaller chunks to build a larger chunk. Lemma 5 and Corollary 4 ensure that the prediction only depends on a small set of values of g. More specifically, we will use chunks sizes with power of two based on the binary form of the number of operations so far and merge chunks when needed. A binary-counter-like argument can be applied to achieve a amortized time of $O(\log^2 n)$ per operation.

We maintain chunks of changes based on the binary representation of the number of operations we have seen so far. Assume we've just performed the k-th operation. Let $b(k, i)$ be the i-th bit from right in the binary representation of k. We can write $k = \sum_{i:b(k,i)=1} 2^i$ and we are going to use chunks of size 2^i for $b(k, i) = 1$, where larger chunks are closer to the start of operations. Let $i_1 < i_2 < \cdots < i_q$ be the i's such that $b(k, i) = 1$, and $d(l) = \sum_{j=1}^{l} 2^{i_j}$. Let $s(j) = k - d(j)$ and $e(j) = k - d(j - 1)$. $(s(j), e(j)]$ represents the range of the j-th chunk. In other words, we split the operations into chunks of size $2^{i_q}, 2^{i_{q-1}}, \ldots, 2^{i_1}$. We maintain chunks of changes as $H(i_j) = f^{(e(j))} - f^{(s(j))}$. See Figure 1 for an example. Moreover, for each chunk, we maintain an array of predicted queries $A(i_j)$ on the chunk $H(i_j)$, so that $A(i_j, 2^{i_j} + c) = \sum_p H(i_j)(p + c^{(e(j))} + c)g(p)$ for $-2^{i_j} \le c \le 2^{i_j}$. We also maintain an array of I so that $I(i_j) = c^{(k)} - c^{(e(j))}$.

Assume we have A, H, and I after n operations, the $(n+1)$-st operation can be performed easily: Query should output $\sum_p f^{(n)}(p + c^{(n)})g(p)$, and we have:

$$\begin{aligned}
\sum_p f^{(n)}(p + c^{(n)})g(p) &= \sum_p \left(\sum_{i_j : b(n,i_j)=1} f^{(e(j))} - f^{(s(j))} \right) (p + c^{(n)})g(p) \\
&= \sum_{i_j : b(k,i_j)=1} \sum_p (f^{(e(j))} - f^{(s(j))})(p + c^{(e(j))} + I(i_j))g(p) \\
&= \sum_{i_j : b(k,i_j)=1} \sum_p H(i_j)(p + c^{(e(j))} + I(i_j))g(p) \\
&= \sum_{i_j : b(k,i_j)=1} A(i_j, 2^{i_j} + I(i_j)).
\end{aligned}$$

So we can answer query by performing a summation over A. UPDATE is handled by simply documenting the change. INCP and DECP are handled by incrementing and decrementing p, respectively. ROTATELEFT and ROTATERIGHT are handled by incrementing and decrementing all entries of I, respectively.

After each operation, an additional maintaining step is performed to ensure A, H, I contain the correct information for the following operations. Assume that we want to maintain the data structure after the n-th operation, we record the change in f as $\Delta f = f^{(n)} - f^{(n-1)}$. Observe that $A(i)$ doesn't change if $b(n,i) = b(n-1,i)$. Let $i^* = \min\{i : b(n,i) = 1\}$, then only $A(i)$'s such that $i \le i^*$ change. Moreover, $A(i)$ becomes empty if $i < i^*$. Similar phenomenon can be seen in the increment of a binary counter. We expect:

$$\begin{aligned}
A(i^*, 2^{i^*} + c) &= \sum_p H(i^*)(p + c^{(n)} + c)g(p) \\
&= \sum_p (f^{(n)} - f^{(n-2^{i^*})})(p + c^{(n)} + c)g(p) \\
&= \sum_p \left(\Delta f + \sum_{j=0}^{i^*-1} f^{(n-2^j)} - f^{(n-2^{j+1})} \right) (p + c^{(n)} + c)g(p) \\
&= \sum_p \left(\Delta f + \sum_{j=0}^{i^*-1} H(j) \right) (p + c^{(n)} + c)g(p).
\end{aligned}$$

So we can merge Δf and $\{H(j) : j < i^*\}$ to get $H(j^*) = f^{(n)} - f^{(n-2^{i^*})}$, and compute $A(i^*, 2^{i^*} + c) = \sum_p H(i^*)(p + c^{(n)} + c)g(p)$ for all $-2^{i^*} \le c \le 2^{i^*}$. This is almost a convolution. To see it clearer, let $L = \min \operatorname{supp} H(i^*)$ and $R = \max \operatorname{supp} H(i^*)$, we can rewrite the expression as $A(i^*, 2^{i^*} + c) = \sum_{p=L}^{R} H(i^*)(p)g(p - c^{(n)} - c)$. We now want to shift the origin so that $\operatorname{supp} H$ can start at 0 to conform to the definition of a convolution. Let $H'(p) = H(i^*)(L + p)$ for $0 \le p \le R - L$, and $H'(p) = 0$ otherwise. Let $g'(p) = g(-p + L - c^{(n)} + 2^{i^*+1})$, for $0 \le p \le 3 \cdot 2^{i^*}$, and $g'(p) = 0$ otherwise. we have:

$$\begin{aligned}
A(i^*, 2^{i^*} + c) &= \sum_{p=L}^{R} H(i^*)(p)g(p - c^{(n)} - c) \\
&= \sum_{p=0}^{R-L} H(i^*)(L + p)g(p - c + L - c^{(n)}) \\
&= \sum_{p=0}^{R-L} H'(p)g'(2^{i^*+1} + c - p) \\
&= H' * g'[2^{i^*+1} + c]
\end{aligned}$$

Where $H' * g'$ is the discrete convolution of H' and g'.

By Lemma 3, $R - L \le 2^{i^*}$, so $0 \le 2^{i^*+1} + c - p \le 3 \cdot 2^{i^*}$. We can treat both H' and g' as circular vectors of size $3 \cdot 2^{i^*}$, and compute $H' * g'[2^{i^*} + c]$ for all $-2^{i^*} \le c \le 2^{i^*}$ by the Fast Fourier Transform. We then use the result to construct $A(i^*)$. We set $I(i^*) = 0$, and clear all $A(i)$, $H(i)$, and $I(i)$ where $i < i^*$.

Now that we have a way to maintain A, H, and I, the following Lemma shows that the QUERY operation always succeed, i.e., $-2^i \le I(i) \le 2^i$.

▶ **Lemma 7.** *After the n-th operation, for any i_j such that $b(n, i_j) = 1$, $|c^{(n)} - c^{(e(j))}| \le 2^{i_j}$.*

Proof. $|c^{(n)} - c^{(e(j))}| \leq \max_{e(j)\leq k\leq n} c^{(k)} - \min_{e(j)\leq k\leq n} c^{(k)} \leq n - e(j) = d(j-1) \leq \sum_{k=0}^{i_j-1} 2^k = 2^{i_j}$. ◀

We now analyze the complexity of the data structure:

▶ **Theorem 8.** *There is a data structure that supports n operations for Dynamic Convolution with Local Updates and Queries in $O(\log^2 n)$ amortized time and $O(n)$ space.*

Proof. Without considering the additional maintaining step after each operation, it is easy to see a single IncP, DecP, or Update takes $O(1)$ time. Likewise a single RotateLeft, RotateRight, or Query takes $O(\log n)$ time since the number of bits in the binary representation of n is $O(\log n)$.

In the maintaining step, whenever the i-th bit in the binary representation of n changes from 0 to 1, we need to merge changes of total size $O(2^i)$, run FFT on two arrays of size $3 \cdot 2^i$, and clear histories of total size $O(2^i)$. These in all take $O(2^i \log(2^i))$ time where the bottleneck is FFT. Like the analysis in binary counter, i-th bit flips from 0 to 1 in every 2^i operations, so the total time spent on maintaining step is: $\sum_{i=0}^{\log n} \frac{n}{2^i} O(2^i \log(2^i)) = O\left(n \sum_{i=0}^{\log n} i \log(2)\right) = O(n \log^2 n)$.

Adding two parts, the total time for n operation is $O(n \log^2 n)$, yielding an amortized time of $O(\log^2 n)$ per operation.

At any point, the space used by the data structure is linear to the size of A, H, I, even during the maintaining step. So the space complexity is $O\left(\sum_{i=0}^{\log n} 2^i\right) = O(n)$. ◀

▶ **Corollary 9.** *Let $g : \mathbb{Z} \to \mathbb{R}$ be a fixed function where we can evaluate $g(x)$ in constant time. Let Q be a queue with elements in $\mathbb{R}$. There is a data structure that supports n operations in $O(\log^2 n)$ amortized time and $O(n)$ space, where each operation is either pushing to the tail of Q, popping from the head of Q or querying $\sum_{i=1}^{|Q|} Q(i)g(|Q| - i)$.*

Proof. We use two instances of data structure $I_1 = (f_1, g_1, p_1, c_1)$ and $I_2 = (f_2, g_2, p_2, c_2)$ from Theorem 8. We make $g_1(x) = g_2(x) = g(-x)$. Whenever we want to perform a push x, we perform Update(x), RotateLeft and IncP on I_1, and perform RotateLeft on I_2. Whenever we want to perform a pop x, we perform Update$(-x)$ and IncP on I_2. Whenever we want to query, we query both I_1 and I_2 and return the sum. Each operation to the queue expands to constant number of operations on I_1 and I_2, so the running time is still $O(\log^2 n)$ amortized and the space is $O(n)$. ◀

4 Computing Shapley Value of Mean Width in 3-D

4.1 Classification of Cases

Let X be a convex polyhedron in 3-D, Lemma 1 becomes:

$$M_1(X) = \frac{1}{2} \sum_{e\in E(X)} l(e)\psi(e), \tag{4}$$

where $E(X)$ is the set of edges of X and $l(e)$ is the length of edge e.

Let $N \subset \mathbb{R}^3$ be a point set. Let n be the number of points in the point set. The mean width we are considering is $M_1(\text{conv}(N))$. Given a permutation π, and a point $p \in N$, for convenience we define $C(\pi, p) = \text{conv}(P_N(\pi, p) \cup \{p\})$ and $C'(\pi, p) = \text{conv}(P_N(\pi, p))$. We then define the contribution of p under permutation π to be:

$$\Delta(\pi, p) = M_1(C(\pi, p)) - M_1(C'(\pi, p)). \tag{5}$$

When there are at least three points in N, we have $\psi(e) = 1/2 - \chi(e)$ for all $e \in E(N)$. In the case where N is an edge e, we have $\psi(e) = 1$. It is also clear that a single point has width 0. So for $p \in N$, when considering the Shapley value $\phi(p)$, we can classify the permutations into three cases:

- Case 1: There is one point before p in the permutation. i.e., after inserting p, the point set forms a line segment.
- Case 2: There are two points before p in the permutation. i.e., after inserting p, the point set forms a triangle.
- Case 3: There are three or more points before p in the permutation.

In other words, we can write $\phi(p)$ as:

$$\phi(p) = \sum_{\pi} \frac{\Delta(\pi, p)}{n!} = \sum_{\pi:\text{Case 1}} \frac{\Delta(\pi, p)}{n!} + \sum_{\pi:\text{Case 2}} \frac{\Delta(\pi, p)}{n!} + \sum_{\pi:\text{Case 3}} \frac{\Delta(\pi, p)}{n!}. \tag{6}$$

For Case 1, we can rewrite the summation as:

$$\begin{aligned}\sum_{\pi:\text{Case 1}} \frac{\Delta(\pi, p)}{n!} &= \sum_{\substack{q \in P \\ q \neq p}} \sum_{\pi:\pi=(q,p,\dots)} \frac{\Delta(\pi, p)}{n!} = \sum_{\substack{q \in P \\ q \neq p}} \sum_{\pi:\pi=(q,p,\dots)} \frac{d(p,q)/2}{n!} \\ &= \sum_{\substack{q \in P \\ q \neq p}} \frac{d(p,q)}{2} \Pr(\pi = (q, p, \dots)) = \sum_{\substack{q \in P \\ q \neq p}} \frac{d(p,q)}{2} \frac{1}{n(n-1)}. \end{aligned} \tag{7}$$

Here $d(\cdot, \cdot)$ is the Euclidean distance function. So for any p, we can compute the summation in $O(n)$ time by enumerating q. And it takes $O(n^2)$ in total to compute for every p.

For Case 2, we can rewrite the summation as:

$$\begin{aligned}\sum_{\pi:\text{Case 2}} \frac{\Delta(\pi, p)}{n!} &= \sum_{\substack{q,r \in P \\ p,q,r \text{ are distinct}}} \sum_{\pi:\pi=(q,r,p,\dots)} \frac{\Delta(\pi, p)}{n!} \\ &= \sum_{\substack{q,r \in P \\ p,q,r \text{ are distinct}}} \sum_{\pi:\pi=(q,r,p,\dots)} \frac{\frac{d(p,q)+d(r,q)+d(p,r)}{4} - \frac{d(r,q)}{2}}{n!} \\ &= \sum_{\substack{q,r \in P \\ p,q,r \text{ are distinct}}} \frac{d(p,q) + d(p,r) - d(r,q)}{4} \Pr(\pi = (q, r, p, \dots)) \\ &= \sum_{\substack{q,r \in P \\ p,q,r \text{ are distinct}}} \frac{d(p,q) + d(p,r) - d(r,q)}{4} \frac{1}{n(n-1)(n-2)}. \end{aligned} \tag{8}$$

Like Case 1, we can compute the summation in $O(n^2)$ time by enumerating q and r. And it takes $O(n^3)$ in total to compute for every p.

In Case 3, each edge we are considering has two faces attached to it. In other words, each edge can be characterized by the common edge shared by two triangles formed by four points. We can denote an edge by (q, r, t_1, t_2), where (q, r) is the edge and is the common edge of $\triangle qrt_1$ and $\triangle qrt_2$. Without loss of generality, we assume unordered tuples when writing (q, r) and (t_1, t_2) to avoid double-counting. The exterior angle is completely determined by the quadruple. In case the convex hull has only 3 points, we allow $t_1 = t_2$. We can then apply Lemma 1 and express $M_1(\text{conv}(N'))$ for a point set N' as:

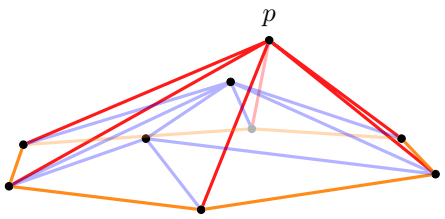

Figure 2 Change of edges when p is inserted, showing only the part visible to p. Red edges: edges added to the convex hull. Blue edges: edges removed from the convex hull. Orange edges: edges with angle changed.

$$M_1(\text{conv}(N')) = \frac{1}{2}\sum_{e\in E(\text{conv}(N'))} l(e)\psi(e) = \frac{1}{2}\sum_{\substack{e=(q,r,t_1,t_2)\\ q,r,t_1,t_2\in N'\\ \triangle qrt_1,\triangle qrt_2 \text{ are faces of } \text{conv}(N')}} l(e)\psi(e)$$
$$= \frac{1}{2}\sum_{\substack{e=(q,r,t_1,t_2)\\ q,r,t_1,t_2\in N'}} d(q,r)\psi(e) I_{\text{conv}(N')}(q,r,t_1,t_2)$$

where we use an indicator variable:

$$I_X(q,r,t_1,t_2) = \begin{cases} 1 & \triangle qrt_1, \triangle qrt_2 \text{ are faces of convex polyhedron } X \\ 0 & \text{otherwise} \end{cases}$$

Conditioning on π being Case 3, we can write $\Delta(\pi,p)$ as:

$$\Delta(\pi,p) = \frac{1}{2}\sum_{\substack{e=(q,r,t_1,t_2)\\ q,r,t_1,t_2\in N}} l(e)\psi(e)\left(I_{C(\pi,p)}(q,r,t_1,t_2) - I_{C'(\pi,p)}(q,r,t_1,t_2)\right)$$

We can write in the form of expectation as:

$$\sum_{\pi:\text{Case 3}} \frac{\Delta(\pi,p)}{n!} = \mathbb{E}_\pi[\Delta(\pi,p)|\pi:\text{Case 3}]\Pr(\pi:\text{Case 3})$$
$$= \frac{1}{2}\sum_{\substack{e=(q,r,t_1,t_2)\\ q,r,t_1,t_2\in N}} l(e)\psi(e)\mathbb{E}_\pi\left[I_{C(\pi,p)}(q,r,t_1,t_2) - I_{C'(\pi,p)}(q,r,t_1,t_2)\middle|\pi:\text{Case 3}\right]\Pr(\pi:\text{Case 3})$$

Observe that $I_{C(\pi,p)}(q,r,t_1,t_2) = 1$ implies π being Case 3. The summation reduces to:

$$\frac{1}{2}\sum_{q,r\in N}\sum_{\substack{t_1,t_2\in N\\ e=(q,r,t_1,t_2)}} l(e)\psi(e)\mathbb{E}_\pi[I_{C(\pi,p)}(q,r,t_1,t_2) - I_{C'(\pi,p)}(q,r,t_1,t_2)] \tag{9}$$

Now consider the impact when inserting p to $P_N(\pi,p)$. The convex hull does not change if p is inside $C'(\pi,p)$. Otherwise if we treat $C'(\pi,p)$ as an opaque object, all the faces visible to p will be removed in $C(\pi,p)$ and a pyramid-like cone with apex at p will be added to $C(\pi,p)$. In terms of edges, there are three types of edges: edges removed, edges added and edges with angle changed. See Figure 2 for an illustration.

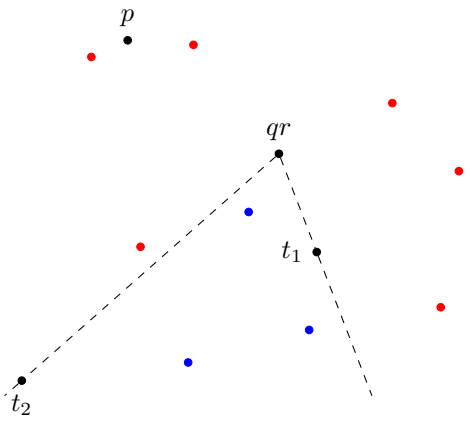

Figure 3 Projection of N along qr. Edge qr gets removed when forming $\text{conv}(P_N(\pi,p)\cup\{p\})$ if and only if (a) p is not in the cone formed by qr, t_1 and t_2, (b) q, r, t_1 and t_2 appear before p in π and (c) No point outside the cone (red point) appears before p in π.

As we are using a 4-tuple to represent an edge in the summation, edges with angles changed can be seen as removal and addition with different (t_1,t_2). We have:

$$I_{C(\pi,p)}(q,r,t_1,t_2) - I_{C'(\pi,p)}(q,r,t_1,t_2)$$
$$= \begin{cases} -1 & (q,r,t_1,t_2) \text{ is an edge of } C'(\pi,p) \text{ and visible to } p \\ 1 & (q,r,t_1,t_2) \text{ is an edge of } C(\pi,p) \text{ and } p\in\{t_1,t_2\} \\ 1 & (q,r,t_1,t_2) \text{ is an edge of } C(\pi,p) \text{ and } p\in\{q,r\} \\ 0 & \text{otherwise} \end{cases}$$

The first three cases are correspondent to blue+orange, orange and red edges in Figure 2 respectively. So we can write:

$$\begin{aligned} &\mathbb{E}_\pi[I_{C(\pi,p)}(q,r,t_1,t_2) - I_{C'(\pi,p)}(q,r,t_1,t_2)] \\ =&\Pr((q,r,t_1,t_2) \text{ is an edge of } C(\pi,p) \text{ and } p\in\{t_1,t_2\}) \\ &+\Pr((q,r,t_1,t_2) \text{ is an edge of } C(\pi,p) \text{ and } p\in\{q,r\}) \\ &-\Pr((q,r,t_1,t_2) \text{ is an edge of } C'(\pi,p) \text{ and visible to } p) \end{aligned} \tag{10}$$

This gives us a way to split the final summation in Equation 9 further into 3 summations. We will show how to compute them efficiently in the following subsection.

4.2 Handling Case 3

The idea is to enumerate edges (q,r) and compute $\sum_{\substack{t_1,t_2\in N \\ e=(q,r,t_1,t_2)}} l(e)\psi(e)\Pr(\cdot)$ for all p where $\Pr(\cdot)$ is one of the three probabilities in Equation 10.

(q,r,t_1,t_2) is an edge of $C'(\pi,p)$ and visible to p

Given a pair of points (q,r), we look at the projection of N along the direction of qr.

▶ **Lemma 10.** *(q,r,t_1,t_2) is an edge of $C'(\pi,p)$ and visible to p if and only if*
(a) *p is not in the cone formed by qr, t_1 and t_2.*
(b) *q, r, t_1 and t_2 appear before p in π.*
(c) *No point outside the cone appears before p in π.*

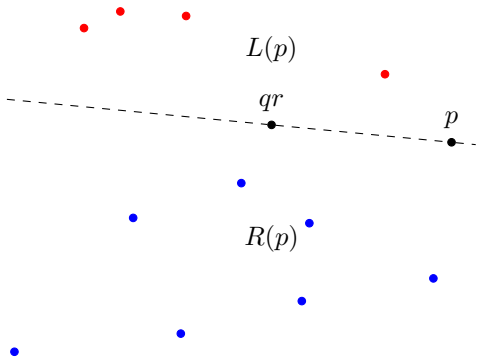

Figure 4 Partition of points based on whether a point is left to $p - qr$ or right to $p - qr$. Red points are left, and blue points are right.

Proof. (b) is immediate as we need q, r, t_1 and t_2 to be in $P_N(\pi, p)$ so that the edge can be in $C'(\pi, p)$. Given (b), (q, r, t_1, t_2) is an edge of $C'(\pi, p)$ if and only if $\triangle qrt_1$ and $\triangle qrt_2$ form two supporting planes of $P_N(\pi, p)$ if and only if no point outside the cone appears before p in π. Finally, (q, r, t_1, t_2) is visible to p if and only if pq and pr are completely outside $C'(\pi, p)$ if and only if p is not in the cone formed by qr, t_1 and t_2. See Figure 3 for an example. ◀

Treat qr as the origin and let $p_1, p_2, \ldots, p_{n-2}$ be the rest of the points in N sorted by polar angles relative to qr. In other words, $p_1, p_2, \ldots, p_{n-2}$ is the order of points when we sweep a ray starting from qr around counterclockwise, initially to the direction of positive x-axis. Let θ_i be the polar angle of p_i. For convenience, we treat the sequence $p_1, p_2, \ldots, p_{n-2}$ as a cyclic array, in the sense that $p_{n-1} = p_1$. Moreover, when we iterate through the sequence, θ_i is non-decreasing. In other words, when we iterate $p_1, p_2, \ldots, p_{n-1}, p_n, \ldots$, although p_{n-1} and p_1 are the same point, we treat $\theta_{n-1} = \theta_1 + 2\pi$.

Let $W_{qr}(p_i, p_j)$ be the number of points in the cone formed by qr, p_i and p_j. Let $g(i) = 4!(n-5-i)!/(n-i)!$. For a point p outside the cone, Lemma 2 gives us:

$$\Pr((q, r, p_i, p_j) \text{ is an edge of } C'(\pi, p) \text{ and visible to } p) = g(W_{qr}(p_i, p_j)) \tag{11}$$

Let $S(i)$ be the set of pairs (p_j, p_k) such that p_i is not in the cone formed by qr, p_j and p_k. We assume $j \le k$ and the cone is formed by sweeping from p_j to p_k counterclockwise. For a given point p_i, Lemma 10 and Equation 11 gives:

$$\begin{aligned}
&\sum_{\substack{t_1, t_2 \in N \\ e=(q,r,t_1,t_2)}} l(e)\psi(e) \Pr((q, r, t_1, t_2) \text{ is an edge of } C'(\pi, p) \text{ and visible to } p_i) \\
&= \sum_{(p_j, p_k) \in S(i)} d(q, r) \left(\frac{1}{2} - \frac{\theta_k - \theta_j}{2\pi} \right) g(W_{qr}(p_j, p_k))
\end{aligned}$$

Splitting the summation we get:

$$d(q, r) \left(\sum_{(p_j, p_k) \in S(i)} \frac{1}{2} g(W_{qr}(p_j, p_k)) - \sum_{(p_j, p_k) \in S(i)} \frac{\theta_k - \theta_j}{2\pi} g(W_{qr}(p_j, p_k)) \right) \tag{12}$$

We now show how to compute $\sum_{(p_j, p_k) \in S(i)} \frac{\theta_k - \theta_j}{2\pi} g(W_{qr}(p_j, p_k))$ for all i. For each point p, we partition the point set by whether a point is left to $p - qr$ or right to $p - qr$. As shown in Figure 4, we define:

$$L(p) = \{p' \in N : (p - qr) \times (p' - qr) > 0\} \quad R(p) = \{p' \in N : (p - qr) \times (p' - qr) < 0\} \tag{13}$$

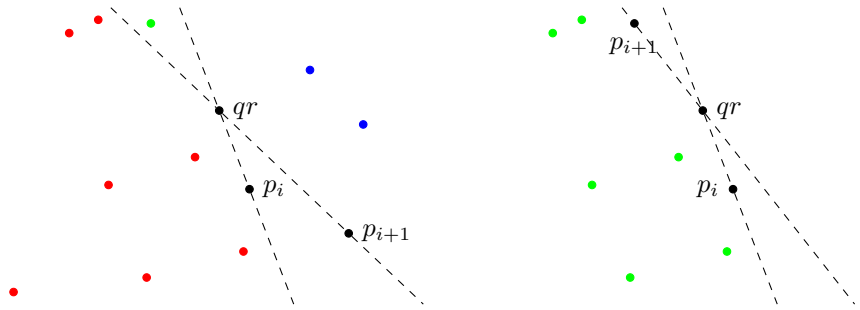

Figure 5 When changing from $S(i)$ to $S(i+1)$, pairs formed between p_{i+1} and $L(p_{i+1}) \cup \{p_{i+1}\}$ are removed and pairs formed between p_i and $R(p_i) \cup \{p_i\}$ are added. Left: $p_{i+1} \in L(p_i)$. Right: $p_{i+1} \in R(p_i)$.

Consider the difference between $S(i)$ and $S(i+1)$. It is easy to see:

$$\begin{aligned}\sum_{(p_j,p_k)\in S(i+1)} \frac{\theta_k-\theta_j}{2\pi} g(W_{qr}(p_j,p_k)) = &\sum_{(p_j,p_k)\in S(i)} \frac{\theta_k-\theta_j}{2\pi} g(W_{qr}(p_j,p_k)) \\ &+ \sum_{p_j\in R(p_i)\cup\{p_i\}} \frac{\theta_i-\theta_j}{2\pi} g(W_{qr}(p_i,p_j)) \\ &- \sum_{p_j\in L(p_{i+1})\cup\{p_{i+1}\}} \frac{\theta_j-\theta_{i+1}}{2\pi} g(W_{qr}(p_j,p_{i+1})) \end{aligned} \tag{14}$$

as demonstrated in Figure 5. We can further write:

$$\begin{aligned}&\sum_{p_j\in R(p_i)\cup\{p_i\}} \frac{\theta_i-\theta_j}{2\pi} g(W_{qr}(p_i,p_j)) \\ =&\theta_i \sum_{p_j\in R(p_i)\cup\{p_i\}} \frac{1}{2\pi} g(W_{qr}(p_i,p_j)) - \sum_{p_j\in R(p_i)\cup\{p_i\}} \frac{\theta_j}{2\pi} g(W_{qr}(p_i,p_j)) \end{aligned} \tag{15}$$

We will show how to compute $\sum_{p_j\in R(p_i)\cup\{p_i\}} f(j)g(W_{qr}(p_i,p_j))$ for an arbitrary function f and for all i at the same time. Equation 15 can then be computed by using $f(j) = 1/2\pi$ and $f(j) = \theta_j/2\pi$, respectively.

If we sort the points in $R(p_i)\cup\{p_i\}$ by polar angles as $q_1, q_2, \ldots, q_l$, where $l = |R(p_i)\cup\{p_i\}|$, it is easy to see $W_{qr}(p_i, q_j) = l - j$ for $1 \le j \le l$. In other words, the number of points within the cone formed by qr, p_i, and q_j is equal to the distance between q_j and p_i in the sequence $p_1, p_2, \ldots$.

Now we consider an instance of the data structure from Corollary 9. We use $g(i) = 4!(n-5-i)!/(n-i)!$ as the function used by the data structure. We start by choosing an arbitrary point p_i and consider a ray opposite to $p_i - qr$. We sweep this ray counterclockwise until hitting p_i, for each point p_j hit by the ray, we perform a push $f(j)$ to the data structure. After we hit p_i, we perform a query and the result is exactly $\sum_{p_j\in R(p_i)\cup\{p_i\}} f(j)g(W_{qr}(p_i,p_j))$. Next we sweep the ray to p_{i+1} and pop all the points that are left to p_{i+1}. They should all appear at the head of the data structure. We then perform a query and the result is exactly $\sum_{p_j\in R(p_{i+1})\cup\{p_{i+1}\}} f(j)g(W_{qr}(p_i,p_j))$. We keep sweeping, popping and querying until coming back to p_{i+n-3} which is p_{i-1}. During this process, each point gets pushed and popped at most twice and we perform $n-2$ queries. So the running time is $O(n \log^2 n)$ and the space complexity is $O(n)$ according to Corollary 9.

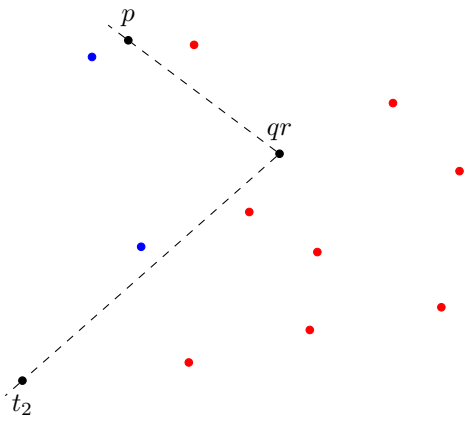

Figure 6 Projection of N along qr. Edge qr gets added when forming $\text{conv}(P_N(\pi, p) \cup \{p\})$ if and only if (a) q, r and t_2 appear before p in π (b) No point outside the cone formed by qr, p and t_2 (red point) appears before p in π.

Hence we can compute $\sum_{p_j \in R(p_i) \cup \{p_i\}} \frac{\theta_i - \theta_j}{2\pi} g(W_{qr}(p_i, p_j))$ for all i in $O(n \log^2 n)$ time and $O(n)$ space. With the same idea but sweeping the other way around, we can also compute $\sum_{p_j \in L(p_{i+1}) \cup \{p_{i+1}\}} \frac{\theta_j - \theta_{i+1}}{2\pi} g(W_{qr}(p_j, p_{i+1}))$ for all i in the same time and space complexity.

Next we only need to compute $\sum_{(p_j,p_k) \in S(i)} \frac{\theta_k - \theta_j}{2\pi} g(W_{qr}(p_j, p_k))$ for a single i and then used the pre-computed result to update to next $\sum_{(p_j,p_k) \in S(i+1)} \frac{\theta_k - \theta_j}{2\pi} g(W_{qr}(p_j, p_k))$ in constant time. Like in the previous case, we need to compute summations of the form $\sum_{(p_j,p_k) \in S(i)} f(j) g(W_{qr}(p_j, p_k))$. We again use an instance of data structure from Corollary 9 and use the same g as above. We start by pushing p_{i+1} and sweep counterclockwise. For each point hit, we first pop all the points that are left to the point, push the point to the data structure and finally perform a query. We stop after hitting p_{i+n-3} which is p_{i-1}. The sum of all the queries will then be $\sum_{(p_j,p_k) \in S(i)} f(j) g(W_{qr}(p_j, p_k))$. In this procedure, each point is pushed and popped at most once, and $n-3$ queries are made. So it takes $O(n \log^2 n)$ time and $O(n)$ space to compute for a single i. After that it takes $O(n)$ time to compute for all i by transitioning from i to $i+1$ in constant time.

Using the same idea, we can compute the other part of Equation 12 in $O(n \log^2 n)$ time and $O(n)$ space as well.

For a fixed pair (q, r), it takes $O(n \log n)$ to sort other points by polar angles on the projected plane. And it takes $O(n)$ to pre-compute g. Hence, it takes $O(n \log^2 n)$ time and $O(n)$ space to compute $\sum_{\substack{t_1,t_2 \in N \\ e=(q,r,t_1,t_2)}} l(e)\psi(e) \Pr(\cdot)$ for all p for the case where (q, r, t_1, t_2) is edge of $C'(\pi, p)$ and visible to p. And it takes $O(n^3 \log^2 n)$ time and $O(n)$ space overall by enumerating all pairs of (q, r).

(q, r, t_1, t_2) is an edge of $C(\pi, p)$ and $p \in \{t_1, t_2\}$

Again we fix a pair (q, r), and look at the projection of N along the direction of qr. Without loss of generality, assume $p = t_1$ in this case.

▶ **Lemma 11.** *(q, r, p, t_2) is edge of $C(\pi, p)$ if and only if*

(a) *q, r and t_2 appear before p in π.*

(b) *No point outside the cone formed by qr, p and t_2 appears before p in π.*

The proof will be almost the same as the proof for Lemma 10. See Figure 6 for an illustration.

In this case

$$\Pr((q,r,p,t_2) \text{ is an edge of } C(\pi,p)) = \frac{3!(n-4-W_{qr}(p,t_2))!}{(n-W_{qr}(p,t_2))!} \tag{16}$$

We have a slightly different $g(i) = 3!(n-4-i)!/(n-i)!$. Any point except p can form a cone with p. So we can write

$$\begin{aligned}
&\sum_{\substack{t_2 \in N \\ e=(q,r,p_i,t_2)}} l(e)\psi(e) \Pr((q,r,p,t_2) \text{ is an edge of } C(\pi,p)\ ) \\
=&\sum_{p_j \in R(p_i)} d(q,r)\left(\frac{1}{2} - \frac{\theta_i - \theta_j}{2\pi}\right) g(W_{qr}(p_i,p_j)) \\
+&\sum_{p_j \in L(p_i)} d(q,r)\left(\frac{1}{2} - \frac{\theta_j - \theta_i}{2\pi}\right) g(W_{qr}(p_j,p_i))
\end{aligned} \tag{17}$$

In Subsection 4.2, we've shown how to compute summations with very similar form as summations in Equation 17. The only differences are that we take $p_j \in R(p_i)$ instead of $p_j \in R(p_i) \cup \{p_i\}$ and we have a slightly different g here. But clearly these don't change the asymptotic running time. Hence it takes $O(n^3 \log^2 n)$ time in total and $O(n)$ space by using the same method.

(q,r,t_1,t_2) is an edge of $C(\pi,p)$ and $p \in \{q,r\}$

Without loss of generality, assume $p = q$. We look at the projection of N along the direction of pr.

▶ **Lemma 12.** *(p,r,t_1,t_2) is edge of $C(\pi,p)$ if and only if*

(a) *r, t_1 and t_2 appear before p in π.*

(b) *No point outside the cone formed by pr, t_1 and t_2 appears before p in π.*

The proof will be almost the same as the proof for Lemma 10. See Figure 7 for an illustration.

In this case

$$\Pr((p,r,t_1,t_2) \text{ is an edge of } C(\pi,p)) = \frac{3!(n-4-W_{qr}(t_1,t_2))!}{(n-W_{qr}(t_1,t_2))!} \tag{18}$$

And we use $g(i) = 3!(n-4-i)!/(n-i)!$. In this case, any pair (t_1,t_2) with $t_1 \neq t_2$ will contribute to result. So we can write

$$\begin{aligned}
&\sum_{\substack{t_1,t_2 \in N \\ e=(p,r,t_1,t_2)}} l(e)\psi(e) \Pr((p,r,t_1,t_2) \text{ is an edge of } C(\pi,p)) \\
=&\frac{1}{2}\sum_{p_k} \sum_{p_j \in R(p_k)} d(p,r)\left(\frac{1}{2} - \frac{\theta_k - \theta_j}{2\pi}\right) g(W_{qr}(p_k,p_j)) \\
+&\frac{1}{2}\sum_{p_k} \sum_{p_j \in L(p_k)} d(p,r)\left(\frac{1}{2} - \frac{\theta_j - \theta_k}{2\pi}\right) g(W_{qr}(p_j,p_k))
\end{aligned} \tag{19}$$

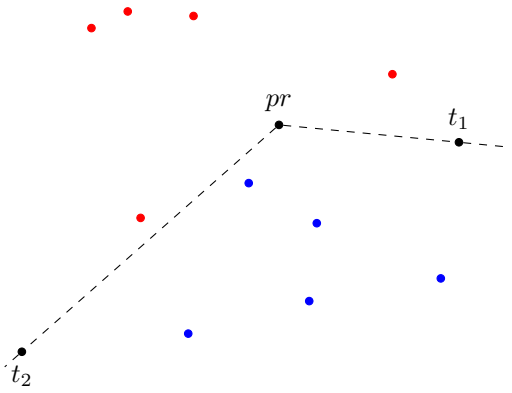

Figure 7 Projection of N along pr. Edge pr gets added when forming $\text{conv}(P_N(\pi, p) \cup \{p\})$ if and only if (a) r, t_1 and t_2 appear before p in π (b) No point outside the cone formed by pr, t_1 and t_2 (red point) appears before p in π.

We have a factor of $\frac{1}{2}$ because each pair is counted twice. In the previous case, all the inner summations have been pre-computed. So for a fixed (p, r), the summation can be computed in $O(n)$ time given previous computation. Hence in total it takes $O(n^3)$ time and no additional space to compute this case.

Now that all the cases take no more than $O(n^3 \log^2 n)$ time and $O(n)$ space to compute, we present our main theorem:

▶ **Theorem 13.** *Shapley values for mean width for a point set in 3-D can be computed in $O(n^3 \log^2 n)$ time and $O(n)$ space.*

5 Discussion

We have presented an algorithm to compute Shapley values of a point set in 3-D with respect to the mean width of its convex hull. We provided an efficient algorithm based on a data structure for a variant of dynamic convolution. We believe the data structure may be of independent interest.

Our algorithm naturally extends to higher dimension to compute Shapley values for $M_{d-2}(\text{conv}(P))$ for a d-dimensional point set P. This relies on the fact that the orthogonal space of a $(d-2)$-face is a plane. In general, it takes $O(n^d \log^2 n)$ time to compute the Shapley values. It is also known that for a convex polytope X, $M_{d-1}(X)$ is equivalent to the $(d-1)$-volume of the boundary of X up to some constant [20]. Hence the algorithm by Cabello and Chan [7] with natural extension can be used to compute Shapley values for $M_{d-1}(\text{conv}(P))$ in $O(n^d)$ time. It would be natural to ask whether there are efficient algorithms to compute $M_i(\text{conv}(P))$ in general.

We can also consider ϵ-coreset of Shapley values for geometric objects. Let P be a set of geometric objects and v be a characteristic function on P. We can define the ϵ-coreset $\widetilde{P}$ to be a weighted set of geometric objects such that, for any geometric objects x, $(1-\epsilon)\phi_{P\cup\{x\}}(x) \leq \phi_{\widetilde{P}\cup\{x\}}(x) \leq (1+\epsilon)\phi_{P\cup\{x\}}(x)$ where ϕ_N means that the underlying player set for the Shapley value is N. Intuitively, $\phi_{P\cup\{x\}}(x)$ means the contribution x makes when x is added as an additional player. Does $\widetilde{P}$ exist? If so, what is the upper and lower bound of its size? How fast can we find a coreset?

References

1 Pankaj K. Agarwal, Sariel Har-Peled, Subhash Suri, Hakan Yıldız, and Wuzhou Zhang. Convex hulls under uncertainty. In Andreas S. Schulz and Dorothea Wagner, editors, *Algorithms – ESA 2014*, pages 37–48, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg. doi:10.1007/978-3-662-44777-2_4.

2 Deepak Ajwani, Saurabh Ray, Raimund Seidel, and Hans Raj Tiwary. On computing the centroid of the vertices of an arrangement and related problems. In Frank Dehne, Jörg-Rüdiger Sack, and Norbert Zeh, editors, *Algorithms and Data Structures*, pages 519–528, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. doi:10.1007/978-3-540-73951-7_45.

3 David Alonso-Gutiérrez and Joscha Prochno. On the Gaussian behavior of marginals and the mean width of random polytopes. *Proceedings of the American Mathematical Society*, 143(2):821–832, 2015. doi:10.1090/S0002-9939-2014-12401-4.

4 Mauricio Alvarez-Manilla, Abbas Edalat, and Nasser Saheb-Djahromi. An extension result for continuous valuations. *Journal of the London Mathematical Society*, 61(2):629–640, 2000. doi:10.1112/S0024610700008681.

5 Boris Aronov and Matthew J. Katz. Batched point location in SINR diagrams via algebraic tools. *ACM Trans. Algorithms*, 14(4):41:1–41:29, August 2018. doi:10.1145/3209678.

6 Karoly J. Böröczky, Ferenc Fodor, Matthias Reitzner, and Viktor Vígh. Mean width of random polytopes in a reasonably smooth convex body. *Journal of Multivariate Analysis*, 100(10):2287–2295, 2009. doi:10.1016/j.jmva.2009.07.003.

7 Sergio Cabello and Timothy M. Chan. Computing Shapley Values in the Plane. In Gill Barequet and Yusu Wang, editors, *35th International Symposium on Computational Geometry (SoCG 2019)*, volume 129 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 20:1–20:19, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.SoCG.2019.20.

8 Satya R. Chakravarty, Manipushpak Mitra, and Palash Sarkar. *A Course on Cooperative Game Theory*. Cambridge University Press, 2015. doi:10.1017/CBO9781107415997.

9 Timothy M. Chan. A (slightly) faster algorithm for Klee's measure problem. *Computational Geometry*, 43(3):243–250, 2010. Special Issue on 24th Annual Symposium on Computational Geometry (SoCG'08). doi:10.1016/j.comgeo.2009.01.007.

10 Martin Fink, John Hershberger, Nirman Kumar, and Subhash Suri. Hyperplane Separability and Convexity of Probabilistic Point Sets. In Sándor Fekete and Anna Lubiw, editors, *32nd International Symposium on Computational Geometry (SoCG 2016)*, volume 51 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 38:1–38:16, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.SoCG.2016.38.

11 Gudmund Skovbjerg Frandsen, Johan P. Hansen, and Peter Bro Miltersen. Lower bounds for dynamic algebraic problems. *Information and Computation*, 171(2):333–349, 2001. doi:10.1006/inco.2001.3046.

12 Alfred Gray. *Steiner's Formula*, pages 209–229. Birkhäuser Basel, Basel, 2004. doi:10.1007/978-3-0348-7966-8_10.

13 Hugo Hadwiger. *Vorlesungen über inhalt, Oberfläche und Isoperimetrie*, volume 93. Springer-Verlag, 2013. doi:10.1007/978-3-642-94702-5.

14 Sergiu Hart. Shapley value. In John Eatwell, Murray Milgate, and Peter Newman, editors, *Game Theory*, pages 210–216. Palgrave Macmillan UK, London, 1989. doi:10.1007/978-1-349-20181-5_25.

15 Lingxiao Huang, Jian Li, Jeff M. Phillips, and Haitao Wang. epsilon-kernel coresets for stochastic points. In Piotr Sankowski and Christos Zaroliagis, editors, *24th Annual European Symposium on Algorithms (ESA 2016)*, volume 57 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 50:1–50:18, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ESA.2016.50.

16 Roland Huber. Continuous valuations. *Mathematische Zeitschrift*, 212(1):455–477, January 1993. doi:10.1007/BF02571668.

17 Daniel A. Klain. A short proof of Hadwiger's characterization theorem. *Mathematika*, 42(2):329–339, 1995. doi:10.1112/S0025579300014625.

18 Stefan Langerman. On the complexity of halfspace area queries. *Discrete & Computational Geometry*, 30(4):639–648, October 2003. doi:10.1007/s00454-003-2856-2.

19 Maarten Löffler and Marc van Kreveld. Largest and smallest convex hulls for imprecise points. *Algorithmica*, 56(2):235, March 2008. doi:10.1007/s00453-008-9174-2.

20 Roger E. Miles. Poisson flats in Euclidean spaces. part I: A finite number of random uniform flats. *Advances in Applied Probability*, 1(2):211–237, 1969. doi:10.2307/1426218.

21 Josef S. Müller. On the mean width of random polytopes. *Probability Theory and Related Fields*, 82(1):33–37, June 1989. doi:10.1007/BF00340011.

22 Roger B. Myerson. *Game Theory: Analysis of Conflict*. Harvard University Press, 1991. doi:10.2307/j.ctvjsf522.

23 John H. Reif and Stephen R. Tate. On dynamic algorithms for algebraic problems. *Journal of Algorithms*, 22(2):347–371, 1997. doi:10.1006/jagm.1995.0807.

24 Alvin E. Roth, editor. *The Shapley Value: Essays in Honor of Lloyd S. Shapley*. Cambridge University Press, 1988. doi:10.1017/CBO9780511528446.

25 Rolf Schneider. *Convex Bodies: The Brunn–Minkowski Theory*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2 edition, 2013. doi:10.1017/CBO9781139003858.

26 Subhash Suri, Kevin Verbeek, and Hakan Yıldız. On the most likely convex hull of uncertain points. In Hans L. Bodlaender and Giuseppe F. Italiano, editors, *Algorithms – ESA 2013*, pages 791–802, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. doi:10.1007/978-3-642-40450-4_67.

27 Eyal Winter. The Shapley value. In *Handbook of Game Theory with Economic Applications*, volume 3, chapter 53, pages 2025–2054. Elsevier, 2002. doi:10.1016/S1574-0005(02)03016-3.

28 Jie Xue, Yuan Li, and Ravi Janardan. On the expected diameter, width, and complexity of a stochastic convex-hull. In Faith Ellen, Antonina Kolokolova, and Jörg-Rüdiger Sack, editors, *Algorithms and Data Structures*, pages 581–592, Cham, 2017. Springer International Publishing. doi:10.1007/978-3-319-62127-2_49.