Annales Univ. Sci. Budapest., Sect. Comp. 47 (2018) 197–209

SOME MATHEMATICAL PROPERTIES OF THE PERFORMANCE MEASURES APPLIED FOR POINT CLOUD DATABASES

István Csabai, László Dobos,
Attila Kiss and János M. Szalai-Gindl

(Budapest, Hungary)

Communicated by András Benczúr

(Received March 30, 2018; accepted June 20, 2018)

Abstract. Data megatsunami remains a genuine challenge in a variety of scientific domains which typically focus on multidimensional objects. A distributed architecture is needed to store massive amounts of data. In such an environment, the optimization for data placement strategy is an important issue which has been of concern to researchers for several decades. To this purpose, tiling algorithms were presented in our previous work which can optimize data partitioning for load balancing and nearest neighbor search. These are based on a multidimensional histogram which can be utilized for data placement. Two performance measures were also introduced which can support the comparison of the efficiency of methods. This paper contributes by extending them with a new measure and examines the mathematical properties of these measures.

1. Introduction

The scientific world is working with vast amounts of data. For example, the Millennium XXL cosmological N-body simulation needs 100 TB space [1].

Key words and phrases: Multi-dimensional histogram partitioning, data placement, load balancing.

2010 Mathematics Subject Classification: 68M14, 68M20.

The project has been supported by the Hungarian National Research, Development and Innovation Office under grant no. OTKA NN 114560 and by the European Union, co-financed by the European Social Fund (EFOP-3.6.3-VEKOP-16-2017-00002).

There are areas that have even greater needs. "Projections indicate that by 2020, more than 60 PB of archived data will be accessible to astronomers", according to [3]. Nowadays, geoinformatics also has petabyte-scaled geospatial data volumes (see e.g. [9]). In order to cope with the data megatsunami, to store and manage them, data should be distributed across multiple servers.

An important aspect is that data updates are rare. In this paper, we assume that we have static multidimensional data, a shared nothing environment and a global index structure. Furthermore, we pay particular attention to sparse but strongly clustered points on a hierarchy of scales which form so-called point clouds such as cosmological N-body simulations, intersections of road networks etc. These point clouds can often be mapped to continuous $\mathcal{D}$-dimensional parameter space. This representation could be useful for specific queries and operations including nearest neighbor search, box queries, cluster analysis and outlier identification. Special support is needed for the aforementioned challenges in the data storage layer and leading to the concept of the point cloud database [6].

Mass scientific data ingestion processes typically consist of parallel Extract-Transform-Load (ETL). A multidimensional histogram is to be formed during the extraction or transformation phase based on the spatial attributes of objects, before the data are distributed to the servers. Using this prior information about approximate estimate of the data distribution, three methods were proposed for grouping histogram bins and assigning them to servers in our previous work [12].

Two performance measures were also introduced for the comparison of the efficiency of methods. This paper is its extension with a new performance measure and some mathematical properties. Moreover, we give a method for bin shell traversal, which is related to the computation of the proximity search-related measure.

This paper is organized as follows. Used symbols are described in Sec. 2. Sec. 3 provides details about two different measures of goodness of a tiling with reference to proximity search and box queries. Furthermore, it contains a method for histogram bin shell traversal which is one of the most time-consuming part of the computation of proximity search-related measure. Finally, Sec. 4 presents a summary of our results.

2. Used symbols

We assume continuous $\mathcal{D}$-dimensional parameter space which can be thought of as a vector space with the Euclidean norm. We restrict our attention to a search space which is a bounding box of r data points $\{d_l\}_{l=1}^{r}$ in $\mathbb{R}^{\mathcal{D}}$. We utilize regular binning (equal bin width) for the histogram construction and write n

for the value of the histogram resolution so there are $n^{\mathcal{D}}$ bins: $\{B_j\}_{j=1}^{n^{\mathcal{D}}}$. If the histogram is considered as an array, a bin index is denoted by $(x_1, x_2, \ldots, x_{\mathcal{D}})$ which means the spatial position of the bin in the histogram. We will use the notation $w(B_j)$ for the contained point amounts (weight) of bin B_j, $\bar{B} = r/n^{\mathcal{D}}$ for the mean weight of bins. Let s be the number of servers $S_1, S_2, \ldots, S_s$ ($s \geq 2$). An $\mathcal{A} : \{B_j\}_{j=1}^{n^{\mathcal{D}}} \to \{S_i\}_{i=1}^{s}$ bin assignment function (or assignment) will take the value of S_i if the bin B_j is assigned to the server S_i. In this paper, we only consider surjective bin assignment functions (which are typically non-injective). We denote the number of data points (weight) and the number of bins belonging to a server S_i by $w(S_i)$ and $b(S_i)$, respectively. The mean weight of servers is defined simply as $\bar{S} = r/s$.

3. Performance measures

The histogram bins should be grouped and assigned to servers in the following manner:

- the server weights must be roughly the same (ensuring load balance of the servers),
- we want to keep clusters of data points together as much as possible (preserving data locality).

Furthermore, we must take into account a variety of query types. One of them is the similarity queries where the user retrieves the result set relative to a query point. It includes exact match and nearest neighbor queries. Also, these are closely linked to intention to bring points together from a cluster. Another query type is the box queries where retrieved data come from a given hyper-rectangle so query criteria include an interval for each dimension. However, different aims cannot be supported at the same time so there is a trade-off between them.

When methods are applied for a given histogram we would like to compare them. Therefore, we attempt to quantify the results of a tiling technique in terms of a selected viewpoint. We defined performance measures in [12] which take values between 0 (worst) and 1 (best). The k-nearest neighbor measure is briefly presented for the self-containment of this paper, a new performance measure is introduced and some properties of them will be given in this section.

3.1. *k*-nearest neighbor measure

A k^{th} nearest neighbor of a given datum point d in $\{d_l\}_{l=1}^{r}$ has the k^{th} smallest distance among all distances between d and $d' \in \{d_l\}_{l=1}^{r}$. We repeat only the most important definitions from [12].

Definition 3.1. A shell of a histogram bin B is any hypersphere drawn around the center of B which passes through any other bin center. To ease terminology, we will use term "a bin is on a shell" if the center of this bin is on the shell.

Definition 3.2. The bin shell difference between B_1 and B_2 is the number of shells centered on B_1 which are contained by the shell defined by B_1 and B_2.

We will discuss traversing histogram bins in the order of bin shells in Sec. 3.2.

Suppose the results of a given method define a bin assignment function $\mathcal{A}$. Let us successively take shells around B_j and accumulate bins on the shells until the bins contain at least k data points taken together. Let $K^*(B_j, k)$ denote the set of these bins. We define a subset of $K^*(B_j, k)$, written as $K(B_j, k, \mathcal{A})$ or K for brevity, which has such bins for which $\mathcal{A}$ takes the same value as for B_j.

Definition 3.3. The k-nearest neighbor measure of a certain bin with respect to an assignment $\mathcal{A}$ for the value of k is

$$\kappa(B_j, k, \mathcal{A}) = \min\left\{1, \frac{w(K)}{k}\right\}. \tag{3.1}$$

Definition 3.4. The k-nearest neighbor measure of a certain server with respect to an assignment $\mathcal{A}$ for the value of k is

$$\kappa(S_i, k, \mathcal{A}) = \frac{1}{b(S_i)} \sum_{B_j \in S_i} \kappa(B_j, k, \mathcal{A}). \tag{3.2}$$

Definition 3.5. The k-nearest neighbor measure of an assignment $\mathcal{A}$ for the value of k is

$$\kappa(k, \mathcal{A}) = \frac{1}{s} \sum_{i=1}^{s} \kappa(S_i, k, \mathcal{A}) = \frac{1}{s} \sum_{i=1}^{s} \left(\frac{1}{b(S_i)} \sum_{B_j \in S_i} \kappa(B_j, k, \mathcal{A}) \right) \tag{3.3}$$

3.2. Bin shell traversal

The computation efficiency of k-nearest neighbor measure is important if hierarchical tiling is applied (see [12]). We want to search a quick method for bin shell traversal which is one of the most time-consuming part of this measure computation.

This section presents a shell traversing method starting from a given bin to determine the order of bin shells. For simplicity, it deals with the case where the histogram grid is a regular rectangular one but it can easily be generalized

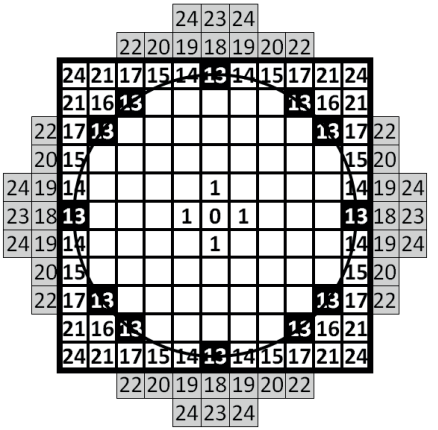

Figure 1. Black square contains histogram bins. The shell differences from the central bin are displayed. As $25 = 0^2+5^2 = 3^2+4^2$, if the central bin has index (x_1, x_2), the indices of black bins are $(x_1 - 3, x_2 + 4), (x_1 + 0, x_2 + 5), (x_1 + 3, x_2 + 4) \dots$ etc. because of the squared distance between them. The black bins are on the thirteenth shell.

to the general case. The squared Euclidean distance between the centers of the side neighbor bins can be taken to be 1 (with the appropriate measurement unit). Therefore, if the squared distance of two arbitrary bin centers is taken we get a non-negative integer. Centers can be identified by bin indices of the histogram. Accordingly, suppose it is possible to find $\mathcal{D}$ square numbers relatively quickly which sum to given a natural number then bin indices of a shell can be located in a faster way than brute force search. To be specific, let us consider Fig. 1. It depicts a histogram bounded by black square where some shell differences from the central bin are given (see Def. 3.2). (Note: the grey bins are beyond the histogram but their shell differences from the central bin are less than or equal to the shell differences between the central bin and corners which make them necessary for the complete picture.) Suppose the central bin has index (x_1, x_2) and the squared distance between itself and black bins is 25. As $25 = 0^2 + 5^2 = 3^2 + 4^2$, the indices of black bins are $(x_1-3, x_2+4), (x_1+0, x_2+5), (x_1+3, x_2+4) \dots$ etc. Furthermore, take a series of integers $1, 2, \dots$, we can write them in the form $j_1^2+\dots+j_{\mathcal{D}}^2$ where $j_1, \dots, j_{\mathcal{D}}$ are integers. Thus we can determine shells, one after another. Note: not all natural numbers can be represented as the sum of two or three square numbers based on Fermat's theorem on sums of two square numbers and Legendre's three-square theorem [7]. The first theorem states that an odd prime p can be expressed as the sum of two square numbers if and only if $p = 4k + 1$ with

integer k. The second theorem states that a natural number m cannot be expressed as the sum of three square numbers if and only if $m = 4^k(8l + 7)$ with integers k, l. However, every natural number can be expressed as the sum of $\mathcal{D}$ integer square numbers with space dimension $\mathcal{D} > 3$ because of Lagrange's four-square theorem [7] which states that every natural number can be expressed as the sum of four square numbers. Therefore, bin indices on the i^{th} shell can be computed by the sum of $\mathcal{D}$ square numbers representation of i in this case although this does not mean that finding bins with a given shell difference becomes easy. The issue is how to compute them quickly.

For $\mathcal{D} = 2$, we will apply the classic Cornacchia's algorithm (see for instance [2]). This method provides only primitive solutions where the two integers are relative primes (see Alg. 1). However, we need every solution for all bin indices of a given shell. As we will see, for this, the prime factorization is needed. For the sake of completeness, the details of the method are given. The following proposition is easily seen and its proof is left to the reader:

Proposition 3.1. *For prime p, primitive solutions to $p = x^2 + y^2$, if any, include all solutions.*

We proceed to show how to construct solutions for a composite number using primitive solutions of its prime factors. Let $norm(a + bi)$ be the square of the absolute value of $a + bi \in \mathbb{Z}[i]$ where $\mathbb{Z}[i]$ stands for Gaussian integers as usual. We denote by $\mathbb{Z}^+$ the set of the positive integers.

Theorem 3.2 ([5, Theorem 9.2])**.** *A prime p in $\mathbb{Z}^+$ is composite in $\mathbb{Z}[i]$ if and only if it is a sum of two squares.*

Proof. The proof is found in [5]. ■

Corollary 3.1 ([5, Corollary 9.3])**.** *If a prime p in $\mathbb{Z}[i]$ is composite, and $p \neq 2$, then up to unit multiple p has exactly two Gaussian prime factors, which are conjugate and have norm p.*

Proof. The proof is found in [5]. ■

Remark 3.1. The prime $p = 2$ is the product of the square of $(1 + i)$ by a unit: $2 = -i(1 + i)^2$ where $1 + i$ is a Gaussian prime thus it has only one Gaussian prime factor up to unit multiple. However, $2 = (1+i)(1-i)$ hence it is also a product of a Gaussian prime and its conjugate (and these have norm 2). Consequently, if a prime integer $p > 0$ is a sum of two square numbers, then there exist exactly two conjugates $P, \overline{P} \in \mathbb{Z}[i]$ with norm p.

Let the factorization of integer m be $p_1 \cdot p_2 \cdots p_t$ where $p_1, p_2, \ldots, p_t$ are primes (which are not necessarily different) and suppose each prime factor

p_j is 2 or may be written of the form $4k + 1$ with integer k. Therefore, Cornacchia's algorithm can applied to them because of Fermat's theorem on sums of two square numbers to determine the form $a_j^2 + b_j^2$. So $p_j = P_j \cdot \overline{P_j} = (a_j + b_j i)(a_j - b_j i)$ and $norm(P_j) = norm(\overline{P_j}) = a_j^2 + b_j^2$. Moreover, $m = p_1 \cdot p_2 \cdots p_t = norm(Q_1) \cdot norm(Q_2) \cdots norm(Q_t)$ where $Q_j \in \{P_j, \overline{P_j}\}$ and $m = norm(Q_1 \cdot Q_2 \cdots Q_t)$ because of the multiplicative property. Note: $norm(Q_1) \cdot norm(Q_2) = norm(Q_1 \cdot Q_2)$ may be interpreted as the famous Brahmagupta–Fibonacci identity. Furthermore, $Q_1 \cdot Q_2 \cdots Q_t$ is a unique Gaussian integer (up to unit multiple) which has (unique) norm in the form $a^2 + b^2$. Finally, if all possible ways of multiplying Gaussian integers $Q_1, Q_2, \ldots, Q_t$ are computed and their norms are taken, then we obtain all solutions for the composite number m. Note: only relatively small integer will need to be factored down to its prime factors so the algorithm can be performed within a reasonable time. No general efficient algorithm is known for the $x^2 + y^2$ decomposition of an arbitrary natural number [4].

Algorithm 1: Cornacchia's algorithm (special case) based on [2]

input : A natural number m
output: All the primitive solutions (x, y) of $m = x^2 + y^2$

```
r_0 = m;
for t ← 1 to m/2 do
    if t^2 ≡ −1 (mod m) then
        r_1 ← m;
        r_2 ← t;
        while r_2^2 ≥ m do
            tmp ← r_2;
            r_2 ← r_1 (mod r_2);
            r_1 ← tmp;
        end
        if r_1^2 > m then
            (r_2, r_1 (mod r_2)) is a primitive solution;
        end
    end
end
```

For $\mathcal{D} > 2$, we can apply backtracking decomposition method (see Alg. 2) which can be found in Schorn's technical report [11]. It also has tolerable time cost for small integer. We remark that there is more faster way for certain dimensions (e.g. $\mathcal{D} = 3, 4$) see for instance [10] and [11] for more details. Furthermore, even if we can find at least a solution for the prime factors with space dimension $\mathcal{D} \neq 2, 4, 8$ efficiently, it would not give the result for the gen-

eral problem. This is because there is no generalization of the Brahmagupta–Fibonacci identity where the multiplication of two number decomposition produces $\mathcal{D}$ square numbers on the other side of the equation based on Hurwitz's theorem (see [8]).

Algorithm 2: Backtracking decomposition method on the basis of the Schorn's report [11]

input : A natural number m, the dimension $\mathcal{D}$
output: All the solutions $(x_1, x_2, \ldots, x_{\mathcal{D}})$ for $m = x_1^2 + x_2^2 + \cdots + x_{\mathcal{D}}^2$

```
if D = 2 then
    give the prime factorization of m;
    apply Alg. 1 for prime factors and Brahmagupta–Fibonacci
     identity, as described in the text;
else
    for i ← ⌊√m⌋ to 1 do
        apply this algorithm recursively for m − i² and D − 1;
        if the result is not empty then
            the output with i² is a solution;
        end
    end
end
```

3.3. Box measure

A $\mathcal{D}$ dimensional box query I is a subset of $B \subset \mathbb{R}^{\mathcal{D}}$ which can be written as $I = [l_1, u_1] \times \cdots \times [l_{\mathcal{D}}, u_{\mathcal{D}}]$, where the $[l_d, u_d]$ are ordinary intervals within the bounding box of the data points. Without restricting generality, we assume that the 'granularity' of box queries is the same as the histogram bins, i.e. the sides of the query box are aligned on the histogram grid. If the resolution of the histogram is equal along all axes, there are $\binom{n+1}{2}^{\mathcal{D}}$ different such box queries. Let us denote data points selected by a box query on a given server by $I \cap S_i$, while the number of data points selected by $w(I)$ and $w(I \cap S_i)$.

Suppose the processing time (or workload) of a box query I on a given server S_i is proportional to $I \cap S_i$. This workload is expressed as $\frac{w(I \cap S_i)}{w(S_i)}$. To quantify the load balancing performance of a given assignment $\mathcal{A}$ with respect to a box query I we define the server speed

$$\beta_i(I) = 1 - \frac{w(I \cap S_i)}{w(S_i)}, \tag{3.4}$$

and the box measure with respect to I

$$\beta(I) = \frac{1}{s}\sum_{i=1}^{s}\beta_i(I). \tag{3.5}$$

(Note: $\beta(I)$ takes values between 0 (worst) and 1 (best).) Considering all possible box queries with equal probability regardless of volume we average over all possible queries to get

$$\beta = \frac{1}{\binom{n+1}{2}^{\mathcal{D}}}\sum_{I}\beta(I) = 1 - \frac{1}{s\cdot\binom{n+1}{2}^{\mathcal{D}}}\sum_{I}\sum_{i=1}^{s}\frac{w(I\cap S_i)}{w(S_i)}. \tag{3.6}$$

For an optimally balanced assignment $\mathcal{A}$, i.e. $w(S_i) = \bar{S}$, the box measure is

$$\beta = 1 - \frac{1}{r\cdot\binom{n+1}{2}^{\mathcal{D}}}\sum_{I} w(I). \tag{3.7}$$

It follows from the fact that $r = s\cdot\bar{S}$ and $\sum_{i=1}^{s} w(I\cap S_i) = w(I)$ because the assigned bin set of the servers are disjoint.

The following proposition will be important.

Proposition 3.3. *A bin B_j is contained in exactly $c_j := \prod_{d=1}^{\mathcal{D}}(x_d+1)\cdot(n-x_d)$ distinct box queries where x_d is the d^{th} value of index $(x_1, x_2, \ldots, x_{\mathcal{D}})$ of B_j.*

Proof. The bin B_j with index $(x_1, x_2, \ldots, x_{\mathcal{D}})$ is contained by box query $I = [l_1, u_1]\times\cdots\times[l_{\mathcal{D}}, u_{\mathcal{D}}]$ where $l_d \le x_d, x_d + 1 \le u_d$ for all $d = 1, \ldots, \mathcal{D}$. It is easy to count all possibilities for the lower and upper bounds because their values can be chosen independently of each other and along each dimension. For dimension d, there are $x_d + 1$ options for l_d and $n - x_d$ options for u_d. The rest of the proof is straightforward. ■

By the previous Prop. 3.3, we get the following results:

$$c_{\min} = \prod_{d=1}^{\mathcal{D}}(0+1)\cdot(n-0) = n^{\mathcal{D}} \tag{3.8}$$

$$c_{\max} = \prod_{d=1}^{\mathcal{D}}\left(\left\lfloor\frac{n}{2}\right\rfloor + 1\right)\cdot\left(n - \left\lfloor\frac{n}{2}\right\rfloor\right) = \left(\left\lceil\frac{n}{2}\right\rceil\right)^{2\mathcal{D}} \tag{3.9}$$

We have that:

$$\sum_{I} w(I) = \sum_{B_j} c_j\cdot w(B_j) = \sum_{B_j} w(B_j)\cdot\prod_{d=1}^{\mathcal{D}}(x_d+1)\cdot(n-x_d), \tag{3.10}$$

where $(x_1, x_2, \ldots, x_{\mathcal{D}})$ is the index of the bin B_j.

Proposition 3.4. *$\sum_I w(I)$ attains its maximum (minimum) value if the coefficient of $c_{\max}$ ($c_{\min}$) is r.*

Proof. We give the proof only for the maximal case. The other case can be done analogously. Suppose, contrary to our claim, that there is a weight distribution $\{w(B_j)\}$ which is better. Then:

$$c_{\max} \cdot r < \sum_j c_j \cdot w(B_j),$$
$$r < \sum_j \frac{c_j}{c_{\max}} \cdot w(B_j) \leq \sum_j w(B_j) = r,$$

a contradiction. ■

Combining Eq. (3.10) with Prop. 3.4 we obtain:

Corollary 3.2. *For an optimally balanced assignment $\mathcal{A}$:*

$$1 - \left(\frac{2 \cdot \left\lceil \frac{n}{2} \right\rceil^2}{(n+1) \cdot n}\right)^{\mathcal{D}} \leq \beta \leq 1 - \left(\frac{2}{n+1}\right)^{\mathcal{D}}.$$

If $\mathcal{D}$ is large enough then β is around 1 because

$$\lim_{n\to\infty} \left(1 - \left(\frac{2 \cdot \left\lceil \frac{n}{2} \right\rceil^2}{(n+1) \cdot n}\right)^{\mathcal{D}}\right) = \lim_{n\to\infty} \left(1 - \left(\frac{2 \cdot \frac{n^2}{4}}{(n+1) \cdot n}\right)^{\mathcal{D}}\right) =$$
$$= \lim_{n\to\infty} \left(1 - \left(\frac{n}{2 \cdot (n+1)}\right)^{\mathcal{D}}\right) =$$
$$= 1 - \frac{1}{2^{\mathcal{D}}}.$$

Therefore with sufficiently large $\mathcal{D}$, the results of the optimization for load balancing also implies server speed optimality for box queries, as one may have expected.

For uniformly distributed dataset, we treat $w(B_j)$ as a random variable whose distribution is binomial: $w(B_j) \sim B(r, 1/n^{\mathcal{D}})$, thus the expected value of any bin weight is

$$\mathbb{E}[w(B_j)] = \frac{r}{n^{\mathcal{D}}} = \bar{B}. \tag{3.11}$$

Proposition 3.5. *For uniformly distributed dataset,*

$$\mathbb{E}\left[\sum_I w(I)\right] = \bar{B} \cdot \left(\frac{n \cdot (n+1) \cdot (n+2)}{6}\right)^{\mathcal{D}}.$$

Proof. Throughout the proof, $C(I)$ denotes the number of contained bins of $I = [l_1, u_1] \times \cdots \times [l_{\mathcal{D}}, u_{\mathcal{D}}]$ which is the size of I, i.e., $\prod_{i=1}^{\mathcal{D}}(u_i - l_i)$. From Eq. 3.11 it follows that $\mathbb{E}[w(I)]$ is equal to the product of $C(I)$ and $\bar{B}$ (by the linearity of expected value operator) and therefore

$$\mathbb{E}\left[\sum_I w(I)\right] = \bar{B} \cdot \sum_{l_1=0}^{n-1} \sum_{u_1=l_1+1}^{n} \cdots \sum_{l_{\mathcal{D}}=0}^{n-1} \sum_{u_{\mathcal{D}}=l_{\mathcal{D}}+1}^{n} \prod_{i=1}^{\mathcal{D}}(u_i - l_i) =$$

$$= \bar{B} \cdot \sum_{l_1=0}^{n-1} \sum_{u_1=l_1+1}^{n} (u_1 - l_1) \cdots \sum_{l_{\mathcal{D}}=0}^{n-1} \sum_{u_{\mathcal{D}}=l_{\mathcal{D}}+1}^{n} (u_{\mathcal{D}} - l_{\mathcal{D}}).$$

It is easy to check that

$$\sum_{l_d=0}^{n-1} \sum_{u_d=l_d+1}^{n} (u_d - l_d) = \frac{n \cdot (n+1) \cdot (n+2)}{6}$$

for any $d = 1, \ldots, \mathcal{D}$ (i.e., the sums is equal to the n^{th} tetrahedral number) and the proof is complete. ■

Proposition 3.6. *For an optimally balanced assignment $\mathcal{A}$ and uniformly distributed dataset, the expected value of the box measure can be explicitly computed by the formula:*

$$\mathbb{E}[\beta] = 1 - \left(\frac{n+2}{3n}\right)^{\mathcal{D}}.$$

Proof. By using Eq. (3.7) and Prop. 3.5,

$$\mathbb{E}[\beta] = 1 - \frac{1}{r \cdot \binom{n+1}{2}^{\mathcal{D}}} \mathbb{E}\left[\sum_I w(I)\right] =$$

$$= 1 - \frac{1}{n^{\mathcal{D}} \cdot \bar{B} \cdot \binom{n+1}{2}^{\mathcal{D}}} \cdot \bar{B} \cdot \left(\frac{n \cdot (n+1) \cdot (n+2)}{6}\right)^{\mathcal{D}} = 1 - \left(\frac{n+2}{3n}\right)^{\mathcal{D}}.$$

■

4. Conclusions

Our previous work presented two performance measures which can support the comparison of the efficiency of data placement methods. In this work, a new performance measure was introduced and we studied mathematical characteristics of this measure. We found a proof for the correlation between the optimization for load balancing and for box queries with sufficiently large $\mathcal{D}$.

In Sec. 3.3, the results showed that the box measure (or its expected value) is very close to 1 in several cases. It is interesting how close to 1 it can be. This issue needs to be addressed in the future. Finally, we presented a method for histogram bin shell traversal which is associated to the Gaussian integers.

References

[1] **Angulo, R.E., V. Springel, S.D.M. White, A. Jenkins, C.M. Baugh and C.S. Frenk,** Scaling relations for galaxy clusters in the Millennium-XXL simulation, *Monthly Notices of the Royal Astronomical Society*, **426(3)** (2012), 2046–2062.

[2] **Basilla, J.M.,** On the solution of $x^2 + dy^2 = m$, *Proceedings of the Japan Academy, Series A, Mathematical Sciences*, **80(5)** (2004), 40–41.

[3] **Berriman, G.B. and S.L. Groom,** How will astronomy archives survive the data tsunami?, *Communications of the ACM*, **54(12)** (2011), 52–56.

[4] **Carella, N.A.,** Lagrange's theorem on the minimal set of squares, `https://arxiv.org/abs/1108.6246v1`, (2011).

[5] **Conrad, K.,** The gaussian integers, `http://www.math.uconn.edu/~kconrad/blurbs/ugradnumthy/Zinotes.pdf`

[6] **Dobos, L., I. Csabai, J.M. Szalai-Gindl, T. Budavári and A.S. Szalay,** Point cloud databases, in: C.S. Jensen, H. Lu, T.B. Pedersen, C. Thomsen, K. Torp (Ed.) *Conference on Scientific and Statistical Database Management, SSDBM '14* (Aalborg, Denmark, June 30 - July 02, 2014), Proceedings of the 26th International Conference on Scientific and Statistical Database Management, ACM, New York, 2014, 33:1–33:4.

[7] **Grosswald, E.,** *Representations of Integers as Sums of Squares*, Springer-Verlag, New York, 1985.

[8] **Hurwitz, A.,** Über die Komposition der quadratischen Formen, *Mathematische Annalen*, **88(1)** (1922), 1–25.

[9] **Lee, J.-G. and M. Kang,** Geospatial big data: challenges and opportunities, *Big Data Research*, **2(2)** (2015), 74–81.

[10] **Rabin, M.O. and J.O. Shallit,** Randomized algorithms in number theory, *Communications on Pure and Applied Mathematics*, **39(1)** (1986), 239–256.

[11] **Schorn, P.,** An experimental comparison of algorithms for decomposing an integer into a sum of three squares, `ftp://ftp.inf.ethz.ch/pub/software/xyz/papers/SquareDecomposition.ps`

[12] **Szalai-Gindl, J.M., L. Dobos and I. Csabai,** Tiling strategies for distributed point cloud databases, in: A. Choudhary, K. Wu, F. Rusu, G. Trajcevski, A. Shoshani, B. Dong, B. Zhang (Ed.) *Conference on Scientific and Statistical Database Management, SSDBM '17* (Chicago, IL, USA, June 27–29, 2017), Proceedings of the 29th International Conference on Scientific and Statistical Database Management, ACM, New York, 2017, 32:1–32:6.

I. Csabai and L. Dobos
Department of Physics of Complex Systems
Eötvös Loránd University
Budapest
Hungary
csabai@complex.elte.hu
dobos@complex.elte.hu

A. Kiss and J. M. Szalai-Gindl
Department of Information Systems
Eötvös Loránd University
Budapest
Hungary
kiss@inf.elte.hu
szalaigindl@inf.elte.hu