

# Investigating Binary Partition Power in Metric Query

Richard Connor<sup>1</sup>, Alan Dearle<sup>1</sup> and Lucia Vadicamo<sup>2</sup>

<sup>1</sup>*School of Computer Science, University of St Andrews, St Andrews, KY16 9SS, Scotland*

<sup>2</sup>*Institute of Information Science and Technologies, CNR, Pisa, Italy*

## Abstract

It is generally understood that, as dimensionality increases, the minimum cost of metric query tends from  $O(\log n)$  to  $O(n)$  in both space and time, where  $n$  is the size of the data set. With low dimensionality, the former is easy to achieve; with very high dimensionality, the latter is inevitable.

We previously described BitPart as a novel mechanism suitable for performing exact metric search in “high(er)” dimensions. The essential tradeoff of BitPart is that its space cost is linear with respect to the size of the data, but the actual space required for each object may be small as  $\log_2 n$  bits, which allows even very large data sets to be queried using only main memory. Potentially the time cost still scales with  $O(\log n)$ . Together these attributes give exact search which outperforms indexing structures if dimensionality is within a certain range.

In this article, we reiterate the design of BitPart in this context. The novel contribution is an in-depth examination of what the notion of “high(er)” means in practical terms. To do this we introduce the notion of *exclusion power*, and show its application to some generated data sets across different dimensions.

## Keywords

similarity search, metric indexing, metric search, exclusion power

## 1. Introduction

Searching a database for objects that are most similar to a query object is a fundamental task in many database application domains, for example data mining, knowledge discovery, information extraction, and machine learning [1].

In [2] we described BitPart as a mechanism suitable for performing exact metric search in “high(er)” dimensions. While we did not define “high(er)”, the evidence in the paper shows that it gives its best relative performance for spaces with dimensionality of between 10 and 30. BitPart’s space cost is linear with respect to the size of the data, but the actual space required for each object may be small as  $\log_2 n$  bits, which allows representations of very large data sets to fit in main memory. Potentially the time cost still scales with  $O(\log n)$  time. Together these attributes give exact search which outperforms indexing structures as dimensionality increases within a certain range<sup>1</sup>.

---


SEBD 2022: The 30th Italian Symposium on Advanced Database Systems, June 19-22, 2022, Tirrenia (PI), Italy

✉ rchc@st-andrews.ac.uk (R. Connor); al@st-andrews.ac.uk (A. Dearle); lucia.vadicamo@isti.cnr.it (L. Vadicamo)

🆔 0000-0003-4734-8103 (R. Connor); 0000-0002-1157-2421 (A. Dearle); 0000-0001-7182-7038 (L. Vadicamo)



© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

<sup>1</sup>Note that due to the *curse of dimensionality* [3] exact metric search rarely outperforms a sequential scan when dimensionality exceeds 10 [4]. For high-dimensional data, approximate methods are usually employed, e.g [5, 6].

The BitPart mechanism uses a set of binary partitions defined over the finite data set, where each partition is represented by a bitmap. If the total set of bitmaps is perfectly balanced and perfectly orthogonal, then the probability of a selection of  $k$  bitmaps being able to exclude an arbitrary element of the data is  $1 - \frac{1}{2^k}$ , and the expected number of non-excluded values becomes very small if  $k \approx \log_2 n$ . While we can come close to achieving this for low-dimensional data, this becomes increasingly challenging as dimensionality increases.

In [2] two issues are noted as requiring further investigation. The first is how to select appropriate pivots in order to find orthogonal exclusion partitions. The second is the interaction between partition *balance* and the efficacy of exclusion; this is the subject of this article.

To do this we introduce the notion of *exclusion power*, and show its application to some generated data sets across different dimensions. Exclusion power provides the ability to: fundamentally understand the exclusions that are possible in different datasets, to understand when one approach will outperform another and, to tune the BitPart algorithm to optimally accommodate data sets of different dimensions.

The novel contribution of this article is the quantification of *exclusion power*, which can be measured for a given binary partition structure with respect to a *balancing factor*  $\tau$ . We observe that, in low dimensions, partitions are best balanced evenly, approximately splitting the search space into two equal parts. In higher dimensions however such balanced partitions perform very badly, and highly skewed partitions have much greater power. We are not aware of any previous work which attempts to quantify this effect, although it is well known within the “folklore” of metric search researchers.

## 2. Formal Context

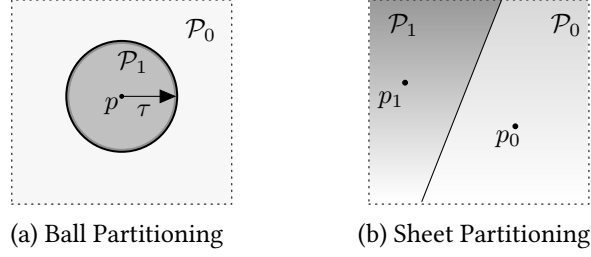
Formally we are interested in searching a (large) finite set of objects  $S$  which is a subset of an infinite set  $U$ , where  $(U, d)$  is a metric space [1]. The general requirement is to efficiently find members of  $S$  which are similar to an arbitrary member of  $U$  given as a query, where the distance function  $d$  gives the only way by which any two objects may be compared. The simplest type of similarity query is the *range search* query: for some threshold  $t$ , based on a query  $q \in U$ , the solution set is  $\mathcal{Q}(q, t) = \{s \in S \mid d(q, s) \leq t\}$ .

The essence of metric search is to spend time pre-processing the finite set  $S$  (i.e., indexing it) so that solutions to queries can be efficiently calculated. In all cases distances between members of  $S$  and selected reference objects (*pivot*) are calculated during pre-processing. At query time the relative distances between the query and the same pivot objects can be used to make deductions about which data values may, or may not, be candidate solutions to the query.

Many metric indexes employ binary partitions of the space  $U$ , i.e., partitions of the form  $\mathcal{P} = \{\mathcal{P}_0, \mathcal{P}_1\}$  where  $\mathcal{P}_0 \cup \mathcal{P}_1 = U$  and  $\mathcal{P}_0 \cap \mathcal{P}_1 = \emptyset$ . In the context of metric search, binary partitions are defined by partition rules that typically rely on computing the distance  $d$  of the data objects to a set of pivots. Notable examples include *ball* and *sheet partitioning* (see, e.g., Figure 1). A ball partition is defined by a pivot  $p \in U$  and a covering radius  $\tau \in \mathbb{R}^+$ , so that

$$\mathcal{P}_0 = \{u \in U \mid d(u, p) > \tau\} \quad \mathcal{P}_1 = \{u \in U \mid d(u, p) \leq \tau\}$$

A sheet partition is defined by two pivots  $p_1, p_0 \in U$  so that the boundary of the partition



**Figure 1:** Examples of a Ball Partitioning defined by a pivot  $p \in U$  and a covering radius  $\tau \in \mathbb{R}^+$ , and a Sheet Partitioning associated to two pivots  $p_1, p_0 \in U$ .

regions is the hyperplane equidistant from the two reference objects:

$$\mathcal{P}_0 = \{u \in U \mid d(u, p_1) > d(u, p_0)\} \quad \mathcal{P}_1 = \{u \in U \mid d(u, p_1) \leq d(u, p_0)\}$$

Informally we refer to  $\mathcal{P}_0$  as "outside" and  $\mathcal{P}_1$  as "inside".

Under certain circumstances it is possible to deduce that, for a given query, one of the subsets of the partition cannot (or conversely must) contain solutions to the query. Typically, the triangle inequality and the distances to the pivots are exploited to determine bounds on distances between data objects (*distance constraints*), which are used at query time for excluding certain regions from the searching process.

The exclusion of a region of a partition occurs when the query ball does not intersect the partition boundary. For a ball region, the condition for non-intersection of the query ball and the boundary is

$$|d(p, q) - \tau| > t \quad (1)$$

For a sheet region, the non-intersection condition is

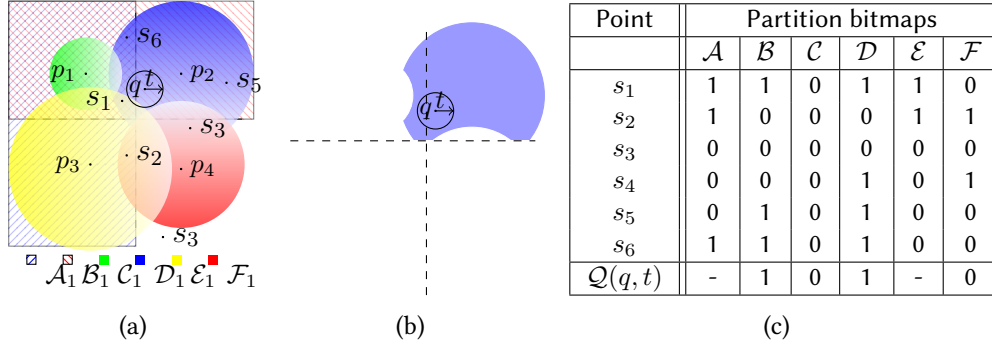
$$|d(p_1, q) - d(p_0, q)| > 2t \quad (2)$$

If the respective non-intersection condition occurs, the implication is that any possible solution to the query lies on one or other side of the partition boundary, and therefore either  $\mathcal{P}_0$  or  $\mathcal{P}_1$  may be excluded from the search for solutions.

### 3. The BitPart Mechanism

The core of the technique is to define a large set of binary partitions over the original space; the data is stored as a set of bitmaps according to containment with these regions. At query time, the query is assessed against this containment information, but without reference to the original data representation.

For each region, one of three possible conditions may be determined: (a) the solution set  $\mathcal{Q}(q, t)$  must be fully contained in the region, or (b) there is no intersection between the region and the solution set, or (c) neither of these is the case. In either case (a) or (b), the containment information stored may be useful with respect to solving the query, and is used as part of the query computation. In case (c) however the containment information is of no value, and is not



**Figure 2:** (a) A partitioning of a data space along with a query  $q$  with threshold  $t$ ; (b) the regions containing the solution set; (c) the data representation according to containment of regions.

accessed as a part of the computation. This approach maximises the effectiveness of memory used for calculation against the original data representation: the amount of memory required depends on the cardinality of the original data, but not on the size of its individual objects.

**An Example.** Figure 2 shows a simple example within the 2D plane, comprising four reference objects  $p_1$  to  $p_4$  and a set of six regions defined by them. The regions are respectively:  $\mathcal{A}_1$ , the area to the left of the line between  $p_1$  and  $p_2$ ;  $\mathcal{B}_1$ , the area above the line between  $p_1$  and  $p_3$ ; and the areas within the variously sized circles drawn around each  $p_1$  to  $p_4$ , labeled  $\mathcal{C}_1$ ,  $\mathcal{D}_1$ ,  $\mathcal{E}_1$  and  $\mathcal{F}_1$  respectively. Note that each regional boundary defines a binary partition of the total space of the form  $\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_0\}$ , where  $\mathcal{P}_0 = U \setminus \mathcal{P}_1$ . Thus in Figure 2, six binary partitions ( $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_0\}$ ,  $\mathcal{B} = \{\mathcal{B}_1, \mathcal{B}_0\}$ ,  $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_0\}$ , etc) are considered. For each partition  $\mathcal{P}$  we generate a bitmap of  $n$  bits, where  $n$  is the number of data points, which stores the logical containment information of each data object  $s_i$ , that is 1 if  $s_i \in \mathcal{P}_1$  and 0 if  $s_i \notin \mathcal{P}_1$  (Figure 2c). Thus a bit being set (to 1) corresponds to that object being in the region and a member of  $\mathcal{P}_1$ .

Figure 2a also shows a range query  $q$  drawn with a threshold  $t$ . It can be seen that all solutions to this query must lie within the area highlighted in Figure 2b. The circle around the query intersects with two regional boundaries ( $\mathcal{A}_1$  and  $\mathcal{E}_1$ ), and so no information is available with respect to these; however it is completely contained within two of the defined regions ( $\mathcal{B}_1$  and  $\mathcal{D}_1$ ), and fails to intersect with the final two ( $\mathcal{C}_1$  and  $\mathcal{F}_1$ ). Such containment and intersection is derivable only from the measurement of distances between the query and the reference objects, the definition of the regions, and the search radius. Here for example the possible solution area shown is determined using only the four distance calculations  $d(q, p_1) \dots d(q, p_4)$ . It can be seen that the only possible solutions to the query are those which match on all non-intersecting fields; in this case, the objects  $s_1$ ,  $s_5$  and  $s_6$  depicted in Figure 2. This is equivalent to the Conjunctive Normal Form (CNF) expression  $\mathcal{B} \wedge \neg \mathcal{C} \wedge \mathcal{D} \wedge \neg \mathcal{F}$ . This expression covers the set of all possible solutions to the query and hints at how the data can be stored using bit vectors.

### 3.1. Data structures

Before describing the query algorithm, we describe the data structures used and how they are initialised. A set of enumerated pivots  $p_1$  to  $p_m$  is selected from  $S$ . We define a set of binary partitions within  $U$ , each of the form  $\mathcal{P} = \{\mathcal{P}_0, \mathcal{P}_1\}$ , defined using the distance function  $d$ . There may be many more regions than reference objects; for example a set of  $m$  reference objects defines at least  $\binom{m}{2}$  sheet regions, plus  $m$  ball regions.

We now define the notion of an *exclusion zone* (EZ) as a containment map of  $S$  based on a given partition; this is the information we will use at query time to perform exclusions and derive a candidate set of solutions. We impose an ordering on  $S$ , then for each  $s_i$  map whether it is a member of the region  $\mathcal{P}_1$  or otherwise. This logical containment information is best stored in a bitmap of  $n$  bits, where  $n = |S|$ . One such exclusion zone is generated per partition.

### 3.2. Overview of the Query Algorithm

The query process comprises three distinct phases:

**Phase 1** in which the query is checked against the regional definitions, and the two types of regions with non-intersecting boundaries are identified. Initially, the distance from the query  $q$  to each pivot  $p_i$  is measured. For each partition, it can be established if the query ball intersects with the boundary of the partition. If there is no intersection, any solution to the query must lie on one or other side of the partition's boundary, so the exclusion zone is brought into the query calculation in one of two sets depending on whether the query solutions are fully contained within  $\mathcal{P}_1$  or  $\mathcal{P}_0$ . We name these sets of bitmaps  $B_1$  and  $B_0$ , which are carried forwards to the computation in Phase 2.

**Phase 2** in which the regions identified are used to conduct a series of logical operations, thus identifying a set of candidate solutions for the query.

The second phase comprises the manipulation of the bitmaps deriving from the first phase to identify a set of candidate solutions. This may be efficiently achieved by a series of bitwise operations over these bitmaps<sup>2</sup>. It can be seen now that any solution is guaranteed to be identifiable from the bitmap deriving from the bitwise calculation:

$$\left( \bigwedge_{b \in B_1} b \right) \wedge \left( \neg \left( \bigvee_{b \in B_0} b \right) \right) \quad (3)$$

**Phase 3** in which the candidate solutions are checked against the original data set to identify the exact solution to the query. This requires a sequential scan of the single bitmap resulting from Phase 2 and checking the data corresponding to any remaining set bits using the original space and distance metric in order to produce an exact solution to the query.

---

<sup>2</sup>On modern processors many of these bitwise operations can be performed in single instructions, each of which may be performed in parallel.

## 4. Partition analysis

To unify our treatment of different kinds of binary partitions with their associated distance constraints, we introduce the concept of a binary partition characterised by a single *partition function*  $f : U \rightarrow \mathbb{R}$  and a *balancing factor*  $\tau \in \mathbb{R}$  with the following properties:

**Property 1:**  $\mathcal{P}_0 = \{s \in U \mid f(s) > \tau\}$  and  $\mathcal{P}_1 = \{s \in U \mid f(s) \leq \tau\}$

**Property 2:**  $d(s, s') \geq |f(s) - f(s')|$  for all  $s, s' \in U$  (the *distance lower-bound* property)

Note that if  $f(s) = \tau$ , then  $s$  is on the partition boundary and by convention we include the partition boundary in  $\mathcal{P}_1$ . The function  $f$  is used both for determining the regions  $\mathcal{P}_0, \mathcal{P}_1$ , and for deriving the exclusion rules to be used at query time. By exploiting the distance lower-bound provided by the function  $f$ , we have that  $|f(q) - \tau| > t$  is a sufficient condition to exclude one of the two partitions from the search. Specifically,

- if  $f(q) \leq \tau - t$  then  $\mathcal{Q}(q, t) \subset \mathcal{P}_1$  and  $\mathcal{P}_0$  can be excluded
- if  $f(q) > \tau + t$  then  $\mathcal{Q}(q, t) \subset \mathcal{P}_0$  and  $\mathcal{P}_1$  can be excluded

Note that different functions  $f$  may be used to determine the same regions  $\mathcal{P}_0, \mathcal{P}_1$  but with different distance lower-bounds, and thus different exclusion rules. For example, let's consider the Euclidean space  $(\mathbb{R}^n, \ell_2)$ ,  $\tau = 0$ ,  $f_1(s) = \frac{d(s, p_1) - d(s, p_0)}{2}$  and  $f_2(s) = \frac{d(s, p_1)^2 - d(s, p_0)^2}{2d(p_1, p_0)}$  for two fixed pivot  $p_1$  and  $p_0$ . Both the functions  $f_1$  and  $f_2$  produce the same sheet partitioning, i.e.,  $\mathcal{P}_1 = \{s \in \mathbb{R}^n \mid d(s, p_1) \leq d(s, p_0)\}$  and  $\mathcal{P}_0 = \{s \in \mathbb{R}^n \mid d(s, p_1) > d(s, p_0)\}$ . However, in [7] we proved that the exclusion rule determined by  $f_1$  (*Hyperbolic exclusion*) is weaker than the one determined by  $f_2$  (*Hilbert Exclusion*), and the latter is valid only on the large class of Supermetric Spaces [8].

In the following we show how general ball partitioning and sheet partitioning can be described within this formalism.

**Ball Partitions** A ball partition can be characterised by the function  $f(s) = d(s, p)$  and the constant  $\tau$ . In this case  $\mathcal{P}_1 = \{s \in U \mid f(s) = d(s, p) \leq \tau\}$ , and  $\mathcal{P}_0 = \{s \in U \mid f(s) = d(s, p) > \tau\}$ , so Property 1 is clear. Property 2 follows from the triangle inequality:

$$d(s, p) \leq d(s, s') + d(s', p), \quad \text{and} \quad d(s', p) \leq d(s', s) + d(s, p)$$

which give that  $d(s, s') \geq |d(s, p) - d(s', p)| = |f(s) - f(s')|$  for any  $s, s' \in U$ .

The balance of the partition is affected by selecting different values for  $\tau$  which we refer to as different balances.

**Sheet Partitions** Sheet partitions are defined with respect to two reference points  $p_1$  and  $p_0$ . Usually, a simple binary partition is defined according to whichever of  $p_1$  and  $p_0$  is the nearer value with respect to the metric  $d$ . However it has been noted [9, 10] that such partitions may also be subject to a balancing factor  $\alpha$ , so that

$$\mathcal{P}_1 = \{s \in U \mid d(s, p_1) \leq d(s, p_0) + \alpha\}, \mathcal{P}_0 = \{s \in U \mid d(s, p_1) > d(s, p_0) + \alpha\}$$

In this case, we can use  $f(s) = (d(s, p_1) - d(s, p_0))/2$  which gives property 1 above for the constant  $\tau = \alpha/2$ . Moreover from the triangle inequality of the space we have that

$$d(s, p_1) \leq d(s, s') + d(s', p_1) \quad \text{and} \quad d(s', p_0) \leq d(s, s') + d(s, p_0)$$

for any  $s, s' \in U$ , which gives  $d(s, s') \geq \frac{d(s, p_1) - d(s, p_0) - d(s', p_1) + d(s', p_0)}{2} = f(s) - f(s')$ . By symmetry, we also have  $d(s, s') > f(s') - f(s)$ , therefore  $d(s, s') > |f(s) - f(s')|$ .

Like ball partitions, the balance of the partition is affected by selecting different values for  $\tau$ .

#### 4.1. Exclusion power

The *exclusion power* of a partition gives a quantification of the effectiveness of that partition with respect to a given metric search task. It is clear that this is likely to be quite different depending on the partition type, the selection of reference objects, and the chosen value  $\tau$ .

The meaning of exclusion power is based on the probability, for arbitrarily selected objects  $q$  and  $s$ , and a distance  $t$ , of being able to demonstrate that  $d(q, s) > t$ . As we will show, for low dimensional spaces the maximum exclusion power of any partition structure is likely to coincide with a value of  $\tau$  which causes the partition to bisect the finite search space  $S$ , i.e.  $|\mathcal{P}_0| \approx |\mathcal{P}_1|$ . However as we will see, in higher dimensional spaces this may be a very bad choice.

For a range query  $\mathcal{Q}(q, t)$  we define the exclusion power of the partition  $\mathcal{P} = \{\mathcal{P}_0, \mathcal{P}_1\}$  as the probability of excluding one generic element  $s$  on the basis only of the data partition to which it belongs:

$$P(s \in \mathcal{P}_0) \cdot P(\mathcal{Q}(q, t) \cap \mathcal{P}_0 = \emptyset) + P(s \in \mathcal{P}_1) \cdot P(\mathcal{Q}(q, t) \cap \mathcal{P}_1 = \emptyset) \quad (4)$$

$$= P(s \in \mathcal{P}_0) \cdot P(\mathcal{Q}(q, t) \subset \mathcal{P}_1) + P(s \in \mathcal{P}_1) \cdot P(\mathcal{Q}(q, t) \subset \mathcal{P}_0) \quad (5)$$

The exact calculation of the probabilities  $P(\mathcal{Q}(q, t) \subset \mathcal{P}_i)$ , for  $i = 0$  or  $1$ , is not straightforward for a general binary partition in a metric space, however, if we know that the partition can be characterised by a function  $f$  and a balancing factor  $\tau$ , we can estimate a lower-bound of these probabilities<sup>3</sup> and thus obtain an approximation of the exclusion power as

$$P(f(s) > \tau) \cdot P(f(q) \leq \tau - t) + P(f(s) \leq \tau) \cdot P(f(q) > \tau + t) \quad (6)$$

Let  $\text{CDF}(x)$  be the cumulative distribution function of  $f(s)$  for  $s \in S$  then the exclusion power can be expressed as

$$g(\tau, t) = (1 - \text{CDF}(\tau)) \cdot \text{CDF}(\tau - t) + \text{CDF}(\tau) \cdot (1 - \text{CDF}(\tau + t)) \quad (7)$$

where  $\tau$  is the partition balancing factor and  $t$  in the query distance threshold

Estimates of exclusion power can be made with respect to a representative sample of the data set, and indeed of the query set if that is different<sup>4</sup>. A large enough sample will give a good estimate of the probability of an individual datum falling within each of the subsets.

<sup>3</sup>In general,  $f(q) \leq \tau - t \Rightarrow \mathcal{Q}(q, t) \subset \mathcal{P}_1$ , therefore  $P(f(q) \leq \tau - t) \leq P(\mathcal{Q}(q, t) \subset \mathcal{P}_1)$ . Similarly,  $f(q) > \tau + t \Rightarrow \mathcal{Q}(q, t) \subset \mathcal{P}_0$ , and thus  $P(f(q) > \tau + t) \leq P(\mathcal{Q}(q, t) \subset \mathcal{P}_0)$ . The equality holds for some cases, e.g., the ball partitioning, where it can be proved that  $f(q) \leq \tau - t \Leftrightarrow \mathcal{Q}(q, t) \subset \mathcal{P}_1$  and  $f(q) > \tau + t \Leftrightarrow \mathcal{Q}(q, t) \subset \mathcal{P}_0$ .

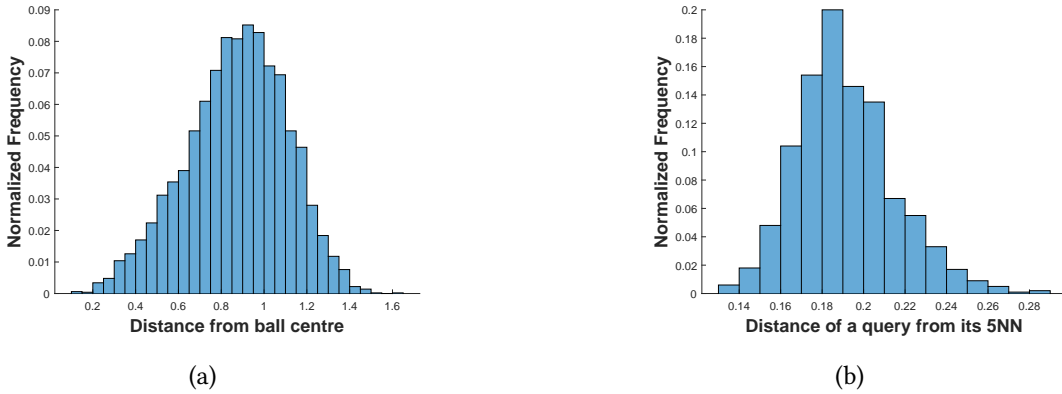
<sup>4</sup>We assume from here on that the distribution of data and query is the same

## 5. Power graphs

To understand the effect of different values of  $\tau$  an exclusion power graph may be constructed which is plotted across the range of  $\tau$  for a fixed value of  $t$ . This allows the optimum value of  $\tau$  to be deduced for a range query with threshold  $t$ .

We illustrate the construction of power graphs for 5 dimensional data in Section 5.1 and for 20 dimensional data in 5.2.

### 5.1. Example: single-pivot ball partitioning over 5 dimensional data



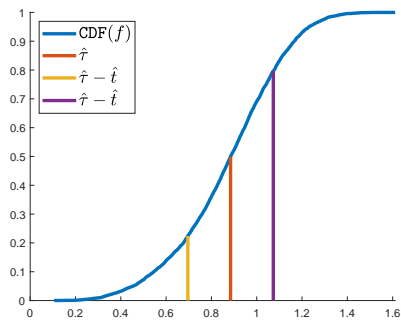
**Figure 3:** Typical inter-point and query threshold distances from 5-dimensional Euclidean data

Figure 3 shows histograms of two sets of sampled distances from a randomly generated five-dimensional Euclidean space. The left-hand side of the figure shows the distribution of 5,000 *witness* points in terms of distance from a randomly selected pivot point,  $p$ . The right-hand side shows the distribution of the 5NN (fifth nearest-neighbour) distances for a set of 1,000 queries; this distance representing a query threshold necessary to return a small proportion of the data, and thus useful values for  $t$ .

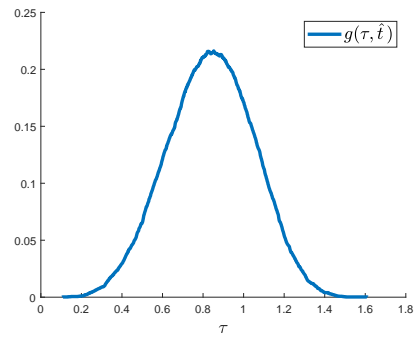
Based on this information, we will pick example values of  $\hat{\tau} = 0.89$  and  $\hat{t} = 0.19$ , being the median values in the distributions of  $\tau$  and  $t$  values, respectively. We can now consider the exclusion power of these values applied to the partition thus formed, centered around  $p$ .

The left-hand side of Figure 4 shows the Cumulative Probability Density function (CDF) with the X-axis values of  $\hat{\tau}$ ,  $\hat{\tau} - \hat{t}$ , and  $\hat{\tau} + \hat{t}$  highlighted. Given that  $\hat{\tau}$  has been selected as the median distance from  $p$ , the value of the CDF at  $\hat{\tau}$  is 0.5. Therefore if an exclusion occurs, half of the data set will be excluded and its exclusion power is:  $\text{CDF}(\hat{\tau} - \hat{t}) \cdot 0.5 + (1 - \text{CDF}(\hat{\tau} + \hat{t})) \cdot 0.5$ . The more general form of this equation (i.e., Equation 7) hints how the exclusion power can be displayed for a single partition, as shown on the right-hand side of Figure 4. Here, for a fixed value of  $t$  (again 0.19), the power of the partition is calculated over the whole range of distances that have been measured between the witness points and the point  $p$ . As can be seen, the maximum power is achieved when the partition is split at the median distance, i.e.  $\hat{\tau} = 0.89$  in this example. With this partition, and this value of  $\tau$ , the probability of excluding an arbitrary datum for an arbitrary query is around 0.21.





(a) CDF of  $f(s) = d(s, p)$  varying  $s$



(b) Exclusion power graph over  $\tau$  (fixing  $t = \hat{t}$ ). The exclusion power for  $\tau = \hat{\tau}$  is 0.23.

**Figure 4:** CDF and Power Graph for 5-dimensional data

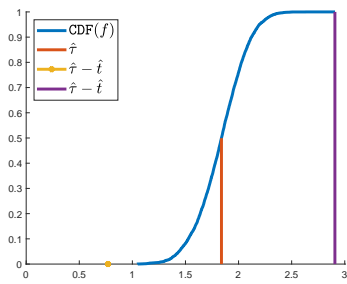
## 5.2. Example: single-pivot ball partitioning over 20 dimensional data



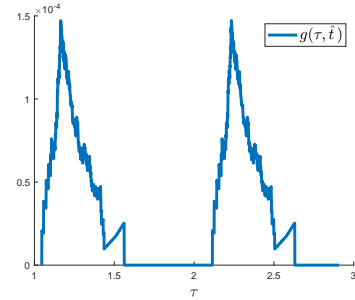
**Figure 5:** Typical inter-point and query threshold distances from 20-dimensional data

The left-hand side of Figure 5 shows a histogram of 5,000 distances measured with respect to a randomly selected element  $p$  of a 20-dimensional uniformly generated Euclidean space. The right-hand side shows a histogram (for 1,000 queries selected from the same set) of the distances to the 5th-nearest neighbour with respect to this same set of 5,000 values. Figure 6 shows the same information in the form of cumulative probability density functions. The left-hand side of Figure 6 shows the CDF function superimposed with a line corresponding to the medium value for  $\tau$ . The value of  $t$  for this data is 1.07.

Finally, although we have no space here to provide in-depth analysis, Figure 7 shows exclusion power graphs using sheet partitions over both 5D and 20D spaces. It can be seen that the graph patterns are quite similar.

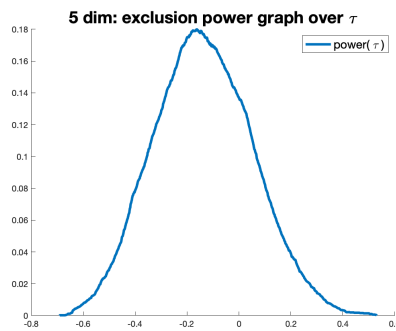


(a) CDF of  $f(s) = d(s, p)$  varying  $s$

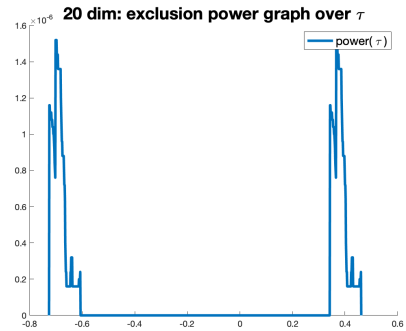


(b) Exclusion power graph over  $\tau$  (fixing  $t = \hat{t}$ ). The exclusion power for  $\tau = \hat{\tau}$  is 0

**Figure 6:** CDF and Power Graph for 20-dimensional data



(a) A sheet partition in 5D space



(b) A sheet partition of 20 dimensions

**Figure 7:** Exclusion Power graphs for sheet partitions in 5 and 20 dimensional Euclidean space

### 5.3. Use of Exclusion Power analysis

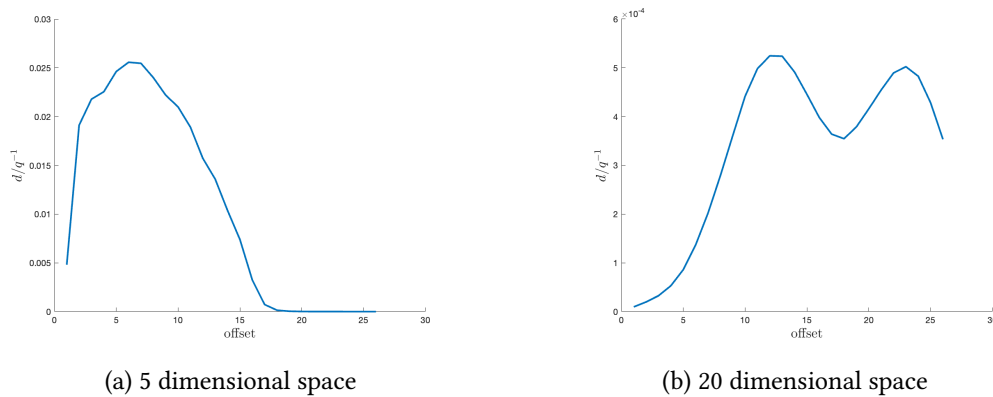
Figures 4 and 6 demonstrate the exclusion power in 5 and 20 Dimensions respectively. In the 5 dimensional case we can observe that setting a  $\hat{\tau}$  value of 0.89 (close to the mean of the distribution of measured distances) will work extremely well. With such a value the exclusion power is maximised. By contrast, choosing the mean in the case of 20D data will work very badly. As can be seen in Figure 6 such a value is unlikely to result in any successful exclusions. These diagrams explain behaviour observed by many researchers into metric search that choosing unbalanced indexing structures often works better for higher dimensional data.

The establishment of a unified partition interface using  $f$  and  $\tau$  allows similar graphs to be produced for sheet partitions (e.g., Figure 7). The mechanistic value of these graphs is that the value or values of  $\tau$  corresponding to the maximum power can be automatically extracted on a per-partition basis, and used in an implementation of BitPart. In the case of low dimensional data a single offset can be used for each ball where as for high(er) dimensional data two offsets can be used. Furthermore, as described earlier a particular implementation can use as few or as many exclusion zones that as necessary, as may be predicted by the power value at the optimum

value.

In Section 6 these offsets are applied to a BitPart implementation to demonstrate their effect.

## 6. Experimental Validation using BitPart



**Figure 8:** Empirical Inverse distances per query for 5D and 20D data for different values of  $\tau$ . These charts plot both ball and sheet partitions, which have different ranges of  $\tau$ ; we have therefore normalised these within the *offset* axis. The Y axis shows the inverse of the number of residual distance computations, i.e. those not excluded by use of the BitPart mechanism (higher is better).

In order to validate the above results we ran experiments against a BitPart implementation<sup>5</sup> using uniformly generated data in 5 and 20 dimensions with Euclidean distance. Each contained one million data points, and 1000 queries were executed. In all experiments fixed threshold distances  $t$  were used and set to values which correspond to one-millionth of the data volume. In order to measure exclusion power, the 1000 queries were run 25 times for differing values of  $\tau$  for both sheet and ball partitions. The 5D data uses 20 pivot points and the 20D data 100.

In [2] we described how a selection of  $m$  reference points allows for  $\binom{m}{2}$  sheet and  $m$  ball partitions to be produced. After the above analysis, we used this number for the 5D space, but in the 20D space we generated twice that number, corresponding to the twin peaks in the power graphs. Note that this does not require any extra distance calculations at query time, as the number of reference points is not increased. Furthermore there is not necessarily any increase in memory usage, as partitions are typically stored in secondary memory and only fetched when the query ball does not intersect with the partition, which requires only the evaluation of the  $f$  function to determine. For each experiment a suitable range is calculated for  $\tau$  for both balls and sheets, ensuring that for all partitions both  $\mathcal{P}_0$  and  $\mathcal{P}_1$  are non-empty, and the whole calculation was executed with different values of  $\tau$  across this range.

Results are shown in Figure 8. In order to visualize these results, the number of residual (i.e. Phase 3) distance calculations per query was measured, this number giving a good proxy for the value of the exclusion mechanism. The plots show the inverse of these counts. Peaks similar to those shown in the power graphs in Figures 4 and 6 can be seen in these experimental results.

<sup>5</sup>The experimental code may be found in <https://bitbucket.org/richardconnor/metricbitblaster.git>

## 7. Conclusions

In this paper, we have reiterated the design of the BitPart indexing and query mechanisms. The novel contribution of this paper is an exposition of how dimensionality effects the efficacy of this and other metric search mechanisms. We have demonstrated how the distribution of distances changes with the dimensionality of the data and how in turn this affects the ability to perform effective exclusions and thus efficient queries. In particular we have explored the notion of *balance* and how that affects exclusion. To do this we have introduced the notion of *exclusion power*, and shown its application to some Euclidean data sets in different dimensions.

This analysis gives fundamental insights into what data can be queried effectively and what cannot. Furthermore it demonstrates how index and search algorithms can be tuned to compensate for data sets of arbitrary dimensions.

We have demonstrated that the phenomena explored in mathematical terms are manifested in synthetically generated data sets and queries over them using an implementation of BitPart. We believe that the results in this paper may be applied to other metric search and indexing mechanisms. Our future plans include investigating these phenomena in real life data sets and exploring the outstanding issue of partition orthogonality.

## References

- [1] P. Zezula, G. Amato, V. Dohnal, M. Batko, Similarity search: the metric space approach, volume 32, Springer Science & Business Media, 2006.
- [2] A. Dearle, R. Connor, Bitpart: Exact metric search in high(er) dimensions, Information Systems 95 (2021) 101493.
- [3] V. Pestov, Indexability, concentration, and vc theory, Journal of Discrete Algorithms 13 (2012) 2–18.
- [4] R. Weber, H.-J. Schek, S. Blott, A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces, in: Proc. of 24th VLDB, volume 98, Morgan Kaufmann, 1998, pp. 194–205.
- [5] G. Amato, C. Gennaro, P. Savino, MI-File: Using inverted files for scalable approximate similarity search, Multimedia Tools and Applications (2014) 1333–1362.
- [6] D. Novak, M. Batko, P. Zezula, Metric index: An efficient and scalable solution for precise and approximate similarity search, Information Systems 36 (2011) 721–733.
- [7] R. Connor, F. A. Cardillo, L. Vadicamo, F. Rabitti, Hilbert exclusion: improved metric search through finite isometric embeddings, ACM Transactions on Information Systems (TOIS) 35 (2016) 1–27.
- [8] R. Connor, L. Vadicamo, F. A. Cardillo, F. Rabitti, Supermetric search, Information Systems 80 (2019) 108–123.
- [9] R. Connor, Reference point hyperplane trees, in: International Conference on Similarity Search and Applications, Springer, 2016, pp. 65–78.
- [10] J. Lokoč, T. Skopal, On applications of parameterized hyperplane partitioning, in: Proceedings of the Third International Conference on Similarity Search and Applications, 2010, pp. 131–132.