Journal of Discrete Algorithms 7 (2009) 49–61

Contents lists available at ScienceDirect

Journal of Discrete Algorithms

www.elsevier.com/locate/jda

Counting distance permutations

Matthew Skala *

David R. Cheriton School of Computer Science University of Waterloo, Waterloo, Ontario, Canada N2L 3G1

ARTICLE INFO

Article history:
Available online 26 September 2008

Keywords:
Metric space
Nearest neighbour
Voronoi diagram
Distance permutation

ABSTRACT

Distance permutation indexes support fast proximity searching in high-dimensional metric spaces. Given some fixed reference sites, for each point in a database the index stores a permutation naming the closest site, the second-closest, and so on. We examine how many distinct permutations can occur as a function of the number of sites and the size of the space. We give theoretical results for tree metrics and vector spaces with L_1, L_2, and L_∞ metrics, improving on the previous best known storage space in the vector case. We also give experimental results and commentary on the number of distance permutations that actually occur in a variety of vector, string, and document databases.

1. Introduction

Let $\langle S, d\rangle$ be a metric space with points S and distance function d. Given k points $x_1, x_2, \ldots, x_k$ in S, called the sites, the distance permutation of a point y in S, denoted by Π_y, is the unique permutation on $\{1, 2, \ldots, k\}$ such that if $i < j$ then $d(x_{\Pi_y(i)}, y) < d(x_{\Pi_y(j)}, y)$ or $d(x_{\Pi_y(i)}, y) = d(x_{\Pi_y(j)}, y)$ and $\Pi_y(i) < \Pi_y(j)$. In other words, Π_y is the permutation that sorts the site indices into order of increasing distance from y, using order of increasing index to break ties. This definition was first introduced by Chávez, Figueroa, and Navarro [7]. In this work we consider the number of distinct distance permutations that occur in a space, that is $|\{\Pi_y \mid y \in S\}|$, and the maximum value of this count over all choices of k sites.

The question of counting distance permutations arises from attempts to improve index data structures for proximity searching. Many kinds of data, including images, text documents, genetic sequences, and audio and video clips, are native to high-dimensional metric spaces, in which it is expensive to compute distance. For instance, the SIFT local descriptor technique described by Lowe, although successful at recognizing images containing the same object, requires processing each image into a set of potentially hundreds of keypoints and then computing distances and vector transforms on sets of keypoints [16]. The word space model, used for studying semantic relations in text, converts words into context vectors with thousands or millions of dimensions [24].

Proximity search data structures attempt to organise the points in a database to answer distance-based queries efficiently. In a k-nearest neighbour (kNN) query, the task is to find the k points in the database nearest to a query point. In a range query, the input is a sphere and the task is to return all database points inside the sphere. Approximate variants of these kinds of problems also exist. It is normally assumed that evaluating the metric is an expensive task, so data structures and algorithms are designed to minimise the number of evaluations of the metric even if that comes at significant cost elsewhere.

The naive algorithm for proximity search measures the distance from the query point to each object in the database in turn, requiring as many distance measurements as there are objects in the database. The challenge for a data structure is to answer the query with fewer distance measurements. If we can find an excuse to skip over a subset of points in

* Tel.: +1 519 888 4567 35351; fax: +1 519 885 1208.
E-mail address: mskala@ansuz.sooke.bc.ca.

1570-8667/$ – see front matter
doi:10.1016/j.jda.2008.09.011

the database without computing their distances explicitly, that will speed up the search. Many existing data structures for proximity search, such as VP-trees and GH-trees, work that way. In these structures, the points are organised into trees and the search algorithm attempts to exclude subtrees from examination by applying the triangle inequality [29,31].

Another approach stores precomputed data for individual points, so that even though the points are considered one at a time, they can sometimes be excluded without actually computing the distance. AESA is the prototype for this kind of technique. It stores the complete quadratic-sized matrix of pairwise distances among database points [30]. But storing index data quadratic in the size of the database only seems appealing because it exploits our definition of cost, which considers only search time: AESA pays a high cost in precomputation and storage instead. For this reason, pure AESA is seldom used in practical applications. A practical data structure must be much smaller.

Micó, Oncina, and Vidal improve on AESA by storing only part of the distance matrix: distances from each database point to k chosen points instead of all the n points in the database [20]. The resulting technique is called LAESA. The space requirement becomes $\Theta(kn)$ instead of $\Theta(n^2)$; and with a suitably chosen k, which can be significantly less than n, the resulting search algorithm is almost as efficient for searching as AESA.

Chávez, Figueroa, and Navarro suggest a further improvement [7]. Instead of storing the actual distances from each database point to the k chosen points, which we call the "sites" for consistency with the Voronoi diagram literature, they store only permutations of the sites: which site is closest to each database point, which one is second-closest, and so on. We call these objects distance permutations to emphasise their connection with existing work on permutation metrics, combinatorics of permutations, and so on. Other authors have referred to them as proximity preserving orders.

The experimental results of Chávez, Figueroa, and Navarro show that distance permutations provide enough information to do an efficient search, comparable to LAESA, while consuming much less storage space. They claim a reduction in storage space requirement from $O(nk \log n)$ bits for LAESA, to $O(nk \log k)$ [7]. The same authors with Paredes extend the concept further to create an algorithm called improved AESA (iAESA), in which distance permutations are also used to select pivot elements, providing a further improvement in search speed over AESA [11]. We focus on the storage space improvement, which is unique to the distance permutation representation; the enhanced pivot selection of iAESA seems applicable even to the older LAESA data structure by computing the distance permutations on demand.

We might ask, out of the $k!$ unrestricted permutations of k sites, how many can actually occur. For general metric spaces, the answer is all of them; for any k there always exists a metric space with a choice of k sites such that every permutation π of the sites has some point with π as its distance permutation. Any L_p space with $k-1$ dimensions suffices by Theorem 6 below.

However, many practical spaces have structural limitations (for instance, small dimensionality) under which the set of all permutations that can be distance permutations is much smaller than $k!$. Then a distance permutation can be stored in fewer bits than an unrestricted permutation, and the index can be made even smaller without changing the search performance. In particular, in the d-dimensional Euclidean case the storage space requirement is reduced to $\Theta(nd \log k)$, an improvement on the previous best known theoretical result. Smaller storage space is valuable in itself, but it also points to the limitations of distance permutation-based algorithms like iAESA [11]. Because only a few distance permutations are possible, that limits how much benefit in reduced search time can ever come from storing and using distance permutations.

2. Distance permutations as Voronoi cells

The cells of Voronoi diagrams correspond to classes of distance permutations. For instance, in the conventional nearest-neighbour Voronoi diagram of Fig. 1, the cell at left contains all the points closer to A than to B, C, or D. Those are exactly the points whose distance permutation begins with A. Many generalisations of Voronoi diagrams have been studied,

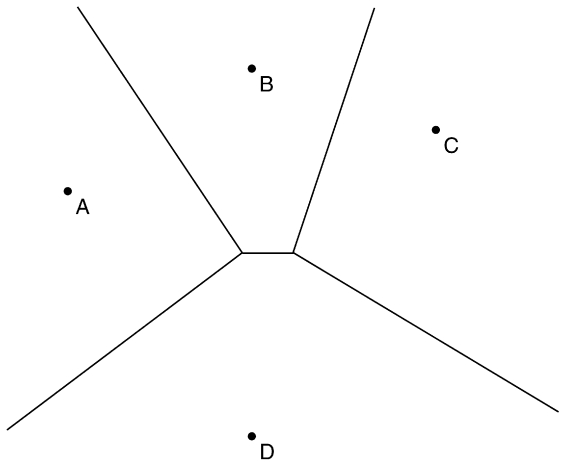

Fig. 1. Euclidean Voronoi diagram.

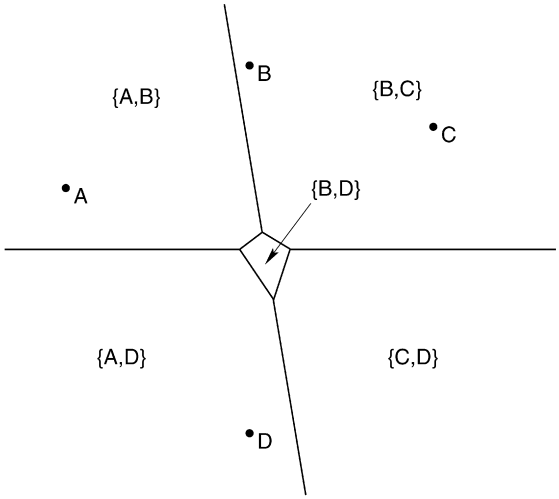

Fig. 2. Second-order Euclidean Voronoi diagram.

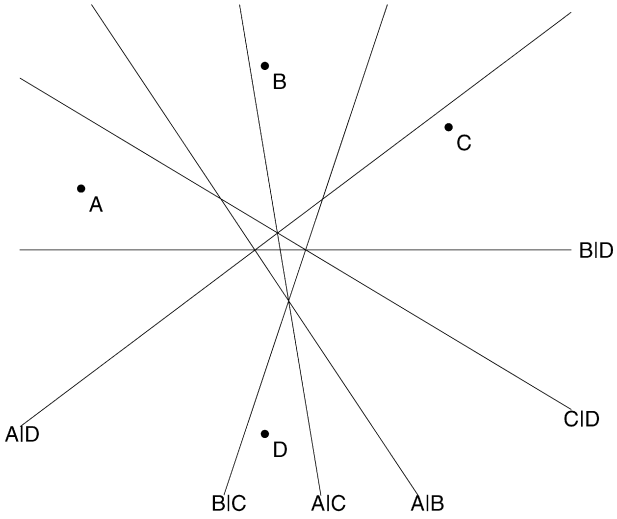

Fig. 3. Bisectors of four points in the Euclidean plane.

including *higher-order* diagrams in which the cells correspond to the set of k nearest neighbours instead of just the one very nearest neighbour [3]. An example for $k = 2$ is shown in Fig. 2. Here the small cell in the middle corresponds to distance permutations beginning with B and D, in either order.

If we consider the entire distance permutation, and consider order to be significant, we can divide the space into a distinct cell for each permutation and get a diagram like that in Fig. 3. All the cell boundaries of the previous two diagrams are included in this one, because the division according to distance permutation is a refinement of the division according to closest site, or closest k sites. Also, the boundaries in Fig. 3 consist exactly of the six (that is, $\binom{4}{2}$) lines that bisect pairs of sites. For each pair of sites, a point is closer to one or the other depending on whether it falls on one side or the other of the corresponding line; its position relative to all six lines defines its distance permutation. Because bisectors are useful in other spaces too, we give a general definition and notation for them:

Definition 1. The *bisector* of two points x and y, denoted by $x|y$, is the set of all points z such that $d(x, z) = d(y, z)$.

An example system of bisectors in a non-Euclidean metric is shown in Fig. 4. Here we show the six bisectors of four points in the plane using the L_1 (Manhattan) metric. Our question of how many distance permutations occur in a space can be interpreted as asking how many cells occur in this type of generalised Voronoi diagram.

If points can be on either side of each of six bisectors in Fig. 3, that suggests there should be $2^6 = 64$ cells, evidently impossible when there are only $4! = 24$ permutations of the four sites; and in fact, the diagram only contains 18 cells, not even one for each permutation. The fact that these are bisectors in Euclidean space and not arbitrary subsets of the

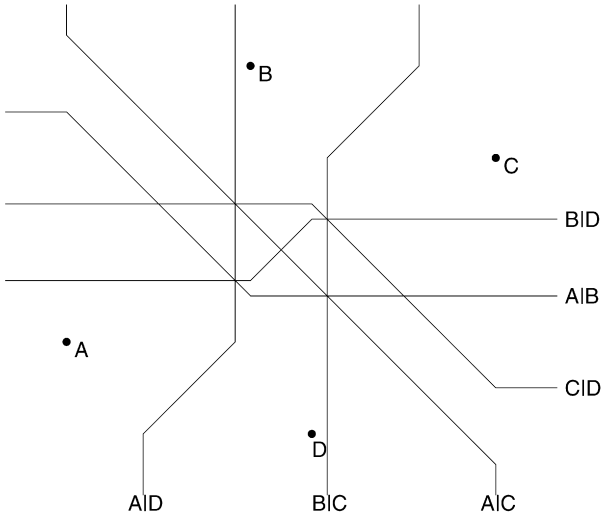

Fig. 4. Bisectors of four points in the L_1 plane.

plane limits the number of cells. Note that the system of bisectors in Fig. 4, with the L_1 metric, also produces 18 cells corresponding to 18 distance permutations, but they are not the same 18 distance permutations. Some permutations exist in each diagram that are not in the other.

Arrangements of hyperplanes, which include bisector systems in Euclidean space, create combinatorial objects called *oriented matroids*, and those are well-studied [5]. Unfortunately, most of the relevant results are inapplicable to bisectors in more general spaces. Many authors including Grünbaum [13] and Mandel [18] have applied oriented matroids to arrangements of pseudolines and pseudospheres (respectively), which describe intersections of generalised hyperplanes that are not necessarily flat. Arrangements of pseudolines as currently defined retain the restriction that each pair of pseudolines must intersect in exactly one point, using the projective plane if necessary to force parallel lines to intersect; and arrangements of pseudospheres have a similar, higher-dimensional requirement for well-behaved intersections. The bisector system shown in Fig. 4 does not have that property, and the associated sign vectors do not form an oriented matroid. Santos successfully generates a Delaunay oriented matroid from a point arrangement in non-Euclidean space by considering the triangulation of the points instead of their bisectors, but his main result is specific to two dimensions, and the connection to our question about bisectors is not clear [25].

Icking and others investigate the behaviour of bisectors with convex distance functions in two and three dimensions, and show a number of surprising results, including that three spheres in general position in 3-dimensional L_4 space can intersect at four distinct points [14], and that the combinatorial structure around the one-dimensional bisector of three points can be different for different connected components of the bisector [15]. Note that there being more than one connected component in a bisector in the first place is a deviation from the intuitive behaviour of Euclidean bisectors. They survey other problematic results on non-Euclidean bisectors and comment on "the surprising, really abnormal, structure of the bisectors which behave totally different[ly] from what is known for the Euclidean distance" [15].

3. Tree metrics

First we consider distance permutations in tree metric spaces. These spaces have a simple definition and notable applications in approximation of other metrics [4].

Definition 2. A *tree metric space* is a set S and distance function d such that there is a tree T with S as its vertex set, and for any $x, y \in S$, d is the number of edges in the unique path from x to y in T. Then d is called a *tree metric*. If T is instead a weighted tree, with a positive real weight associated with each edge, and $d(x, y)$ is the sum of the edge weights on the path from x to y, then d is a *weighted tree metric*. By setting all weights equal to 1, every tree metric is a weighted tree metric.

Terminology used to describe tree metrics varies, and many authors assume the definition without stating it precisely [2,17]. There are also other definitions in use, including those that assume a finite number of points [1], and those that define tree metrics as all metrics satisfying the "four-point condition" that for every set of four distinct points $\{x, y, z, t\}$ we have $d(x, y) + d(z, t) \leqslant \max\{d(x, z) + d(y, t), d(x, t) + d(y, z)\}$. That condition permits the points to be a proper subset of the vertices of the tree [19]. Topological studies of tree metrics sometimes turn the edges into homomorphic images of real intervals and allow points anywhere along the edges, which creates a fundamentally different kind of space [10]. We reserve

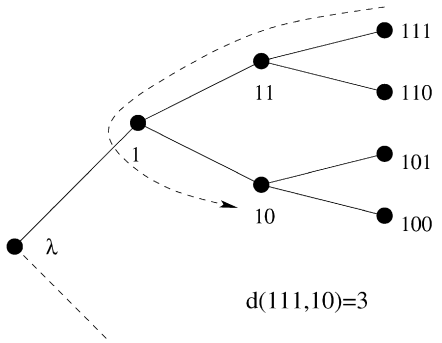

Fig. 5. The prefix metric is a simple tree metric.

the term tree metric space for the spaces satisfying Definition 2, following Lynn, Prabhakaran, and Sahai, whose work on obfuscated neighbourhoods (robust hashes) does not define tree metrics rigorously but assumes the ability to traverse a tree metric one edge at a time finding a point at each step [17]. As Buneman shows, any finite metric space satisfying the four-point condition must also be a subset of a tree metric space satisfying Definition 2 [6].

The prefix metric gives an especially convenient tree metric space; it names points with strings, and the distance is easy to calculate from the strings. Here is the formal definition:

Definition 3. The *prefix distance* between two strings x and y is the minimal number of edits to transform one string into the other, where an edit consists of adding or removing a letter at the right-hand end of the string.

The distance between two strings in the prefix metric is the sum of their lengths, minus twice the length of their longest common prefix. Fig. 5 shows an example. It can be thought of as measuring the distance between two items organised in an hierarchical structure labelled with strings, such as books in a library; longer common prefix of LC or Dewey decimal call numbers implies more closely related content.

Theorem 4. *For k sites in a space with a (possibly weighted) tree metric, there can be at most $\binom{k}{2}+1$ distinct distance permutations.*

Proof. Let d be the tree metric. For any three vertices x, y, and z with $x \neq y$, consider whether $d(x,z) \leqslant d(y,z)$. There is exactly one edge, and it happens to be on the path between x and y, where the statement is true at one endpoint and not the other. Removing that edge splits the tree into two connected components, one containing all vertices z where the statement is true and one containing all vertices where it is false. Repeat that procedure setting x and y to every pair chosen from the k sites. The resulting components correspond to the distinct distance permutations that can occur. There are at most $\binom{k}{2}+1$ of them. □

Furthermore, the bound of Theorem 4 is easily achievable in spaces like that of the prefix metric, where long paths are abundant.

Corollary 5. *The bound of $\binom{k}{2}+1$ distinct distance permutations is achievable in a tree metric space that contains a path of 2^{k-1} edges with the same weight.*

Proof. Label the vertices along the path sequentially from one end with the integers 0 to 2^{k-1}. Let the sites, in order, be the vertices labelled 0 and $2, 4, 8, \ldots, 2^{k-1}$. Note that there are $2^{k-1}+1$ vertices, all of which have labels, but we have chosen only k of those to be sites. Now the midpoint of the vertices 0 and 2^i for any $i \geqslant 1$ will fall on the vertex labelled 2^{i-1}; and the midpoint of the vertices labelled 2^i and 2^j will fall on the vertex labelled $2^{i-1}+2^{j-1}$. All those $\binom{k}{2}$ midpoint vertices are distinct (easily seen by examining the binary representations of their indices), and the edges from them to their higher-numbered neighbours are the distinct splitting edges of Theorem 4. Removing those edges separates the tree into $\binom{k}{2}+1$ connected components corresponding to the $\binom{k}{2}+1$ distinct distance permutations. Note that the midpoint vertices follow their lower-numbered neighbours in the division because of the tiebreaking rule, which considers lower-indexed sites, which are the lower-labelled sites by our choice, to be closer in case of ties. □

The proof is based on the fact that every edge in a tree is a cut-edge. When we split up the tree into distance permutations by cutting on all the bisectors, the number of components increases by at most one for each bisector. It is possible to design a tree metric with extremely uneven edge weights, or no sufficiently long paths, so that the bound of Theorem 4 is unachievable; and in a finite space, k could be chosen large enough that $\binom{k}{2}+1$ is more than the number of points in the

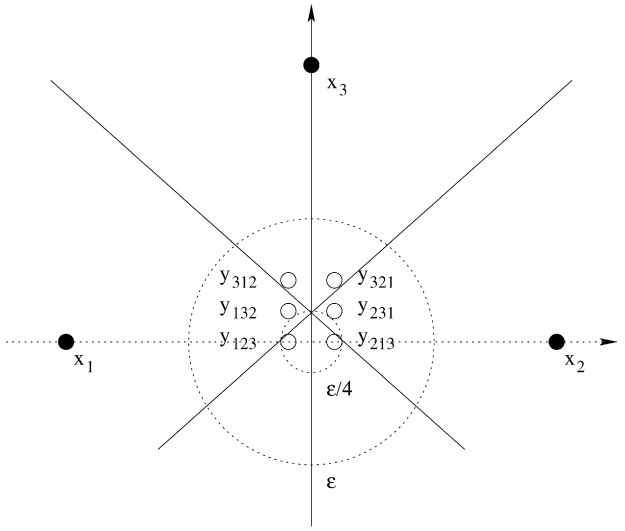

Fig. 6. Achieving all permutations in the vicinity of the origin.

space and thus could not possibly be achieved. However, those are exceptional cases. In general, for practical tree metrics such as the prefix metric, long paths are plentiful and the bound of $\binom{k}{2}+1$ is easily achieved.

4. Real vectors with L_p metrics

Euclidean spaces are familiar and widely used, so it is natural to examine metric space questions there. We also consider the other Minkowski L_p metrics, defined for points $\mathbf{x} = \langle x_1, x_2, \ldots, x_n\rangle$ and $\mathbf{y} = \langle y_1, y_2, \ldots, y_n\rangle$ by $d(\mathbf{x},\mathbf{y}) = (\sum_{i=1}^{n} |x_i - y_i|^p)^{1/p}$ for real $p \geqslant 1$ or $d(\mathbf{x},\mathbf{y}) = \max_{i=1}^{n} |x_i - y_i|$ for $p = \infty$. These spaces are a simple generalisation of Euclidean space and share many of its properties; in particular, the L_2 metric is the Euclidean metric. Let $N_{d,p}(k)$ represent the maximum number of distinct distance permutations generated by k sites in the space of d-dimensional real vectors with the L_p metric.

First of all, it is possible to make all $k!$ permutations occur in sufficiently high dimension. The construction places points with care at approximately unit distance from the origin, one on each coordinate axis and an additional one on the opposite side on the first axis, as shown in Fig. 6. All permutations are forced to occur inside a small sphere centred on the origin, giving the following theorem.

Theorem 6. *In d-dimensional real vector space with any L_p metric, k sites can be chosen such that all $k!$ distinct distance permutations exist, for any $k \leqslant d+1$. That is, $N_{d,p}(k) = k!$ for $d \geqslant k-1$ and any $p \geqslant 1$.*

Proof. For $k = 1$ the question is trivial: zero-dimensional space has only one point, we choose it as the site, and it has the single distance permutation consisting of itself. For $k \geqslant 2$ we prove a somewhat stronger statement by induction on k, namely that for any integer $k \geqslant 2$ and real $\epsilon > 0$, there exist k sites $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$ in $(k-1)$-dimensional L_p space such that for any permutation $\pi : \{1, 2, \ldots, k\} \to \{1, 2, \ldots, k\}$, there is a point $\mathbf{y}_\pi$ such that

$$\Pi_{\mathbf{y}_\pi} = \pi \tag{1}$$

$$d(\mathbf{0}, \mathbf{y}_\pi) < \epsilon \tag{2}$$

$$\left|1 - d(\mathbf{x}_i, \mathbf{y}_\pi)\right| < \epsilon \tag{3}$$

$$d(\mathbf{x}_i, \mathbf{y}_\pi) \neq d(\mathbf{x}_j, \mathbf{y}_\pi) \quad \text{if } \mathbf{x}_i \neq \mathbf{x}_j. \tag{4}$$

In other words, with $k-1$ dimensions we can achieve all $k!$ permutations (1) with points that are near the origin (2), almost exactly unit distance from all the sites (3), and not equidistant from any two sites (4).

Basis case. For $k = 2$, let $\mathbf{x}_1 = \langle -1\rangle$, $\mathbf{x}_2 = \langle 1\rangle$. Then where the two permutations are denoted by 12 and 21, we have $\mathbf{y}_{12} = \langle -\epsilon/2\rangle$ and $\mathbf{y}_{21} = \langle \epsilon/2\rangle$. These points are easily seen to meet the conditions (1)–(4).

Inductive step. For $k > 2$ and some $\epsilon > 0$, assume that there exist $k-1$ sites $\mathbf{x}'_1, \mathbf{x}'_2, \ldots, \mathbf{x}'_{k-1}$ in $(k-2)$-dimensional space such that for any permutation $\pi' : \{1, 2, \ldots, k-1\} \to \{1, 2, \ldots, k-1\}$, there is a point $\mathbf{y}'_{\pi'}$ such that $\Pi_{\mathbf{y}'_{\pi'}} = \pi'$ (1), $d(\mathbf{0}, \mathbf{y}'_{\pi'}) <$

$\epsilon/4$ (2), $|1 - d(\mathbf{x}'_i, \mathbf{y}'_{\pi'})| < \epsilon/4$ (3), and $d(\mathbf{x}'_i, \mathbf{y}'_{\pi'}) \neq d(\mathbf{x}'_j, \mathbf{y}'_{\pi'})$ if $\mathbf{x}'_i \neq \mathbf{x}'_j$ (4). This is simply the statement currently being proved, with one less dimension and ϵ divided by four.

Let $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{k-1}$ be the sites $\mathbf{x}'_1, \mathbf{x}'_2, \ldots, \mathbf{x}'_{k-1}$ extended to one more dimension by appending a zero component to each, and let $\mathbf{x}_k = \langle 0, 0, \ldots, 0, 1+\epsilon/4\rangle$; that is, we are adding one dimension and placing a new site on the newly-introduced coordinate axis at distance $1 + \epsilon/4$.

Let π be an arbitrary permutation of the k site indices and π' be π with k removed; for instance, if $k = 5$ and $\pi = 12543$ then π' would be 1243. Let $\mathbf{y}$ represent $\mathbf{y}'_{\pi'}$ augmented with one more component (to make it $(k-1)$-dimensional) and let z represent the value of the last component of $\mathbf{y}$. Consider the distance permutation of $\mathbf{y}$ as we increase z from $-\epsilon/2$ to $3\epsilon/4$. In all cases the distance permutation of $\mathbf{y}$ with respect to the first $k-1$ sites will be π', because the distance permutation is determined by inequalities of the form $d(\mathbf{x}_i, \mathbf{y}) \leqslant d(\mathbf{x}_j, \mathbf{y})$, each distance is the $1/p$ power of a sum of pth powers of per-component differences, and we are changing one of those per-component differences that is added equally to all the distances. All the functions involved are monotonic, so the inequalities continue to hold as we vary z.

Note 1. In the case of the L_∞ metric we depend on the fact that the per-component difference for the last component is smaller than any of the distances from $\mathbf{y}$ to sites and so does not enter into the maximum that defines the metric. We can ensure this by assuming ϵ less than $1/2$, so that $1 - \epsilon > \epsilon$; we are free to do that because the statement we are proving always holds for larger ϵ if it holds for small ϵ.

When $z = -\epsilon/2$, the distance $d(\mathbf{x}_k, \mathbf{y})$ must be at least $1 + 3\epsilon/4$ because that is the last per-component difference. But all the distances $d(\mathbf{x}_i, \mathbf{y})$ for $i < k$ must be less than $1 + 3\epsilon/4$ by the triangle inequality, because $d(\mathbf{x}_i, \mathbf{y}') < 1 + \epsilon/4$ by the inductive assumption and $d(\mathbf{y}', \mathbf{y}) = -z = \epsilon/2$ by definition. Therefore when $z = -\epsilon/2$, $\mathbf{y}$ is strictly farther from $\mathbf{x}_k$ than any other site, and the distance permutation of $\mathbf{y}$ ends with k.

On the other hand, when $z = 3\epsilon/4$, the distance $d(\mathbf{x}_k, \mathbf{y})$ must be less than $1 - \epsilon/4$, because it is at most the last per-component difference of $1 - \epsilon/2$, plus $d(\mathbf{0}, \mathbf{y}'_\pi) < \epsilon/4$ by the inductive assumption. The distance $d(\mathbf{x}_i, \mathbf{y})$ for all $i < k$ must be at least $1 - \epsilon/4$ because it must be at least $d(\mathbf{x}_i, \mathbf{y}')$ by the construction and $d(\mathbf{x}_i, \mathbf{y}')$ is at least $1 - \epsilon/4$ by the inductive assumption. Therefore when $z = 3\epsilon/4$, $\mathbf{y}$ is strictly nearer to $\mathbf{x}_k$ than any other site, and the distance permutation of $\mathbf{y}$ begins with k.

By choosing a value of z between those two extremes, we can find a value of $\mathbf{y}$ where k appears in any position in the distance permutation; and since this holds for any permutation π' of the first $k-1$ sites, we can find a $\mathbf{y}_\pi$ for any permutation π of the k sites, giving (1), a point for every permutation. By doing this we are perturbing each $\mathbf{y}'$ by at most $3\epsilon/4$ from its original position which was within $\epsilon/4$ of the origin, so each $\mathbf{y}$ remains within ϵ distance of the origin (2); similarly, the distance from each $\mathbf{y}$ to each site must be in the interval $1 \pm \epsilon$ (3); and by our choices of z, all the distances to sites are distinct at each $\mathbf{y}$ (4). Therefore the theorem holds for k sites.

By induction, the theorem holds for all values of k. □

A classical problem (often stated in terms of cutting a cake, or a cheese) asks how many pieces can be formed by cutting d-dimensional Euclidean space with m hyperplanes of dimension $d-1$ in general position. Price shows that where $S_d(m)$ represents the number of pieces formed by m cuts in d-dimensional Euclidean space, then $S_d(0) = S_0(m) = 1$; and $S_d(m) = S_d(m-1) + S_{d-1}(m-1)$ for $d, m > 0$ [23]. His proof is an induction that follows the structure of the recurrence relation: when we add the mth hyperplane to an arrangement that already contains $S_d(m-1)$ pieces, then the new hyperplane is itself a $(d-1)$-dimensional space cut up by the $m-1$ existing hyperplanes into $S_{d-1}(m-1)$ pieces, and each of those partitions off a new piece in the original d-dimensional space, proving the recurrence. It also follows easily that $S_d(m) = \Theta(m^d)$ [23].

The Euclidean cake-cutting problem provides a starting point for counting the pieces formed by bisectors in real vector spaces. Since there are $\binom{k}{2}$ bisectors between k sites, if the bisectors were in general position relative to each other then we would have the number of distance permutations in Euclidean space equal to the number of pieces formed by $\binom{k}{2}$ hyperplanes, or $N_{d,2}(k) = S_d(\binom{k}{2})$. Since the bisectors are not in general position, the actual number of distance permutations is less; but that remains as an upper bound, giving $N_{d,2}(k) = O(k^{2d})$ because $\binom{k}{2}$ is $\Theta(k^2)$ and $S_d(m)$ is $\Theta(m^d)$. That result will be extended to other metrics in Theorem 9, but first we give an exact result for the Euclidean case.

Theorem 7. *In d-dimensional Euclidean space,*

$$N_{0,2}(k) = N_{d,2}(k) = 1, \tag{5}$$

$$N_{d,2}(k) = N_{d,2}(k-1) + (k-1)N_{d-1,2}(k-1). \tag{6}$$

Proof. Zero-dimensional space contains only one point and so can only contain one piece, and with only one site, there are no bisectors and the space remains undivided. Therefore $N_{0,2}(k) = N_{n,2}(1) = 1$.

For the general case we extend the line of reasoning used by Price [23]. Consider the space with n dimensions that already contains $k-1$ sites, their bisectors, and the resulting pieces. It contains, by definition, $N_{n,2}(k-1)$ pieces. Adding

Table 1
Number of distance permutations $N_{d,2}(k)$ in Euclidean space.

	$k=2$	3	4	5	6	7	8
$d=1$	2	4	7	11	16	22	29
2	2	6	18	46	101	197	351
3	2	6	24	96	326	932	2311
4	2	6	24	120	600	2556	9080
5	2	6	24	120	720	4320	22212
6	2	6	24	120	720	5040	35280
7	2	6	24	120	720	5040	40320
8	2	6	24	120	720	5040	40320
9	2	6	24	120	720	5040	40320
10	2	6	24	120	720	5040	40320

	$k=9$	10	11	12
$d=1$	37	46	56	67
2	583	916	1376	1992
3	5119	10366	19526	34662
4	27568	73639	177299	392085
5	94852	342964	1079354	3029643
6	212976	1066644	4496284	16369178
7	322560	2239344	12905784	62364908
8	362880	3265920	25659360	167622984
9	362880	3628800	36288000	318540960
10	362880	3628800	39916800	439084800

one more site adds a group of $k-1$ bisectors. The first of those is a $(n-1)$-dimensional space cut by the existing bisectors of $k-1$ sites into (by definition) $N_{n-1,2}(k-1)$ pieces, and each of those pieces creates a new piece in the n-dimensional space as well.

The second of the $k-1$ new bisectors appears to be cut by the existing bisectors and also the one we just added. However, the intersection of the first new bisector and the second new bisector is exactly the same set as the intersection of the second new bisector with some other bisector that already existed. Letting a and b be sites added earlier and x be the new site, then we have $a|x \cap b|x = a|b \cap b|x$ by the transitivity of equality. So intersections between bisectors in the same group need not be counted; they are always equal to the intersections already counted between bisectors in the new group and bisectors in earlier groups.

Therefore each of the $k-1$ new bisectors in the new group, not just the first, adds exactly $N_{n-1,2}(k-1)$ pieces. There are also by definition $N_{n,2}(k-1)$ pieces that existed before we added the latest site. Therefore we have the recurrence relation (6). □

Numerical results are shown in Table 1. Note the factorials that appear in the lower triangle, corresponding to Theorem 6. For the one-dimensional case, the formula reduces to $\binom{k}{2}+1$, which is equal to the value for tree metrics from Theorem 4. The proof of Theorem 7 takes the same general approach used by Price [23]. The complication is that because equality is transitive, some of the intersections among bisectors must coincide. With three sites A, B, and C, $A|B \cap B|C \subseteq A|C$. Accounting for those intersections and the resulting missing pieces leads to Theorem 7. Bounds on $N_{d,2}(k)$ then follow by induction:

Corollary 8. *The function $N_{d,2}(k)$ satisfies*:

$$N_{d,2}(k) \leqslant k^{2d}, \tag{7}$$

$$N_{d,2}(k) = \frac{k^{2d}}{2^d d!} + o\left(k^{2d}\right). \tag{8}$$

Therefore, the distance permutation in a Euclidean space can be stored in $\Theta(d \log k)$ bits.

Proof. The proof for (7) is by induction on k. The result holds trivially for $k=1$. Then we have $N_{n,2}(k) = N_{n,2}(k-1) + (k-1)N_{n-1,2}(k-1)$, and substituting in the inductive hypothesis gives $N_{n,2}(k) \leqslant k^{2n}$. Then the space to store a distance permutation is $\lg N_{n,2}(k)$ bits, so $2n \lg k$ is an upper bound.

For (8) we use induction on n. It holds trivially for $n=0$. Let a_n and b_n represent the leading two coefficients of the polynomial in k that defines $N_{n,2}(k)$; then we have:

$$\begin{aligned} N_{n,2}(k) &= a_n k^{2n} + b_n k^{2n-1} + o\left(k^{2n-1}\right) \\ &= a_n (k-1)^{2n} + b_n (k-1)^{2n-1} + (k-1) a_{n-1} (k-1)^{2n-2} + o\left(k^{2n-1}\right) \\ &= a_n k^{2n} - 2n a_n k^{2n-1} + b_n k^{2n-1} + a_{n-1} k^{2n-1} + o\left(k^{2n-1}\right). \end{aligned}$$

The sum of the coefficients for the k^{2n-1} term must be b_n by definition, so we have $b_n = -2na_n + b_n + a_{n-1}$. Solving for a_n gives $a_n = a_{n-1}/(2n)$, and with $a_0 = 1$ from the basis case we have $a_n = 1/(2^n n!)$. □

With other L_p metrics, the situation is more complicated. Consider the two-dimensional L_1 case shown in Fig. 4. A bisector in this space generally consists of an orthogonal line with a diagonal kink in the middle. In the Euclidean plane, two bisectors either coincide, intersect at exactly one point, or do not intersect at all; and if they are in general position relative to each other, they must intersect at exactly one point. But here, two bisectors can be in general position relative to each other and still fail to intersect, like $A|D$ and $B|C$; or they can intersect at exactly two points, like $A|B$ and $C|D$. There are also many degenerate cases possible, in which the intersection might be for instance two disjoint rays, or a ray with a line segment attached. Higher dimensions are even worse. Because the intersections are not well-behaved in non-Euclidean metrics, we cannot treat each bisector as a space of the same type, subject to the overall result as part of an induction.

However, the difficult combinatoric issues come from seeking an exact and general answer. In the two special cases of L_1 and L_∞ spaces, which happen to be of great practical interest, we can prove a new asymptotic bound with elementary results. For $p \in \{1, 2, \infty\}$, bisectors are piecewise linear. That is, each bisector consists of a union of subsets of hyperplanes; and the maximum number of hyperplanes per bisector is a function of the dimension. For instance, in two-dimensional L_1 space as seen in Fig. 4, each bisector is a union of subsets of at most three lines. Then cutting up d-dimensional L_p space with the bisectors of k points can yield no more pieces than cutting up d-dimensional Euclidean space with $O(f(d)k^2)$ hyperplanes in general position; that gives the following result.

Theorem 9. *The function $N_{d,p}(k)$ satisfies*:

$$N_{d,1}(k) = O\left(2^{2d^2}k^{2d}\right) \tag{9}$$

$$N_{d,2}(k) = O\left(k^{2d}\right) \tag{10}$$

$$N_{d,\infty}(k) = O\left(2^{2d}d^{2d}k^{2d}\right). \tag{11}$$

All three of these are $O(k^{2d})$ for constant d.

Proof. The case of the L_2 metric is already covered by Corollary 8. For the other two, consider a pair of sites $\mathbf{x}$ and $\mathbf{y}$, and let $\mathbf{z}$ be on their bisector; then $d(\mathbf{x}, \mathbf{z}) = d(\mathbf{y}, \mathbf{z})$. We will show that for each value of $p \in \{1, 2, \infty\}$, the bisector is a subset of the union of some flat hyperplanes, with an upper bound on the number of hyperplanes determined only by the number of dimensions n. Subscripts denote individual components of the vectors.

For the L_1 metric, we have $d(\mathbf{x}, \mathbf{z}) = |x_1 - z_1| + |x_2 - z_2| + \cdots + |x_n - z_n|$, which is $\pm(x_1 - z_1) \pm (x_2 - z_2) \pm \cdots \pm (x_n - z_n)$ for some choice of the signs dependent on the component values. Thus $d(\mathbf{x}, \mathbf{z})$ is equal to one of 2^n linear functions of $\mathbf{x}$ and $\mathbf{z}$. Similarly, $d(\mathbf{y}, \mathbf{z})$ is equal to one of 2^n linear functions of $\mathbf{y}$ and $\mathbf{z}$. The set of points at which $d(\mathbf{x}, \mathbf{z}) = d(\mathbf{y}, \mathbf{z})$ is thus a subset of the set of points at which at least one of the functions for $d(\mathbf{x}, \mathbf{z})$ equals at least one of the functions for $d(\mathbf{y}, \mathbf{z})$; therefore it must be a subset of the union of 2^{2n} hyperplanes.

For the L_∞ metric, we have $d(\mathbf{x}, \mathbf{z}) = \max\{|x_1 - z_1|, |x_2 - z_2|, \ldots, |x_n - z_n|\}$, which is $\pm(x_i - z_i)$ for some choice of the sign and the index i dependent on the component values. So, similarly to the L_1 case, $d(\mathbf{x}, \mathbf{z})$ is equal to one of $2n$ linear functions of $\mathbf{x}$ and $\mathbf{z}$, and $d(\mathbf{y}, \mathbf{z})$ is equal to one of $2n$ linear functions of $\mathbf{y}$ and $\mathbf{z}$. The bisector is a subset of the union of $4n^2$ hyperplanes.

Since each bisector is a subset of the union of some hyperplanes, we can only increase the number of cells in an arrangement of bisectors if we expand each bisector to be the entire union instead of a proper subset. In the cake analogy, that is like extending a cut to slice all the way through the cake instead of only through the first layer. Assuming the hyperplanes to be in general position can also only increase the number of cells. With k sites, there are $\binom{k}{2} = \Theta(k^2)$ bisectors, and by Price's result the number of cells for m hyperplanes in general position in n dimensions is $\Theta(m^n)$ [23]. Combining those with the upper bound on number of hyperplanes per bisector given above, the theorem follows. □

This result gives an asymptotic improvement in the bound on storage space for distance permutations, because a general permutation of k sites would require $\Theta(k \log k)$ bits. When the number of points in the database is large in comparison to the number of permutations, the bound can be achieved simply by storing the full permutations in a separate table and storing the index numbers into that table alongside the points. For smaller databases a more sophisticated structure may be possible, taking into account the special structure of the set of permutations. The practical consequence of the limit on number of permutations is that adding sites costs very little in index space requirement, once the number of sites is significant compared to the number of dimensions. On the other hand, it also suggests that once we have about twice as many sites as dimensions, there is little value in adding more sites; the distance permutation contains little more information.

We emphasise that the bounds for L_1 and L_∞ are very loose with respect to d. In applications we are generally given a space and cannot change it, whereas we have the opportunity to choose the number of sites k. For that reason Theorem 9 is aimed primarily at determining the asymptotic behaviour with respect to k, not d. The counting procedure assumes every hyperplane intersects with and is in general position relative to all other hyperplanes. In fact many hyperplanes will

be parallel to each other, or have their extents limited, such that they do not intersect. Fewer intersections lead to fewer cells. In Fig. 4 we see an example from L_1 where the number of distance permutations is 18, exactly equal to the number from L_2 space despite the 2^{2d^2} term in the bound; that was the largest number obtainable in informal computer-graphics experiments, and it seems intuitively clear that the number of permutations should be approximately the same for all the L_p metrics. Then we face the question of whether the Euclidean bound might actually be a bound for all L_p spaces, or a practical estimate even if not strictly a bound. That question is examined in the experiments.

5. Experimental results

Because we are interested in worst-case storage space of data structures, our theoretical results focus on computing the maximum possible number of distance permutations that could occur in any data set. That is also the best case, in one sense, for permutation-based similarity search algorithms like iAESA: having as many distinct permutations in the index as possible means that maximum information can be extracted from the index without needing distances to be computed at search time. However, in a real database which may not fill the space completely, the number of distance permutations actually occurring may be significantly less than the theoretical maximum.

Fig. 7 shows two ways a distance permutation could fail to be included in the database. The grey box represents the range of values present in the database, and the circles represent individual database points. Some cells of the generalised Voronoi diagram may not happen to contain any database points, and in that case their permutations will not appear. A large enough database would be expected to hit all such cells. But other cells, like the cross-hatched ones at the right of the figure, may lie entirely outside the range of database values. Those permutations will never appear no matter how large the database grows, if data values stay range-limited.

To examine such issues, we implemented distance permutations for the SISAP library of Figueroa, Navarro, and Chávez [12], as a new index type called `distperm`. Our `distperm` code is a minor modification of the library's `pivots` index type. The library's `iaesa` index type uses distance permutations internally, but as part of a more sophisticated algorithm, making it harder to modify for counting permutations. Our `build-distperm-*` programs write out the permutations in ASCII as a side effect of index generation, so that the number of unique permutations can easily be counted with `sort | uniq | wc`.

We used our code to count the number of unique distance permutations for a variety of metric spaces including randomly-generated vectors and the sample databases supplied with the library. Results on the SISAP library's sample databases [12] are shown in Table 2, and for vectors (10^6 uniformly chosen from the unit cube) in Table 3. Because the result for vectors depends on the random choice of sites, we ran each vector experiment 100 times, and show both the mean and maximum number of distance permutations observed, for selected values of k, the number of sites.

These data sets were chosen for consistency with others' work. The benefits to the field from test data standardisation are obvious, and use of the SISAP library in particular was a stated requirement for the workshop at which we announced these results [9]. As we and others have noted (and as seen in our present results) the properties of the uniform distribution may be significantly different from those of more realistic data sets [26]. However, partly because its high dimensionality tends to push the limits of indexing systems, the uniform distribution remains the standard for testing index data structures [7,20,22,30].

The most obvious feature of these results is that the numbers are so small. For instance, with the sample database `long`, which contains feature vectors extracted from news articles, with 12 sites there are only 261 distinct distance permutations.

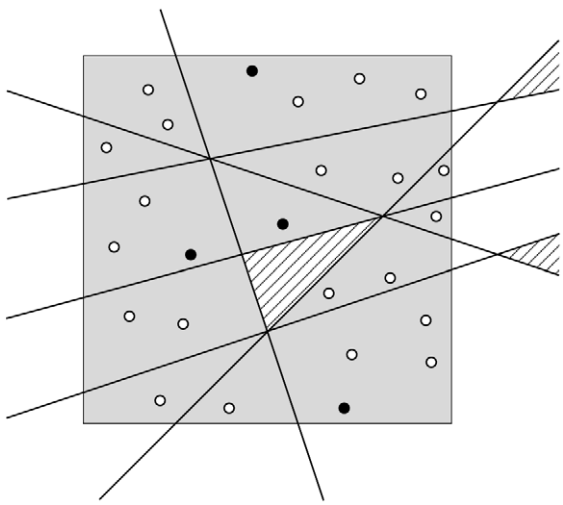

Fig. 7. The database may not hit every cell.

Table 2
Number of distance permutations for the SISAP sample databases.

Database	n	ρ	$k=3$	4	5	6	7	8
Dutch	229 328	7.159	6	24	119	577	2693	11 566
English	69 069	8.492	6	24	120	645	2211	7140
French	138 257	10.510	6	24	118	475	2163	8118
German	75 086	7.383	6	24	119	517	1639	4839
Italian	116 879	10.436	6	24	120	653	3103	10 872
Norwegian	85 637	5.503	6	24	118	632	2530	7594
Spanish	86 061	8.722	6	24	118	598	2048	5428
listeria	20 660	0.894	4	11	19	29	49	85
long	1265	2.603	5	10	22	47	51	98
short	25 276	808.739	6	24	111	508	2104	6993
colors	112 544	2.745	6	18	44	96	200	365
nasa	40 150	5.186	6	24	115	530	1820	3792
Database	n	ρ	$k=9$	10	11	12		
Dutch	229 328	7.159	34 954	74 954	116 817	163 129		
English	69 069	8.492	16 212	28 271	38 289	45 744		
French	138 257	10.510	19 785	35 903	58 453	81 006		
German	75 086	7.383	10 154	19 489	30 347	43 208		
Italian	116 879	10.436	27 843	45 754	71 921	90 316		
Norwegian	85 637	5.503	15 147	25 872	42 992	57 988		
Spanish	86 061	8.722	13 357	23 157	39 443	54 628		
listeria	20 660	0.894	206	510	952	1145		
long	1265	2.603	114	163	252	261		
short	25 276	808.739	13 792	20 223	23 102	23 940		
colors	112 544	2.745	796	1563	2800	4408		
nasa	40 150	5.186	7577	13 243	19 066	24 154		

Table 3
Number of distance permutations for uniform random vectors.

	d	ρ	mean perms			max perms		
			$k=4$	8	12	$k=4$	8	12
L_1	1	1.00	7.00	29.00	66.99	7	29	67
	2	2.00	14.86	261.71	1436.87	18	305	1541
	3	3.00	21.08	1464.98	16 398.42	24	1923	19 658
	4	4.00	23.56	4832.90	81 304.91	24	6661	100 133
	5	5.00	23.92	9909.68	218 714.35	24	13 573	258 874
	6	6.00	24.00	15 937.97	399 705.65	24	20 666	485 317
	7	7.00	24.00	21 593.99	580 001.49	24	30 086	661 262
	8	8.00	24.00	25 261.27	720 120.79	24	33 637	788 347
	9	9.00	24.00	28 730.97	811 518.59	24	37 198	872 023
	10	10.00	24.00	31 418.99	878 756.82	24	37 667	935 715
L_2	1	1.00	7.00	28.99	66.99	7	29	67
	2	2.21	15.35	271.79	1456.10	18	312	1583
	3	3.52	21.35	1360.24	14 605.82	24	1664	16 326
	4	4.88	22.74	3970.11	67 709.09	24	5247	77 766
	5	6.27	23.50	8043.95	181 511.81	24	11 277	226 874
	6	7.68	23.86	13 089.65	343 377.92	24	22 644	439 620
	7	9.09	23.91	15 891.40	504 358.71	24	30 652	615 441
	8	10.50	23.99	20 431.39	646 276.54	24	35 694	796 775
	9	11.92	24.00	22 891.22	729 070.09	24	34 037	864 896
	10	13.35	24.00	26 128.61	817 225.75	24	39 417	924 472
L_∞	1	1.00	6.99	28.97	66.95	7	29	67
	2	2.23	13.81	237.53	1317.41	18	298	1528
	3	3.59	18.90	1222.09	12 805.30	24	1888	17 441
	4	5.05	21.73	3665.22	56 767.84	24	5688	73 315
	5	6.58	22.67	7133.63	149 166.98	24	12 566	235 359
	6	8.17	23.73	10 772.22	252 573.87	24	20 988	352 150
	7	9.80	23.72	14 774.93	371 777.13	24	27 150	574 611
	8	11.47	23.73	16 489.73	475 934.17	24	29 989	683 855
	9	13.17	23.85	19 999.01	567 307.71	24	33 293	730 139
	10	14.90	24.00	23 159.34	637 689.81	24	34 984	770 769

That is not just because the database is small. It contains 1265 points, much less than $\sqrt{12!}$, so if the distance permutations were chosen uniformly from all possible permutations we should expect no collisions and 1265 distinct permutations. The observed number is less by a factor of about 4.8. Similar effects show up in the listeria and colors databases at the

maximum permutation length, and in many of the sample databases for shorter permutations: the number of permutations observed is often much less than the number of database points even when $k!$ is larger still.

By comparing numbers from Table 2 with the values for Euclidean spaces in Table 3, we see that `colors` has a few more distance permutations than a two-dimensional uniform distribution in Euclidean space. The `nasa` database has as many distance permutations as a Euclidean uniform distribution with between three and four dimensions, ignoring the values for $k = 12$ because there the permutations appear to be limited by the number of points in the database. The dictionary databases vary, but seem equivalent to Euclidean uniform distributions with up to six dimensions. And the `listeria` database, despite having plenty of points, seems equivalent to a Euclidean uniform distribution with just under two dimensions. In this way we can characterise the dimensionality of a database in a highly general way.

Comparison to the intrinsic dimensionality ρ, defined by Chávez and Navarro as mean squared divided by twice the variance of distance between two random points [8], seems natural but may not be meaningful. The intrinsic dimensionality depends heavily on the probability distribution [26], whereas the number of distinct distance permutations depends only on which points can exist at all. Thus, no firm relationship between ρ and distance permutations can ever exist. Two different distributions with the same support can have different ρ values and the same maximum number of distance permutations. The ρ statistic also describes distances only among random points chosen from the entire space, which will usually be far apart; an indexing data structure's behaviour with small query radius may be better described by other measures of dimensionality, such as the D_q dimensions. The D_q dimensions describe how probability density increases with radius at small radii [21,28]. Nonetheless, ρ is a convenient way to describe distributions in metric spaces, and we give ρ values for reference only based on the assumption of choosing points uniformly from the databases.

The random-vector experiments suggest that all three tested metrics produce comparable numbers of distance permutations. There is a general downward trend in number of permutations from L_1 to L_2 and from L_2 to L_∞. Database size interferes with comparison to the Euclidean maximum when that approaches and exceeds 10^6, but we can see from smaller permutations and dimensions that the Euclidean maximum is seldom achieved even when it could be. For instance, with four-dimensional vectors and permutations of length 12, the Euclidean value from Table 1 is 392 085, and the largest number of permutations we saw with any L_p metric was 100 133. The Euclidean value describes the number of generalised Voronoi cells for all non-degenerate choices of sites, so all 392 085 permutations could be achieved with database points in the right places; and the average of about 10 database points per permutation observed suggests that we cannot have missed very many of the cells intersecting the unit cube. It seems the usual case is for many distance permutations to be associated with cells that nowhere intersect the unit cube.

A notable result not shown in Table 3 is that in three-dimensional L_1 space, the experiment found a database and choice of five sites giving 108 distinct distance permutations in the test database, exceeding the limit of 96 for Euclidean space. Even more than 108 permutations may exist because the experiment only counted permutations represented in the database. Therefore the hypothesis that the Euclidean limit applies to all L_p spaces is false. The exceptional sites are:

$$\begin{aligned}
\mathbf{x}_1 &= \langle 0.205281, 0.621547, 0.332507 \rangle, \\
\mathbf{x}_2 &= \langle 0.053421, 0.344351, 0.260859 \rangle, \\
\mathbf{x}_3 &= \langle 0.418166, 0.207143, 0.119789 \rangle, \\
\mathbf{x}_4 &= \langle 0.735218, 0.653301, 0.650154 \rangle, \\
\mathbf{x}_5 &= \langle 0.527133, 0.814207, 0.704307 \rangle.
\end{aligned} \tag{12}$$

Similar counterexamples were found for three-dimensional spaces with L_1 and $k = 6$, L_∞ and $k = 5$, and for four-dimensional space with L_1 and $k = 6$. These prove that $N_{n,p}(k) = N_{n,2}(k)$ is not true in general.

6. Conclusions

We have described the problem of counting how many distance permutations are possible in a space, and given exact solutions for tree metrics and Euclidean spaces. For the L_1 and L_∞ metrics on real vectors, we have given an asymptotic analysis, which is sufficient to improve the best previous bound. We have also implemented permutation counting in the SISAP library [12], and given experimental results on the number of distance permutations found in the sample databases. The experimental results suggest a novel way of estimating the dimensionality of databases.

Acknowledgements

This paper is an expanded version of an extended abstract we presented at SISAP'08 [27], and includes results also described in the author's PhD thesis [28]. The comments of the anonymous reviewers are gratefully acknowledged.

References

[1] R. Agarwala, V. Bafna, M. Farach, B. Narayanan, M. Paterson, M. Thorup, On the approximability of numerical taxonomy (fitting distances by tree metrics), in: Proceedings of the Seventh Annual ACM–SIAM Symposium on Discrete Algorithms (SODA'96), ACM/SIAM, Atlanta, Georgia, USA, 1996.

[2] N. Alon, R.M. Karp, D. Peleg, D.B. West, A graph-theoretic game and its application to the *k*-server problem, SIAM Journal on Computing 24 (1) (1995) 78–100.
[3] F. Aurenhammer, Voronoi diagrams—a survey of a fundamental geometric data structure, ACM Computing Surveys 23 (3) (1991).
[4] Y. Bartal, Probabilistic approximations of metric spaces and its algorithmic applications, in: 37th Annual Symposium on Foundations of Computer Science (FOCS'96), IEEE, 1996.
[5] A. Björner, M. Las Vergnas, B. Sturmfels, N. White, G.M. Ziegler, Oriented Matroids, second ed., Cambridge University Press, 1999.
[6] P. Buneman, A note on the metric properties of trees, Journal of Combinatorial Theory, Series B 17 (1974) 48–50.
[7] E. Chávez, K. Figueroa, G. Navarro, Proximity searching in high dimensional spaces with a proximity preserving order, in: A. Gelbukh, A. de Álbornoz, H. Terashima-Marín (Eds.), 4th Mexican International Conference on Artificial Intelligence (MICAI'05), Springer, 2005.
[8] E. Chávez, G. Navarro, Measuring the dimensionality of general metric spaces, Tech. Rep. TR/DCC-00-1, Department of Computer Science, University of Chile, 2000.
[9] E. Chávez, G. Navarro (Eds.), Proceedings of the 1st International Workshop on Similarity Search and Applications (SISAP'08), Apr. 11–12, 2008, IEEE Computer Society, Cancún, Mexico, 2008.
[10] A.W.M. Dress, Trees, tight extensions of metric spaces, and the cohomological dimension of certain groups: A note on combinatorial properties of metric spaces, Advances in Mathematics 53 (3) (1984) 321–402.
[11] K. Figueroa, E. Chávez, G. Navarro, R. Paredes, On the least cost for proximity searching in metric spaces, in: C. Àlvarez, M. Serna (Eds.), Experimental and Efficient Algorithms: 5th International Workshop (WEA'06), Springer, 2006.
[12] K. Figueroa, G. Navarro, E. Chávez, Metric spaces library, online, http://www.sisap.org/?f=library, Accessed November 24, 2007.
[13] B. Grünbaum, Arrangements and Spreads, Conference Board of the Mathematical Sciences Regional Conference Series in Mathematics, vol. 10, American Mathematical Society, Providence, 1971.
[14] C. Icking, R. Klein, N.-M. Lê, L. Ma, Convex distance functions in 3-space are different, Fundamenta Informaticae 22 (4) (1995) 331–352.
[15] C. Icking, R. Klein, N.-M. Lê, L. Ma, F. Santos, On bisectors for convex distance functions in 3-space, in: 11th Canadian Conference on Computational Geometry (CCCG '99), University of British Columbia, 1999.
[16] D.G. Lowe, Distinctive image features from scale-invariant keypoints, International Journal of Computer Vision 60 (2) (2004) 91–110.
[17] B. Lynn, M. Prabhakaran, A. Sahai, Positive results and techniques for obfuscation, in: C. Cachin, J. Camenisch (Eds.), Advances in Cryptology: International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'04), Springer, Interlaken, Switzerland, 2004.
[18] A. Mandel, Topology of oriented matroids, Ph.D. thesis, University of Waterloo, 1982.
[19] J. Matoušek, On embedding trees into uniformly convex Banach spaces, Israel Journal of Mathematics 114 (1) (1999) 221–237.
[20] L. Micó, J. Oncina, E. Vidal, A new version of the nearest-neighbour approximating and eliminating search algorithm (AESA) with linear preprocessing time and memory requirements, Pattern Recognition Letters 15 (1) (1994) 9–17.
[21] E. Ott, Chaos in Dynamical Systems, second ed., Cambridge University Press, Cambridge, UK, 2002.
[22] R. Paredes, E. Chávez, Using the *k*-nearest neighbor graph for proximity searching in metric spaces, in: M.P. Consens, G. Navarro (Eds.), String Processing and Information Retrieval, 12th International Conference (SPIRE'05), Springer, 2005.
[23] D.J. Price, Some unusual series occurring in *n*-dimensional geometry, The Mathematical Gazette 30 (1946) 149–150.
[24] M. Sahlgren, The word-space model, Ph.D. thesis, Stockholm University, 2006, http://www.sics.se/~mange/TheWordSpaceModel.pdf.
[25] F. Santos, On Delaunay oriented matroids for convex distance functions, Discrete & Computational Geometry 16 (2) (1996) 197–210.
[26] M. Skala, Measuring the difficulty of distance-based indexing, in: M.P. Consens, G. Navarro (Eds.), String Processing and Information Retrieval: 12th International Conference (SPIRE'05), Springer, 2005.
[27] M. Skala, Counting distance permutations, in: Chávez and Navarro [9], pp. 69–76.
[28] M.A. Skala, Aspects of metric spaces in computation, Ph.D. thesis, University of Waterloo, 2008, http://ansuz.sooke.bc.ca/books/thesis.pdf.
[29] J.K. Uhlmann, Satisfying general proximity/similarity queries with metric trees, Information Processing Letters 40 (1991) 175–179.
[30] E. Vidal Ruiz, An algorithm for finding nearest neighbors in (approximately) constant time, Pattern Recognition Letters 4 (1986) 145–157.
[31] P.N. Yianilos, Data structures and algorithms for nearest neighbor search in general metric spaces, in: Fourth Annual ACM/SIGACT–SIAM, Symposium on Discrete Algorithms (SODA'93), ACM/SIAM, 1993.