Counting Distance Permutations

Matthew Skala
David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1
mskala@ansuz.sooke.bc.ca

Abstract

A distance permutation index supports fast proximity searching in a high-dimensional metric space. 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 spaces.

1. Introduction

Proximity searching is important for databases of images, text documents, genetic sequences, and audio and video clips, among others. These objects are often native to high-dimensional 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 [11]. The word space model, used for studying semantic relations in text, converts words into context vectors with thousands or millions of dimensions [17].

The naive algorithm for a range or kNN query 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 while doing fewer distance measurements. The data structure stores some precomputed information about the relationships among database objects, which can be used to infer things about the distance to the query object without measuring it directly. For instance, if it is known that the distance obeys the triangle inequality, the query measures the distance from query x to database point y as 3, and the data structure stores the fact that the distance from y to another database point z is 1, then the query can infer without measuring that the distance from x to z is at least 2.

There is a trade-off as to how much information the data structure should store. At one extreme, AESA stores *all* the pairwise distances among database points, paying a quadratic cost in space complexity but gaining significant speed [16]. Near the other extreme, a data structure can still gain some speed by storing as little information as the identities of the k nearest neighbours of each database point, with no other information about distances [14]. In this work we consider a choice between those two extremes: an approach introduced by Chávez, Figueroa, and Navarro that stores a permutation of k sites for each database point, representing which of the k sites is closest, which is second-closest, and so on [4].

In many spaces of interest, not all permutations can actually occur, and so the space requirement for this type of index can be reduced without further compromising the performance measured by number of distance evaluations. We also discuss experimental results on the number of permutations found in the SISAP sample databases [7]. Formal proofs of the theoretical results are sketched or omitted in the interest of giving a more detailed discussion and the experimental results.

1.1. Indexing with distance permutations

The query process must, explicitly or implicitly, evaluate the distance from the query to each point in the database to determine whether that database point should be returned in the result. If we can find an excuse

0-7695-3101-6/08 $25.00 © 2008 IEEE
DOI 10.1109/SISAP.2008.15

to skip over a subset of points in 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 trees, the search attempts to exclude subtrees from examination [20, 21].

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. Because the measure of cost is the number of distance function evaluations during the search, a technique that reduces those can be valuable even if it still does a linear examination of all points in the database. AESA is the prototype for this kind of technique [16]. 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 [13]. 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 [4]. 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. Their experimental results show that these permutations contain enough information to do an efficient search, while consuming much less storage space. They claim a reduction in storage space requirement from $O(nk\log n)$ bits for LAESA, to $O(nk\log k)$ [4]. 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 [6].

A formal definition of distance permutations follows. Note that other authors refer to these objects as proximity preserving orders; we call them distance permutations to emphasize their connection with existing work on permutation metrics, combinatorics of permutations, and so on.

Definition 1. Given k points $x_1, x_2, \ldots, x_k$, called the **sites**, in some space with distance function d, the **distance permutation** of a point y, 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 $\Pi_y(i) < \Pi_y(j)$. That is, 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.

In some spaces, the number of distance permutations that can actually occur may be significantly less than the $k!$ permutations of k sites; as a result, the 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 [6]. 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.

1.2. Generalized Voronoi diagrams

The cells of Voronoi diagrams correspond to classes of distance permutations. For instance, in the conventional nearest-neighbour Voronoi diagram of Fig. 1(a), 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 generalizations of Voronoi diagrams have been studied, including *higher-order* diagrams in which the cells correspond to the set of k nearest neighbours instead of just the one very nearest neighbour [1]. An example for $k = 2$ is shown in Fig. 1(b). 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. 2(a). 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. 2(a) 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 gen-

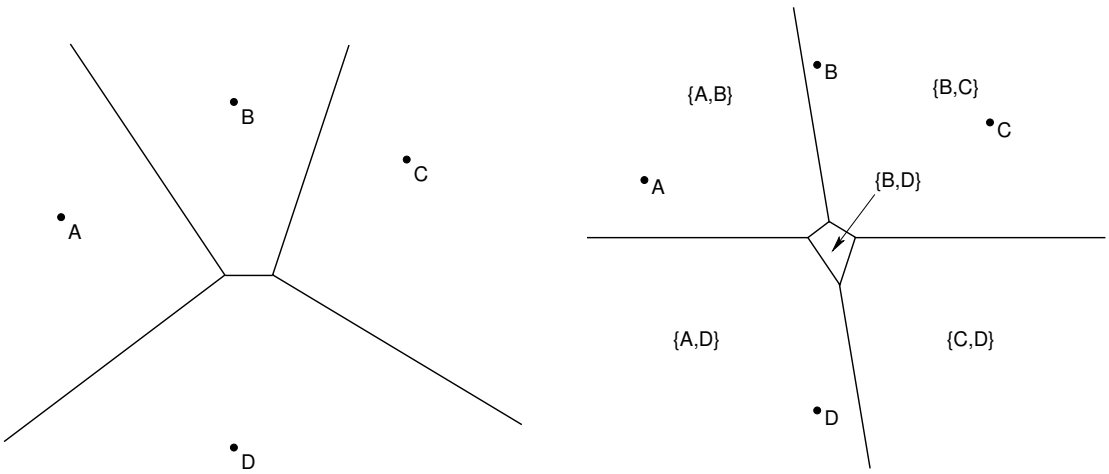

Figure 1. Euclidean Voronoi diagrams

eral definition and notation for them:

Definition 2. The bisector of two points x and y, denoted by $x|y$, is the set of all points z such that $d(x,z) = d(y,z)$.

If points can be on either side of each of six bisectors in Fig. 2(a), 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 plane limits the number of cells.

Arrangements of hyperplanes, which include bisector systems in Euclidean space, create combinatorial objects called *oriented matroids*, and those are well-studied [3]. Unfortunately, most of the relevant results are inapplicable to bisectors in more general spaces. Many authors including Grünbaum [8] and Mandel [12] have applied oriented matroids to arrangements of pseudolines and pseudospheres (respectively), which describe intersections of generalized 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. 2(b) 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 [18].

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 [9], and that the combinatorial structure around the one-dimensional bisector of three points can be different for different connected components of the bisector [10]. They survey other problematic results on 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." [10]

2. Tree metrics

Consider first a space with a tree metric. That is, each point in the space is associated with a vertex in a (possibly infinite) tree, and the distance between two points is the number of edges in the path between them. A natural extension is to place positive real weights on the edges and let the metric be the sum of weights on the path between two points. One simple tree metric is the prefix metric between strings, which 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. Tree metrics are commonly used in creating well-behaved approximations of arbitrary metrics [2].

Theorem 1. For k sites in a space with a (possibly

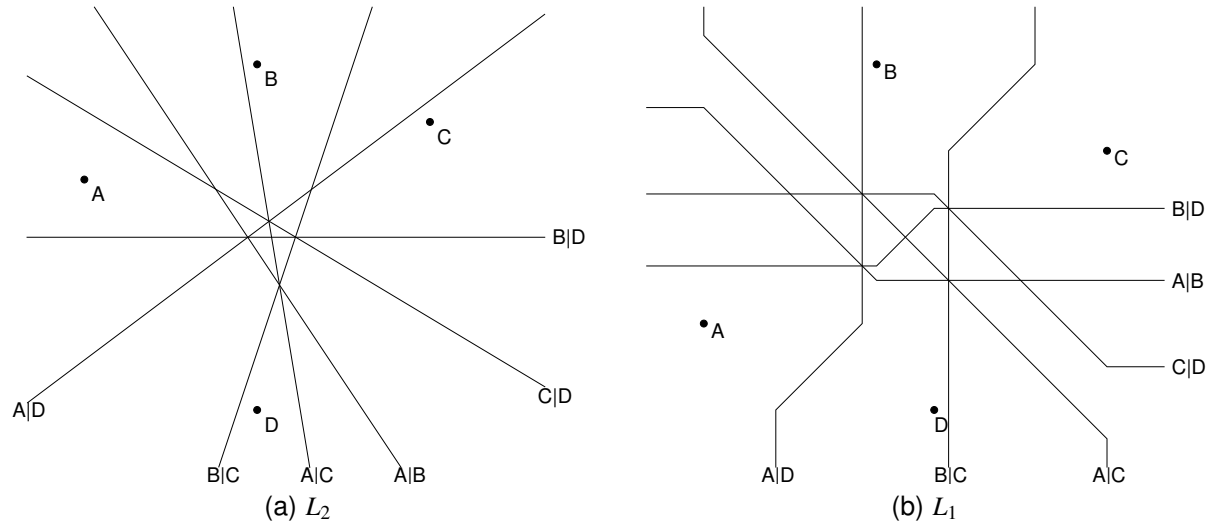

Figure 2. Bisectors of four points

weighted) tree metric, there can be at most $\binom{k}{2}+1$ distinct distance permutations.

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 Thm. 1 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 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.

3. 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 $x = \langle x_1, x_2, \ldots, x_n \rangle$ and $y = \langle y_1, y_2, \ldots, y_n \rangle$ by $d(x,y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p}$ for real $p \geq 1$ or $d(x,y) = \max_{i=1}^{n} |x_i - y_i|$ for $p = \infty$. These spaces are a simple generalization 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. All permutations are forced to occur inside a small sphere centred on the origin, giving the following theorem.

Theorem 2. 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 $k \leq d+1$. That is, $N_{d,p}(k) = k!$ for $d \geq k-1$ and any $p \geq 1$.

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 $d, m > 0$ [15]. His proof is an induction that follows the structure of the recurrence relation: when we add the m-th 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 $S_d(m) = \Theta(m^d)$ [15].

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 $N_{d,2}(k) = S_d\left(\binom{k}{2}\right)$. Since the bisectors are not in gen-

eral 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 Thm. 4, but first we give an exact result for the Euclidean case.

Theorem 3. In d-dimensional Euclidean space,

$$N_{0,2}(k) = N_{d,2}(k) = 1 \tag{1}$$
$$N_{d,2}(k) = N_{d,2}(k-1) + (k-1)N_{d-1,2}(k-1)\,. \tag{2}$$

Numerical results are shown in Tbl. 1. Note the factorials that appear in the lower triangle, corresponding to Thm. 2. For the one-dimensional case, the formula reduces to $\binom{k}{2}+1$, which is equal to the value for tree metrics from Thm. 1. The proof of Thm. 3 takes the same general approach used by Price [15]. 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 Thm. 3. Bounds on $N_{d,2}(k)$ then follow by induction:

Corollary 1. The function $N_{d,2}(k)$ satisfies:

$$N_{d,2}(k) \leq k^{2d} \tag{3}$$
$$N_{d,2}(k) = \frac{k^{2d}}{2^d d!} + o(k^{2d})\,. \tag{4}$$

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

With other L_p metrics, the situation is more complicated. Consider the two-dimensional L_1 case shown in Fig. 2(b). 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. 2(b), 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 4. The function $N_{d,p}(k)$ satisfies:

$$N_{d,1}(k) = O(2^{2d^2}k^{2d}) \tag{5}$$
$$N_{d,2}(k) = O(k^{2d}) \tag{6}$$
$$N_{d,\infty}(k) = O(2^{2d}d^{2d}k^{2d})\,. \tag{7}$$

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

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. The practical consequence 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.

Experiments with interactive computer graphics raise the question of whether non-Euclidean L_p metrics ever actually give more permutations than the Euclidean bound; it was not easy to find four sites to give 18 permutations for Fig. 2(b), and we could not find any configuration with more than 18. As the results in the next section show, it is in fact possible for L_1 vectors to exceed the limit for L_2.

4. 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 uniformly, the number of distance permutations actually occurring may be significantly less than

	k:										
	2	3	4	5	6	7	8	9	10	11	12
d: 1	2	4	7	11	16	22	29	37	46	56	67
2	2	6	18	46	101	197	351	583	916	1376	1992
3	2	6	24	96	326	932	2311	5119	10366	19526	34662
4	2	6	24	120	600	2556	9080	27568	73639	177299	392085
5	2	6	24	120	720	4320	22212	94852	342964	1079354	3029643
6	2	6	24	120	720	5040	35280	212976	1066644	4496284	16369178
7	2	6	24	120	720	5040	40320	322560	2239344	12905784	62364908
8	2	6	24	120	720	5040	40320	362880	3265920	25659360	167622984
9	2	6	24	120	720	5040	40320	362880	3628800	36288000	318540960
10	2	6	24	120	720	5040	40320	362880	3628800	39916800	439084800

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

the theoretical maximum. As seen in Fig. 2, some of the cells in the generalized Voronoi diagram are much smaller than others, and may not contain any objects in a real database.

To examine these issues, we implemented distance permutations for the SISAP library of Figueroa, Navarro, and Chávez [7], 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`. Source code (in C, with the same distribution terms as the original library) is available.

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 sample databases are shown in Tbl. 2. We show results for vectors (10^6 uniformly chosen from the unit cube) in Tbl. 3. Because the result for vectors depends on the random choice of sites, we ran each vector experiment 20 times, and show both the mean and maximum number of distance permutations observed, for selected values of k, the number of sites.

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, out of the 479001600 general permutations of 12 objects. To some extent this can be explained by the small number of points in the database; but the other sample databases also show small numbers of distance permutations. These results suggest that an even greater space saving is possible, but also that the distance permutations do not contain much information: the search algorithm can only benefit to a limited degree from using them.

By comparing numbers from Tbl. 2 with the theoretical values for Euclidean spaces in Tbl. 1, we see that `colors` has a few more distance permutations than would a Euclidean space with two dimensions. The `nasa` database has a few more distance permutations than a Euclidean space with three dimensions, ignoring the values for $k > 10$ 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 spaces with up to four dimensions. And the `listeria` database, despite having plenty of points, seems equivalent to a Euclidean space with between one and two dimensions. In this way we can characterize the dimensionality of a database in a highly general way.

Comparison to the intrinsic dimensionality ρ, defined by Chávez and Navarro as mean squared divded by twice the variance of distance between two random points [5], seems natural but may not be meaningful. The intrinsic dimensionality depends heavily on the probability distribution [19], 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 exist. Nonetheless, we give experimental ρ values based on the assumption of choosing points uniformly from the databases. As the tables show, databases with larger ρ tend to have more distance permutations.

A notable result not shown in Tbl. 3 is that in three-dimensional L_1 space, the experiment found a database and choice of five sites giving 108 distinct dis-

database	n	ρ	k: 3	4	5	6	7	8	9	10	11	12
Dutch	229328	7.159	6	24	119	577	2693	11566	34954	74954	116817	163129
English	69069	8.492	6	24	120	645	2211	7140	16212	28271	38289	45744
French	138257	10.510	6	24	118	475	2163	8118	19785	35903	58453	81006
German	75086	7.383	6	24	119	517	1639	4839	10154	19489	30347	43208
Italian	116879	10.436	6	24	120	653	3103	10872	27843	45754	71921	90316
Norwegian	85637	5.503	6	24	118	632	2530	7594	15147	25872	42992	57988
Spanish	86061	8.722	6	24	118	598	2048	5428	13357	23157	39443	54628
listeria	20660	0.894	4	11	19	29	49	85	206	510	952	1145
long	1265	2.603	5	10	22	47	51	98	114	163	252	261
short	25276	808.739	6	24	111	508	2104	6993	13792	20223	23102	23940
colors	112544	2.745	6	18	44	96	200	365	796	1563	2800	4408
nasa	40150	5.186	6	24	115	530	1820	3792	7577	13243	19066	24154

Table 2. Number of distance permutations for sample databases.

			mean perms			max perms		
	d	ρ	$k=4$	8	12	$k=4$	8	12
L_1	1	1.00	7.00	29.00	67.00	7	29	67
	2	2.00	14.65	268.45	1398.20	18	290	1532
	3	3.00	20.85	1405.20	16143.70	24	1804	18239
	4	4.00	23.95	4705.35	82253.85	24	5663	94537
	5	5.00	24.00	10390.85	220231.20	24	13573	258874
	6	6.00	24.00	16073.50	394466.85	24	20234	471375
	7	7.00	24.00	20811.65	569807.35	24	27824	653015
	8	8.00	24.00	26999.10	728040.20	24	33637	770929
	9	9.00	24.00	30309.60	809393.30	24	37198	845181
	10	10.00	24.00	30715.55	884013.40	24	35698	917237
L_2	1	1.00	7.00	28.95	67.00	7	29	67
	2	2.21	15.25	268.05	1440.15	18	298	1539
	3	3.52	19.60	1332.75	14584.20	24	1568	15929
	4	4.88	22.70	4214.20	67179.40	24	5079	75850
	5	6.27	23.90	7515.05	182253.50	24	10471	208301
	6	7.68	23.75	13824.25	348609.25	24	18693	402685
	7	9.09	23.95	17349.30	502957.40	24	23944	613857
	8	10.50	23.95	20244.80	657013.80	24	34866	796775
	9	11.92	24.00	21936.45	730146.10	24	28635	851775
	10	13.35	24.00	25562.25	815217.05	24	33097	905490
L_∞	1	1.00	7.00	29.00	67.00	7	29	67
	2	2.23	13.15	233.15	1314.10	18	278	1485
	3	3.59	18.05	1219.75	13152.25	24	1712	16162
	4	5.05	23.50	3664.60	54838.10	24	4912	70354
	5	6.58	22.80	6488.30	150360.55	24	12566	213951
	6	8.17	23.35	11314.00	265706.25	24	17837	352150
	7	9.80	23.70	14384.65	357331.00	24	23983	466484
	8	11.47	24.00	15433.55	496952.75	24	26906	610841
	9	13.17	24.00	18494.20	569572.75	24	27160	714881
	10	14.90	24.00	22415.00	648613.15	24	34281	770769

Table 3. Number of distance permutations for random vectors.

tance permutations, exceeding the limit of 96 for three-dimensional Euclidean space. Therefore the hypothesis that the Euclidean limit applies to all L_p spaces is false. The exceptional sites are:

$$\begin{aligned} x_1 &= (0.205281, 0.621547, 0.332507) \\ x_2 &= (0.053421, 0.344351, 0.260859) \\ x_3 &= (0.418166, 0.207143, 0.119789) \\ x_4 &= (0.735218, 0.653301, 0.650154) \\ x_5 &= (0.527133, 0.814207, 0.704307)\,. \end{aligned} \tag{8}$$

5. Conclusion and Open Problems

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 [7], 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. The same questions in other spaces remain open problems.

References

[1] F. Aurenhammer. Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Computing Surveys, Sept 1991*, 23(3), 1991.

[2] Y. Bartal. Probabilistic approximations of metric spaces and its algorithmic applications. In *37th Annual Symposium on Foundations of Computer Science (FOCS'96)*, pages 184–193. IEEE, 1996.

[3] A. Björner, M. Las Vergnas, B. Sturmfels, N. White, and G. M. Ziegler. *Oriented Matroids*. Cambridge University Press, second edition, 1999.

[4] E. Chávez, K. Figueroa, and G. Navarro. Proximity searching in high dimensional spaces with a proximity preserving order. In A. Gelbukh, Á. de Albornoz, and H. Terashima-Marín, editors, *4th Mexican International Conference on Artificial Intelligence (MICAI'05)*, pages 405–414. Springer, 2005.

[5] E. Chávez and G. Navarro. Measuring the dimensionality of general metric spaces. Technical Report TR/DCC-00-1, Department of Computer Science, University of Chile, 2000.

[6] K. Figueroa, E. Chávez, G. Navarro, and R. Paredes. On the least cost for proximity searching in metric spaces. In C. Àlvarez and M. Serna, editors, *Experimental and Efficient Algorithms: 5th International Workshop (WEA'06)*, pages 279–290. Springer, 2006.

[7] K. Figueroa, G. Navarro, and E. Chávez. *Metric Spaces Library*. Online `http://www.sisap.org/?f=library`. Accessed November 24, 2007.

[8] B. Grünbaum. *Arrangements and Spreads*. Number 10 in Conference Board of the Mathematical Sciences Regional Conference Series in Mathematics. American Mathematical Society, Providence, June 1971.

[9] C. Icking, R. Klein, N.-M. Lê, and L. Ma. Convex distance functions in 3-space are different. *Fundamenta Informaticae*, 22(4):331–352, 1995.

[10] C. Icking, R. Klein, N.-M. Lê, L. Ma, and F. Santos. On bisectors for convex distance functions in 3-space. In *11th Canadian Conference on Computational Geometry (CCCG'99)*, pages 119–123. University of British Columbia, 1999.

[11] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, Nov. 2004.

[12] A. Mandel. *Topology of Oriented Matroids*. PhD thesis, University of Waterloo, 1982.

[13] L. Micó, J. Oncina, and 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):9–17, 1994.

[14] R. Paredes and E. Chávez. Using the k-nearest neighbor graph for proximity searching in metric spaces. In M. P. Consens and G. Navarro, editors, *String Processing and Information Retrieval, 12th International Conference (SPIRE'05)*, pages 127–138. Springer, 2005.

[15] D. J. Price. Some unusual series occurring in n-dimensional geometry. *The Mathematical Gazette*, 30:149–150, 1946.

[16] E. V. Ruiz. An algorithm for finding nearest neighbors in (approximately) constant time. *Pattern Recognition Letters*, 4:145–157, 1986.

[17] M. Sahlgren. *The Word-Space Model*. PhD thesis, Stockholm University, 2006. Online `http://www.sics.se/~mange/TheWordSpaceModel.pdf`.

[18] F. Santos. On Delaunay oriented matroids for convex distance functions. *Discrete & Computational Geometry*, 16(2):197–210, 1996.

[19] M. Skala. Measuring the difficulty of distance-based indexing. In M. P. Consens and G. Navarro, editors, *String Processing and Information Retrieval: 12th Internatio nal Conference (SPIRE'05)*, pages 103–114. Springer, 2005.

[20] J. K. Uhlmann. Satisfying general proximity/similarity queries with metric trees. *Information Processing Letters*, 40:175–179, Nov. 25, 1991.

[21] 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)*, pages 311–321. ACM/SIAM, 1993.