Aspects of Metric Spaces in Computation

by

Matthew Adam Skala

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2008

AUTHOR'S DECLARATION FOR ELECTRONIC SUBMISSION OF A THESIS

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Matthew Adam Skala

Abstract

Metric spaces, which generalise the properties of commonly-encountered physical and abstract spaces into a mathematical framework, frequently occur in computer science applications. Three major kinds of questions about metric spaces are considered here: the intrinsic dimensionality of a distribution, the maximum number of distance permutations, and the difficulty of reverse similarity search. Intrinsic dimensionality measures the tendency for points to be equidistant, which is diagnostic of high-dimensional spaces. Distance permutations describe the order in which a set of fixed sites appears while moving away from a chosen point; the number of distinct permutations determines the amount of storage space required by some kinds of indexing data structure. Reverse similarity search problems are constraint satisfaction problems derived from distance-based index structures. Their difficulty reveals details of the structure of the space. Theoretical and experimental results are given for these three questions in a wide range of metric spaces, with commentary on the consequences for computer science applications and additional related results where appropriate.

Acknowledgements

This work for supported by an NSERC Postgraduate Scholarship during the first two and a half years. My thanks to all the usual suspects: my supervisor Ian Munro for his support and patience; my family members for offering shoulders to cry on; and the library and other staff at the University of Waterloo for providing an environment where I could get my work done. Thanks also to the good people of CTRL-A, Infinite Circle, HealthDoc/Inkpot, #ook, and #nerdsholm for helping me stay sane through what turned out to be a much longer and more stressful program than I ever expected.

Nou paha e ka inoa
E kaʻikaʻikū ana
A kau i ka nuku
E hapahapai aʻe.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction (0,1)

The idea of objects existing in some sort of space is fundamental to human beings' understanding of the universe. Not only are real-life phenomena intimately connected to the space-time described by physics, but abstract concepts are routinely imagined as existing in a conceptual space. This imagination is implicit in language that uses geometric and spatial terms to describe things other than physical space: we may speak of a discussion going off on a tangent, friends being close, a joke that goes too far, or ideas being on one or the other hand. Psychological theories posit that an association between locations and directions in physical space, and abstract ideas in our minds, may be fundamental to cognition [131, 163].

The spatial metaphor is also fundamental to many computer applications, especially in the realm (which is another spatial term) of databases. Objects in a database may be imagined as points in a space, which also imposes a spatial meaning on queries. Typically the answers to a query will all be clustered in a definite region of the space; and the query may even be defined in terms of a region of the space. This dissertation presents results on several computer science problems related to searching in abstract spaces.

We primarily consider three basic questions: intrinsic dimensionality, number of distance permutations, and the difficulty of reverse similarity search. In this introductory chapter we present general metric spaces, and each of the problems, with comments on the history of relevant previous work. There follow chapters for different kinds of metric spaces, and the answers to our questions for each of them. The discussion of real vectors is split into three chapters to keep their lengths manageable and resolve interdependencies between the real vector and Hamming string results.[1] Relevant previous work for the individual spaces is

[1] Chapter 8 depends on Chapter 5 which depends on Chapters 2 and 3.

covered in the respective spaces' chapters, along with some results on other problems specific to particular spaces. We close with some notes on other results of interest.

1.1 Metric spaces and similarity search

First of all, what is an abstract space of the kind we are studying? Many kinds of space exist that generalise in different ways the familiar physical space of human reality. The present work is primarily concerned with metric spaces, which can be glossed as spaces where there are points with distances between them, and the distances are reasonably well-behaved.

Definition 1.1
Let S and $d : S \times S \to \mathbb{R}$ be a function that may or may not satisfy these properties, for all $x, y, z \in S$:

$$d(x,y) \geq 0 \tag{1.1}$$
$$d(x,x) = 0 \tag{1.2}$$
$$d(x,y) \neq 0 \text{ if } x \neq y \tag{1.3}$$
$$d(x,y) = d(y,x) \tag{1.4}$$
$$d(x,z) \leq d(x,y) + d(y,z). \tag{1.5}$$

triangle inequality

distance function
point
metric
metric space
pseudometric
quasimetric
semimetric
prametric

The property (1.5), which is of particular importance to our work, is called the *triangle inequality*.

Any function $d : S \times S \to \mathbb{R}$ used to express some concept of distance will be called a *distance function*. Elements of the set S will be called *points*. If the distance function satisfies all the above properties, then it is a *metric* and the pair $\langle S, d \rangle$ is a *metric space*. Other kinds of distance functions are defined by relaxing one or more of the properties: without (1.3), it is a *pseudometric*; without (1.4), a *quasimetric*; without (1.5), a *semimetric*; and without any of those (leaving only (1.1) and (1.2)), a *prametric*[2] [10, 121, 200].

1.1.1 Metric spaces

Most properties of metric spaces are ones we would intuitively associate with travel among points: a journey cannot take less than no distance, two points with

[2]The definition of notation such as $\mathbb{R}$ is deferred to Section 1.2 to avoid interrupting the general introduction and motivation; nothing very unusual will be used before then.

zero distance between them must be identical, and a journey (along the same route) in either direction must be of the same length. The triangle inequality is the real key to the definition of metric spaces: it says that stopping at a third point on the trip from one to another cannot ever result in a shorter trip than just going directly between the two points. That property is common to all the familiar kinds of spaces we might consider. It is strong enough that we can use it to infer useful things about points based on their distances from each other, while still being weak enough to permit the existence of a rich variety of metric spaces. A few examples follow.

Example 1.2
Ordinary physical space, with distances measured as by a ruler, is a metric space: there are points, there is a distance between any two points, all the distances are nonnegative, distance is the same in either direction, two points have distance zero if and only if they are the same point, and the triangle inequality applies.

Example 1.3
Any set S is a metric space, using the *equality metric* defined by $d(x,y) = 0$ if and only if $x = y$, $d(x,y) = 1$ otherwise. It is easy to verify that this satisfies Definition 1.1. Such a space is called a *discrete space*.

equality metric

discrete space

Example 1.4
The set of all 43252003274489856000 legal configurations [28, page 761] of a Rubik's Cube is a metric space, with the distance between two configurations being the minimum number of moves required to transform one to the other. The maximum distance between any two points in this space is known to be at most 26 [130]. Berlekamp, Conway, and Guy give a lower bound of 18 [28, page 767] and, in a result apparently published only on an electronic mailing list, Reid gives a lower bound of 20 [174]. Hofstadter also gives some commentary of interest on the Cube and variations [103, pages 301–363]. Note that legal configurations are configurations reachable by twisting the starting "solved" configuration. Configurations only reachable by taking apart and reassembling the Cube do not count unless one also counts the disassembly-and-reassembly operation as a move, in which case this becomes just another (large, finite) discrete space.

Fréchet introduced spaces like these in his 1906 thesis on functional analysis, defining spaces in which a *voisinage* ("vicinity") function, which he notated as (A,B), had the properties that $(A,B) = (B,A) \geq 0$, $(A,B) = 0$ if and only if $A = B$, (A,B) tended to zero if A and B tended to each other, and a relaxed form of the

triangle inequality held: if $(A,B) \leq \epsilon$ and $(B,C) \leq \epsilon$, then $(A,C) \leq f(\epsilon)$ where $\lim_{\epsilon \to 0^+} f(\epsilon) = 0$ [78, page 18]. We note that right from the start, the study of metric spaces has been combined with the study of relaxed versions like this one, which is clearly designed for proving limits but does not quite restrict spaces as far as the metric spaces of Definition 1.1.

Fréchet went on to define the *écart des deux éléments* ("variation of two elements") to satisfy the full triangle inequality $(A,B) \leq (A,C) + (C,B)$, making it a metric under the modern definition [78, page 30]. Hausdorff later gave such spaces their current name of *metrische Räume*, that is, "metric spaces" [99, page 211] [100]. Hausdorff used both overbar notation (as in $\overline{xz} \leq \overline{xy} + \overline{yz}$) and the function notation we prefer ($d(A,C) \leq d(A,B) + d(B,C)$ [99, page 291]). Around the time of Fréchet's work, Minkowski was developing a geometry for the unified space-time entity postulated by Lorentz and Einstein, based on an innovative distance function that allowed negative values and used them to describe the distinction between space-like and time-like directions [154]. Minkowski's name is now applied to a class of metrics on real vectors which we discuss in detail in Chapters 2, 3, and 8.

Metric spaces generalise an important and useful concept, and so they have applications in a wide variety of mathematical fields. In topology, every metric defines a unique topology for its space, and such spaces often have interesting or useful topological properties [10, 200]. Differential geometry considers analysis in metric spaces [209]. In coding theory, the Hamming metric (subject of Chapter 5) is central to the study of channel errors [169]. Linear algebra studies vector norms, which are related to inner products and also generate metrics on the vectors [102].

Practical metric space applications arise when computation is applied to problems that involve a concept of distance among things. Geographic information systems are entirely concerned with spatial questions, expressed not only in the three-dimensional Euclidean space of ordinary experience, but also more abstract conceptual spaces describing things like travel time between locations [77]. Database applications use metric spaces not only for geographic questions but also anywhere that similarities and differences between data objects are relevant. As a result, there is a massive literature on representing objects as points in metric spaces [142, 178], transforming the spaces for ease of processing [56], and especially on searching in metric spaces. Multiple surveys, software packages, and conferences cover the question of metric space searching [42, 71, 101, 220].

Note 1.5

The space for a given application, including both the points and the metric, is typically imposed by the application. We do not get to choose a nice metric. At best we might try to substitute a convenient metric

for the application's metric and then argue that the consequences of the substitution are not too bad. Also, the metric for a space may be expensive to compute. As a result, it is often an important goal for data structures to minimise the number of times the metric must be computed, even if that means doing more work elsewhere.

The issue of not being able to choose the metric is significant because it creates a need for data structures and theoretical work applicable to *general* metric spaces. In general, we must assume that the metric will be expensive and badly-behaved, with the properties guaranteed by Definition 1.1 but not necessarily any others.

Example 1.6
Local descriptor techniques are successful at detecting similar images, or objects in common between images, despite changes in lighting, movement, image compression, and other transformations [7, 142]. Comparing two images with local descriptors involves scanning each image for certain features to make a list of descriptors—an expensive operation which can at least be done as a precomputation, roughly analogous to a dimension reduction—and then searching for similar descriptors in common between the two images. In the case of the SIFT technique described by Lowe, there are typically a few hundred descriptors per image, each a 128-component vector [142]. The search for matching descriptors is itself much like a similarity search problem, but it must be done just to compare a single pair of images. Any practical data structure for searching images based on local descriptors must somehow avoid doing a linear number of full image-to-image distance measurements.

1.1.2 Other abstract spaces

Metric spaces are not the only way to formalise these kinds of studies. The most natural distance measure for a given space may not be a metric. All the relaxed versions mentioned in Definition 1.1 see some amount of use. The compression distance is one example of a non-metric distance function of interest in bioinformatics applications. It describes the amount of information in one string conditional on the other, as measured by a data compression program. Assuming optimal compression (taking each string to a compressed length equal to its Kolmogorov complexity [141]) the compression distance would be a metric, but since Kolmogorov complexity is uncomputable, real-life data compression programs are used to approximate it instead. The compression programs may give far from ideal performance [91] and their estimates do not necessarily obey the

almost metric

properties of a metric (in particular, symmetry and the triangle inequality) [25, 140]. Sahinalp and others have studied *almost metrics*, in which the triangle inequality holds to within a constant factor; they show that compression distance obeys that relaxed definition, and that some data structures designed for metric spaces are still useful with almost metrics [177].

Another way to describe the relationship between two points is with a measure of similarity rather than distance, typically on a scale where the measure takes a finite maximum value for identical points and a minimum value for points that are unrelated to each other. With two Euclidean vectors $\mathbf{x}$ and $\mathbf{y}$, the quantity $\mathbf{u}\cdot\mathbf{v}/|\mathbf{u}||\mathbf{v}|$, which is equal to the cosine of the angle between the vectors, expresses similarity on a scale of -1 to 1. It takes the value ± 1 if one vector is a scalar multiple of the other (according to the sign of the scalar) and 0 if they are orthogonal. Correlation coefficients used in statistics express relations between variables on a similar scale of -1 to 1 [62, pages 215–217].

Similarity measures used in text processing applications include what has become known as the Dice coefficient, originally proposed for comparing biological species. It counts how many features (such as occurrence in a given sample location) the species have in common, normalising the result to be between 0 and 1; two species are considered similar if they tend to occur in the same locations [63]. Sokal and Sneath review that and other similarity coefficients used in biological taxonomy [198]; and Adamson and Boreham apply the Dice coefficient to similarity of strings, letting the features be presence or absence of two-letter substrings [2]. Other similarity measurements for strings and documents come from using different similarity coefficients, longer substrings, or words as features instead of pure substrings.

In applications like search engines, where a measurement of the relationship between documents is exposed to users, it may be easier for the users to understand a similarity measure on a finite scale than a distance measure with unknown or complicated units. Kondrak defines a function called n-gram similarity as a further development from the Dice coefficient, with n-gram distance as a modification of the similarity. He takes the position that the similarity view of this measure is "conceptually simpler than n-gram distance" [128]. We are not convinced there is a meaningful difference.

In a context like plagiarism detection where the interesting thing that can be said about two documents is where they are the same, not where they differ, a measure of similarity may be more natural. The MOSS plagiarism detection system, for instance, starts with a robust hash, formed by robustly selecting ordinary hashes of n-grams from the documents, and then expresses similarity between documents as the raw number of matching n-gram hashes. Schleimer, Wilkerson, and Aiken describe the system and report that MOSS users find a

sharp correlation between plagiarism and number of matches over a constant threshold, with the threshold dependent on the type of documents [183].

However, in this work we consider primarily a distance point of view, and primarily metric spaces in particular, because of the usefulness of strict properties like the triangle inequality that may be harder to define or use in a similarity context. Depending on the properties of the similarity, it may be possible to define a metric as some function of similarity, or similarity as some function of a metric, to create an equivalence between the two. Li and others nod to the similarity view by defining their "similarity metric" [140] to take values between 0 and 1 so that it can be easily converted to a similarity by subtracting it from 1. Any metric d can be converted to a similarity score on a scale of 0 to 1 by computing $1/(1+d)$; the result will be 1 for identical points and approach 0 for distant points.

1.1.3 Similarity search and geometry

Even when considered from a metric point of view, the two most common metric-space search problems are generally called *similarity search* problems. They are defined as follows. similarity search

Definition 1.7 (Range search)
Given a database of points in some metric space, a query point q, and a real $r > 0$, find all the points z in the database such that $d(q,z) \leq r$. range search

Definition 1.8 (k-Nearest Neighbour (kNN) search)
Given a database of points in some metric space, a query point q, and an integer $k > 0$, find the k points in the database nearest to q. kNN search

The simplest way to solve a similarity search problem would be to just compute all the distances from the query point to points in the database, in linear time. Algorithmic work on these problems focuses on doing precomputation to build an index data structure on which the searching operation can run more efficiently. Hjaltason and Samet review similarity search from a practical perspective [101] and Chávez and others review the subject from a theoretical perspective [42].

Some techniques for searching in metric spaces depend on the geometric properties of special spaces. In two and three dimensions, quadtrees [104, 211] and octrees [111] are popular in applications like graphics [75] and finite element analysis [172]. The obvious generalisation of this technique is seldom applied to higher dimensions, however, because each node requires space exponential in the number of dimensions. The kd-tree described by Bentley provides improves performance in higher dimensions from a similar technique by considering only one dimension per node so that the tree remains binary [26]. The R-tree of

Guttman [95] is similar, and has many variants, including $R*$-trees [23], R^+-trees [186], and SR-trees [119]. The hybrid tree of Chakrabarti and Mehrotra uses overlapping subtrees instead of a strict split, to guarantee balance properties [38]. The pyramid trees of Berchtold, Böhm, and Kriegel are specifically designed for high-dimensional vector spaces, making use of the geometry of such spaces (in particular, the tendency for one vector component to dominate the others) [27].

However, general metric spaces do not provide the geometry required by those techniques. For a general metric space with no other assumptions, it is necessary to use a *distance-based* approach that indexes points solely on the basis of their distance from each other. Burkhard and Keller [35] offered one of the first such index structures, now known as a BK-tree for their initials, in 1973. In a BK-tree, the metric is assumed to have a few discrete return values, each internal node contains a vantage point, and the subtrees correspond to the different values of the metric.

distance-based

1.1.4 VP-trees

Yianilos describes a VP-tree (for "vantage point"), which resembles a binary BK-tree with the metric values at each node simplified down to a binary threshold [217]. It also resembles the binary search trees widely used for sorted lookups in a single dimension [50, pages 244–280]. Each internal node contains a *vantage point* and two subtrees of points divided up according to their relationship to the vantage point. Instead of dividing points according to whether their key is greater than or less than the node's, as we would in a single-dimensional binary search tree, the VP-tree stores a radius in each node and the two subtrees correspond to points that are or are not within the radius; so the tree divides space according to spheres about the vantage points. The geometric situation at one node is shown in Figure 1.1.

vantage point

Definition 1.9

VP-tree

A *VP-tree* for a metric space S is a binary tree data structure in which each leaf stores a set of points and each internal node stores a point $v \in S$ and real radius $r \geq 0$, as well as its left and right child subtrees. At each internal node, all points appearing in the left subtree must be on or inside the sphere of radius r centred on v, and all points appearing in the right subtree must be outside that sphere.

For a balanced tree, the radius should be the median of distances from the vantage point among points in the tree. This approach necessitates calculating those median distances, making it not directly suitable for the dynamic applications served by single-dimensional balanced search tree structures, but it has the

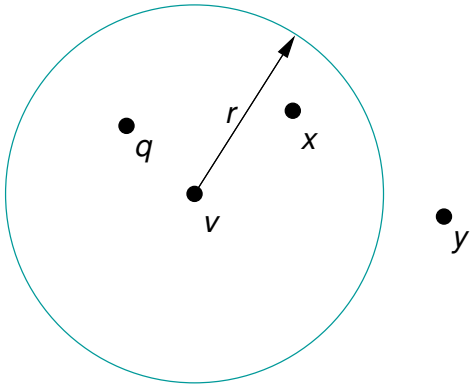


Figure 1.1: How the VP-tree divides space.

advantage of guaranteeing balance as long as the distribution of queries is close to the distribution of objects in the database. It also appears at first glance to be nicely efficient: just like a conventional low-dimensional binary search tree, there is one comparison to one vantage point made at each node.

Searching the VP-tree proceeds by descending through the nodes, using the triangle inequality and the inequalities defining the subtrees to prove bounds on how far the points in a subtree must be from the query. Then subtrees that provably cannot contain the query results can be pruned from further examination. Hjaltason and Samet describe this kind of algorithm in detail [101]; it applies to all space-dividing data structures in general, not just VP-trees.

Where v is the vantage point, for each x in the left subtree (within r distance of v) and each y in the right subtree (at least r distance from v), we can compute the distance $d(q,v)$ for a query point q and then the triangle inequality gives us these bounds, which are used to prune the search:

$$\begin{aligned} d(q,v) - r \leq & d(q,x) \leq d(q,v) + r \\ r - d(q,v) \leq & d(q,y)\,. \end{aligned}$$

1.1.5 GH-trees

The VP-tree has an apparent problem: spheres may be far from ideal shapes for dividing the search space. Intuitively, the problem is that a sphere's surface is curved: if our query point is outside the sphere (which happens half the time, assuming a balanced tree), then after removing the contents of the sphere from consideration, the remaining points may be near, far, or at about the same distance from the query point as were the points in the sphere; eliminating the

sphere tells us little about the distance to the remaining points from the query point.

The "generalised hyperplane" tree (*GH*-tree) introduced by Uhlmann attempts to divide space in a more useful way [205]. If we had to divide Euclidean space as neatly as possible, the obvious choice would be to use a hyperplane (that is, the constant-value set of a linear function). In general metric spaces we cannot define hyperplanes so easily, but Uhlmann describes a generalised hyperplane capturing one essential property of the Euclidean hyperplane. Given two points u and v, the generalised hyperplane between them is the set of all points in the space that are equidistant from u and v. The *GH*-tree, then, is a binary space-partitioning tree with a generalised hyperplane at each node.

Definition 1.10

GH-tree

A *GH-tree* for a metric space S is a binary tree data structure in which each leaf stores a set of points and each internal node stores two points $u, v \in S$, as well as its left and right child subtrees. Any point x appearing in a subtree must be in the left subtree if $d(u,x) \leq d(v,x)$ and in the right subtree otherwise.

generalised hyperplane

Figure 1.2 illustrates how a node of the *GH*-tree partitions space. The point e represents any arbitrary point on the *generalised hyperplane*, that is, equidistant from u and v. Suppose the query point q is, as shown, closer to u than v. Then because $d(u,e) = d(v,e)$, we have:

$$\begin{aligned} d(q,v) - d(q,u) &= d(q,v) - d(q,u) + d(u,e) - d(v,e) \\ &= (d(q,v) - d(v,e)) + (d(u,e) - d(q,u)) \\ &\leq 2d(q,e) \text{ (by the triangle inequality).} \end{aligned}$$

If we have found an x that is close to the query point, such that $d(q,x) < (d(q,v) - d(q,u))/2$, then we know that no point y on the other side of the generalised hyperplane can possibly be closer to q, and so in a simple nearest-neighbour search, we can prune that subtree. Similar pruning occurs in other kinds of similarity search on *GH*-trees.

Note 1.11

The *VP*- and *GH*-tree data structures are not covered further in this work. We give definitions and descriptions for them only to motivate similarity search and in particular our definitions of VPREVERSE and GHREVERSE (Definitions 1.27 and 1.29), which are constraint satisfaction problems formed by the constraints implicit in the data structures. But our results on VPREVERSE and GHREVERSE relate to those constraint satisfaction

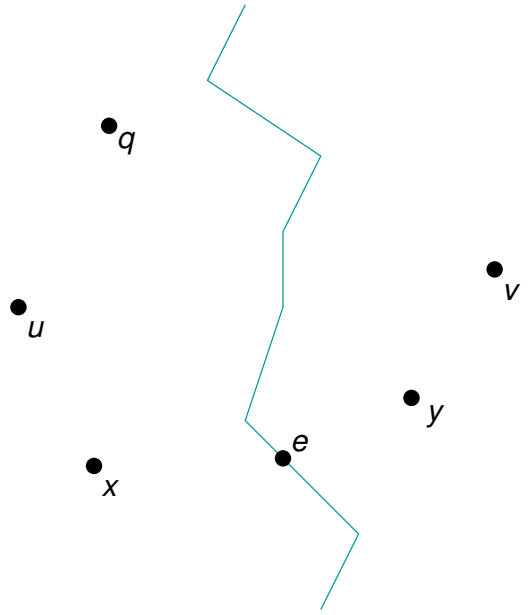


Figure 1.2: How the *GH*-tree divides space.

problems, not directly to the data structures. In particular, the tree metric spaces of Chapter 4 are not *VP*- or *GH*-trees.

1.1.6 Other data structures for similarity search

Instead of organising the database primarily into a tree structure and pruning the tree to narrow down the search, a data structure could exclude objects one at a time on the basis of some index information stored with each object. The amount of processing required to satisfy a query in such a structure might be linear, but if it means saving some computations of the metric, it can still provide an advantage. Since the metric may be expensive (Note 1.5), the usual cost model for these kinds of data structures counts only the number of invocations of the metric. Even a large amount of other data and computing on the side can be excused in the name of avoiding metric computations.

The AESA technique (Approximating and Eliminating Search Algorithm) of Vidal Ruiz carries that approach to an extreme: it precomputes and stores all the pairwise distances among database objects [176]. Then the distance from a query point to any object in the database can be used with the triangle inequality to exclude other objects as possible answers to the query. This approach works well in the sense of answering queries with very few distance computations;

however, it requires index space quadratic in the number of database objects and so becomes impractical for databases of any significant size. Shasha and Wang describe a technique that similarly keeps a quadratic-sized matrix of distances, but instead of precomputing them all, they start with lower bounds, initially very loose, and update the bounds with the triangle inequality as queries are applied and better estimates (or exact measurements) become available [187]. Further improvements on AESA are discussed in Section 1.4.

Paredes and Chávez describe a different approach to storing limited data: instead of storing the exact distances to a limited set of pivot elements, their k-nearest neighbour graph technique stores the identities, not the distances, of the k other database objects closest to each database object [164]. Search in such a database proceeds by heuristically applying rules to infer which objects could be in the result set based on as few distance computations as possible. Compare this approach to the pyramid-trees of Berchtold, Böhm, and Kriegel, which use the identity of the greatest-magnitude vector component to choose a one-dimensional tree to contain the object [27]. Both techniques depend on storing a clue as to which measurement or measurements of the object will be most useful in further evaluation.

The approach of storing some data about each point to eliminate points one at a time can also be combined with tree-based approaches. A typical example is the Geometric Near-neighbour Access Tree (GNAT) described by Brin [31]. In a GNAT, each internal node stores some number of vantage points, and subtrees descend from the node based on the nearest vantage point (as in a GH-tree generalised to k vantage points per node), but the internal nodes also store, for each subtree, its range of distances to each vantage point. Each internal node then resembles a miniature AESA data structure. The size of the node is $\Theta(k^2)$, but there are many more opportunities to prune subtrees than with a simple generalised GH-tree.

1.2 Notation and organisation

Important notational conventions will be described again when they are used, but we collect them here as well for ease of reference.

1.2.1 General mathematics

log
lg
$\mathbb{R}$
factorial
gamma function

We use log for the natural logarithm, base $e \approx 2.71828$, and lg for the base-2 logarithm. If $y = \log x$ then $x = e^y$ and $\lg x = y / \log 2$. The set of real numbers is denoted by $\mathbb{R}$. The *factorial* of an integer n is denoted by $n!$, and the *gamma function* (generalised factorial) of a real z by $\Gamma(z)$. For integers, $\Gamma(z) = (z-1)!$.

Note 1.12
The gamma function exists for complex z but we only use it on positive real z. It is well-behaved for such inputs. In particular, the value between positive integer inputs increases smoothly from one factorial to the next, and going half-way to the next positive integer multiplies the result by approximately the square root of the next integer. A stronger version of this property will be stated and used in the proof of Corollary 2.11.

The following definition gives three other extensions of the factorial concept which we use occasionally.

Definition 1.13
The *multinomial coefficient* for four terms is given by [85, 88, page 168] multinomial coefficient

$$\binom{n}{i,j,k,n-i-j-k} = \frac{n!}{i!j!k!(n-i-j-k)!}; \tag{1.6}$$

and the *double factorial* of an integer n is given by [9, pages 544–545] !!

$$n!! = \begin{cases} 1{\cdot}3{\cdot}5{\cdot}\cdots{\cdot}n & \text{odd } n > 0, \\ 2{\cdot}4{\cdot}6{\cdot}\cdots{\cdot}n & \text{even } n > 0, \\ 1 & n \in \{-1, 0\}\,. \end{cases} \tag{1.7}$$

Note that $n!!$ is quite different from $(n!)!$. Finally, $(x)_n$ denotes the *rising factorial*, or Pochhammer symbol, $x(x+1)\cdots(x+n+1)$. rising factorial

Note 1.14
As Weisstein describes, the standard notation for rising factorial is different in different fields; in combinatorics it is often denoted by $x^{(n)}$ whereas in the theory of special functions it is often denoted by $(x)_n$ [213]. We follow Abramowitz and Stegun in using $(x)_n$ [1, page 256], and we use it at all only in the following definition of hypergeometric functions.

Definition 1.15
As described by Oberhettinger [160], the Gaussian *hypergeometric function* ${}_2F_1(a,b;c;x)$ is given by hypergeometric function

$${}_2F_1(a,b;c;x) = \sum_{n=0}^{\infty} \frac{(a)_n (b)_n}{(c)_n n!} x^n\,. \tag{1.8}$$

That describes a power series in which the ratio between successive coefficients is a rational function (quadratic over quadratic) of the index.

O, o, Θ, Ω, ω

We use the standard asymptotic notation (O, o, Θ, Ω, ω) throughout. The case of ω is less common than the others and may be unfamiliar; note that $f(n) = \omega(g(n))$, read "$f(n)$ is little-omega of $g(n)$," if $f(n)$ is greater than any constant multiple of $g(n)$ for sufficiently large n. Just as o is a strict version of O, ω is a strict version of Ω. It is also convenient to have right-arrow for convergence to within lower-order terms.

Definition 1.16

If

$$\lim_{n\to+\infty} \frac{f(n)}{g(n)} = 1 \tag{1.9}$$

$\rightarrow$

then we write $f(n) \to g(n)$. Note that $f(n) \to g(n)$ is a stronger statement than $f(n) = \Theta(g(n))$ because it implies that the constant is 1.

Although the need to remain consistent with other authors sometimes forces exceptions to this policy, we attempt to follow a consistent practice for parentheses and brackets. Angle brackets are for sequences with a specific order; so

$\langle\cdot\rangle$

$\langle 1, 2, 3\rangle$ is a vector with three components, not equal to $\langle 3, 2, 1\rangle$. Curly braces are

$\{\cdot\}$

for sets; so $\{1, 2, 3\} = \{3, 2, 1\}$ is a set. Round parentheses and square brackets

$(\cdot)$

are for grouping operations, as in $[(1 + 2) \cdot 3] = 9$, and occasionally to denote the

$[\cdot]$

open and closed bounds of intervals, as in $\frac{1}{2} \in [0, 1)$. Special brackets may be used for the arguments of some functions and function-like operators, as a clue to the special types of the arguments; for instance, $\max\{S\}$ operates on a set S, and $\mathrm{E}[X]$ operates on a random variable X. We follow the general rule of big letters for big ideas: a real x might be a component in a vector $\mathbf{x}$ which is in a set X which is part of a class $\mathcal{X}$, although we seldom use that many levels of abstraction simultaneously. To reduce confusion among things made of smaller things, we

element
component
letter

note that sets contain *elements*, vectors contain *components*, and strings contain *letters*.

1.2.2 Probability and statistics

probability
expectation
variance

We write $\Pr[\mathcal{E}]$ for the *probability* of an event $\mathcal{E}$, and $\mathrm{E}[X]$ and $\mathrm{V}[X]$ for the *expectation* and *variance* of a random variable X, respectively. Where $\mathcal{X}$ is the set of values X can assume, we have

$$\mathrm{E}[X] = \sum_{x\in\mathcal{X}} x \Pr[X = x] \tag{1.10}$$

for a discrete random variable, or where $f(x)$ is the probability density function of a continuous random variable, then

$$\mathrm{E}[X] = \int_{\mathcal{X}} x f(x)\, dx\,. \tag{1.11}$$

Then variance can be defined as

$$\mathrm{V}[X] = \mathrm{E}[(X - \mathrm{E}[X])^2]\,. \tag{1.12}$$

The following computational formula [62, page 110] is invaluable when dealing with variance:

$$\mathrm{V}[X] = \mathrm{E}[X^2] - \mathrm{E}^2[X]\,. \tag{1.13}$$

Following the notation used by Arnold, Balakrishnan, and Nagaraja [13], we write $X \overset{d}{=} Y$ if X and Y are identically distributed, $X(n) \xrightarrow{d} Y$ if the distribution of $X(n)$ converges to the distribution of Y as n goes to positive infinity, and $X(n) \overset{d}{\longleftrightarrow} Y(n)$ if the distributions of both X and Y depend on n and converge to each other. $\overset{d}{=}$ $\xrightarrow{d}$ $\overset{d}{\longleftrightarrow}$

Especially when discussing the L_∞ metric, which is defined in terms of the maximum function, it is convenient to define for any real random variate Z random variates $\max^{(k)}\{Z\}$ (read "max over k from Z") and $\min^{(k)}\{Z\}$ ("min over k from Z") realized as random variables $\max_i^{(k)}\{Z\}$. and $\min_i^{(k)}\{Z\}$ respectively. Each $\max_i^{(k)}\{Z\}$ is the maximum, and each $\min_i^{(k)}\{Z\}$ the minimum, of k random variables from Z. $\max^{(k)}\{Z\}$ $\min^{(k)}\{Z\}$

1.2.3 Vectors

Points in general spaces are denoted by italic letters like other variables, such as x, y, z. For points as real vectors in particular, we use bold like $\mathbf{x}, \mathbf{y}, \mathbf{z}$, with subscripted italics like x_1, x_2, x_3 for individual components of a vector. As mentioned earlier, we use angle brackets to enclose the components of a vector when writing the vector out explicitly, as in $\langle 1, 2, 3\rangle$. In a few cases subscripted bold is used for individual vectors within a family of vectors; for instance, the $\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3$ defined in Chapter 8 are unit vectors along the first three axes, not components of a vector $\mathbf{u}$. Indices start from 1. The zero vector is represented by $\mathbf{0}$. zero vector ($\mathbf{0}$)

The Minkowski L_p metrics are the subject of Chapters 2, 3, and 8 and discussed in detail there. The basic definition is that where $\mathbf{x} = \langle x_1, x_2, \ldots, x_n\rangle, \mathbf{y} = \langle y_1, y_2, \ldots, y_n\rangle$, the L_p *metric* $d_p(\mathbf{x}, \mathbf{y})$ is defined by L_p metric

$$d_p(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p} \tag{1.14}$$

for real $p \geq 1$ or

$$d_\infty(\mathbf{x}, \mathbf{y}) = \max_{i=1}^{n} |x_i - y_i| \tag{1.15}$$

for $p = \infty$.

1.2.4 Strings

string
alphabet
Σ
α
binary
empty string
letter

Many of our spaces have *strings* over some *alphabet* as their points. We generally use Σ to represent the alphabet and α as an arbitrary element of Σ. *Binary* strings are strings for which $\Sigma = \{0,1\}$. The *empty string* is denoted by λ. The elements of a string or an alphabet are called *letters* even if we happen to denote them with numerals.[3]

concatenation
repetition

Definition 1.17
In the context of strings, juxtaposition denotes *concatenation* and exponentiation denotes *repetition*. For instance, if $x = 100$ and $y = 011$ then $xy = 100011$; and the notation 1^3 refers to the string 111, not the number 1 (one cubed). Similarly, $\alpha^0 = \lambda$. The metaphor is that concatenation is like multiplication.

By choosing any interval of the indices in a string we can extract a substring, and by choosing any subset of the indices we can extract a subsequence. These two concepts are similar, but the distinction is important.

substring
subsequence

Note 1.18
A *substring* is contiguous; a *subsequence* is not necessarily contiguous. All substrings are subsequences but not all subsequences are substrings. Thus BANANA has AAA as a subsequence but not as a substring, whereas it has NAN as both.

$\text{lcs}(x, y)$

The notation $\text{lcs}(x, y)$ denotes the longest common contiguous substring between x and y, a concept used frequently in Chapter 7. The longest common possibly-discontiguous subsequence is also an important concept, but we do not define a specific notation for it.

1.2.5 Computational complexity

$\mathcal{P}$
$\mathcal{NP}$
$\mathcal{NPC}$
$\mathcal{UP}$

We use $\mathcal{P}$ to denote the class of polynomial-time decision problems: problems for which a yes- or no-instance can be recognized in polynomial time by a deterministic universal Turing machine. Similarly, $\mathcal{NP}$ is the class of nondeterministic polynomial-time problems; problems in $\mathcal{NP}$ have polynomial-sized certificates verifiable in polynomial time. An $\mathcal{NP}$-hard problem is one to which any problem in $\mathcal{NP}$ can be reduced in polynomial time, and a problem that is both $\mathcal{NP}$-hard and in $\mathcal{NP}$ is in $\mathcal{NPC}$, the class of $\mathcal{NP}$-complete problems. Finally, we use $\mathcal{UP}$ to denote the class of unique-certificate polynomial-time problems. These abbreviations are standard, but mentioned here for reference.

[3] Letters like 0 and 1 are printed in a different typeface from numbers like 0 and 1, but it should also be clear from context which one is meant.

1.3 Dimensionality measurement

The familiar space of human experience is basically Euclidean space with three dimensions. Any point can be uniquely identified with three real numbers. Present-day models of physics allow for physical space-time to be non-Euclidean, and to have more dimensions, as many as 26 in the case of bosonic string theory [221]. More abstract spaces used in linear algebra also associate points with tuples called vectors, of real or perhaps complex numbers, with defined rules for measuring distances among points. In a linear-algebra vector space, the number of components in each vector is an intrinsic property of the space, invariant over multiple representations of the space. For instance, points in Euclidean three-space have three coordinates each no matter whether they are represented with Cartesian, cylindrical, or spherical coordinates. The number three is the number of dimensions, or *dimensionality* of the space, and differentiates it from (for instance) the two-dimensional Euclidean plane.

dimensionality

At first glance it appears that the number of dimensions of a space is simply the number of components in the vectors that describe points. That is an inadequate definition because it is too closely tied to the vector representation. Not all spaces are naturally represented as vectors in the first place. For instance, in the Rubik's Cube space of Example 1.4, it would seem more natural to represent a point as the permutation between its cubelet positions and the cubelet positions of the "solved" state, with some information about rotation of cubelets. We could write that as a list of numbers, but how long the list would be would depend on how we chose to represent a concept like "the red and green side cubelet has been moved down and to the right by one quarter-turn."

It would be preferable that the metric defined on the original space correspond to some metric appropriate to vectors; but with any naive translation from Cube positions to vectors, the fewest-moves metric between two vector-represented Cube positions would end up being something along the lines of "first, transform the vectors back to a more natural representation of Cube positions; then count the minimal number of moves..." Also, there might be multiple representations for a given point, corresponding for instance to rotating the entire Cube without twisting it (which would not normally count as a move); then the metric space property that $d(x, y) = 0$ if and only if $x = y$ would be violated. We would be faced with requiring the vectors to be some kind of canonical representation instead of just any vectors following the encoding scheme.

Even in a space with a well-agreed vector representation, there are issues of whether a data set uses all the dimensions that may exist. For instance, consider a meteorological data set consisting of triples of temperature, humidity, and dew point. That seems to have three dimensions. But dew point happens to be a

calculated function of temperature and humidity, uniquely determined by them at least up to effects smaller than measurement error. If the three numbers were plotted in a three-dimensional graph, they would all fall on a smooth surface immersed in the three-dimensional space. It seems that in some meaningful sense that is a two-dimensional data set notwithstanding that it happens to be represented as three-dimensional vectors. The portion of the set of all three-dimensional vectors actually occupied by data values can be described by a probability distribution governing how likely a given combination of temperature, humidity, and dew point would be to occur in the data. That leads naturally to the idea that in a metric space application, we also have a probability distribution. From the application's point of view the distribution is part of the space.

native distribution

Definition 1.19
The probability distribution associated with a space in a given application is called the *native distribution* of the space. Unless otherwise specified, any time we talk about drawing points from the space, that means drawing points independently and identically distributed from the native distribution of the space.

effective dimension

The effective or intrinsic number of dimensions in a data set is determined not only by the representation of points, but by how those points are distributed. For instance, four-dimensional vectors that happen to be uniformly distributed along a one-dimensional line segment might be expected to behave very much like one-dimensional real numbers distributed on an interval, and not much like four-dimensional vectors chosen uniformly from a four-dimensional hypercube. Given that we can freely translate data among multiple equivalent representations, we can ask what kind of dimensionality is invariant among the representations. Mandelbrot claims that "*effective dimension* [is] a notion that *should not* be defined precisely." [146, page 17, italics his]; but we propose to find a definition for it anyway.

This question came to our attention as a result of work on robust hashing. A secure robust hash [51, 79] is designed to recognise points that are close to a secret point without revealing the secret point until a "close" point has been found. If the points exist in for instance n-dimensional Euclidean space, then someone searching for the secret point can start with an initial guess and then explore increasing neighbourhoods around the initial guess, hitting the secret point after $\Theta(l^n)$ attempts where l is the distance from the initial guess to the secret point. The number of dimensions determines the difficulty of finding the secret point. But this is an adversarial application, where it must be assumed that attackers will transform points into whatever representation they find most advantageous—that is, the one with fewest dimensions. So the important question is not how

many dimensions did we use in our own representation, but rather what is the smallest number of dimensions the attacker will be able to use while still correctly representing the data? That question depends not only on the points in the space, but also the metric, and the probability distribution of points the attacker expects for our choice of secret point.

A similar issue is also important in database indexing, where it has become known as "the curse of dimensionality" [24, 41, 110]. As dimensionality increases, parameters of interest to similarity search increase exponentially. For instance, as discussed above, quadtrees in two dimensions become octrees in three dimensions, and an analogous data structure quickly becomes unworkable in higher dimensions because the branching factor doubles for each added dimension. Even distance-based data structures, with no direct dependence on the geometry of the space, show rapidly declining performance as dimensionality increases, and performance in high-dimensional spaces is an important design goal for distance-based data structures.

To compare performance of data structures among spaces of varying dimensionality, we need some way of describing the dimensionality of a space. The measure of dimensionality should be applicable even to spaces that are not vector spaces; it should capture the idea of how many actual or effective dimensions are contained in a native distribution that might be represented in a higher-dimensional space; and it should correlate with the observed difficulty of indexing in the space. We can start looking for ways to measure the dimensionality of metric spaces by listing known properties of high-dimensional spaces and finding ways to measure those. These properties, when suitably formalised, can be proved to apply to high-dimensional Euclidean and other well-behaved spaces; but more importantly, they are empirically observed to be typical of the spaces that we tend to think of as high-dimensional.

1. High-dimensional objects require many bits to write down.
2. It is difficult to cover a high-dimensional space with small spherical subsets; in particular, such a covering must cover some points many times over.
3. The volume of a sphere in high-dimensional space increases rapidly with its radius.
4. Points chosen from a high-dimensional distribution tend to be equidistant from each other.

The first property comes from the simple definition of vector dimension as the number of components; if we generalise that to arbitrary objects that can be represented as binary bits, it seems natural to ask how many bits we need per

object. However, that approach is tied to the particular representation chosen unless we make it the length of the minimal representation—which would make the question equivalent to Kolmogorov complexity, and uncomputable [141]. It also ignores differences among spaces with the same points and different metrics; whereas difficulty of indexing or robust hashing in a space depends very much on the metric.

Because topology studies invariant properties of spaces, it is one place to look for a satisfactory definition of dimensionality. Indeed, topologists have defined and studied in detail a number of different concepts of dimensionality, and our second property, about covering with spherical subsets, is a simplified description of one of them. Introductory works like that of Kinsey give more precise detail; she discusses building a cell complex to model a space, at which point the dimension can be observed from the kinds of cells needed to build the complex [123, page 61]. Fedorchuk devotes an entire book part to topological dimension theory [67]. Topological dimension considers the metric on a space, and it can be applied to the set of data values that actually occur (treating that set as a space in its own right) rather than only to the entire representation space, so it seems both more relevant to indexing and more invariant to representation than Kolmogorov complexity. However, topological dimension still does not consider the native distribution of the space. There can also be difficulties applying it to spaces in which the distance is discretised, such as the Hamming-distance space.

We do not consider topological issues in much detail in the present work because of those limitations. However, two definitions from topology will become important in our work on D_q dimensions, so we reproduce them here. Kinsey gives more detail on the implications of these definitions [123].

Definition 1.20

open set

A subset O of a metric space S is called an *open set* if every $x \in O$ has a neighbourhood entirely contained in O.

Definition 1.20 may seem unhelpful because it simply shifts the definitional problem to another target: we know what an open set is given a neighbourhood, but what is a neighbourhood? In topological work, the definition of neighbourhood is considered to be part of the space; one of the equivalent definitions of a topological space is as a set of points and a collection of their neighbourhoods. For our work on metric spaces, neighbourhoods can be defined in terms of the metric: a neighbourhood of a point x is the interior of a sphere centred on x, that is, for some $\epsilon > 0$ an *ϵ-neighbourhood* of x is the set of points y such that $d(x, y) < \epsilon$, and a *neighbourhood* as such is any ϵ-neighbourhood. These definitions formalise for general metric spaces the familiar notions of neighbourhoods and open sets used in real analysis.

ϵ-neighbourhood

neighbourhood

Definition 1.21

A space S is *compact* if every open cover of S has a finite subcover. That is, if $\mathcal{C}$ is a set of open sets in S such that the union of all sets in $\mathcal{C}$ is equal to S, then there must be a finite subset $\mathcal{F}$ of $\mathcal{C}$ such that the union of all the sets in $\mathcal{F}$ is equal to S. compact

Compactness describes, in abstract topological terms without direct reference to coordinates or a metric, something like boundedness. Indeed, basic theorems in topology relate compactness of spaces to boundedness and certain other properties, like the convergence of Cauchy sequences [10, page 60]. The subtleties of these definitions are relevant to some of the more badly-behaved spaces studied in topology, and generally not important for computational spaces where points are represented by explicit data structures on which we can compute distances with actual computer software.

The two remaining properties we mentioned for high-dimensional metric spaces can both be defined, and quantified, in terms of the probability distribution of distance between two random points from the space. As such, they consider not only the points and the space but also the native distribution. Changes of representation that do not affect the metric, or do not affect it much, also have no or little effect on the probability distribution of distance between two random points, so a dimensionality measurement based on this probability distribution should be immune to representation changes.

The volume of a sphere in d-dimensional Euclidean space increases as the d-th power of the radius; that is, exponentially with dimension. There is an intuitive link between rapidly increasing sphere volume and points tending to be equidistant: because the volume of a sphere in high-dimensional space is much larger than the volume of a sphere with even slightly smaller radius, that means most of the volume is contained at or just below the surface. The deep interior of the sphere is much smaller than the surface. So considering how the native distribution appears from the point of view of one point, if we draw increasing spheres until one contains most of the native distribution, we will find that most of the distribution ends up near the surface of the sphere just because there is vastly more space there.

On the fourth point, about points tending to be equidistant, we emphasise that this contemplates a form of dimensionality deeper than the specific representation chosen. It is possible to imagine that data may be represented in a high- or infinite-dimensional space (in some sense) while still behaving like low-dimensional data (in some other sense). Indeed, that seems to be the usual case for high-dimensional representations actually encountered in practice, such as the word space model of documents as described by Sahlgren [178]. Documents correspond to vectors with thousands of components, but they have

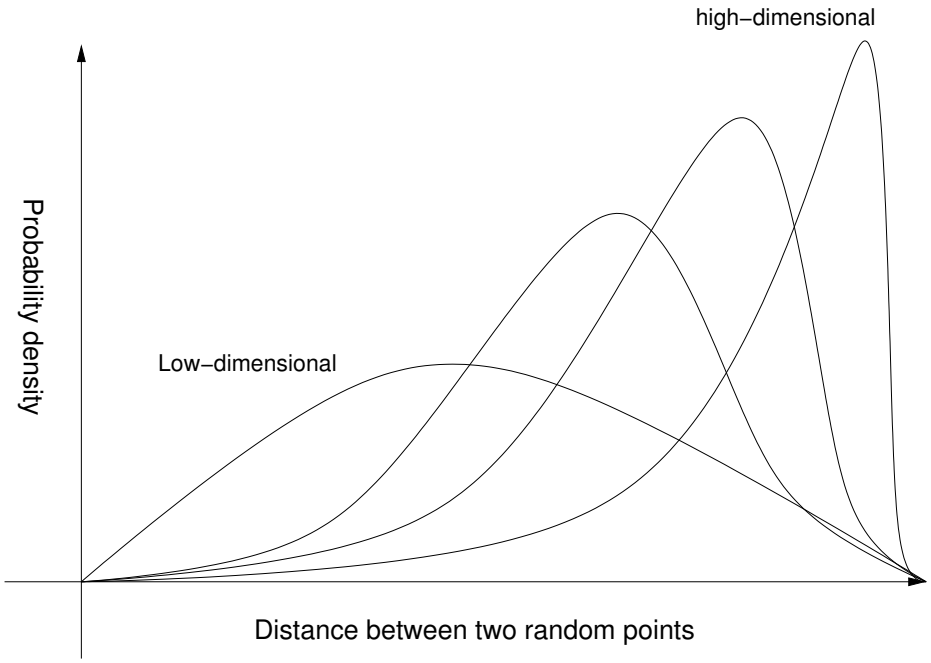


Figure 1.3: Distance distribution changes with dimensionality.

indexing properties similar to those of randomly chosen vectors with far fewer compoents. It may be said, then, that documents in the word-space model are not really high-dimensional. They can be called low-dimensional because they behave like other low-dimensional things—and it is that kind of dimensionality we seek to measure. Situations where the representation may have a much higher dimensionality than the data can include vectors where components are highly correlated to each other, and native distributions that produce strong clumping behaviour.

In Figure 1.3 we see illustrative probability density functions for the distance between two points chosen from several spaces of varying dimensionality. Two phenomena can be observed corresponding to our high-dimensional metric space properties. First, the density on the lower tail of the curve, corresponding to the amount of probability density inside a randomly chosen sphere, increases more sharply (like a higher-degree polynomial) for the higher-dimensional curves. Second, the peaks of the curves are sharper for higher dimensions, with less variance in relation to the mean: points have more tendency to be equidistant.

Others have introduced dimensionality measures based on each of these properties. The D_q dimensions, which generalise several dimensionality measures used in fractal geometry and chaos theory, express the tendency of higher-

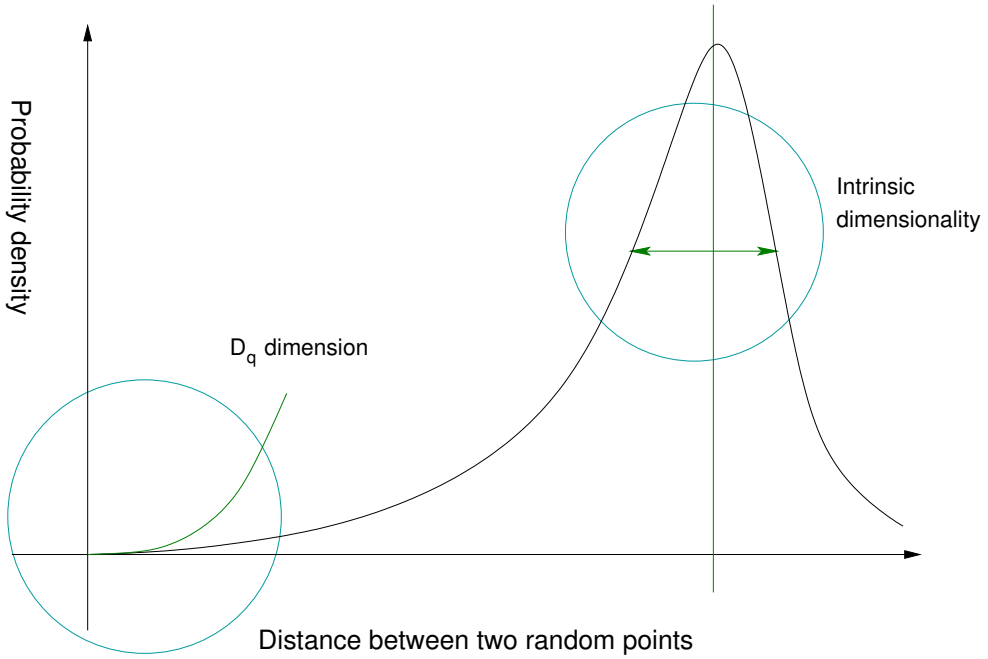


Figure 1.4: Intrinsic dimensionality describes the average case, while D_q dimension describes the limit for small distances.

dimensional probability distributions to look like higher-degree polymonials; it essentially answers the question "What power law does the distribution look like at short distances?" The intrinsic dimensionality measures the tendency for random points to be equidistant; noting that discrete spaces can only be searched by linear search and thus can be seen as having very high dimensionality, intrinsic dimensionality answers the question "How much does this space look like a discrete space?" As shown in Figure 1.4, the two measurements are complementary, measuring different parts of the probability distribution.

1.3.1 D_q dimension

Suppose we start with a small sphere in a space and evaluate the probability that a point chosen from the native distribution will be within that sphere. If we increase the radius of the sphere, the probability increases, until it becomes a certainty if the sphere encompasses the entire support of the distribution. For many common well-behaved distributions, the probability for small-radius spheres increases with some power of the radius; and for distributions in spaces where dimensionality is easy to define, the power seems to correspond to the dimensionality. For instance, in k-dimensional Euclidean space with the native

distribution uniform on the unit hypercube, the probability increases with the k-th power of the radius. If for some space we can find a value of k such that probability exhibits this behaviour, we can say that in some sense the space has k dimensions.

That approach to defining dimensionality is the basis for several measures of dimensionality used in the studies of chaos and dynamical systems. Many things can go wrong while looking for a value of k. In particular, it could be that the probability does not show polynomial behaviour in the limit of small distances. It could be that it shows polynomial behaviour but with a different exponent k depending on the centre we chose. If we average over a random selection of the centre, then it may depend on the distribution we use to choose the centre. The metric might be discrete-valued (like edit distance, for instance), so that the idea of limiting behaviour for small spheres is not meaningful.

The exponent might turn out not to be an integer—but although counterintuitive, that situation is not necessarily a problem. Some distributions and spaces really do display behaviour in some sense intermediate between two integer dimensions, and the insight that non-integer dimensions can be meaningful is the basis for the study of fractals, pioneered by Benoit B. Mandelbrot in the 1970s and 1980s.

The D_q dimension examines the power-law behaviour of the distance probability distribution, addressing many of the mentioned issues [162, Section 3.3]. The definition is often described in terms of coordinate-aligned boxes of size ϵ, but we have used general open balls in order to address general metric spaces without coordinate axes.

Definition 1.22

D_q dimension

For a compact metric space S and a real radius $\epsilon > 0$, let $\{B_1, B_2, \ldots, B_n\}$ be a minimum-size (necessarily finite by compactness; see Definition 1.21) cover of S by open balls of radius at most ϵ, let x represent a random point drawn from the native distribution of S, and for $q \geq 0$ define the D_q *dimension* in general as the following limit, if it exists:

$$D_q = \frac{-1}{1-q} \lim_{\epsilon \to 0^+} \frac{\log \sum_{i=1}^{n} \Pr[x \in B_i]^q}{\log \epsilon} \tag{1.16}$$

By convention, when $q = 0$ we use $0^0 = 0$ and for $q = 1$, we use the limit of D_q as q goes to 1.

For specific values of q, the D_q dimension reduces to other cases that have been independently described. In particular, D_0 (for subsets of Euclidean space) is the *box-counting dimension*, describing dimensionality of a set without reference to the probability distribution over it. The *Hausdorff dimension* has a

box-counting dimension

Hausdorff dimension

complicated measure-theoretic definition, but turns out to be equal to D_0 in the cases ordinarily encountered [162, pages 100–103]. The term *fractal dimension* is often applied interchangeably to the D_0 and Hausdorff dimensions even though they can be theoretically distinct; Mandelbrot, who coined the term, writes that Hausdorff dimension is "a fractal dimension," leaving open the possibility that other measures of dimensionality could also be fractal dimensions [146, page 15].

fractal dimension

Evaluating D_0 from a specific point results in the *pointwise dimension*, which can depend on the point we chose; and then the D_1 dimension is the expected pointwise dimension for a point chosen from the native distribution, also called the *information dimension*. Most relevant for our study of distance-based indexing, the D_2 dimension, called the *correlation dimension*, describes the growth of probability density for the distance between two points chosen from the native distribution. As Grassberger and Procaccia describe, correlation dimension is also especially convenient for empirical measurements of chaotic systems [89].

pointwise dimension

information dimension

correlation dimension

Much work has been done on the relationships between D_q for different values of q. One important result is that D_q must be nonincreasing with increasing q. On the other hand, it is an empirical observation that in practical systems, D_q is generally constant or nearly constant regardless of q [162, pages 79–80]. The relatively rare exceptions, where D_q decreases in a significant way with increasing q, are called *multifractals* and they are of significant interest in the theory of dynamical systems. Mandelbrot's well-known book on fractals is credited with popularising the idea of fractional dimensions in chaotic dynamical systems [146]. Ott describes much of the subsequent development of the field at an introductory level [162]. Pesin gives a more detailed presentation, including the subtler topological and measure-theoretic issues [168]. Young proves connections between Hausdorff dimension, entropy, and the Lyapunov exponents, which measure the tendency for dynamical systems to amplify small changes in initial conditions [219].

multifractals

The D_q dimension is attractive for studies of distance-based indexing structures, especially trees, because it describes the behaviour of distances in the limit for small distance. To prove asymptotic behaviour of data structures, we are generally interested in the limit for large numbers of points in the database, which translates to small distances between them. The distances encountered while searching the bottom leaves of the tree will be the ones described by D_q (especially D_2) dimension, so this dimension should be useful for proving bounds on index behaviour. Faloutsos and Kamel use that approach to analyse $R*$-trees, giving an estimate of search performance based on the D_1 dimension and experimental results supporting the accuracy of the estimate [66].

However, the limitation to spaces where arbitrarily small distances are mean-

ingful (roughly equivalent to the complete spaces defined in topology) is a significant limitation. It rules out use of D_q dimension without severe modification on important spaces like strings with Hamming or edit distance. Since real-life data sets are necessarily limited to a finite number of objects, it is also difficult to compute D_q dimension on practical data as opposed to theoretically-defined probability distributions; at best we can approximate it by looking at the smallest distances available in our data set, essentially plotting the probability density for smaller and smaller distances until the data runs out and then drawing a line on the graph and hoping it represents the limiting behaviour. That is the approach others have generally used for applying D_q dimension to real-world data sets [66, 89, 162].

1.3.2 Intrinsic dimensionality

The other natural way to examine the distance distribution is to look at the main body and measure the tendency for points to be equidistant. In higher-dimensional spaces, the distance between two random points is more likely to be close to its mean, so a statistic measuring that likelihood can be used to compare and describe spaces. Chávez and Navarro define such a measure, which they call intrinsic dimensionality [40].

Definition 1.23

Where S is a space and μ and σ^2 are the mean and variance of the distance between two random points from native distribution in that space, the *intrinsic dimensionality* of S, denoted by ρ, is given by [40]

intrinsic dimensionality

$$\rho = \mu^2/2\sigma^2 . \tag{1.17}$$

The formula (1.17) may appear arbitrary, but it follows naturally from the concept it is designed to measure. The measure of dimensionality should grow as the distribution becomes less variable, so σ^2 is in the denominator. Scaling all distances by a constant should have no effect, so μ^2 is in the numerator to make such scaling cancel out. Dimensional analysis of the formula also supports squaring μ: the units of μ will be the units of the metric (for instance, metres), but the units of σ^2 will be the metric's units squared (for instance, metres squared) and they should cancel out to make ρ a unitless number. The factor of 2 in the denominator scales the result to be equal to number of vector components in some common cases, as we shall prove in Chapter 2. By this definition discrete spaces have high intrinsic dimensionality (increasing linearly with the number of points), even though in topological terms, discrete spaces are zero-dimensional.

Note 1.24

In the case where the native distribution consists of always selecting the same point, then the mean and variance of the distance are zero, and the intrinsic dimensionality is formally undefined (division by zero). It is convenient to define $\rho = 0$ for this case. That value is intuitively reasonable—a single point seems like it should be zero-dimensional—and it is consistent with evaluating the formula from Theorem 1.1 in the limit as $\Pr[x = y]$ approaches 1.

Intrinsic dimensionality is an attractive way of describing spaces because it is easy to compute, both in theory and in practice. In the present work we give both kinds of results: theoretical proofs of the value of ρ for specified distributions, and experimental measurements for actual databases. Unlike D_q dimensions, which are based on the behaviour at arbitrarily small distances and so can only be approximated in real-life experiments, ρ is defined by basic summary statistics and can be computed easily from a sample.

However, for intrinsic dimensionality to be useful we must not only know its numerical value, but be able to draw conclusions about other things based on that value. To draw conclusions from intrinsic dimensionality requires a link between the number and questions like performance of similarity search. Chávez and Navarro introduce some theoretical results of that kind in their original paper introducing intrinsic dimensionality. In particular, they prove bounds on the performance of several kinds of distance-based index structures for metric spaces in terms of ρ [40]. However, most work that uses intrinsic dimensionality relies instead on the practical observation that it does correlate with increased difficulty of indexing and with other measures of dimensionality, for which links to indexing difficulty are known. For instance, Mao and others describe its practical use in a biological context without going into the theoretical link between large values of ρ and hard bounds on the algorithms [147].

The question of how much a distribution varies about its mean is of course interesting in the general statistical context, not only for indexing databases. The precise form of Definition 1.23 seems to be original with Chávez and Navarro, but similarly-intended statistics have been thoroughly studied. In particular, the *coefficient of variation* σ/μ equal to $1/\sqrt{2\rho}$, and the squared coefficient of variation, are well-known [126, page 107]. Handbooks of statistical distributions give formulas for coefficient of variation for many well-known distributions, from which intrinsic dimensionality would be easy to calculate [115, 116, 114]. However, the distributions that actually occur in our database problems often are not of the well-studied forms for which those results apply. For example, the Euclidean distance between vectors with independent and identical normal components ends up having a chi distribution, representing the square root of a

coefficient of variation

chi-squared variable. The chi distribution has relatively few published results; and for L_p metrics with finite p other than the Euclidean metric, it becomes the general p-th root of a sum of p-th powers, with even fewer published results. We consider those cases, and a number of others, in Chapter 2. It may seem intuitive that these kinds of results should be well-known already, but the actual results, let alone detailed presentations like ours, are absent from the usual sources.

It is an observation that easy spaces to index have small ρ, and the statistic increases as the spaces become harder to index. Chávez and Navarro use a result of Yianilos to argue that intrinsic dimensionality should be proportional to the number of vector components for vectors chosen uniformly at random from hypercubes [40, 218]. They also give calibration data consisting of experimental results on the ρ values for vector spaces. We give a more detailed theoretical and experimental examination of these questions in a paper presented at SPIRE'05 [190]; those results are included in Chapter 2 and similar results for other spaces are given throughout the present work, forming the first of the three main studies in this dissertation. The overall situation is that intrinsic dimensionality does indeed correlate, both in theory and in practice, with other attributes of spaces we think of as high-dimensional; but some individual spaces display counterintuitive behaviour, and in some cases the relationship between ρ and other features we might think of as dimensionality, may not be linear.

The intrinsic dimensionality statistic may have other useful applications beyond indexing. In particular, scale-free graphs [17] are a current topic in the study of systems like semantic networks [201] and the Internet [139]. These are graphs in which the degree sequence of vertices follows a power-law distribution, and as described by Li and others, the coefficient of variation of that distribution has important consequences for interesting properties of the graph [139]. The distribution of distance between nodes in a network is not precisely the same thing as the distribution of degrees of individual nodes; but distance between nodes is certainly an important topic for scale-free graphs. The fact that essentially the same statistic (coefficient of variation as opposed to intrinsic dimensionality) is already used for a similar purpose (describing the kinds of connections that exist) suggests that scale-free graphs may also yield applications for this kind of study.

Tables 1.1 and 1.2 summarise the new intrinsic dimensionality results in the present work. Chapter 2 describes results for vector spaces, where in most typical cases the intrinsic dimensionality turns out to be asymptotically linear in the number of vector components. We give an asymptotic analysis for random vectors in which all components are independent and identically distributed, with L_p metrics. In the case of L_1, the asymptotic approximations become exact. In the case of L_∞, the asymptotic behaviour is not necessarily linear, and in particular, it turns out to be $\Theta(\log^2)$ when the components are normally distributed. We

also give more detail on several cases of multivariate normal distributions in Euclidean space, as summarised in the table.

For non-vector spaces the results are more diverse. Discrete spaces are easy, and analysed in this introductory section. We consider tree metrics, Hamming distance, Levenshtein distance (the usual form of edit distance), and Superghost distance (which we introduce, in Chapter 7); each of those is described in detail in its own chapter. For tree metrics, ρ is between a constant and linear in the number of points, depending on the distribution. For the string metrics it shows a variety of behaviours. We give exact results for Hamming distance. The theoretical issues for Levenshtein edit distance are complicated, and connect with much previous work on statistical behaviour of that distance; we discuss the previous work and give experimental results suggesting that intrinsic dimensionality for this space is approximately $\Theta(n^{5/4})$ in string length for uniformly chosen equal-length strings. The Superghost metric implicates many of the same issues, but we can at least prove a lower bound: $\rho = \Omega(n^2/\log^2 n)$. Finally, we examine the relationship between intrinsic and D_q dimensionality, showing that no relationship necessarily exists because we can construct a space with both chosen arbitrarily; and we give experimental results for some practical databases.

We close this section by proving simple intrinsic dimensionality results for discrete spaces, which do not have a chapter of their own.

Theorem 1.1
If S is a discrete space, then where x and y are random points from the native distribution of S and $q = \Pr[x = y] < 1$, the intrinsic dimensionality of S is given by $\rho = (1-q)/2q$.

Proof The result follows almost trivially from the definition. The distance between two random points is a Bernoulli random variable, equal to 0 with probability q and 1 with probability $1-q$. Then its mean is $1-q$, its variance is $q(1-q)$, and substitution into the definition of ρ gives the result. ∎

Corollary 1.2
If S is a discrete space comprising n points with the native distribution uniform, then $\rho = (n-1)/2$, and this is the greatest possible intrinsic dimensionality for any finite space with n points.

Vectors of n iid real components

	L_p, finite p	L_∞
general	Theorem 2.1: $\rho \to \left\lceil \frac{p^2(\mu'_p)^2}{2(\mu'_{2p} - (\mu'_p)^2)} \right\rceil n$ where μ'_k is the k-th raw moment of $\lvert X - Y \rvert$.	Theorem 2.5: ρ approaches one of three expressions, depending on limit behaviour of $\lvert X - Y \rvert$; **not** necessarily linear.
uniform	Theorem 2.6: $\rho \to \left\lceil \frac{4p+2}{p+5} \right\rceil n$	Theorem 2.7: $\rho \to \left\lceil \frac{1}{2 - \pi/2} \right\rceil n$
normal	Theorem 2.8: $\rho \to \left\lceil \frac{p^2\Gamma^2\left(\frac{p+1}{2}\right)}{2\left(\sqrt{\pi}\Gamma\left(p+1/2\right) - \Gamma^2\left(\frac{p+1}{2}\right)\right)} \right\rceil n$	Theorem 2.9: $\rho \to \frac{12}{\pi^2}\log^2 n$

By Corollary 2.2, the approximations for large n above are exact for all n in the case of the L_1 metric.

Multivariate normal vectors in Euclidean space

Theorem 2.10: When all nonzero variances equal,

$$\rho = \frac{1}{2} \cdot \frac{\Gamma^2\left((n+1)/2\right)}{\Gamma(n/2)\Gamma\left((n+2)/2\right) - \Gamma^2\left((n+1)/2\right)}.$$

Theorem 2.12: With two variances σ_1^2, σ_2^2, and $\tau = (\sigma_1^2 - \sigma_2^2)/(\sigma_1^2 + \sigma_2^2)$,

$$\rho = \left(\frac{8}{\pi} \left((1+\tau)(1-\tau)\, {}_2F_1\left(3/4, 5/4; 1; \tau^2\right) \right)^{-2} - 2 \right)^{-1}.$$

Subsection 2.2.3: With variances $\sigma_1^2, \sigma_2^2, \ldots, \sigma_n^2$,

$$\rho \approx \frac{1}{2} \cdot \frac{\Gamma^2(\alpha + 1/2)}{\Gamma(\alpha)\Gamma(\alpha+1) - \Gamma^2(\alpha+1/2)} \text{ where } \alpha = \frac{(\sigma_1^2 + \sigma_2^2 + \cdots + \sigma_n^2)^2}{2(\sigma_1^4 + \sigma_2^4 + \cdots + \sigma_n^4)}.$$

Table 1.1: Intrinsic dimensionality results from Chapter 2.

Discrete spaces

Theorem 1.1: $\rho = (1-q)/2q$ where $q = \Pr[x = y]$.

Corollary 1.2: $\rho = (n-1)/2$ when there are n points and uniform distribution, and this is maximum for any distribution.

Tree metrics

Section 4.1: With finite number of points n, ρ can be as small as a constant ($\to 1$) with uniform distribution, or arbitrarily small with general distribution, and as large as linear ($\to n/2$) for uniform distribution, which is maximum for any distribution by Corollary 1.2.

Theorem 4.1: $\rho \to (2n+1-|\Sigma|)^2/|\Sigma|(|\Sigma|-1)$ for uniformly-chosen strings of length n with alphabet Σ and prefix distance.

Strings of n bits with Hamming distance

Section 5.1: $\rho = nq(1-q)/(1-2q+2q^2)$ if each bit is chosen from a Bernoulli distribution with parameter q.

Theorem 5.1: $\rho \to [r/(2r+1)]n$ for uniform distribution on radius-r ball.

Strings of length n with Levenshtein distance

Section 6.1: very difficult theoretical question; experimental results suggest ρ approximately $\Theta(n^{5/4})$.

Strings of length n with Superghost distance

Theorem 7.3: $\rho = \Omega(n^2/\log^2 n)$.

Other results

Theorem 9.2: There is not necessarily any relationship between ρ and D_q.

Section 9.3: Experimental values of ρ for SISAP library databases.

Table 1.2: Intrinsic dimensionality results from Chapters 1, 4–7, and 9.

Proof The value of ρ follows from Theorem 1.1 with $q = 1/n$. Now, suppose that we have a finite space whose metric is not the equality metric. We will modify the distance function, never decreasing the intrinsic dimensionality, until the distance function becomes the equality metric. Some of the intermediate steps may not satisfy the metric space properties, but since the equality metric does, the conclusion that no other metric can give greater intrinsic dimensionality remains valid. Note that the native distribution cannot be such as to always choose the same point; otherwise the distance between two random points would always be zero and the intrinsic dimensionality would be zero by definition (see Note 1.24).

Let $\mathrm{E}[d]$ represent the expected distance between two points chosen from the native distribution, and $\mathrm{E}[d^2]$ the expectation of the square of the distance. The intrinsic dimensionality is given by

$$\rho = \frac{\mathrm{E}^2[d]}{2(\mathrm{E}[d^2] - \mathrm{E}^2[d])}.$$

Observe that intrinsic dimensionality is unchanged if we scale all distances by a constant factor c, because both numerator and denominator increase by c^2 and it cancels out. Scale the distances such that the maximum distance is 1. There must be some distance $d(x,y) < \mathrm{E}[d]$ for distinct points $x \neq y$ (otherwise d would be the equality metric by definition). We will increase $d(x,y)$, and $d(y,x)$ to match, to bring d closer to the equality metric while not decreasing ρ.

Consider the distances between distinct points as a vector $\mathbf{d}$ (it will have $\binom{n}{2}$ components). Quantities like ρ and $\mathrm{E}[d]$ are functions of $\mathbf{d}$. Let prime represent derivative with respect to $d(x,y)$; for instance, $\rho' = \partial\rho/\partial d(x,y)$. Let $\Pr[z]$ represent the native distribution of the space; that is, the probability of drawing z when we draw a point. Now we will compute the sign of ρ'.

$$\begin{aligned}
\mathrm{E}[d] &= \sum_{a \neq b \in S} 1/2 \Pr[a] \Pr[b] d(a,b) \\
\mathrm{E}'[d] &= \Pr[x] \Pr[y] \\
\mathrm{E}[d^2] &= \sum_{a \neq b \in S} 1/2 \Pr[a] \Pr[b] d^2(a,b) \\
\mathrm{E}'[d^2] &= \Pr[x] \Pr[y] 2d(x,y) \\
\rho &= \frac{\mathrm{E}^2[d]}{2(\mathrm{E}[d^2] - \mathrm{E}^2[d])} \\
\rho' &= \frac{\mathrm{E}[d](2\,\mathrm{E}'[d]\,\mathrm{E}[d^2] - \mathrm{E}[d]\,\mathrm{E}'[d^2])}{2(\mathrm{E}[d^2] - \mathrm{E}^2[d])^2} \\
&= \left(\frac{\Pr[x]\Pr[y]}{\mathrm{E}[d^2] - \mathrm{E}^2[d]}\right)^2 (\mathrm{E}[d^2] - \mathrm{E}[d] d(x,y))
\end{aligned}$$

The squared quotient must be nonnegative, so we are left to evaluate the sign of $\mathrm{E}[d^2] - \mathrm{E}[d]d(x,y)$. But since $d(x,y) < \mathrm{E}[d]$ and we know $\mathrm{E}[d^2] - \mathrm{E}^2[d] > 0$ because it is the (necessarily positive) variance of the distribution, then $\mathrm{E}[d^2] - \mathrm{E}[d]d(x,y) > 0$; by subtracting less, we can only increase the expression. Then $\rho' \geq 0$. So by increasing a distance up to the mean distance, we can only increase the intrinsic dimensionality. Repeatedly doing that to all distances less than the mean, leaves us in the limit with a distance function where all distances among distinct pairs of points are equal to 1, which is the equality metric. Therefore the equality metric gives maximum intrinsic dimensionality.

Now, as in Theorem 1.1, the distance between two points is a Bernoulli random variable with parameter q equal to the sum of squared probabilities of individual points, and $\rho = (1-q)/2q$. That is maximised when q is minimised, which occurs when the probabilities of all points are equal. Therefore the discrete space with n points and native distribution uniform has the maximum possible intrinsic dimensionality. ∎

1.4 Distance permutations

The second of the three main questions considered in the present work has to do with the maximum number of distance permutations possible in a space. We begin by defining distance permutations.

Definition 1.25

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.

site

distance permutation

Chávez, Figueroa, and Navarro introduce distance permutations (also called proximity preserving orders) in the context of similarity search, as part of a suggested improvement to the LAESA technique of Micó, Oncina, and Vidal [39, 152]. Our interest in them stems from their possible use as a robust hashing scheme and for revealing differences in the geometry of otherwise-similar spaces.

Any algorithm that answers similarity search queries must evaluate the distance from the query point to all points in the database, at least to the point of knowing for each point in the database whether that point should be included in the answer. Algorithms improve on linear exhaustive search by using data from the index to avoid direct computations of the metric. In the case of metric

tree data structures like the VP- and GH-trees described in Subsections 1.1.4 and 1.1.5, the search algorithm proceeds down the tree using the triangle inequality to prove that subtrees cannot contain any points to be included in the result; then those subtrees are pruned from further consideration.

Another way to speed up the search would be to examine database points exhaustively, but store information about each one allowing it to possibly be excluded without a call to the metric. Since we generally assume that the metric is expensive (Note 1.5) and measure performance in terms of the number of calls to the metric, even a linear search of the entire database can be advantageous if it avoids actually computing the metric too many times.

The classic algorithm of this type is the Approximating and Eliminating Search Algorithm (AESA) of Vidal Ruiz [176]. In that algorithm, the database contains a precomputed and quadratic-sized matrix of all the pairwise distances between database points. Using that precomputed data and the triangle inequality it is often possible to place a bound on the distance from the query to a database point, proving that that database point must or cannot be in the result, without actually computing many distances. However, it is seldom practical to precompute and store a quadratic amount of index data. Indices normally must be linear or smaller.

The next step is to limit the size of the precomputed matrix. The LAESA (Linear AESA) technique of Micó, Oncina, and Vidal is a standard way of doing that: instead of storing all the pairwise distances, k reference sites or pivots are chosen from the database and the index stores the k distances from each database point to those, for a total index size linear for constant k [152]. The resulting search algorithm is known to perform well; almost as well as AESA, and for much less cost.

Chávez, Figueroa, and Navarro introduce distance permutations as a further improvement on LAESA [39]. Instead of storing the actual distances to the k sites, they store the distance permutation as defined above. They demonstrate that most of the information useful to the search algorithm is preserved in the transformation from k real distances to a permutation on the k sites, with a considerable saving in index size (allowing the use of larger k). The same authors with Paredes develop the idea further into an algorithm called iAESA (improved AESA) [70], for which an implementation is available in the SISAP library [71]. In iAESA, distance permutations not only rule objects in and out directly, but also help select the next distance measurement to make for AESA-like narrowing of the search set.

In the present work we show that further space savings are possible without any further compromise of the information content, because in many spaces of interest, the geometry of the space limits how many distance permutations

actually occur and therefore the number of bits needed to store a distance permutation. This combinatorial question is also of theoretical interest because it reveals differences among spaces, like the differences revealed by intrinsic dimensionality. It connects to work in pure mathematics, specifically the field of combinatorial geometry.

Table 1.3 summarises our results. For tree metric spaces, k sites can generate up to $\binom{k}{2} + 1$ distinct distance permutations, and we display cases where the bound is and is not achieved. For strings in general, the question is complicated, especially where the strings may be of unlimited length. We show some bounds for Hamming distance, and use them to give a loose bound for Levenshtein distance as well. For Superghost distance the length of the strings becomes important (the question seems trivial with unlimited-length strings) and we give a bound relating string length to number of permutations. Finally, we give experimental results for some practical database and discuss the distance permutation question in hyperbolic space.

1.5 Reverse similarity search

The third focus of the present work is on constraint satisfaction problems arising from VP- and GH-trees. In either kind of tree, the internal nodes describe constraints on the points that can appear in leaves of each subtree. Every leaf then contains points that satisfy all the constraints of the internal nodes above it. If we consider the set of constraints for a leaf in isolation from the database or tree that generated them, we can ask the basic constraint satisfaction question: is there any point satisfying all the constraints on the list? Could this leaf ever appear and be nonempty in a tree generated from actual data? We call that *reverse similarity search*, because it reverses the process of doing an ordinary search in the tree. Rather than going from a query to a leaf, we go from a description of a leaf to a query (if one exists) that would end up in that leaf.

reverse similarity search

Note 1.26

The reverse similarity search problems studied here are distinct from the reverse nearest neighbour search of Korn and Muthukrishnan. In their problem, a query starts with a point x and finds the other points in a database that have x as their nearest neighbour [129]. That is a close variation of the ordinary kNN search and approached with similar techniques. Our reverse similarity searches are decision problems more closely resembling constraint satisfaction problems like 3SAT.

We begin by defining the problems formally.

Vectors of n reals with L_p metrics

Theorem 3.1: $k!$ achievable when $n \geq k-1$ for all L_p metrics. Theorem 3.2 and Corollary 3.3: recurrence relation giving maximum for Euclidean (L_2) space, $\leq k^{2n}$, $k^{2n}/2^n n! + o(k^{2n})$. Theorem 3.4: $O\left(2^{2n^2}k^{2n}\right)$ for L_1, $O\left(k^{2n}\right)$ for L_2, $O\left(2^{2n}n^{2n}k^{2n}\right)$ for L_∞. Section 3.5: experimental results on distance permutations actually occurring as opposed to maximum, including examples proving the Euclidean limit is not an upper bound for L_1 and L_∞.

Tree metric spaces

Theorem 4.2: $\binom{k}{2}+1$ maximum for any tree space. Corollary 4.3: space containing unweighted path of length 2^{k-1} is sufficient condition for achieving the bound. Examples 4.10 and 4.11: spaces in which the bound is not achieved.

Strings of n bits with Hamming distance

Theorem 5.2: 2^n when $k > n$, $\geq 2^n - n$ when $k = n$. Example 5.4: $2^n - n$ bound from Theorem 5.2 is not tight. Theorem 5.3: $k!$ when $k(k-1) \leq n$, achieving the permutations without resort to tiebreaking.

Strings with Levenshtein distance

Corollary 6.7: strings of length $2n(n+1)$ with Levenshtein can have at least as many distance permutations as strings of length $2n$ with Hamming.

Strings with Superghost distance

Theorem 7.4: k strings of length $O(k \log k)$ can achieve all $k!$ permutations and all the strings achieving the permutations will be of length $O(k^2 \log k)$.

Other spaces

Section 9.3: experimental results for SISAP library databases. Section 9.4: apparently ranges from $\binom{k}{2}+1$ to Euclidean limit in hyperbolic geometry.

Table 1.3: Results for maximum number of distance permutations with k sites.

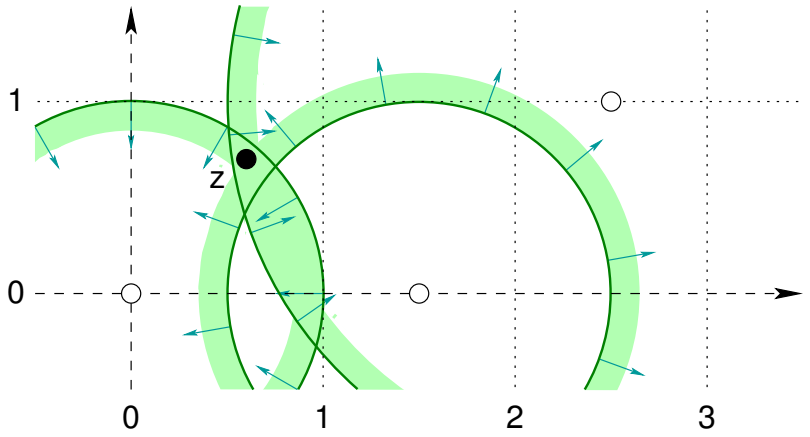


Figure 1.5: A VPREVERSE instance.

VPREVERSE

Definition 1.27 (The VPREVERSE Problem)
In some metric space (S, d), given a set P of ordered triples (x_i, r_i, b_i) with $x_i \in S$, r_i real given to some precision, and $b_i \in \{0, 1\}$, accept if and only if there exists a point $z \in S$ such that for every $(x_i, r_i, b_i) \in P$, $d(z, x_i) \leq r_i$ if and only if $b_i = 1$. Mnemonic for the bit values: 1 looks like "I" and requires points to be Inside the sphere; 0 looks like "O" and requires points to be Outside the sphere.

Example 1.28
Where (S, d) is the Euclidean plane, let P be the set of triples $\{((3/2, 0), 1, 0), ((5/3, 1), 2, 1), ((0, 0), 1, 1)\}$. This is a yes-instance satisfied by the point $z = (0.4, 0.7)$. See Figure 1.5. The point z must be outside the circle centred on $(3/2, 0)$ and inside the other two. Only points in the small curved-sided triangle-like region satisfy the instance.

GHREVERSE

Definition 1.29 (The GHREVERSE Problem)
In some metric space (S, d), given a set P of ordered pairs of points from S, with $n = |P|$, accept if and only if there exists a point $z \in S$ such that $d(z, x_i) \leq d(z, y_i)$ for every $(x_i, y_i) \in P$.

Example 1.30
For the Euclidean plane, let P be the set of pairs of points $\{((2, 1), (0, 0)), ((1, 0), (1, 1)), ((1, 0), (3, 0))\}$. This is a yes-instance satisfied by the point $z = (1.8, 0.3)$. See Figure 1.6. The satisfying point z must be on the shaded side of each of the three lines that define the constraints.

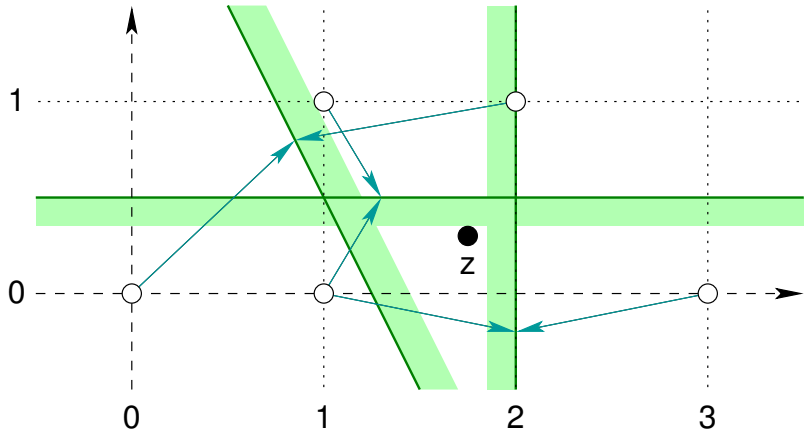


Figure 1.6: A GHREVERSE instance.

The original motivation for considering these problems came from the desire for robust hashing in security applications. Security applications sometimes require the ability to recognise objects without keeping examples for comparison. For instance, a multi-user computer system might wish to recognise users who provide an exactly correct password, without storing an explicit copy of the password that could be vulnerable to intruders [184, 216]. But an object like a fingerprint, which could be distorted by noise or measurement error, requires a fuzzy match; and current techniques store information from which the fingerprint can be reconstructed, thereby opening themselves to attack [175]. Robust hashes are designed to recognise such objects while keeping some of the security properties of existing one-way hashes [51].

One way to build a robust hash would be as a list of predicates—yes or no questions that describe an object. If the predicates are chosen so that for each question and a randomly selected object, either answer is equally likely, and so that the answers to different questions are as nearly as possible independent, then the possibly-complicated native distribution of the space can be transformed into a uniform distribution on binary strings. Nearby points are likely to give the same answers to most of the questions, but distant points are likely to give uncorrelated answers and result in binary strings with no special relationship.

Assuming the robust hash works well, we can store just the bit string as the hash value for a secret object, and recognise other similar objects by their similar hash values. On the other hand, generating a new object (short of guessing at random) to produce or approximate a given hash value, should be difficult. Our study begins with the question of whether such hashes are secure. They are already in use, with experimental rather than theoretical backing, in the Nilsimsa

	GHREVERSE	VPREVERSE	VPREVERSE, equal radii
tree metrics*	$\mathcal{P}$, Thm. 4.5	†	†
Hamming	$\mathcal{NPC}$, Cor. 5.5	$\mathcal{NPC}$, Thm. 5.4	$\mathcal{NPC}$, Thm. 5.4
Levenshtein	$\mathcal{NPC}$, Thm. 6.8	$\mathcal{NPC}$, Thm. 6.8	open
Superghost	$\mathcal{NPC}$, Thm. 7.9	$\mathcal{NPC}$, Thm. 7.9	open
L_2	$\mathcal{P}$, Thm. 8.5	$\mathcal{NPC}$, Thm. 8.3	$\mathcal{NPC}$, Thm. 8.11
$L_p, p \neq 2$	$\mathcal{NPC}$, Thm. 8.9	$\mathcal{NPC}$, Thm. 8.3	$\mathcal{NPC}$, Thm. 8.11
L_∞	$\mathcal{NPC}$, Thm. 8.10	$\mathcal{NPC}$, Thm. 8.4	$\mathcal{NPC}$, Thm. 8.11

* With some exceptions; see Section 4.4.

† Depends on space (Theorem 4.6); may be $\mathcal{P}$ (Definition 4.5) or $\mathcal{NPC}$ (Example 4.9).

Theorem 9.3: DPREVERSE $\Rightarrow$ generalised GHREVERSE $\Leftrightarrow$ GHREVERSE $\Leftarrow$ VPRU.

Table 1.4: Reverse similarity search results.

spam filter [53].

The *VP*- and *GH*-tree data structures suggest forms for lists of predicates to use in such a robust hash. Deciding whether a point can be included in a given leaf consists of answering a series of yes or no questions about it: Is it inside this sphere? Is it outside that sphere? Is it closer to this point than to that one? Such a list of questions naturally forms a candidate robust hash scheme, and then the task of attacking the robust hash's main security guarantee ("it is difficult to construct a point having a chosen hash value") is exactly the VPREVERSE or GHREVERSE problem. Successful reverse similarity search would reverse the hash.

There are two ways the attacker's problem could be hard, and we would like both. We want hard instances of the problem to exist, and we want hard instances to be easy to generate (for instance, by choosing them at random). Our results on reverse similarity search are summarised in Table 1.4. For many spaces the problems turn out to be $\mathcal{NP}$-complete. We also show some reductions connecting variations of the problems in Theorem 9.3.

Having $\mathcal{NP}$-completeness means that some instances are as hard as any problem in $\mathcal{NP}$, which provides the first kind of hardness we might want for security. But that does not mean all instances are hard, and it in fact militates against the hardness of randomly chosen instances. We would like the problems

to exhibit random self-reducibility: the property of any instance being reducible to a reasonable number of randomly chosen instances. In that case we could argue that random instances are as hard as the hardest instances in the problem. But as Feigenbaum and Fortnow show, if there were any $\mathcal{NP}$-complete random self-reducible problem, then the polynomial hierarchy would collapse at the third level [68]. Such a collapse would be a deep, important, and unexpected result in computational complexity theory, and seems too much to hope for. Achieving one of the desired security properties makes it difficult to expect the other.

However, the complexity landscape of VPREVERSE and GHREVERSE has theoretical interest apart from the original security application. As shown in the table, these problems are generally $\mathcal{NP}$-complete, but polynomial-time in some spaces. Among vector metrics, the fact that GHREVERSE is polynomial-time for Euclidean space but not any other L_p metrics highlights the special nature of that space. Among string edit distances, there is an important boundary between prefix distance (with edits allowed at one end of the string; it is a tree metric for which reverse similarity searches are polynomial time) and even very simple modifications such as Superghost distance (in which edits are allowed at both ends of the string). Like the different behaviour of intrinsic dimensionality among different spaces, the differences in reverse similarity search complexity provide a way to classify and distinguish among spaces that otherwise appear similar.

Chapter 2

Real vectors, L_p metrics, and dimensionality

Vectors of real numbers, especially with the Euclidean metric, are among the standard examples of metric spaces. Vector spaces are intuitively easy to understand because their properties closely resemble those of physical space; there is a well-studied and widely-applied theory of linear algebra based on vectors; and they are typically used as test cases for metric-space data structures. In this chapter we focus on the well known class of metrics called the L_p metrics, which generalise the Euclidean metric, and study the intrinsic dimensionality of various distributions of real vectors with these metrics.

intrinsic dimensionality

Recall that Chávez and Navarro define the *intrinsic dimensionality* of a space as $\mu^2/(2\sigma^2)$ where μ and σ^2 are the mean and variance of the distance between two random points drawn from the native distribution of the space [40]. Intrinsic dimensionality is denoted by ρ. As discussed in Subsection 1.3.2, calculating intrinsic dimensionality is equivalent to calculating the coefficient of variation, which is known for many common distributions. However, some of the specific distributions we study are not common well-known distributions. For instance, Theorem 2.8 contemplates the p-th root of a sum of p-th powers of the absolute values of normal variables; that is, a chi distribution as opposed to chi-squared, generalised to arbitrary powers. Even in cases like Theorem 2.5 where the conclusion should follow easily from well-known results, we give original derivations because those are hard to find in the standard literature. We also express the result in terms of intrinsic dimensionality ρ for convenient application to the problems we study.

This chapter uses much of the notation for probability and statistics defined in Subsection 1.2.2. In particular, recall that $\mathrm{E}[Z]$ is the expectation of a random variable Z; and $\mathrm{V}[Z]$ is the variance of a random variable Z. We use an arrow

$f(n) \to g(n)$ to indicate that f approaches g up to lower-order terms (Definition 1.16). In the discussion of the L_∞ metric we use the notation $\max^{(k)}\{Z\}$, read "max over k from Z," for the random variate consisting of the maximum of k independent and identically distributed random variables drawn from the random variate Z; $\min^{(k)}\{Z\}$ is defined analogously for the minimum of k variables from Z. Following the notation used by Arnold, Balakrishnan, and Nagaraja [13], we write $X \stackrel{d}{=} Y$ if X and Y are identically distributed, $X \stackrel{d}{\to} Y$ if the distribution of $X(n)$ converges to the distribution of Y as n goes to positive infinity, and $X \stackrel{d}{\leftrightarrow} Y$ if the distributions of both X and Y depend on n and converge to each other.

$\max^{(k)}\{Z\}$

$\min^{(k)}\{Z\}$

Definition 2.1

L_p metric

Where $\mathbf{x} = \langle x_1, x_2, \ldots, x_n \rangle, \mathbf{y} = \langle y_1, y_2, \ldots, y_n \rangle$, the L_p *metric* $d_p(\mathbf{x}, \mathbf{y})$ is defined by

$$d_p(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p} \tag{2.1}$$

for real $p \geq 1$ or

$$\begin{aligned} d_\infty(\mathbf{x}, \mathbf{y}) &= \lim_{p \to \infty} d_p(\mathbf{x}, \mathbf{y}) \\ &= \max_{i=1}^{n} |x_i - y_i| \end{aligned} \tag{2.2}$$

for $p = \infty$.

The name of Hermann Minkowski is often attached to the L_p metrics by way of the Minkowski Inequality, a basic result in functional analysis, which amounts to the statement that the L_p metrics obey the triangle inequality [153, pages 112–113, 273–274]. Some values of p are especially popular and have specific names of their own:

Manhattan distance
taxicab distance

- L_1 is called the *Manhattan* or *taxicab distance*, because in the integer-coordinate case it measures the distance travelled by a taxicab through a grid-structured city.

Euclidean distance

- L_2 is the familiar *Euclidean distance* calculated by the Pythagorean Theorem.

Chebyshev distance
chessboard distance

- L_∞ is called the *Chebyshev distance* or the *chessboard distance*, the latter because in the integer-coordinate case it measures the number of moves required by a Chess King to get from one point to another.

The unit circles of L_p distance functions for two-dimensional space and several different values of p are shown in Figure 2.1. Note that $L_{1/2}$ is not actually permitted by the L_p metric definition, which requires $p \geq 1$; the defining formula

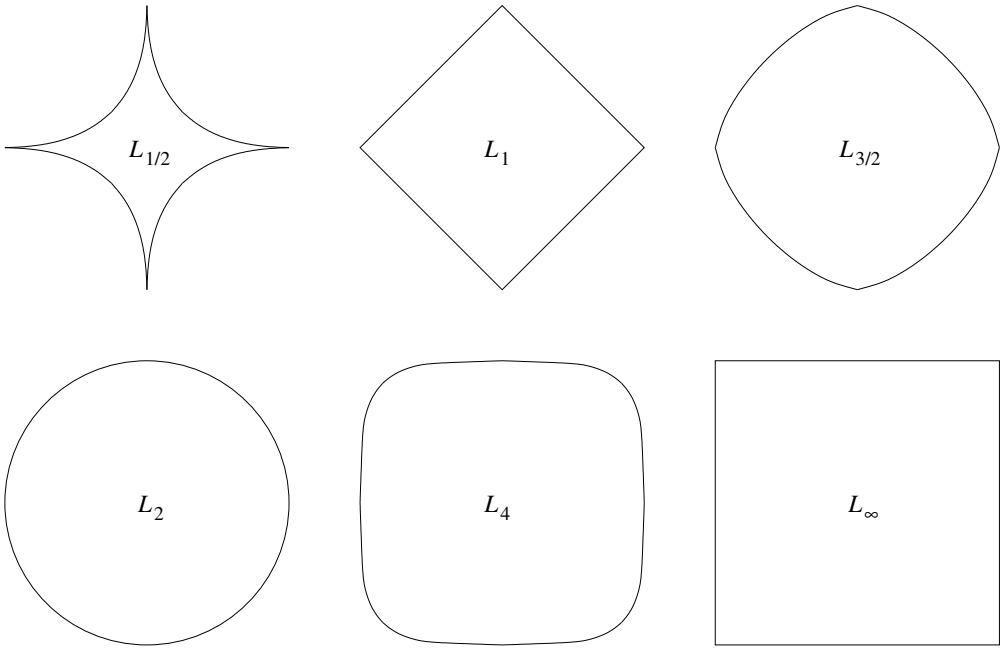


Figure 2.1: Some L_p unit circles.

(2.1) can be evaluated with $p < 1$, but the resulting distance function fails to obey the triangle inequality and so it is not a metric. The failure to satisfy the triangle inequality in this case is closely related to the fact that the $L_{1/2}$ circle, as shown, is not convex.

Circles from L_p, possibly stretched or generalised in other ways, are known as Lamé curves or superellipses. They provide a visually pleasing shape somewhere between a square and a circle, and have seen use in fields such as architecture and font design. Zapf based the Melior typeface on them [103, page 291]. Piet Hein[1] famously designed the traffic roundabout at Sergels Torg, Stockholm, around a stretched $L_{5/2}$ circle; and Balinski and Holt suggested one as a shape for the table in the Vietnam War peace negotiations [83]. The POV-Ray ray tracer offers a three-dimensional superellipsoid as a scene primitive [167, section 2.4.1.11].

Knuth's Computer Modern typeface family, which is the LaTeX default, also includes superellipse-based forms in some of its fonts [127]. Since Computer Modern is nearly universal in computer science documents,[2] most computer scientists see L_p circles every day without consciously knowing it.

[1] As Gardner writes, "he is always spoken of by both names." [83, page 241]

[2] This dissertation is an exception. The main text is set in Bitstream Charter to better survive limited-resolution processes like microfilming.

convex distance function

More general classes of vector distance functions exist. Chew and Drysdale discuss *convex distance functions* in the plane, and the resulting Voronoi diagrams [44]. Convex distance functions are defined by their unit circles: the unit circle must be a convex set containing the origin, and then the length of a vector is the factor by which the unit circle must be scaled to bring the vector onto its boundary. The distance between two vectors $d(\mathbf{x},\mathbf{y})$ is the length of $\mathbf{x}-\mathbf{y}$. Convex distance functions, generalised to arbitrary-dimensional vectors, are popular in the study of Voronoi diagrams, as are restricted versions which require the unit sphere to be a polyhedron, or symmetric about the origin. If it is symmetric about the origin, the convex distance function is also a metric. All the L_p metrics are convex distance functions with symmetric unit spheres, but only in the L_1 and L_∞ cases (and the trivial one-dimensional case) are their unit spheres also polyhedra.

2.1 Asymptotic intrinsic dimensionality with all components independent and identically distributed

When Chávez and Navarro introduced the intrinsic dimensionality statistic ρ, they presented some experimental results for L_p vectors showing that ρ on vectors chosen from the unit cube was well-approximated by a linear function of the number of components n for n between 2 and 20 and $p \in \{1,2,\infty\}$. They also gave an intuitive argument for why it should be linear [40]. The intuitive argument for linear behaviour is based on a result of Yianilos [218, Proposition 2] that only applies to L_p metrics with finite p, and as we will show, the behaviour of ρ is not in fact necessarily linear for L_∞ although it is linear for the uniform-component case of the experiments.

Exact theoretical analysis of ρ is difficult for finite n because it requires computing the distributions of functions of multiple random variables for which no convenient representations are known. We consider some of the easier cases of that in Section 2.2. In the present section, we characterise the asymptotic behaviour of ρ on vectors with all components independent and identically distributed, as the number of components n increases. The results of this section were first presented in an extended abstract at SPIRE'05 [190]. The expositions given here have been revised and expanded to provide a clearer description of the results, especially in the proofs.

Consider the space consisting of $\mathbb{R}^n$ with the L_p metric for some p and the native distribution being vectors with every component drawn independently and identically from some distribution on $\mathbb{R}$. If we let $X = Y$ be random variates representing the component distribution, then we can construct random vectors

$\mathbf{x} = \langle X_1, X_2, \ldots, X_n \rangle$ and $\mathbf{y} = \langle Y_1, Y_2, \ldots, Y_n \rangle$, which are points drawn from the space. We consider the intrinsic dimensionality of such a space, and especially its asymptotic behaviour as n goes to infinity. Letting $D_{p,n}$, itself a random variable, represent the distance between $\mathbf{x}$ and $\mathbf{y}$ under the L_p metric, then the intrinsic dimensionality is $\rho = \mathrm{E}^2[D_{p,n}]/2\mathrm{V}[D_{p,n}]$ [40].

2.1.1 Generally distributed components

The following two results offer a method for computing the asymptotic behaviour of intrinsic dimensionality depending on the distribution of $|X - Y|$, the absolute difference between two components drawn from the component distribution in question. First we consider the case for L_p metrics with finite p, which is relatively straightforward.

Theorem 2.1
For a space of n-component real vectors with independent and identically distributed components as described above, using the L_p metric for finite p, then if μ'_{2p} is defined and finite, the intrinsic dimensionality ρ obeys

$$\rho \to \left[\frac{p^2 (\mu'_p)^2}{2(\mu'_{2p} - (\mu'_p)^2)} \right] n \,, \tag{2.3}$$

where μ'_k is the k-th raw moment of $|X - Y|$, that is, $\mathrm{E}[|X - Y|^k]$.

Proof The L_p distance consists of the p-th root of a sum of p-th powers of absolute differences of components, so first we must describe the distribution of the sum. Let $s = \sum_{i=1}^{n} |X_i - Y_i|^p$. Gut gives a strengthened version of the Central Limit Theorem for independent and identically distributed random variables [94, Theorem 7.5.1], by which the finiteness of μ'_{2p} implies that the sum is uniformly integrable up to order $2p$. Not only does its distribution converge to a normal distribution, but its moments up to the $2p$-th moment also converge to the moments of that distribution. Then we can calculate the expectation and variance as follows:

$$\begin{aligned} \mathrm{E}[s] &= n\,\mathrm{E}[|X_i - Y_i|^p] \\ &= n\mu'_p \\ \mathrm{V}[s] &= n\,\mathrm{V}[|X_i - Y_i|^p] \end{aligned}$$

$$
\begin{aligned}
&= n(\mathrm{E}[|X_i - Y_i|^{2p}] - \mathrm{E}^2[|X_i - Y_i|^p]) \\
&= n(\mu'_{2p} - (\mu'_p)^2)\,.
\end{aligned}
$$

Now, $D_{p,n} = s^{1/p}$. The mean and variance of s both increase linearly with n, so the standard deviation increases according to the square root of n and its ratio to the mean decreases according to the square root of n. For large n, s does not vary much around its mean, and $s^{1/p}$ can be approximated by a linear function of s; then in an application of the delta method (see Gut [94, page 349] and Knight [126, page]) we can move the $1/p$ power outside the expectation to approximate the distribution of $D_{p,n}$. As in the previous step, the condition on existence of high enough moments gives us moment convergence of the needed order.

$$
\begin{aligned}
&\mathrm{E}[s^{1/p}] \to \mathrm{E}[s]^{1/p} = n^{1/p}(\mu'_p)^{1/p} \\
&\mathrm{V}[s^{1/p}] \to \mathrm{V}[s]\left(\frac{d}{ds}s^{1/p}\right)^2\Bigg|_{s=\mathrm{E}[s]} = \frac{\mu'_{2p} - (\mu'_p)^2}{np^2(\mu'_p)^2} n^{2/p}(\mu'_p)^{2/p} \\
&\rho = \frac{\mathrm{E}^2[s^{1/p}]}{2\mathrm{V}[s^{1/p}]} \to \left[\frac{p^2(\mu'_p)^2}{2(\mu'_{2p} - (\mu'_p)^2)}\right] n\,.
\end{aligned}
$$

This corollary follows naturally:

Corollary 2.2
If the space in Theorem 2.1 uses the L_1 metric, then the approximation is exact even for finite n:

$$
\rho = \left[\frac{p^2(\mu'_p)^2}{2(\mu'_{2p} - (\mu'_p)^2)}\right] n\,. \tag{2.4}
$$

Proof When $p = 1$, then $x^{1/p}$ is the identity function, the linear approximation is perfect, and the limits for large n in the proof of Theorem 2.1 become equalities. ■

As for the L_∞ metric, the distance $D_{\infty,n}$ is the maximum of n variables of the form $|X - Y|$. The maximum is a simple example of an "order statistic"; and order statistics are well studied and much is known about them [13, 80]. The cumulative distribution functions of maxima are easy to calculate: if $F(x)$ is the cumulative distribution function of Z, then it follows from the definitions that $F^n(x)$ is the cumulative distribution function of $\max^{(n)}\{Z\}$.

For a large collection of independent and identically distributed random variables, it is known that the maximum obeys something like the Central Limit Theorem. Just as the distribution of a sum tends to one known form (the normal distribution), the distribution of a maximum tends to one of three forms. We say that the random variable W with non-degenerate cumulative distribution function $G(x)$ is the *limiting distribution* of the maximum of Z if there exist sequences $\{a_n\}$ and $\{b_n > 0\}$ such that $F^n(a_n + b_n x) \to G(x)$. The following well-known result describes the distribution of W if it exists at all.

limiting distribution

Theorem 2.3 (Fisher and Tippett, 1928)
If $(\max^{(n)}\{Z\} - a_n)/b_n \xrightarrow{d} W$, then the cumulative distribution function $G(x)$ of W is one of the following, where α is a constant greater than zero [13, Theorem 8.3.1] [72]:

$$G_1(x;\alpha) = \exp(-x^{-\alpha}) \text{ for } x > 0 \text{ and 0 otherwise;} \tag{2.5}$$

$$G_2(x;\alpha) = \exp(-(-x)^{\alpha}) \text{ for } x < 0 \text{ and 1 otherwise; or} \tag{2.6}$$

$$G_3(x) = \exp(-e^{-x})\,. \tag{2.7}$$

The distributions described by (2.5)–(2.7) are called Fréchet type, Weibull type, and Gumbel type respectively [116, Chapter 22]. The criteria for finding which of the three limiting distributions applies, and its parameters, can be complicated, and they are often stated in terms of sufficient conditions rather than a complete characterisation. Also, the sequences of constants a_n and b_n are not necessarily unique. Arnold, Balakrishnan, and Nagaraja give a detailed presentation of these criteria [13], and we will cite specific results as we use them.

The theory of asymptotic order statistics also gives rise to the following lemma. It eliminates the absolute value function when we study the maximum of a large enough collection of random variables. This result in this form is original, but a reasonably obvious application of the cited well-known principles.

Lemma 2.4
If Z is a real variate with distribution symmetric about zero, and W, a_n, and b_n exist such that $(\max^{(n)}\{Z\} - a_n)/b_n \xrightarrow{d} W$, then $\max^{(n)}\{|Z|\} \xleftrightarrow{d} \max^{(2n)}\{Z\}$.

Proof The maximum absolute value among n variables from Z must be either the maximum, or the negative of the minimum. We could find those separately and then see which one is larger. But as explained by Arnold, Balakrishnan, and Nagaraja, the maximum and minimum of a collection of random variables are asymptotically independent [13, Theorem 8.4.3]. Therefore, instead of finding the maximum and minimum of one set of n variables, we could find the maximum of a set of n variables and the minimum of a different set of n variables; or by symmetry, the maximum of $2n$ variables. Symbolically,

$$\begin{aligned}\max^{(n)}\{|Z|\} &\overset{d}{\longleftrightarrow} \max\{\max^{(n)}\{Z\}, -\min^{(n)}\{Z\}\} \\ &\overset{d}{=} \max\{\max^{(n)}\{Z\}, \max^{(n)}\{Z\}\} \\ &\overset{d}{=} \max^{(2n)}\{Z\}\,.\end{aligned}$$

■

Now, given the distribution of $X - Y$ or $|X - Y|$, we can apply the asymptotic theory described by Galambos [80] to determine the limiting distribution for $D_{\infty,n} = \max^{(n)}\{|X - Y|\}$; and if it exists, it will be in one of the three forms stated in Theorem 2.3. We can then integrate to find the expectation and variance, and standard results give acceptable choices for the norming constants a_n and b_n, giving the following theorem.

Theorem 2.5

For random vectors with the L_∞ metric, when Theorem 2.3 applies to $\max^{(n)}\{|X-Y|\}$ and the first two moments (or mean and variance) of $|X-Y|$ are defined and finite, then the intrinsic dimensionality ρ obeys:

$$\rho \to \frac{\left(a_n + b_n\Gamma\left(1 - 1/\alpha\right)\right)^2}{2b_n^2\left(\Gamma\left(1 - 2/\alpha\right) - \Gamma^2\left(1 - 1/\alpha\right)\right)} \text{ for } G_1(x;\alpha); \tag{2.8}$$

$$\rho \to \frac{\left(a_n + b_n\Gamma\left(1 + 1/\alpha\right)\right)^2}{2b_n^2\left(\Gamma\left(1 + 2/\alpha\right) - \Gamma^2\left(1 + 1/\alpha\right)\right)} \text{ for } G_2(x;\alpha); \text{ and} \tag{2.9}$$

$$\rho \to \frac{3(a_n + b_n\gamma)^2}{b_n^2\pi^2} \text{ for } G_3(x); \tag{2.10}$$

where $\gamma = 0.57721\,56649\,015\ldots$, the Euler-Mascheroni constant.

Proof It is required to find the mean and variance of each of the three possible limiting distributions. These distributions are well-known, but they are often presented without details of how to derive their means and variances [13, 80, 116]. We give detailed derivations here.

The cases of (2.8) and (2.9). The first two cases, where the cumulative distribution function of W is as shown in (2.5) and (2.6), are almost identical and we will consider them together. The limiting cumulative distribution function F of $D_{\infty,n}$ is as shown below, taking the upper signs for (2.5) and the lower signs for (2.6).

$$F(x) = \begin{cases} \exp\left[-\left(\pm\frac{x-a_n}{b_n}\right)^{\mp\alpha}\right] & \text{for } (\pm x) > (\pm a_n), \\ 0 \text{ or } 1 & \text{otherwise.} \end{cases}$$

We can differentiate it to find the probability density function, f:

$$\begin{aligned} f(x) &= \frac{d}{dx}F(x) \\ &= \begin{cases} \pm\frac{\alpha}{x-a_n}\left(\pm\frac{x-a_n}{b_n}\right)^{\mp\alpha}\exp\left[-\left(\pm\frac{x-a_n}{b_n}\right)^{\mp\alpha}\right] & \text{for } (\pm x) > (\pm a_n), \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

The existence of the first two moments of $|X - Y|$ implies, by a result of Pickands, that the the first two moments of the maximum $D_{\infty,n}$ converge to the first two moments of the limiting distribution W [108]. We integrate, using the substitution $y = \pm(x - a_n)/b_n$ in both integrals. The substitution replaces x with $\pm(y\,b_n + a_N)$ and dx with $\pm b_n\,dy$, and the lower limit of integration becomes 0. Then we can apply the formula $\int_0^\infty y^n y^{-\alpha}\exp(-y^{-\alpha})\,dx = {}^1\!/_\alpha\Gamma(1 + (n+1)/\alpha)$, which follows from the definition of the gamma function:

$$\begin{aligned} \mathrm{E}[D_{\infty,n}] &\to \pm\int_{\pm a_n}^{\pm\infty} x f(x)\,dx \\ &= \alpha\int_0^\infty (y\,b_n + a_n) y^{\mp\alpha-1}\exp(-y^{\mp\alpha})\,dy \\ &= b_n\Gamma\left(1 \mp {}^1\!/_\alpha\right) + a_n \\ \mathrm{E}[D_{\infty,n}^2] &\to \pm\int_{\pm a_n}^{\pm\infty} x^2 f(x)\,dx \\ &= \alpha\int_0^\infty (y^2 b_n^2 + 2a_n b_n y + a_n^2) y^{\mp\alpha-1}\exp(-y^{\mp\alpha})\,dy \\ &= b_n^2\Gamma\left(1 \mp {}^2\!/_\alpha\right) + 2a_n b_n\Gamma\left(1 \mp {}^1\!/_\alpha\right) + a_n^2\,. \end{aligned}$$

Substituting those expectations into the formula for intrinsic dimensionality gives (2.11), which covers both (2.8) and (2.9):

$$\mathrm{V}[D_{\infty,n}] = \mathrm{E}[D_{\infty,n}^2] - \mathrm{E}^2[D_{\infty,n}]$$

$$
\begin{aligned}
&\to b_n^2\Gamma\left(1\mp{}^2\!/_\alpha\right) + 2a_nb_n\Gamma\left(1\mp{}^1\!/_\alpha\right) + a_n^2 - \left(b_n\Gamma\left(1\mp{}^1\!/_\alpha\right) + a_n\right)^2 \\
&= b_n^2\left(\Gamma\left(1\mp{}^2\!/_\alpha\right) - \Gamma^2\left(1\mp{}^1\!/_\alpha\right)\right) \\
\rho &= \frac{\mathrm{E}^2[D_{\infty,n}]}{2\mathrm{V}[D_{\infty,n}]} \\
&\to \frac{\left(a_n + b_n\Gamma\left(1\mp{}^1\!/_\alpha\right)\right)^2}{2b_n^2\left(\Gamma\left(1\mp{}^2\!/_\alpha\right) - \Gamma^2\left(1\mp\frac{1}{\alpha}\right)\right)}\,. \qquad (2.11)
\end{aligned}
$$

The case of (2.10). The third case proceeds similarly. Where the cumulative distribution function of W is $\exp(-e^{-x})$ as in (2.7), then we find the cumulative distribution function F and probability density function f of $D_{\infty,n}$ as follows:

$$
\begin{aligned}
F(x) &= \exp\left[-\exp\left(-\frac{x-a_n}{b_n}\right)\right] \\
f(x) &= \frac{d}{dx}F(x) \\
&= \frac{1}{b_n}\exp\left(-\frac{x-a_n}{b_n}\right)\exp\left[-\exp\left(-\frac{x-a_n}{b_n}\right)\right]\,.
\end{aligned}
$$

As in the other case, the existence of the first two moments of $|X-Y|$ implies moment convergence of order two for the maximum, so we can find the moments by examining those of W. We express the expectations as integrals and apply the substitution $y = \exp(-(x-a_n)/b_n)$. Then $x = a_n - b_n\log y$, $dx = -{}^{b_n}\!/_y\,dy$, and we get integrals of the form $\int_0^\infty \log^k x e^{-x}\,dx$. That formula is trivially equal to 1 for $k=0$; for higher k it is more difficult, but standard references give $\int_0^\infty \log x e^{-x}\,dx = \gamma$ and $\int_0^\infty \log^2 x e^{-x}\,dx = {}^1\!/_6\pi^2 + \gamma^2$, where γ is the Euler-Mascheroni constant [87, 4.331(1.) and 4.335(1.), page 567]. Then we can compute the variance, and the intrinsic dimensionality follows.

$$
\begin{aligned}
\mathrm{E}[D_{\infty,n}] &\to \int_{-\infty}^{\infty} x f(x)\,dx \\
&= \int_{-\infty}^{\infty} x\frac{1}{b_n}\exp\left(-\frac{x-a_n}{b_n}\right)\exp\left[-\exp\left(-\frac{x-a_n}{b_n}\right)\right] \\
&= \int_0^{\infty} (a_n - b_n\log y)e^{-y}\,dy \\
&= a_n + b_n\gamma \\
\mathrm{E}[D_{\infty,n}^2] &\to \int_{-\infty}^{\infty} x^2 f(x)\,dx \\
&= \int_{-\infty}^{\infty} x^2\frac{1}{b_n}\exp\left(-\frac{x-a_n}{b_n}\right)\exp\left[-\exp\left(-\frac{x-a_n}{b_n}\right)\right]
\end{aligned}
$$

$$= \int_0^\infty (a_n^2 - 2a_n b_n \log y + b^2) e^{-y}\, dy$$

$$= a_n^2 + 2a_n b_n \gamma + \frac{b_n^2 \pi^2}{6} + b_n^2 \gamma^2$$

$$\mathrm{V}[D_{\infty,n}] = \mathrm{E}[D_{\infty,n}^2] - \mathrm{E}^2[D_{\infty,n}] = \frac{\pi^2}{6} b_n^2$$

$$\rho = \frac{\mathrm{E}^2[D_{\infty,n}]}{2\mathrm{V}[D_{\infty,n}]} \to \frac{3(a_n + b_n\gamma)^2}{b_n^2 \pi^2}\,.$$

Note 2.2
The intrinsic dimensionality ρ is not necessarily $\Theta(n)$. For instance, in Theorem 2.9, ρ is $\Theta(\log^2 n)$.

2.1.2 Uniform components

Chávez and Navarro used random vectors with each component chosen uniformly from the interval $[0,1)$ for their experiment, as a typical example of the kind of distribution used for testing metric space data structures [40]. For this distribution, Theorem 2.1 applies easily.

Theorem 2.6
For the space of n-component real vectors with each component uniform in the range $[0,1)$ and the L_p metric for finite p, the intrinsic dimensionality ρ obeys

$$\rho \to \left\lceil \frac{4p+2}{p+5} \right\rceil n\,, \tag{2.12}$$

and this is an equality for $p = 1$.

Proof This is a simple application of Theorem 2.1. When X and Y are uniform real random variates with the range $[0,1)$, then the probability density function of $|X - Y|$ is given by

$$f(x) = \begin{cases} 2 - 2x & \text{for } 0 \le x < 1, \\ 0 & \text{otherwise;} \end{cases} \tag{2.13}$$

and we can find the raw moments by integration and substitute them into the formula as follows:

$$\mu_p' = \int_0^1 x^p (2 - 2x) = \frac{2}{(p+1)(p+2)}$$

$$\mu'_{2p} = \int_0^1 x^{2p}(2-2x) = \frac{1}{(2p+1)(p+1)}$$
$$\rho \to \left[\frac{p^2(\mu'_p)^2}{2(\mu'_{2p} - (\mu'_p)^2)}\right] n = \left[\frac{4p+2}{p+5}\right] n\,.$$

By Corollary 2.2, this is an equality for all n, not just a limit for large n, in the case of the L_1 metric. ∎

For L_p metrics with large p, the slope of the line in Theorem 2.6 approaches 4. From there it would be natural to assume that the L_∞ metric should produce a line with a slope of exactly 4. It does produce a line; but with a much shallower slope. The phenomenon of L_∞ behaving differently from L_p for large finite p is not so strange as it may seem: what is actually happening here is that we are taking two limits, the limit for large n and the limit for large p. Usually, we can take two limits in either order and get the same result, but this is one of the rare cases that introductory calculus texts warn us about: in this case it matters which order we take the limits and we get different results depending on that choice. We prove it by integrating the probability density function to find the cumulative distribution function $F(x) = 2x - x^2$. Then standard results on extreme order statistics give the limiting distribution of $\max^{(n)}\{|X-Y|\}$ and Theorem 2.5 applies.

Theorem 2.7
For the space of n-component real vectors with each component uniform in the range $[0,1)$ and the L_∞ metric, the intrinsic dimensionality ρ obeys

$$\rho \to \left[\frac{1}{2 - \pi/2}\right] n\,. \tag{2.14}$$

This is the same line approached for the $L_{\tilde{p}}$ metric where $\tilde{p} = (1+\pi)/(7-2\pi) = 5.77777\,310519\ldots$.

Proof Integration of the probability density function of $|X-Y|$ given by (2.13) above results in the cumulative distribution function of the per-component absolute differences $F(x)$:

$$F(x) = \int_{-\infty}^{x} 2 - 2t\,dt = \begin{cases} 2x - x^2 & 0 \le x \le 1, \\ 0 & x < 0, \\ 1 & x > 1. \end{cases}$$

Now, $F^{-1}(1) = 1$, and we can compute the following limit, which turns out to be a positive constant power of x:

$$\begin{aligned}\lim_{\epsilon\to 0^+} \frac{1-F(F^{-1}(1)-\epsilon x)}{1-F(F^{-1}(1)-\epsilon)} &= \lim_{\epsilon\to 0^+} \frac{1-2(1-\epsilon x)+(1-\epsilon x)^2}{1-2(1-\epsilon)+(1-\epsilon)^2} \\ &= \lim_{\epsilon\to 0^+} \frac{\epsilon^2 x^2}{\epsilon^2} \\ &= x^2 .\end{aligned}$$

Then as described by Arnold, Balakrishnan, and Nagaraja, the cumulative distribution function of the limiting distribution of a maximum of these per-component absolute differences is of the form shown in (2.6); and the norming constants a_n and b_n so that $(\max^{(n)}\{|X-Y|\}-a_n)/n_b \xrightarrow{d} W$ are given by $a_n = F^{-1}(1)$, $b_n = F^{-1}(1) - F^{-1}(1-n^{-1})$ [13, Theorems 8.3.2(ii), 8.3.4(ii)]. In our case, that gives $a_n = 1$, $b_n = 1/\sqrt{n}$; and the cumulative distribution function of W is $G_2(x;\alpha)$ for $\alpha = 2$. Then we can apply (2.9) from Theorem 2.5 as follows:

$$\begin{aligned}\rho &\to \frac{\left(a_n + b_n\Gamma\left(1+{}^1\!/_\alpha\right)\right)^2}{2b_n^2\left(\Gamma\left(1+{}^2\!/_\alpha\right) - \Gamma^2\left(1+{}^1\!/_\alpha\right)\right)} \\ &= \frac{\left(1+\Gamma\left({}^3\!/_2\right)/\sqrt{n}\right)^2}{2^1\!/_n\left(\Gamma(2) - \Gamma^2\left({}^3\!/_2\right)\right)} \\ &= \left[\frac{\left(1+{}^1\!/_2\sqrt{\pi/n}\right)^2}{2-{}^\pi\!/_2}\right] n \\ &\to \left[\frac{1}{2-{}^\pi\!/_2}\right] n .\end{aligned}$$

Substituting that slope value into (2.12) from Theorem 2.6 gives an equivalent p-value for an L_p metric with the same asymptotic behaviour as L_∞: the $L_{\tilde{p}}$ metric for $\tilde{p} = (1+\pi)/(7-2\pi)$. ∎

This result may seem especially surprising because the expression (2.12) seems simple and well-behaved; in particular, it is monotonic in both p and n. So it is strange that (2.14) should not agree with (2.12) for large p. Note, however, that (2.12) is already the result of taking one limit for n in the proof of Theorem 2.6. We are not claiming that (2.14) is the limit of (2.12), but that they are each limits (in different directions) of ρ. Also, the limiting process in the proof of Theorem 2.6 depends on a convergence assumption (existence of moments) which seems likely to break down for infinite p; it would require infinite-order moments. So we are not even taking a different limit of the same

well-behaved formula, but in some sense looking at a different formula for the L_∞ case. Theorem 2.7 and its disagreement with Theorem 2.6 are solidly supported by the experimental evidence in Section 2.3. It seems clear that these results are true, however strange.

It is striking that $\tilde{p}$'s decimal expansion starts with $5.77777\ldots$, suggesting $^{52}/_{9} = 5.7777\,77777\ldots$, but it does not keep up the pattern past the fifth decimal place. Substituting the value $^{52}/_{9}$ and then solving for π reveals the reason for the suggestive pattern: if $\tilde{p}$ were exactly equal to $^{52}/_{9}$, that would make π exactly equal to the well-known rational approximation $^{355}/_{113}$, which is attributed to the 5th Century mathematician Zǔ Chōngzhī[3] [157].

2.1.3 Normal components

The standard normal distribution is another reasonable choice for the distribution of vector components. For finite-p L_p metrics it produces straightforward linear behaviour; in fact, for the Euclidean metric the limiting slope is conveniently equal to 1. However, for L_∞ we find more complicated behaviour, illuminating a fundamental difference between L_∞ and merely L_p for large p. For the case $p = 2$ the distance has a well-known distribution (see note below) for which mean and variance (implying coefficient of variation and so intrinsic dimensionality) are already known. The use of general p, however, seems to require an original proof.

Note 2.3

The proofs of Theorems 2.8 and 2.10 make use of properties of the chi distribution, also known as the generalised Rayleigh distribution. The chi distribution is distinct from the more commonly-seen chi-squared distribution, although they are related by the fact that the square root of a chi-squared random variable will have a chi distribution. Both distributions are described in detail by Johnson, Kotz, and Balakrishnan [115, Chapter 18].

Theorem 2.8

For the space of n-component real vectors with each component standard normal and the L_p metric for finite p, the intrinsic dimensionality ρ obeys

$$\rho \to \left[\frac{p^2 \Gamma^2 \left(\frac{p+1}{2} \right)}{2 \left(\sqrt{\pi} \Gamma \left(p + {}^1/_2 \right) - \Gamma^2 \left(\frac{p+1}{2} \right) \right)} \right] n \,, \tag{2.15}$$

[3] Pinyin Romanisation. Tsu Chhung-Chih in the modified Wade-Giles system used by the reference [157].

and this is an equality for $p = 1$.

Proof When X and Y are standard normal, then the difference $X - Y$ is a normal random variate with mean equal to zero and variance two. The absolute difference $|X - Y|$ has a half-normal distribution (the same as a chi distribution with one degree of freedom) and its probability density function is given by

$$f(x) = \begin{cases} \frac{\sqrt{2}}{\pi} \exp\left(-\frac{x^2}{2\pi}\right) & x \geq 0, \\ 0 & x < 0. \end{cases}$$

Note that the variance of $X - Y$ is two, because it is the difference of two standard normal variates, and so the given density function incorporates the appropriate scaling.

We can find its raw moments as follows:

$$\begin{aligned} \mu'_k &= \int_{-\infty}^{\infty} x^k f(x)\, dx \\ &= \int_0^{\infty} x^k \frac{\sqrt{2}}{\pi} \exp\left(-\frac{x^2}{2\pi}\right) dx \\ &= \pi^{(k-1)/2} 2^{k/2} \int_0^{\infty} y^{(k-1)/2} e^{-y}\, dy \quad \text{substituting } y = \frac{x^2}{2\pi} \\ &= \pi^{(k-1)/2} 2^{k/2} \Gamma\left(\frac{k+1}{2}\right). \end{aligned} \tag{2.16}$$

Then we can apply Theorem 2.1 to find the intrinsic dimensionality:

$$\begin{aligned} \rho &\to \left\lceil \frac{p^2 (\mu'_p)^2}{2(\mu'_{2p} - (\mu'_p)^2)} \right\rceil n \\ &= \left\lceil \frac{p^2 \pi^{p-1} 2^p \Gamma^2\left(\frac{p-1}{2}\right)}{2\left(\pi^{p-1/2} 2^p \Gamma\left(p - 1/2\right) - \pi^{p-1} 2^p \Gamma^2\left(\frac{p-1}{2}\right)\right)} \right\rceil n \\ &= \left\lceil \frac{p^2 \Gamma^2\left(\frac{p+1}{2}\right)}{2\left(\sqrt{\pi}\Gamma\left(p + 1/2\right) - \Gamma^2\left(\frac{p+1}{2}\right)\right)} \right\rceil n\,. \end{aligned}$$

As in the uniform case, Corollary 2.2 makes this an equality for all n, when $p = 1$. ∎

Unlike the uniform case, the slope of the asymptotic line does not increase monotonically with p. It increases from $1/(\pi-2) \approx 0.87597$ for L_1 to unity for L_2, which is the maximum, but then it decreases rapidly with increasing p, reaching zero in the limit of large p. For the L_∞ metric, ρ is sublinear: $\Theta(\log^2 n)$. The argument for linear behaviour from Yianilos [218, Proposition 2] only applies to finite p. However, this result does at least seem more intuitive than what happens with the uniform distribution. With the normal distribution ρ is linear for finite p with a slope that decreases towards zero, then it becomes sublinear for L_∞. Recall that with the uniform distribution it is linear in all cases, but the L_∞ case appears somewhere in the middle of the finite-p cases, not at the limit for large p.

Theorem 2.9
For the space of n-component real vectors with each component standard normal and the L_∞ metric, the intrinsic dimensionality ρ obeys

$$\rho \to \frac{3}{4\pi^2}\left[4\log n - \log\log 2n + \log {}^4\!/_\pi + 2\gamma\right]^2 \tag{2.17}$$

where γ is the Euler-Mascheroni constant.

Proof The distance between two random points in this space is $\max^{(n)}\{|X - Y|\}$, with X and Y being standard normal. By Lemma 2.4, that approaches $\max^{(2n)}\{X - Y\}$. Then each $X - Y$ variable is normally distributed with mean zero and variance two. It is well known that with suitable constants a_{2n} and b_{2n}, then $(\max^{(2n)}\{X - Y\} - a_{2n})/b_{2n} \xrightarrow{d} W$ where W has the cumulative distribution function described in (2.7) [13, 80, 98]. Arnold, Balakrishnan, and Nagaraja give these values for a_n and b_n for a standard normal distribution:

$$a_n = \sqrt{2\log n} - \frac{1}{2}\cdot\frac{\log(4\pi\log n)}{\sqrt{2\log n}}$$

$$b_n = \frac{1}{\sqrt{2\log n}}\,.$$

Because our variables are not standard normal but have variance two, we must include an additional factor of $\sqrt{2}$ to scale the variables to standard normal, and our use of Lemma 2.4 to remove the absolute value function has the effect of

doubling n, so the actual a_n and b_n such that $(\max^{(n)}\{|X-Y|\} - a_n)/b_n \xrightarrow{d} W$ are

$$a_n = 2\sqrt{\log 2n} - \frac{\log(4\pi \log 2n)}{2\sqrt{\log 2n}}$$
$$b_n = \frac{1}{2\sqrt{\log 2n}}\,.$$

Substituting them into (2.10) and applying Theorem 2.5 gives the intrinsic dimensionality:

$$\begin{aligned}
\rho &\to \frac{3(a_{2n} + b_{2n}\gamma)^2}{b_n^2 \pi^2} \\
&= \frac{3}{\pi^2}(2\sqrt{\log 2n})^2 \left[2\sqrt{\log 2n} - \frac{\log(4\pi \log 2n)}{2\sqrt{\log 2n}} + \frac{\gamma}{2\sqrt{\log 2n}}\right]^2 \\
&= \frac{3}{4\pi^2}\left[4\log n - \log\log 2n + \log {}^4\!/_\pi + 2\gamma\right]^2 .
\end{aligned}$$

Note 2.4

Although the limiting distribution of the maximum is unique, the sequences of norming constants leading to that distribution are not. We use the constants given by Arnold, Balakrishnan, and Nagaraja, because they lead to a relatively simple form for the intrinsic dimensionality [13]. The constants suggested by Hall for faster convergence also work, and produce the same most-significant term for intrinsic dimensionality, but they give more complicated lower-order terms [98]. Readers of those two sources should be warned that they use opposite notational conventions: Arnold, Balakrishnan, and Nagaraja use a for additive constants and b for multiplicative constants, as does Galambos, and we follow them; but Hall uses a and α for multiplicative constants and b and β for additive constants [13, 80, 98].

These results describe behaviour for large n. As discussed in Section 2.3, the convergence of intrinsic dimensionality to these asymptotic values may be slow. In the case of the L_2 metric, it is possible to do better, and find better approximations or even exact results for finite n. That is the subject of the next section.

2.2 Normal components, Euclidean distance, and finite n

Random vectors with all components normally distributed are common in statistics. Especially in Euclidean space, such vectors have special properties that make

their intrinsic dimensionality easy to compute. We consider a few cases here.

2.2.1 All components with the same variance

If the variance is the same in every component, or zero in some components and the same among all the others, then the question is easy: the difference between two random vectors is normal in every nonzero component, with the same variance in each one, so we can scale it to be standard normal in every nonzero components. Then each squared difference is a chi-squared random variable with one degree of freedom, the sum of squared differences is also a chi-squared random variable with as many degrees of freedom as there are nonconstant components, the distance is a chi (as opposed to chi-squared) random variable, and its properties are well-known. The details are in the following proof.

Theorem 2.10
The space of vectors of reals where n components have independent and identical normal distributions with variance σ_X and any remaining components are constant, with the Euclidean distance, has intrinsic dimensionality

$$\rho = \frac{1}{2} \cdot \frac{\Gamma^2((n+1)/2)}{\Gamma(n/2)\Gamma((n+2)/2) - \Gamma^2((n+1)/2)}. \tag{2.18}$$

Proof Let $X = Y$ be normal random variates with variance σ_X^2. Their difference $X - Y$ is normal with mean zero and variance $2\sigma_X^2$; and $(X-Y)/(\sqrt{2}\sigma_X)$ is standard normal. So if $\mathbf{x}$ and $\mathbf{y}$ are random vectors in which n components are independent and identical normal random variables with variance σ_X^2, and the remaining components constant, then each $|X-Y|^2/(2\sigma_X^2)$ has a chi-squared distribution with one degree of freedom, and the sum of n of those has a chi-squared distribution with n degrees of freedom. Where D is the distance between $\mathbf{x}$ and $\mathbf{y}$, $D/(\sqrt{2}\sigma_X)$ has a chi (as opposed to chi-squared) distribution with n degrees of freedom. From the results given there, the mean μ_D and variance σ_D (not to be confused with σ_X) of the distance are:

$$\mu_D = \sigma_X \Gamma((n+1)/2)/\Gamma(n/2) \tag{2.19}$$

$$\sigma_D^2 = \sigma_X^2 \left[\Gamma(n/2)\Gamma((n+2)/2) - \Gamma^2((n+1)/2)\right]/\Gamma^2(n/2) \tag{2.20}$$

Substituting into the definition $\rho = \mu_D^2/(2\sigma_D^2)$, the variance cancels and we are left with the result (2.18). ∎

In Subsection 2.1.3 we derive an expression for the asymptotic intrinsic dimensionality of a similar space of normally-distributed vectors; those results are more general in that they cover arbitrary L_p metrics, but also narrower in that they cover only the limit as n approaches infinity. The following result shows that Theorems 2.8 and 2.10 agree in the case where they both apply.

Corollary 2.11
For the space of vectors of reals where n components are independent and identical normal random variables with variance σ_X and any remaining components are constant, with the Euclidean distance, $\rho = n - 1/2 + o(1)$.

Proof We make use of the following properties of the gamma function; (2.21), which is not trivial because it applies for non-integer z, is given by Davis [57, 6.1.15, page 256], and (2.22) is given by Graham, Knuth, and Patashnik [88, page 602]. (That reference uses the notation $z^{\overline{1/2}}$ for $\Gamma(z+1/2)/\Gamma(z)$ [88, page x].)

$$\Gamma(z+1)/\Gamma(z) = z \tag{2.21}$$

$$\Gamma(z+1/2)/\Gamma(z) = \sqrt{z}(1 - 1/(8z) + o(z^{-1})). \tag{2.22}$$

With the substitution $k = n/2$ we can express (2.18) in terms of $\Gamma(k)$ and cancel to find the limit:

$$\begin{aligned}
\rho &= \frac{1}{2} \cdot \frac{\Gamma((n+1)/2)}{\Gamma(n/2)\Gamma((n+2)/2) - \Gamma^2((n+1)/2)} \\
&= \frac{1}{2} \cdot \frac{\Gamma^2(k)k\left(1 - 1/(4k) + o(k^{-1})\right)}{\Gamma^2(k)k - \Gamma^2(k)k\left(1 - 1/(4k) + o(k^{-1})\right)} \\
&= \frac{1}{2} \cdot \frac{1 - 1/(4k) + o(k^{-1})}{1/(4k) + o(k^{-1})} \\
&= n - 1/2 + o(1).
\end{aligned}$$

■

2.2.2 Exact result for $n = 2$ and distinct variances

Real-life data often approximates a multivariate normal distribution with almost all the variance confined to a few dimensions. That is the model, for instance, typically contemplated by principal component analysis [21]. Theorem 2.10 describes the situation where the variation in the data is divided equally among an integer number of dimensions; for instance, approximately 0.87597 when

$n = 1$ and 1.82990 when $n = 2$. We may ask how ρ evolves as we move from one of those to the other; that is the case analysed in this section. The result and its proof are complicated and it is not clear that they can be pushed beyond two dimensions; however, we give some useful approximations.

Theorem 2.12
For two-component vectors with the components chosen independently from normal distributions with variances σ_1^2, σ_2^2, the intrinsic dimensionality ρ is given by

$$\rho = \left(\frac{8}{\pi} \left((1+\tau)(1-\tau)\, {}_2F_1\left({}^3\!/_4, {}^5\!/_4; 1; \tau^2 \right) \right)^{-2} - 2 \right)^{-1} \tag{2.23}$$

where $\tau = (\sigma_1^2 - \sigma_2^2)/(\sigma_1^2 + \sigma_2^2)$.

Proof Suppose $X_1 = Y_1, X_2 = Y_2$ are normal with variances σ_1^2, σ_2^2. Then $(X_1 - Y_1), (X_2 - Y_2)$ are normal with mean 0 and variances $\sqrt{2}\sigma_1^2, \sqrt{2}\sigma_2^2$. Then $(X_1 - Y_1)^2, (X_2 - Y_2)^2$ have the two-parameter gamma distribution, as described by Johnson, Kotz, and Balakrishnan [115, Chapter 17], with shape parameter ${}^1\!/_2$ and scale parameters σ_1^2, σ_2^2, and each has the probability density function

$$f(x) = \frac{e^{-x/\sigma^2}}{\sigma\sqrt{x\pi}}.$$

These distributions are the same as scaled chi-squared distributions of one degree of freedom [115, Chapter 18].

Where D is the distance between $\mathbf{x}$ and $\mathbf{y}$, the probability density function of D^2 is given by the following, where $I_0(x)$ is the modified Bessel function of the first kind of order 0. We use a formula from Gradshteyn and Ryzhik to do the integration [87, 3.364(1.), page 341].

$$\begin{aligned} f(x) &= \int_0^x \frac{e^{-t/\sigma_1^2}}{\sigma_1\sqrt{\pi t}} \cdot \frac{e^{-(x-t)/\sigma_2^2}}{\sigma_2\sqrt{\pi(x-t)}}\, dt \\ &= \frac{1}{\sigma_1\sigma_2\pi} \int_0^x \frac{\exp\left(-t/\sigma_1^2 - (x-t)/\sigma_2^2\right)}{\sqrt{t(x-t)}}\, dt \\ &= \frac{1}{\sigma_1\sigma_2} \exp\left(-x\frac{\sigma_1^2 + \sigma_2^2}{2\sigma_1^2\sigma_2^2} \right) I_0\left(x\frac{\sigma_1^2 - \sigma_2^2}{2\sigma_1^2\sigma_2^2} \right). \end{aligned}$$

We could find $\mathrm{E}[D^2]$ by integrating $xf(x)$, but it is easier to use the known mean of the gamma distribution, which gives $\mathrm{E}[D^2] = (\sigma_1^2 + \sigma_2^2)/2$. For $\mathrm{E}[D]$, however, we must integrate. We begin by expanding the modified Bessel function into its power series about 0 [161, 9.6.12, page 375].

$$\begin{aligned}\mathrm{E}[D] &= \int_0^\infty \sqrt{x} f(x)\, dx \\ &= \int_0^\infty \frac{\sqrt{x}}{\sigma_1\sigma_2} \exp\left(-x\frac{\sigma_1^2+\sigma_2^2}{2\sigma_1^2\sigma_2^2}\right) I_0\left(x\frac{\sigma_1^2-\sigma_2^2}{2\sigma_1^2\sigma_2^2}\right) \\ &= \int_0^\infty \frac{\sqrt{x}}{\sigma_1\sigma_2} \exp\left(-x\frac{\sigma_1^2+\sigma_2^2}{2\sigma_1^2\sigma_2^2}\right) \sum_{k=0}^\infty \frac{x^{2k}}{(k!2^k)^2}\left(\frac{\sigma_1^2-\sigma_2^2}{2\sigma_1^2\sigma_2^2}\right)^{2k}.\end{aligned}$$

Then we re-arrange the expression, cancelling where possible. The terms that do not depend on x are moved outside the integral.

$$\mathrm{E}[D] = \sum_{k=0}^\infty (\sigma_1\sigma_2)^{-(4k+1)}(\sigma_1^2-\sigma_2^2)^{2k}2^{-4k}(k!)^{-2} \cdot \int_0^\infty \exp\left(-x\frac{\sigma_1^2+\sigma_2^2}{2\sigma_1^2\sigma_2^2}\right) x^{2k+1/2}\, dx\,.$$

That puts the integral into the form $\int \exp(-xa)x^b\, dx$ with a and b constant in relation to x [87, 3.381(4.), page 342]. On integration it produces the gamma function:

$$\mathrm{E}[D] = \sum_{k=0}^\infty \sigma_1^2\sigma_2^2\left(\frac{\sigma_1^2-\sigma_2^2}{2\sigma_1^2\sigma_2^2}\right)^{2k} \cdot (\sigma_1^2+\sigma_2^2)^{-3/2}2^{-2k}2^{3/2}(k!)^{-2}\Gamma\left(2k+{}^3\!/_2\right).$$

Recall that !! denotes the *double factorial* from Definition 1.13, given by

double factorial

$$n!! = \begin{cases} 1\cdot 3\cdot 5\cdot\cdots\cdot n & \text{odd } n > 0, \\ 2\cdot 4\cdot 6\cdot\cdots\cdot n & \text{even } n > 0, \\ 1 & n \in \{-1, 0\}\,. \end{cases} \tag{2.24}$$

We move the terms that do not depend on k out of the summation to make the infinite sum as simple as possible, and replace the gamma function with the formula given by Davis for its value at half-integer arguments, which uses double factorial [57, 6.1.12, page 255].

$$\mathrm{E}[D] = \sigma_1^2\sigma_2^2\left(\frac{2}{\sigma_1^2+\sigma_2^2}\right)^{3/2} \cdot \sum_{k=0}^\infty \left(\frac{\sigma_1^2-\sigma_2^2}{\sigma_1^2+\sigma_2^2}\right)^{2k} \cdot \frac{1}{(k!2^k)^2}\Gamma\left((2k+1)+{}^1\!/_2\right)$$

$$= \sigma_1^2\sigma_2^2 \left(\frac{2}{\sigma_1^2+\sigma_2^2}\right)^{3/2} \cdot \sum_{k=0}^{\infty} \left(\frac{\sigma_1^2-\sigma_2^2}{\sigma_1^2+\sigma_2^2}\right)^{2k} \cdot \frac{1}{(k!2^k)^2} \cdot \frac{1\cdot 3\cdot 5\cdot \cdots \cdot (4k+1)}{2^{2k+1}} \sqrt{\pi}$$

$$= \sigma_1^2\sigma_2^2 \left(\frac{2}{\sigma_1^2+\sigma_2^2}\right)^{3/2} \cdot \frac{\sqrt{\pi}}{2} \sum_{k=0}^{\infty} \left(\frac{\sigma_1^2-\sigma_2^2}{\sigma_1^2+\sigma_2^2}\right)^{2k} \cdot \frac{(4k+1)!!}{2^{4k}(k!)^2}.$$

If we let $\tau = (\sigma_1^2 - \sigma_2^2)/(\sigma_1^2 + \sigma_2^2)$ and let c_k be the coefficient of τ^{2k} in the infinite sum, we can compute the ratio of successive coefficients c_{k+1}/c_k and find that it is a rational function of k. That makes the infinite sum a hypergeometric function of τ^2. It also proves that the series converges for $|\tau| < 1$, because the limit of the rational function turns out to be 1, so the ratio of successive terms is less than one for $\tau^2 < 1$; and with positive σ_1^2 and $\sigma_2^2 \neq \sigma_1^2$, $|\tau| < 1$ always holds so the series always converges.

$$\frac{c_{k+1}}{c_k} = \frac{(4(k+1)+1)!!}{2^{4(k+1)}((k+1)!)^2} \Big/ \frac{(4k+1)!!}{2^{4k}(k!)^2}$$

$$= \frac{\left(k+{}^3\!/_4\right)\left(k+{}^5\!/_4\right)}{(k+1)^2}$$

$$\sum_{k=0}^{\infty} c_k \tau^{2k} = {}_2F_1\left({}^3\!/_4, {}^5\!/_4; 1; \tau^2\right).$$

From the definition of τ it also follows that

$$\frac{(\sigma_1^2+\sigma_2^2)^2}{\sigma_1^2\sigma_2^2} = \frac{4}{(1+\tau)(1-\tau)}.$$

Then we can calculate the intrinsic dimensionality ρ as follows:

$$\rho = \frac{\mathrm{E}^2[D]}{2(\mathrm{E}[D^2] - \mathrm{E}^2[D])}$$

$$= \left(\frac{2\,\mathrm{E}[D^2]}{\mathrm{E}^2[D]} - 2\right)^{-1}$$

$$= \left(\frac{(\sigma_1^2+\sigma_2^2)^4}{2\pi\sigma_1^4\sigma_2^4\left({}_2F_1\left({}^3\!/_4, {}^5\!/_4; 1; \tau^2\right)\right)^2} - 2\right)^{-1}$$

$$= \left(\frac{8}{\pi}\left((1+\tau)(1-\tau)\,{}_2F_1\left({}^3\!/_4, {}^5\!/_4; 1; \tau^2\right)\right)^{-2} - 2\right)^{-1}. \qquad \blacksquare$$

Scaling the original random vectors by a constant will not affect the value of τ and so the intrinsic dimensionality is unchanged by scaling. A plot of this function as a function of τ is shown in Figure 2.2; however, it may be difficult

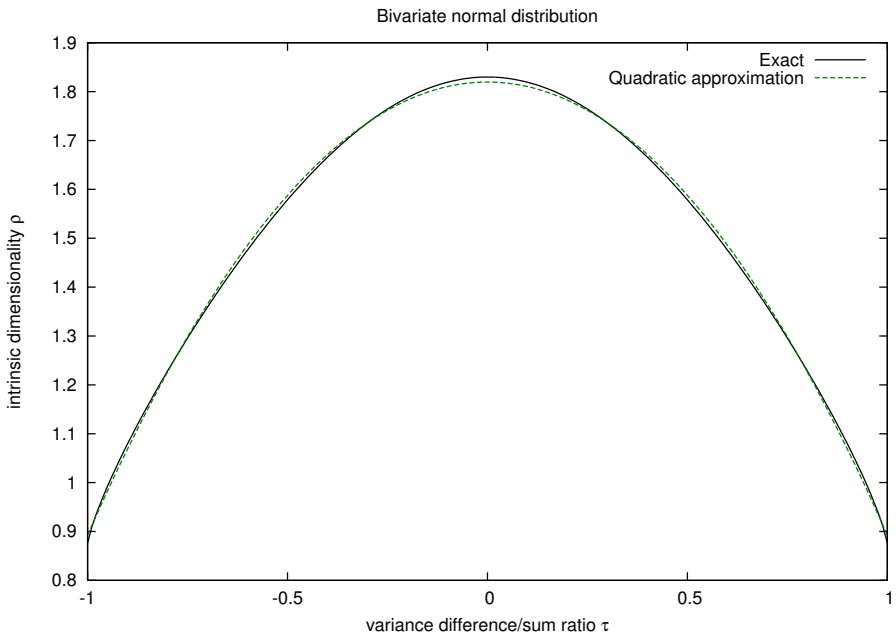


Figure 2.2: Intrinsic dimensionality for the bivariate normal distribution as a function of τ.

to interpret because the substituted variable τ has little intuitive significance; it was chosen to make the algebra easier. In Figure 2.3, the function values are plotted against σ_2^2/σ_1^2 (we have $\tau = (1-\sigma_2^2/\sigma_1^2)/(1+\sigma_2^2/\sigma_1^2)$), so it shows what happens if a random variable in one dimension has a second component added whose variance grows to match and surpass that of the first component.

As can be seen in Figure 2.2, the value of ρ is reasonably approximated by a quadratic function of τ. The approximation shown, obtained by numerical curve-fitting, is

$$\hat{\rho} = 1.81971 - 0.927058\tau^2 .$$

This approximation offers some hope that a simple formula could provide adequate results for higher-dimensional distributions, avoiding the need for an exact analysis.

2.2.3 Approximation for larger n

Consider the more general case, where the components of the vectors are chosen independently from normal distributions with variances not necessarily equal. So far we have exact results for the special cases where there are any number

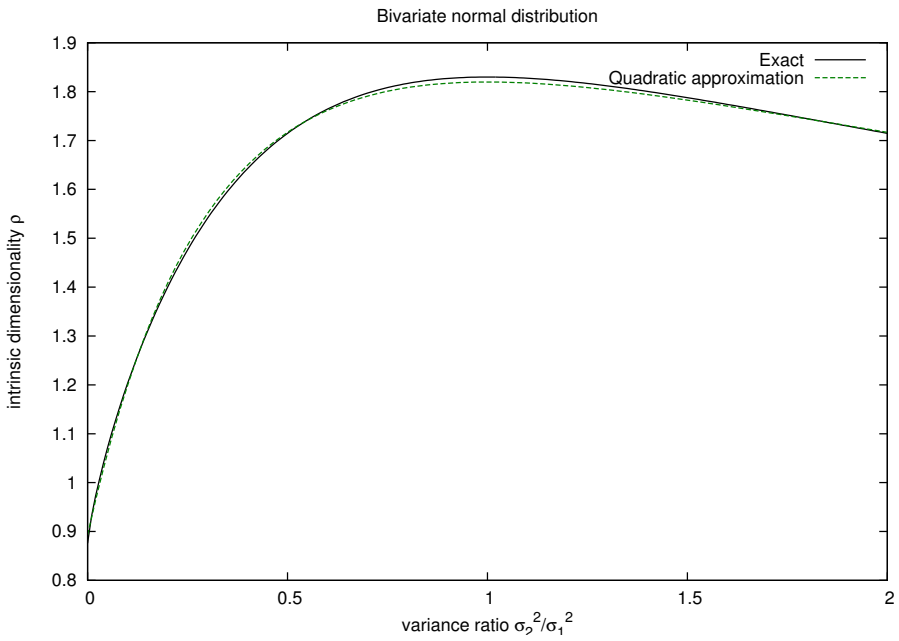


Figure 2.3: Intrinsic dimensionality for the bivariate normal distribution as a function of σ_2^2/σ_1^2.

of normal components but they all have the same variance (Theorem 2.10) and where there are exactly two components with variances not necessarily equal (Theorem 2.12). In all cases we are interested in the distribution of a sum of gamma-distributed random variables. For a large number of random variables (that is, a large number of vector components) the sum approaches a normal distribution, which is also the limit of a gamma distribution for large shape parameter. In the special case of adding two with the same scale parameter the distribution of the sum of two gamma variables is also gamma.

So we might approximate the sum as *always* having a gamma distribution. Given the variances of the components in the vectors, we can calculate exactly what the mean and variance of the sum should be. If we choose the unique gamma distribution with that mean and variance, we have a distribution that approximates the true distribution of the sum. That is the approach used in this section. The approximation is exact if all variances happen to be the same, and even if they do not, it improves as the number of components increases, because by the Central Limit Theorem, the sum approaches a normal distribution, as does the gamma approximation.

Suppose each component of $\mathbf{x}$ and $\mathbf{y}$ is chosen from a normal distribution, independent but not necessarily identically distributed, with per-component vari-

ances $\sigma_1^2, \sigma_2^2, \ldots, \sigma_n^2$. The random variables $(X_i - Y_i)^2$ for integer $i \in \{1, 2, \ldots, n\}$ are gamma distributed, all with shape parameter $1/2$ and with scale parameters $\sigma_1^2, \sigma_2^2, \ldots, \sigma_n^2$. The mean and variance of their sum D^2 are just the sums of the means and variances of the individual variables:

$$\begin{aligned} \mathrm{E}[D^2] &= 1/2(\sigma_1^2 + \sigma_2^2 + \cdots + \sigma_n^2) \\ \mathrm{V}[D^2] &= 1/2(\sigma_1^4 + \sigma_2^4 + \cdots + \sigma_n^4). \end{aligned}$$

If we approximate the distribution of D^2 with a gamma distribution with some parameters α (shape) and β (scale), then we have $\mathrm{E}[D^2] = \alpha\beta$ and $\mathrm{V}[D^2] = \alpha\beta^2$. Then α is given by

$$\alpha = \frac{(\sigma_1^2 + \sigma_2^2 + \cdots + \sigma_n^2)^2}{2(\sigma_1^4 + \sigma_2^4 + \cdots + \sigma_n^4)}. \tag{2.25}$$

Note the resemblance between (2.25) and the definition of ρ. We do not need to find β explicitly (although it would be easy to do so) because it will cancel out later.

The probability density function of the gamma distribution for D^2 is

$$f(x) = \frac{x^{\alpha-1} e^{-x/\beta}}{\beta^\alpha \Gamma(\alpha)}.$$

So the assumption that D^2 has a gamma distribution allows us to approximate $\mathrm{E}[D]$ by integrating $\sqrt{x} f(x)$, as in Theorem 2.12:

$$\mathrm{E}[D] \approx \int_0^\infty \sqrt{x} f(x)\, dx. \tag{2.26}$$

Gradshteyn and Ryzhik give a formula for the integral [87, 3.381(4.), page 342]:

$$\begin{aligned} \int_0^\infty \sqrt{x} f(x)\, dx &= \frac{1}{\beta^\alpha \Gamma(\alpha)} \int_0^\infty x^{(\alpha+1/2)-1} e^{-(1/\beta)x}\, dx \\ &= \frac{1}{\beta^\alpha \Gamma(\alpha)} \cdot \frac{\Gamma(\alpha + 1/2)}{(1/\beta)^{\alpha+1/2}} \\ &= \sqrt{\beta} \frac{\Gamma(\alpha + 1/2)}{\Gamma(\alpha)}. \end{aligned} \tag{2.27}$$

Substituting $\mathrm{E}[D^2] = \alpha\beta$ and (2.27) into the intrinsic dimensionality formula, β cancels and we can multiply through by $\Gamma^2(\alpha)$ to simplify the expression.

$$\rho = \frac{\mathrm{E}^2[D]}{2(\mathrm{E}[D^2] - \mathrm{E}^2[D])}$$

$$\approx \beta \left(\frac{\Gamma(\alpha + 1/2)}{\Gamma(\alpha)} \right)^2 \Bigg/ 2 \left[\alpha\beta - \beta \left(\frac{\Gamma(\alpha + 1/2)}{\Gamma(\alpha)} \right)^2 \right]$$
$$= \frac{1}{2} \cdot \frac{\Gamma^2(\alpha + 1/2)}{\alpha\Gamma^2(\alpha) - \Gamma^2(\alpha + 1/2)}.$$

Finally, we apply $\alpha\Gamma(\alpha) = \Gamma(\alpha + 1)$ to get

$$\rho \approx \frac{1}{2} \cdot \frac{\Gamma^2(\alpha + 1/2)}{\Gamma(\alpha)\Gamma(\alpha + 1) - \Gamma^2(\alpha + 1/2)}. \tag{2.28}$$

Note 2.5

The approximation sign $\approx$ is used here in an informal sense, distinct from the formal approximation up to lower-order terms denoted by $\rightarrow$ and defined in Definition 1.16. The approximations (2.26)–(2.28) represent the values $\mathrm{E}[D]$ and ρ *would* have if D^2 had a gamma distribution. In general, it does not have a gamma distribution; but it does, and the approximations are exact, for the special case of all variances equal (by Theorem 2.10), and in the limit for high dimensions (by the Central Limit Theorem, noting that the normal distibution is the limit of a gamma distribution). We discuss below the question of how good the approximation may be for other cases.

The formula for ρ in (2.28) is the same as (2.18) with the substitution $\alpha = n/2$. That is the value of α when all the n variances are equal, so the approximation is exact in the case of equal variances.

The value α from (2.25) seems to serve as a proxy for the number of dimensions: in the unequal-variances case, it is the number of equal-variance dimensions that would produce an equivalent distribution for D^2, defined by mean and variance. That might make α a contender as a dimensionality measure itself. For any distribution of vectors we could compute α based on the componentwise distributions and attempt to claim that the vector distribution has similar properties to the distribution of vectors with α number of independent and identical standard normal components. An advantage of α is that it takes on the intuitive value of n in the case of n normal components, whereas ρ only does so asymptotically. A disadvantage is that α seems to be specific not only to vectors, but to normally distributed vectors with independent components and the L_2 metric. Whether this number would be useful when applied to other cases is not clear.

How good an approximation is (2.28)? When the variance is distributed equally among the n vector components, it is exact. If the variance is distributed unequally between two components, we have an exact value from Theorem 2.12

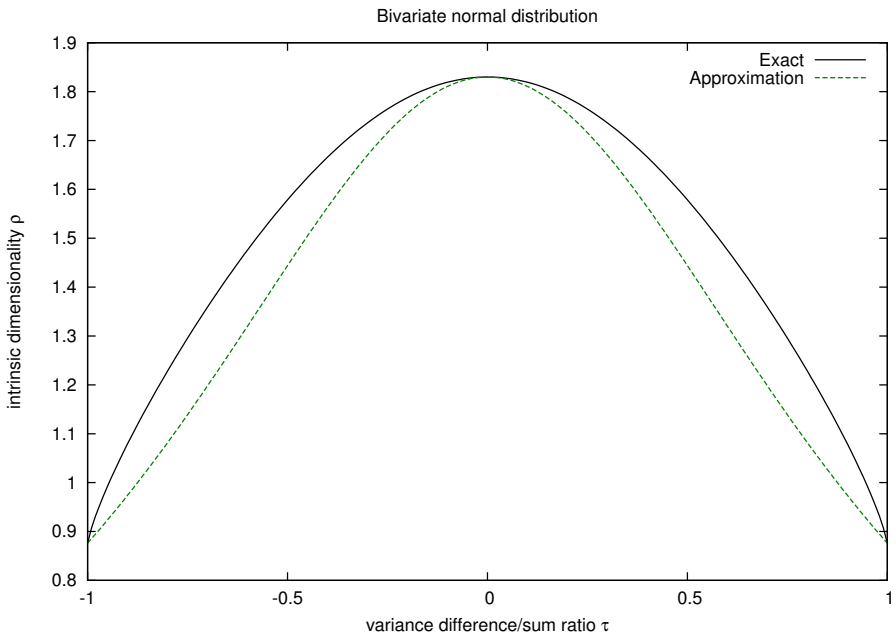


Figure 2.4: Comparison of exact ρ for bivariate normal with its approximation from (2.28).

to compare against, and Figure 2.4 shows both values plotted against the variable τ used earlier. ($\alpha = 1/(\tau^2 + 1)$.) In this comparison, (2.28) does not look particularly impressive; the difference between the two is plotted in Figure 2.5, and is as large as 0.16 in the worst case. However, this the worst case for the approximation: with only two components contributing to the sum, the result will be as far from gamma-distributed as it can be. When the correct intrinsic dimensionality is larger, we should expect the gamma approximation to be better. In Subsection 2.3.2 we present some experimental results for cases that may be more realistic.

2.3 Experimental results with discussion

Most of the results of this chapter are in the form of asymptotic approximations, suggesting the question of how well the approximations hold in practical cases. We present some experimental results on that question here.

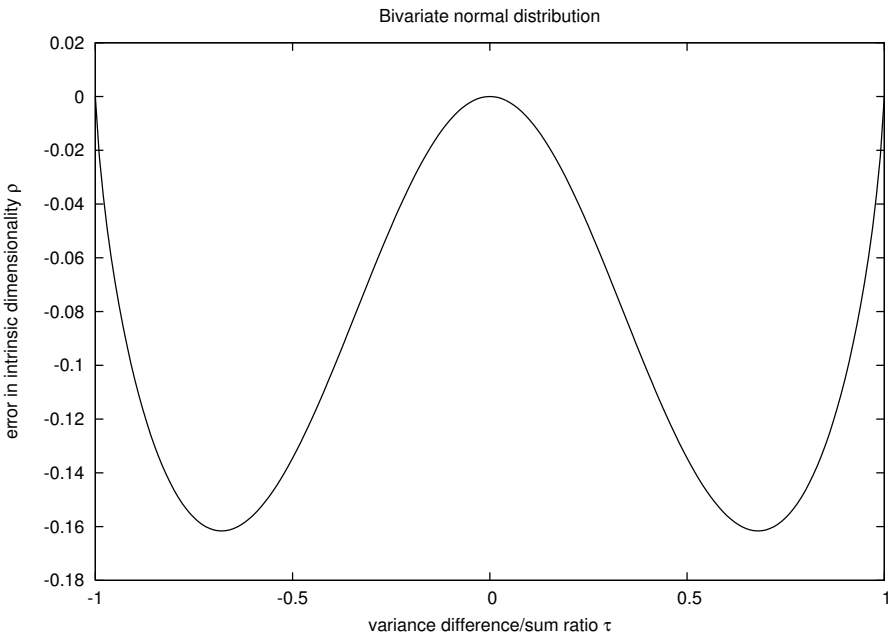


Figure 2.5: Error in the (2.28) approximation.

2.3.1 All components independent and identically distributed

Chávez and Navarro describe an experiment in which, for L_1, L_2, and L_∞ and numbers of components between two and twenty, they chose one million pairs of points in each space and measured the distances to compute the intrinsic dimensionality. They report that the intrinsic dimensionality values fell on lines, as expected [40, Fig. 3]. However, the slopes they reported were much shallower than predicted by our theory in Theorems 2.6 and 2.7. We repeated the experiment and extended it to other distributions, metrics, and vector lengths. These results appeared in SPIRE'05 [190].

Results for vectors of up to 20 components with uniformly-distributed components are shown in Figure 2.6. Note that except for L_1 (for which ρ is exactly a linear function of n by Corollary 2.2), the experimental values do not fall on their asymptotic lines. The L_2 points are close, but the L_{256} and L_∞ points, which appear to coincide, are far from their lines. In Figure 2.7 we see the same experiment extended to vectors of up to 2^{20} (a little over a million) components. Note that that plot has logarithmic axes to accommodate the large dynamic range of both n and ρ. The points for all the metrics are seen to converge on their respective theoretical lines, but the ones for L_{256} do so very slowly. As predicted, L_∞ and $L_{5.778}$ approach the same line and each other, a line quite different from

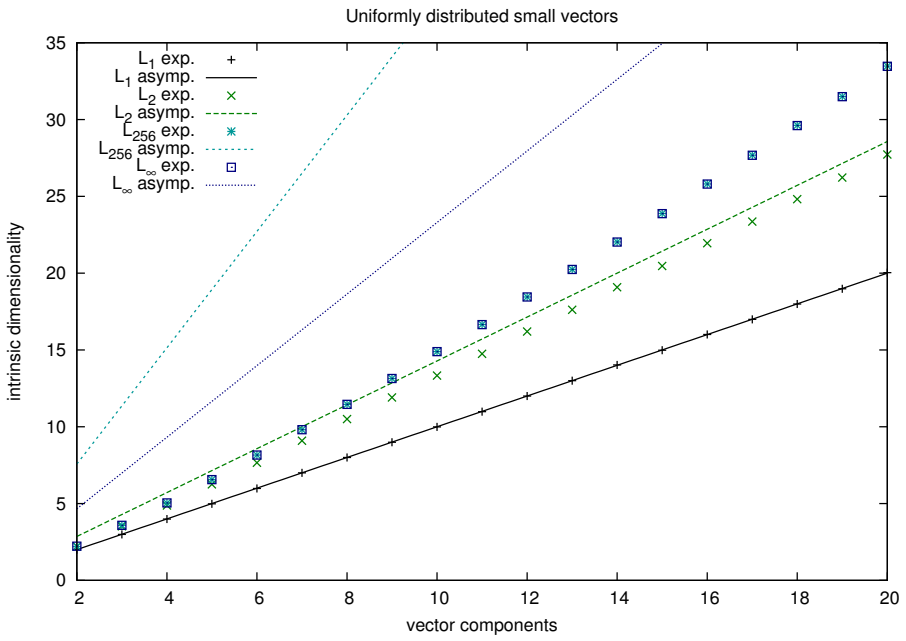


Figure 2.6: Experimental results: short vectors, uniform components.

the one for L_{256}.

A similar set of results for vectors with standard normal components is shown in Figures 2.8 and 2.9. For vectors with up to 20 components, the L_1 and L_2 points seem close enough to their predicted lines, but the L_4 and L_8 points go off in other directions. The L_{16} points seem very close to the L_∞ points. Note that we did not even plot the asymptotic line for L_{16} in Figure 2.8 because with its slope of $^{143}/_{52176} \approx 0.0027407$ it would be lost in the lower border of the plot. In Figure 2.9 we see that for larger vectors the points do approach their respective limiting functions, but L_{16} takes a long time to converge. It is still quite far above the line even with over a million components. In that range L_∞ remains visually slightly off its curve too, possibly because of the negative $\log \cdot \log\log$ term in (2.17), which is next after $\log^2$ in the expansion and still large enough at $n = 2^{20}$ to have a noticeable effect. The maximum of a collection of normal random variables, which essentially describes the L_∞ metric here, is notorious for slow convergence [98].

From these results one can see that normal distributions have smaller values of intrinsic dimensionality than uniform distributions for vectors of the same length; in the case of L_∞ asymptotically so, but even for finite p, the constants are smaller for normal distributions. So the question arises, are normal distributions really easier to index, or is this just a sign that ρ is an unreliable measure? Our

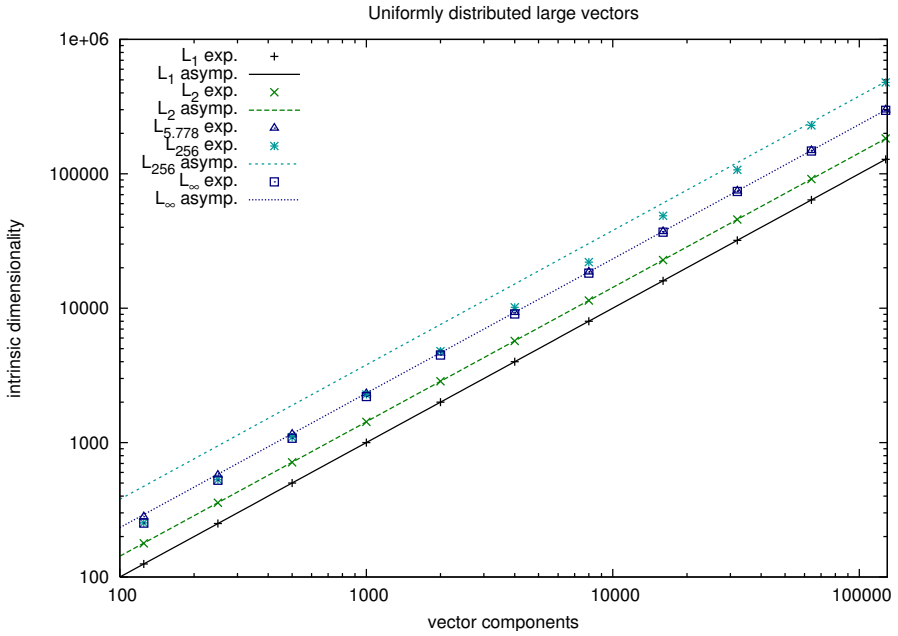


Figure 2.7: Experimental results: long vectors, uniform components.

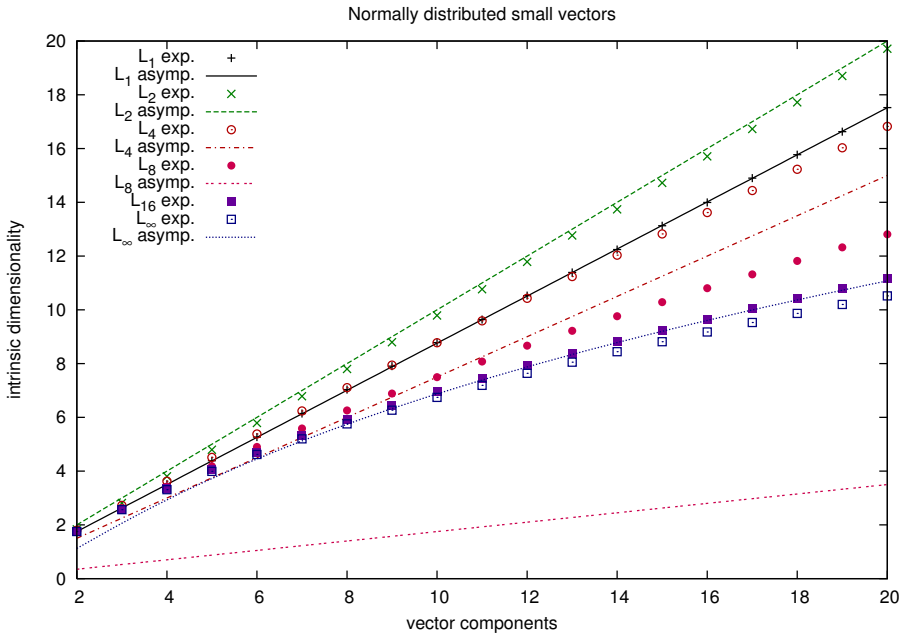


Figure 2.8: Experimental results: short vectors, normal components.

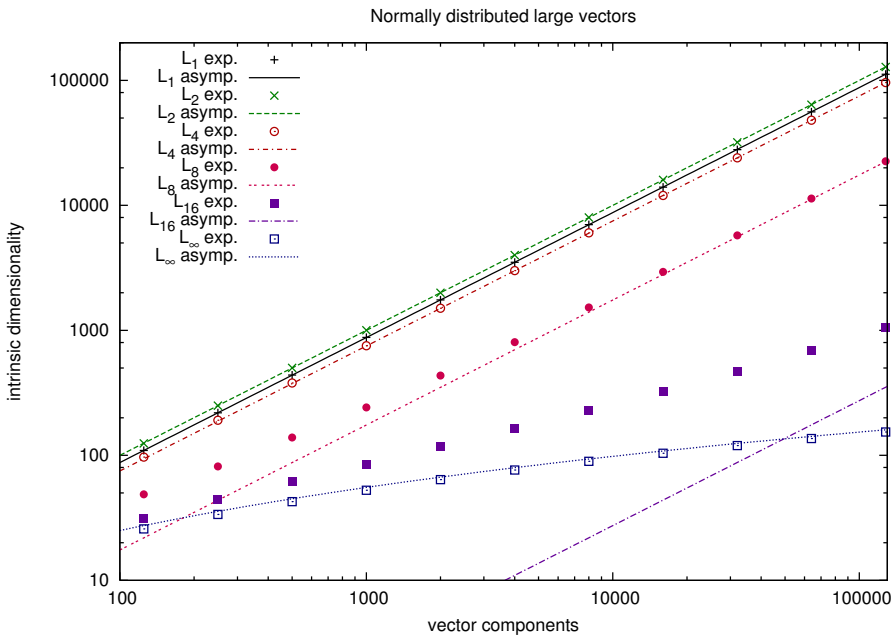


Figure 2.9: Experimental results: long vectors, normal components.

suggestion is that "normal distributions really are easier to index." [190]

In a high-dimensional normal distribution, it will tend to be the case that most components will have small magnitudes, and there will be a few outliers of much larger magnitude. That is what we expect to see whenever we sample from a normal distribution. In that case, when we measure the distance between two vectors (and especially if we use L_p for large p, let alone L_∞) the distance will be dominated by the largest-magnitude components. In fact we could make a reasonable guess at the distance between two vectors by *only* looking at their largest-magnitude components. That is the insight behind the pyramid-tree technique of Berchtold, Böhm, and Kriegel [27]. They organise vectors by greatest-magnitude components, using brute-force search after that, and the resulting data structure works especially well with high-dimensional normal distributions, because with those, most of the story about a point is told by its greatest-magnitude component.

On the other hand, in a high-dimensional uniform distribution, the distribution of componentwise differences is triangular, with much heavier tails than a normal distribution. There will be many components of magnitude comparable to the largest one. The tendency for one component to dominate in the distance calculation is much less strong. Then approximating a vector by one or a few of its components does not work so well; and therefore searching for vectors is a

more difficult task. A measure of dimensionality should rate such a distribution more difficult, as ρ does.

2.3.2 Components independent and normal but not identically distributed

Evaluation of the quality of approximation described in Subsection 2.2.3 is hampered by the fact that we have no general exact result to compare against. However, it is easy to generate random vectors and measure their intrinsic dimensionality experimentally. In Table 2.1 we show the results of such an experiment, designed to simulate the kinds of distributions modelled by principal component analysis. These distributions are generally multivariate normal with one or a small number of components containing most of the variance and much smaller variance in the remaining components.

For each set of variances $\sigma_1^2 \geq \sigma_2^2 \geq \sigma_3^2 \geq 0$ with $\sigma_1^2 + \sigma_2^2 + \sigma_3^2 = 1$, σ_1^2 a multiple of 0.1, and σ_2^2 a multiple of 0.05, we generated 10^7 pairs of points and measured the resulting intrinsic dimensionality. We repeated each experiment 20 times, and the table shows the mean $\bar{\rho}$ and sample standard deviation s of the resulting ρ, along with the approximate value $\hat{\rho}$ predicted by (2.28) and, where applicable, the exact value from Theorem 2.10. The approximation is worst in the case of $\sigma_1^2 = 0.6$, $\sigma_2^2 = \sigma_3^2 = 0.2$, with an error of almost 21%. It generally improves as the intrinsic dimensionality increases.

To model a similar case but with higher dimensionality, we used vectors with ten components, normal distribution in each component, and the variance distributed exponentially: $\sigma_i^2 = (\sigma_1^2)^i$, for different values of σ_1^2. As in the previous experiment, we used 10^7 pairs of points for each distribution and repeated each experiment 20 times. The results are shown in Table 2.2. As expected, the approximation improves as the intrinsic dimensionality increases.

For large enough intrinsic dimensionality it seems clear that (2.28) gives a number close enough to be useful, but real-life data sets often seem to have low dimensionality and the error may be too great for the approximation to be useful there. Further improvement goes beyond the planned scope of the present work, but one approach that might be fruitful would be to seek a better approximation of the sum of chi-squared variables. For instance, the same approach used in Subsection 2.2.3 of fitting the sum with a two-parameter gamma distribution could be used to fit a Gram-Charlier expansion, or a similar one based on the gamma distribution, with the hope of getting a more accurate estimate of $\mathrm{E}[D]$ [115, pages 25–33 and 343].

σ_1^2	σ_2^2	σ_3^2	Thm. 2.10 ρ	(2.28) $\hat{\rho}$	Expt. $\bar{\rho}$	Expt. s
1.00	0.00	0.00	0.87597	0.87597	0.87611	0.00043
0.90	0.10	0.00	1.22836	1.08182	1.22841	0.00055
0.90	0.05	0.05		1.08888	1.28921	0.00061
0.80	0.20	0.00	1.47599	1.32020	1.47626	0.00095
0.80	0.15	0.05		1.35188	1.60854	0.00080
0.80	0.10	0.10		1.36278	1.64567	0.00081
0.70	0.30	0.00	1.66582	1.56332	1.66559	0.00064
0.70	0.25	0.05		1.63818	1.86062	0.00066
0.70	0.20	0.10		1.68650	1.96347	0.00099
0.70	0.15	0.15		1.70323	1.99707	0.00117
0.60	0.40	0.00	1.78769	1.75538	1.78795	0.00106
0.60	0.35	0.05		1.88991	2.03645	0.00114
0.60	0.30	0.10		1.99882	2.19730	0.00100
0.60	0.25	0.15		2.07016	2.29422	0.00103
0.60	0.20	0.20		2.09505	2.32665	0.00134
0.50	0.50	0.00	1.82990	1.82990	1.82991	0.00089
0.50	0.45	0.05		2.02207	2.11547	0.00074
0.50	0.40	0.10		2.20063	2.32352	0.00089
0.50	0.35	0.15		2.34792	2.47522	0.00131
0.50	0.30	0.20		2.44577	2.56780	0.00104
0.50	0.25	0.25		2.48015	2.59963	0.00108
0.40	0.40	0.20		2.58912	2.65400	0.00098
0.40	0.35	0.25		2.70772	2.74209	0.00120
0.40	0.30	0.30		2.74962	2.77156	0.00140

Table 2.1: Comparison of Theorem 2.10 and the approximation from (2.28) with experimental results.

σ_1^2	(2.28) $\hat{\rho}$	Expt. $\bar{\rho}$	Expt. s
0.05	0.97434	1.08528	0.00048
0.10	1.08438	1.25468	0.00057
0.15	1.20817	1.42499	0.00047
0.20	1.34829	1.60404	0.00072
0.25	1.50803	1.79869	0.00093
0.30	1.69159	2.01355	0.00064
0.35	1.90443	2.25524	0.00124
0.40	2.15367	2.53193	0.00157
0.45	2.44870	2.85278	0.00117
0.50	2.80170	3.22979	0.00152
0.55	3.22820	3.67560	0.00144
0.60	3.74694	4.20965	0.00186
0.65	4.37829	4.84617	0.00220
0.70	5.13930	5.59344	0.00250
0.75	6.03296	6.44938	0.00381
0.80	7.03008	7.37458	0.00345
0.85	8.04826	8.28791	0.00382
0.86	8.24279	8.45981	0.00380
0.87	8.43123	8.62514	0.00364
0.88	8.61216	8.78238	0.00421
0.89	8.78420	8.93185	0.00371
0.90	8.94595	9.06999	0.00482
0.91	9.09608	9.20014	0.00409
0.92	9.23335	9.31748	0.00392
0.93	9.35664	9.42141	0.00369
0.94	9.46496	9.51274	0.00508
0.95	9.55749	9.59158	0.00446
0.96	9.63361	9.65407	0.00514
0.97	9.69288	9.70540	0.00296
0.98	9.73511	9.74003	0.00469
0.99	9.76027	9.76178	0.00453

Table 2.2: Comparison of the approximation from (2.28) with experimental results.

Chapter 3

Real vectors: distance permutations

Vectors of real numbers are typical candidates for indexing by distance permutations: by reducing each point in a database to a small identifying code, we can hope to save both time and space in indexing and searching the database. The hope is that similar objects will have similar distance permutations, so by examining the distance permutations we can quickly rule database points out of a query without having to calculate a possibly expensive metric on the database points.

Recall that the *distance permutation* of a point y with respect to k points $x_1, x_2, \ldots, x_k$ called the *sites* is the permutation that sorts the site indices into increasing order of distance from y, breaking ties by placing the lower-index site first (Definition 1.25). If the distance permutation is denoted by Π_y, we have for any indices $1 \leq i < j \leq k$, $d(x_{\Pi_y(i)}, y) < d(x_{\Pi_y(j)}, y)$ or $d(x_{\Pi_y(i)}, y) = d(x_{\Pi_y(j)}, y)$ and $\Pi_y(i) < \Pi_y(j)$.

distance permutation
site

In this chapter we discuss the question of how many distinct values the distance permutation may have over the points in a real vector space. The maximum number of values determines how much space the index needs to store each permutation. We describe the connection between this problem and work on Voronoi diagrams and oriented matroids, and discuss the difficulties encountered in non-Euclidean spaces, particularly L_4. We also give an exact analysis for Euclidean space and an asymptotic upper bound for L_1 and L_∞ spaces, improving on the previous best known storage space bound for a distance permutation index. The results for L_1, L_2, and L_∞ appeared, without the detailed proofs we give here, in SISAP'08 [191]. We begin by defining a notation for the number of distance permutations.

Definition 3.1

Let $N_{n,p}(k)$ represent the maximum number of distinct distance permutations generated by any choice of k sites in the space of n-dimensional real

$N_{n,p}(k)$

vectors with the L_p metric. Where S is the space and $\Pi_{\mathbf{y}}$ is the distance permutation as defined above,

$$N_{n,p}(k) = \max_{\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_k \in S} \left|\{\Pi_{\mathbf{y}} | \mathbf{y} \in S\}\right| . \tag{3.1}$$

3.1 Achieving all permutations

If there are enough dimensions, it is possible to achieve all $k!$ permutations of k sites; and it turns out that $k-1$ dimensions are sufficient. The following result establishes that for all L_p metrics. The concept is intuitive: in $(k-1)$-dimensional space, we can place k points equidistant from the origin and then all distance permutations will occur in the neighbourhood of the origin. The actual proof is more involved (an epsilon-delta induction) because the boundary cases L_1 and L_∞ cause problems if the points are *exactly* equidistant from the origin. Instead, we place them at slightly different distances for each dimension to force the inequalities to be strict.

Theorem 3.1
In n-dimensional real vector space with any L_p metric, k sites can be chosen such that all $k!$ distinct distance permutations exist, for any $k \leq n+1$. That is, $N_{n,p}(k) = k!$ for $n \geq k-1$ and $p \geq 1$.

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 $k \geq 2$ we prove a somewhat stronger statement by induction on k, namely that for any integer $k \geq 2$ 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

$$\Pi_{\mathbf{y}_\pi} = \pi \tag{3.2}$$
$$d(\mathbf{0}, \mathbf{y}_\pi) < \epsilon \tag{3.3}$$
$$|1 - d(\mathbf{x}_i, \mathbf{y}_\pi)| < \epsilon \tag{3.4}$$
$$d(\mathbf{x}_i, \mathbf{y}_\pi) \neq d(\mathbf{x}_j, \mathbf{y}_\pi) \text{ if } \mathbf{x}_i \neq \mathbf{x}_j . \tag{3.5}$$

In other words, with $k-1$ dimensions we can achieve all $k!$ permutations (3.2) with points that are near the origin (3.3), almost exactly unit distance from all the sites (3.4), and not equidistant from any two sites (3.5).

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 (3.2)–(3.5).

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 $\Pi_{\mathbf{y}'_{\pi'}} = \pi'$ (3.2) and $|1 - d(\mathbf{x}_i, \mathbf{y}_\pi)| < \epsilon/4$ (3.4).

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 $\mathbf{x}_k = \langle 0, 0, \ldots, 0, 1 - \epsilon/2 \rangle$; that is, we are adding one dimension and placing a new site on the newly-introduced coordinate axis at distance $1 - \epsilon/2$.

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 vary the value of z from $1 - \epsilon/2$ to $1 + 3\epsilon/4$. 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 $d(\mathbf{x}_i, \mathbf{y}) \leq d(\mathbf{x}_j, \mathbf{y})$, each distance is the $1/p$ power of a sum of p-th 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 3.2**
> 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.

The distance $d(\mathbf{x}_k, \mathbf{y}) < 1 - \epsilon/4$ when $z = 1 - \epsilon/2$ by the triangle inequality, using the distances $d(\mathbf{x}_k, \mathbf{0}) = z = 1 - \epsilon/2$ by definition, and $d(\mathbf{0}, \mathbf{y}) < \epsilon/4$ from the inductive assumption. But the distance from $\mathbf{y}$ to any other site than $\mathbf{x}_k$ must be greater than $1 - \epsilon/4$, again by the inductive assumption. Therefore at $z = 1 - \epsilon/2$, the distance permutation of $\mathbf{y}$ with respect to the k sites $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$ begins with k.

However, when $z = 1 + 3\epsilon/4$, then the distance $d(\mathbf{x}_k, \mathbf{y}) \geq 1 + \epsilon/4$ because the per-component difference between $\mathbf{x}_k$ and $\mathbf{y}$ in the last component is $1 + \epsilon/4$, and the overall distance cannot be any less. By the inductive assumption, the distance from $\mathbf{y}$ to any other site than $\mathbf{x}_k$ must be less than $1 - \epsilon/4$. Therefore at $z = 1 + 3\epsilon/4$,

the distance permutation of $\mathbf{y}$ with respect to the k sites $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k$ ends 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 (3.2), 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 (3.3); similarly, the distance from each $\mathbf{y}$ to each site must be in the interval $1 \pm \epsilon$ (3.4); and by our choices of z, all the distances to sites are distinct at each $\mathbf{y}$ (3.5). Therefore the theorem holds for k sites.

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

3.2 Voronoi diagrams and distance permutations

Voronoi diagrams are, as the title of Aurenhammer's survey of the subject suggests, "a fundamental geometric data structure" [15]. The concept has been independently reinvented many times and its origins go back centuries. Voronoï gave a general n-dimensional definition in 1908 [207], and the popular name "Voronoi diagram" refers to him. Lejeune Dirichlet used the diagrams, limited to two and three dimensions, in his study of quadratic forms in 1850 [135], and the name "Dirichlet tesselation" is also used for them.

cell

The classic Voronoi diagram starts with a set of points called sites and divides the Euclidean plane into regions called *cells* according to which of the sites is closest, as shown in Figure 3.1. In the figure, the points in the cell at left are those points that are closer to A than the B, C, or D.

m-th order Voronoi diagram

There are many ways to generalise Voronoi diagrams, including the use of more dimensions, distance functions other than the Euclidean metric, definitions of the sites as things other than isolated points, and changes to how points are divided into cells. Aurenhammer's 1991 survey describes many of the variations [15]. One of particular interest for our work is the *m-th order Voronoi diagram*, in which the cells correspond to the set of m nearest neighbours among the k sites rather than just the one nearest neighbour. An example for $m = 2$ is shown in Figure 3.2. Note that the same sites from Figure 3.1 are used in this diagram. Each cell corresponds to a set of two sites; for instance, the small cell in the middle of the diagram contains points for which the closest two sites are B and D, in either order.

The cells of m-th order Voronoi diagrams represent classes of distance permutations, by definition. For instance, in Figure 3.1 the cell at left contains all points whose distance permutations start with A, and in Figure 3.2 the cell in the middle contains all points whose distance permutations start with B, D or D, B.

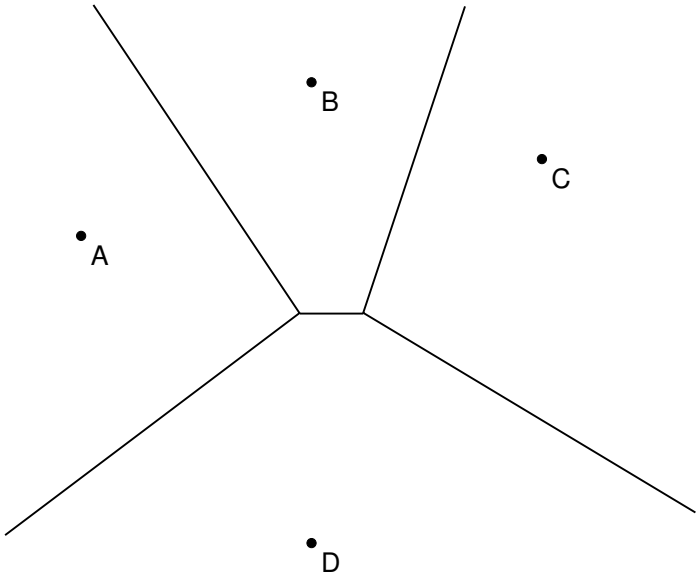


Figure 3.1: A first-order Euclidean Voronoi diagram.

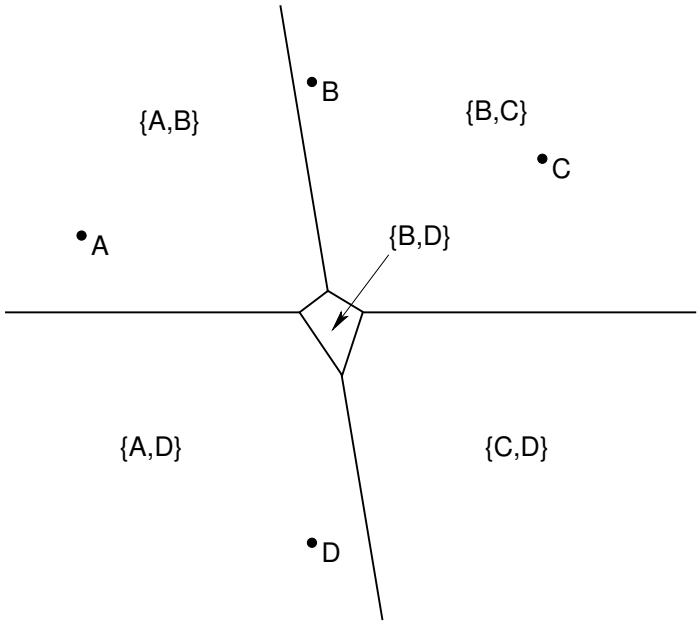


Figure 3.2: A second-order Euclidean Voronoi diagram.

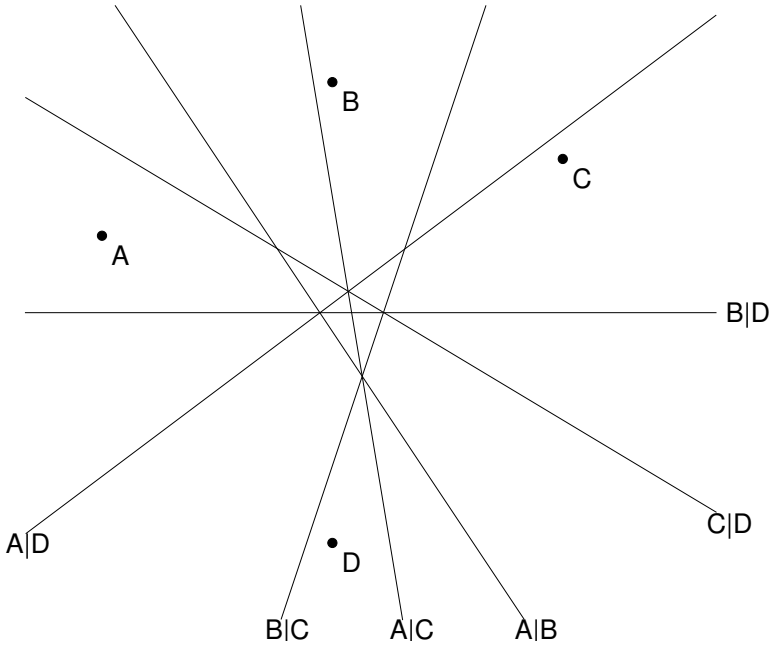


Figure 3.3: Bisectors of four points in Euclidean space.

Suppose we construct a diagram in which each permutation has its own cell. We could do that by taking the union of all the cell boundaries from all the m-th order Voronoi diagrams. Figure 3.3 shows such a diagram. The boundaries shown are exactly the six (that is, $\binom{4}{2}$) lines that bisect pairs of sites. We can uniquely identify the distance permutation of a point by stating, for each of the six pairs of sites, which one is closer; and in Euclidean space, each of those statements divides the space along a flat hyperplane of dimension one less than the space (a line, in the plane). The sets that divide the space are useful in other spaces too, so we define a general name and notation for them:

bisector
bisector system

Definition 3.3

The *bisector* of two points x and y, denoted by $x|y$, is the set of all points z such that $d(x,z) = d(y,z)$. The *bisector system* of a set of sites is the collection of all their pairwise bisectors.

However, answering the "which side?" question for six bisectors in Figure 3.3 suggests that there should be $2^6 = 64$ sets of answers and so 64 cells, evidently impossible when there are only $4! = 24$ permutations of the four sites. Moreover, there are only 18 cells in the arrangement shown. The fact that all the bisectors are straight lines limits the number of cells, and so does the fact that they are the pairwise bisectors of four sites, not just any six lines in arbitrary position relative to each other.

Arrangements of hyperplanes, which include bisector systems in Euclidean space, create combinatorial objects called oriented matroids, and those are well-studied [29]. Although we obtain the count by other methods in the next section, it appears that such methods could be applied to count distance permutations in Euclidean space. Unfortunately, most of the relevant results are inapplicable to bisectors in more general spaces.

Many authors, including Grünbaum [92] and Mandel [145], have applied oriented matroids to generalised hyperplanes that are not necessarily flat, calling them pseudolines and pseudospheres respectively. Grünbaum's pseudolines are defined in two dimensions with the requirement that any two must intersect in exactly one point, using the projective plane if necessary to force parallel lines to intersect at infinity. Mandel's pseudospheres have an analogous higher-dimensional requirement for well-behaved intersections: every intersection must be topologically equivalent to a sphere. The bisector system shown in Figure 3.4 fails to meet those criteria because of the bisectors $A|B$ and $C|D$, whose intersection consists of two points. Oriented matroids can also be described in terms of sign vectors, representing whether a point in the space is on one side, on the other side, or exactly on each bisector; and the sign vectors associated with Figure 3.4 fail to obey the properties defining an oriented matroid. (That sign-vector definition of an oriented matroid is complicated, and omitted here; Björner and others describe it in detail [29].)

So the obvious transformation from a bisector system to an oriented matroid appears unworkable. There may be other ways to apply oriented matroid techniques to bisector systems in non-Euclidean spaces. 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 [181]. The Delaunay oriented matroid is defined in terms of spheres passing through subsets of the points in the arrangement.

Lê studies the question of how many spheres can pass through a set of $n+1$ points in n-dimensional space with various L_p metrics, particularly the case of p equal to an even integer [134]. That bears on the complexity of an ordinary closest-point Voronoi diagram because it is the number of vertices in the diagram for $n+1$ points. Icking, Klein, Lê, and Ma then continue this line of research to describe in detail the number of spheres that can pass through four points in 3-space with various convex distance functions [107]. Whereas in Euclidean 3-space, four points in general position define exactly one sphere passing through all of them, they show that other smooth convex metrics allow for arbitrarily large finite numbers of spheres. In particular, with the L_4 metric, they show that there exists a set of four points that can be perturbed independently in 3-dimensional

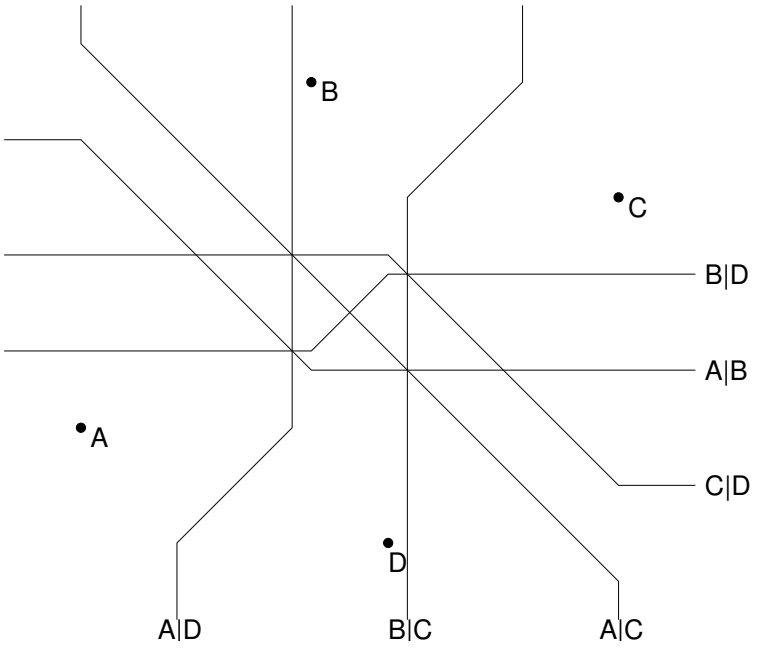


Figure 3.4: Bisectors of four points in L_1 space.

neighbourhoods (that is, the points are in general position) and through which pass exactly three spheres. They display such a set of points. However, they do not give the three spheres, only a proof of the spheres' existence and some intervals for their radii [107]. For their work, the startling result is that such spheres exist at all. The four points are:

$$\begin{array}{rrr} X & Y & Z \\ \hline 0 & 0 & 0 \\ 1 & 1/2 & -2 \\ -1 & -3/2 & 1/3 \\ -3 & -4 & -1/2 \end{array} . \tag{3.6}$$

The three-sphere result is significant for distance permutations because it suggests that a larger number of distance permutations might exist in non-Euclidean metrics. If four sites lie on the surface of a sphere, then by a simple epsilon-delta argument all 24 distance permutations of those sites occur in the neighbourhood of the sphere's centre. If we add a fifth site, we will show in Theorem 3.2 that only 96 of the 120 imaginable distance permutations can actually occur in three-dimensional Euclidean space. Part of the reason seems to be that the sphere centre with 24 permutations clustered around it can only occur in one place relative to the new site. If more than one sphere could pass

through the first four sites, then we would have a cluster of 24 permutations at the centre of each sphere, and placing the fifth site carefully relative to those clusters might allow us to give it a different relationship to each centre (inside some sphere and outside another), and achieve more than the Euclidean limit of 96 distance permutations.

Hoping to gain a better understanding of why L_4 space is so strange, we used simulated annealing to develop approximations for the three spheres whose intersection is given by (3.6), with this result:

centre X	centre Y	centre Z	radius
−16.969316	12.531628	−13.989112	19.5433
−0.4230825	−0.042524706	−4.0244665	4.47108
−0.5513786	−2.2194826	−2.4523801	2.78927

. (3.7)

A POV-Ray visualisation [167] of (3.6) and (3.7) is shown in Figure 3.5, and it provides some intuition of what is going on. The four intersection points lie approximately on a plane, and moreover on an arc of an ordinary Euclidean circle. If it were Euclidean space, then four points exactly on an arc would be a degenerate case, defining an infinite number of spheres that all shared the circle containing the points. Cutting a sphere with a plane in Euclidean space always creates a circle, so we can freely choose the size of the sphere (as long as its radius is no smaller than that of the circle itself) and still find a centre for it which will allow it to contain the specified circle. The Euclidean case is like a child catching a bubble on a circular bubble wand: the same circular loop can catch any sufficiently large bubble.

In the L_4 space, a sphere's intersection with a plane is not a perfect circle, and our four points are only approximately on a plane to begin with. The L_4 sphere has just enough bumpiness that between scaling, moving the centre, and the fact that there are only four points (not a whole circle) to match, we can place a sphere to intersect the four points and still have enough freedom left over to choose one of three centres and deal with the points being in general position. Note that each of the spheres is cut in a qualitatively different way by the approximate plane of the four points. Figure 3.6 shows schematically the least-squares plane and its relationship to each of the three bounding cubes for the L_4 spheres. For the largest L_4 sphere the plane cuts off one corner, approximately passing through the other. For the middle-sized sphere the plane cuts off two corners. For the smallest sphere, it cuts through the centre, separating three corners from the other five. Thus the three spheres are to some extent independent of each other, and we can keep them all fitted to the points while moving the points within small neighbourhoods.

The same authors with Santos investigate the behaviour of bisectors with

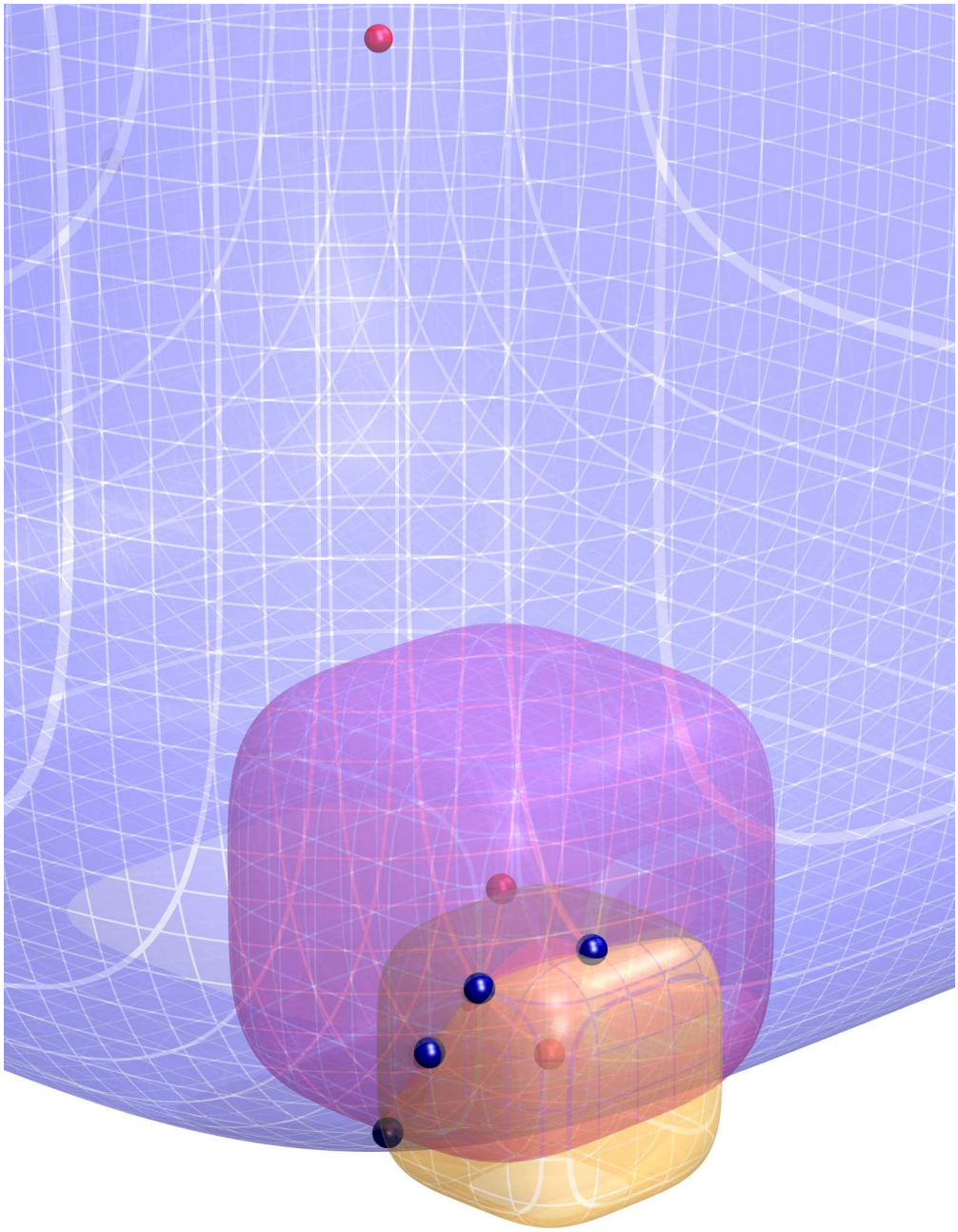


Figure 3.5: Visualisation of the four-point L_4 system.

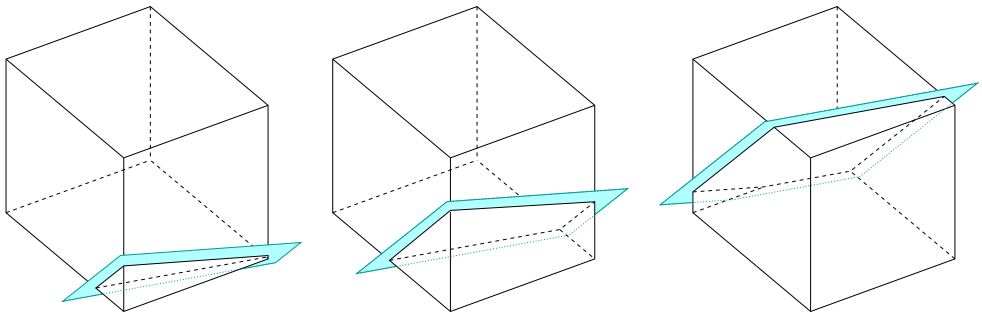


Figure 3.6: How the least-squares plane cuts the bounding cubes.

convex distance functions in two and three dimensions, generalised to sets equidistant from more than two sites [106]. They survey problematic results obtained by other authors and by subsets of their own group, and show further surprising results; in particular, that the combinatorial structure around the one-dimensional bisector of three points can be different for different connected components of the bisector. Note that the fact there can be more than one connected component of the bisector in the first place is already a significant departure from the Euclidean behaviour. As they describe it:

> One of the reasons for the lack of results on Voronoi diagrams for higher dimensions under arbitrary convex distance functions is the surprising, really abnormal, structure of the bisectors which behave totally different[ly] from what is known for the Euclidean distance. [106]

3.3 Euclidean space

Gardner illustrates his concept of "aha! insight" with the classical problem of cutting a circular cheese into eight slices with three cuts of a straight knife and no rearrangement of the pieces between cuts [82]. The solution is shown in Figure 3.7; the necessary "aha!" is that the cheese is three-dimensional. Attempts to solve the puzzle while treating the cheese as a two-dimensional object cut by straight lines will fail because with those restrictions, only seven slices are possible.

Suppose we generalise the question to cutting an object in n-dimensional Euclidean space with m flat cuts, each of which is an $(n-1)$-dimensional hyperplane. If we assume the object is convex and has nonzero measure, we can ignore its exact shape and simply consider the number of cells into which the hyperplanes

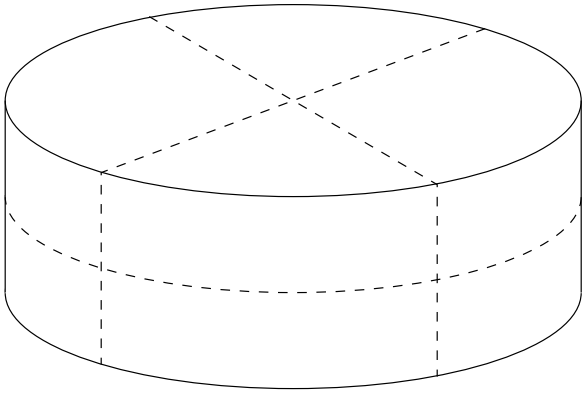


Figure 3.7: Cutting a cheese into eight pieces with three cuts.

divide the entire space, because we can always scale the hyperplane arrangement to put all the bounded cells inside a neighbourhood entirely contained in the object, and get as many slices from the object as we would from the entire space. Let $S_n(m)$ represent the maximum number of cells into which m cuts can divide n-dimensional Euclidean space. Some example values for $n = 2$ (the "pancake problem") are shown in Figure 3.8.

Price gives a straightforward analysis of the general problem, finding that $S_n(0) = S_0(m) = 1$, and $S_n(m) = S_n(m-1) + S_{n-1}(m-1)$ for $n, m > 0$ [171]. He uses an induction that follows the structure of the recurrence relation: when the m-th hyperplane is added to an arrangement that already contains $S_n(m-1)$ pieces, then the new hyperplane is itself a $(n-1)$-dimensional space cut up by the $m-1$ existing hyperplanes into $S_{n-1}(m-1)$ pieces, and each of those partitions off a new piece in the original n-dimensional space, proving the recurrence. Then it follows easily that $S_n(m) = \Theta(m^n)$ [171].

As described in the previous section, the distance permutations for a set of sites correspond to cells in the bisector system of the sites; and since bisectors in Euclidean space are simply hyperplanes, we can apply the result on cells formed by hyperplanes to count the distance permutations. There are $\binom{k}{2}$ bisectors between k sites; so if the bisectors were in general position, we would have $S_n\left(\binom{k}{2}\right)$ distance permutations.

However, the bisectors are not in general position, and the actual number of distance permutations is less. Note that for $n = 2$, $k = 3$, there are $\binom{3}{2} = 3$ bisectors, which suggest $S_2(3) = 7$ cells as shown in Figure 3.8, but there can be only $3! = 6$ distance permutations. However, Price's result for hyperplanes in general position remains an upper bound: $N_{n,2}(k) = O(k^{2n})$ because $\binom{k}{2}$ is $\Theta(k^2)$ and $S_n(m)$ is $\Theta(m^n)$. With a more detailed argument we can get an exact count.

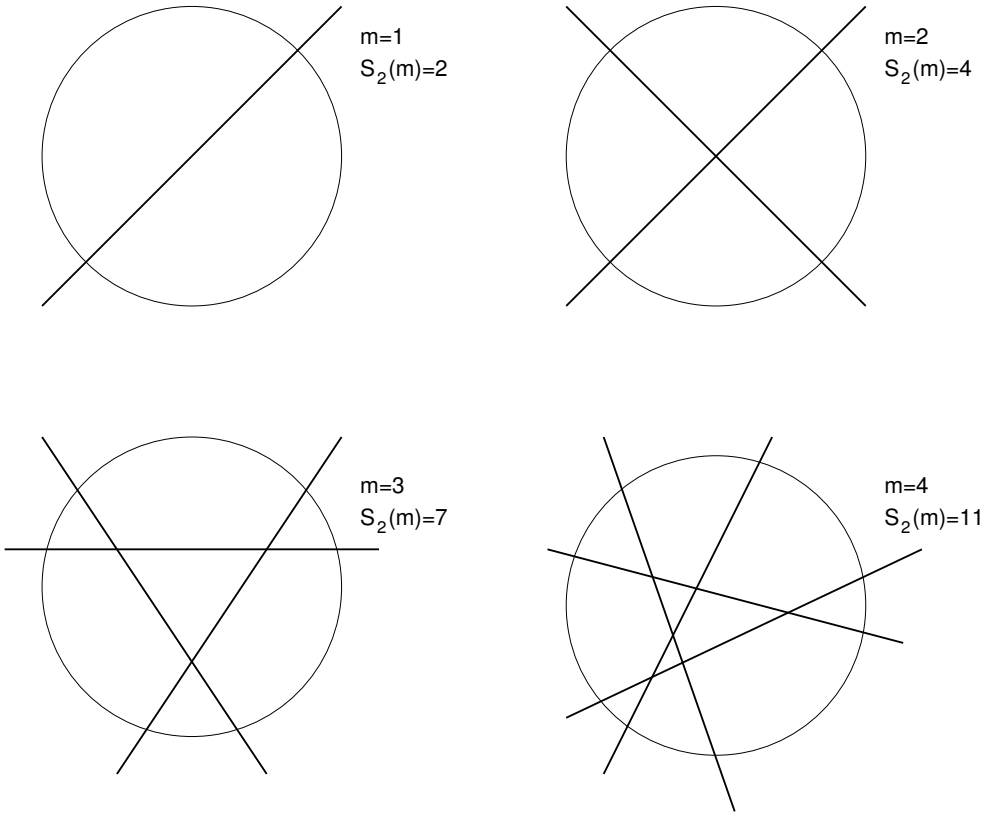


Figure 3.8: Cutting a pancake.

We emphasize that this bound is *nearly always* achieved: like Price's result [171], it does not depend on the arrangement of sites except that they are in general position. In this respect Theorem 3.2 differs from other similar results we will prove for other spaces, where sites matching the bound may be harder to find.

Theorem 3.2
In n-dimensional Euclidean (L_2) space, we have

$$N_{0,2}(k) = N_{n,2}(1) = 1 \tag{3.8}$$
$$N_{n,2}(k) = N_{n,2}(k-1) + (k-1)N_{n-1,2}(k-1)\,. \tag{3.9}$$

Moreover, the upper bound $N_{n,2}(k)$ is always achieved by sites in general position.

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 [171]. 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 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. Let 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 (3.9). ∎

Bounds on $N_{n,2}(k)$ follow from Theorem 3.2 by induction:

Corollary 3.3
The function $N_{n,2}(k)$ satisfies:

$$N_{n,2}(k) \leq k^{2n} \tag{3.10}$$

$$N_{n,2}(k) = \frac{k^{2n}}{2^n n!} + o(k^{2n}). \tag{3.11}$$

Therefore, the distance permutation in Euclidean space can be stored in $2n \lg k$ bits with an appropriate encoding.

Proof The proof for (3.10) is by induction on k. The result holds trivially for $k = 1$. Then we have:

$$\begin{aligned} N_{n,2}(k) &= N_{n,2}(k-1) + (k-1)N_{n-1,2}(k-1) \\ &\leq (k-1)^{2n} + (k-1)(k-1)^{2n-2} \\ &= k(k-1)^{2n-1} \\ &\leq k^{2n}. \end{aligned}$$

The space to store a distance permutation is $\lg N_{n,2}(k)$ bits, so $2n \lg k$ is an upper bound.

For (3.11) 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:

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

The sum of the coefficients for the k^{2n-1} term must be b_n by definition, so (with $a_0 = 1$ from the basis case) we can find a_n:

$$\begin{aligned} b_n &= -2n a_n + b_n + a_{n-1} \\ a_n &= \frac{1}{2n} a_{n-1} \\ &= \frac{1}{2^n n!}. \end{aligned}$$

Numerical values of $N_{d,2}(k)$ are shown in Table 3.1. Note that the lower triangle contains factorials corresponding to Theorem 3.1.

	k:								
	2	3	4	5	6	7	8	9	10
n: 1	2	4	7	11	16	22	29	37	46
2	2	6	18	46	101	197	351	583	916
3	2	6	24	96	326	932	2311	5119	10366
4	2	6	24	120	600	2556	9080	27568	73639
5	2	6	24	120	720	4320	22212	94852	342964
6	2	6	24	120	720	5040	35280	212976	1066644
7	2	6	24	120	720	5040	40320	322560	2239344
8	2	6	24	120	720	5040	40320	362880	3265920
9	2	6	24	120	720	5040	40320	362880	3628800

Table 3.1: Number of Euclidean distance permutations $N_{n,2}(k)$.

Corollary 3.3 implies an asymptotic improvement in the bound on storage space for a distance permutation index, because a general permutation of k sites would require $\Theta(k \log k)$ bits. This means that adding sites costs very little in terms of index space, once the number of sites is significant compared to the number of dimensions. On the other hand, additional sites only have a small effect on the number of possible index values, so it may also suggest that there is little value in adding more sites once we have about twice as many sites as dimensions.

3.4 The L_1 and L_∞ metrics

Other metrics than L_2 make the question of counting distance permutations significantly more complicated. For instance, in the Euclidean plane, a bisector is a line. Two bisectors in general position intersect at exactly one point. In degenerate cases, they either coincide or fail to intersect at all. But in the two-dimensional L_1 space shown in Figure 3.4, a bisector is in general an orthogonal line with a diagonal kink in the middle. Two bisectors in general position may intersect in one point, like $A|B$ and $B|C$; or two distinct points, like $A|B$ and $C|D$; or they may fail to intersect, like $A|D$ and $B|C$; and there are many degenerate intersections possible, such as two disjoint rays, or a ray with a line segment attached. In higher dimensions the number of possibilities grows rapidly.

The technique used in Theorem 3.2 of treating each intersection as a space of the same type and one fewer dimension, which makes it amenable to induction, fails for non-Euclidean metrics. As discussed earlier, oriented matroid theory provides some tools for studying objects like these bisector systems, but current

results do not solve the problem of counting distance permutations.

However, the greatest difficulties come from considering L_p metrics for general p. The special cases of L_1 and L_∞ share one of the useful properties of L_2: their bisectors are piecewise linear. Each bisector is the union of subsets of hyperplanes, with the number of hyperplanes a function of the number of dimensions. Then from elementary results we can obtain bounds on the number of distance permutations; perhaps loose bounds, but tight enough to improve the best previous storage space bounds for a permutation-based index.

Theorem 3.4
The function $N_{n,p}(k)$ satisfies:

$$N_{n,1}(k) = O\left(2^{2n^2}k^{2n}\right) \tag{3.12}$$

$$N_{n,2}(k) = O\left(k^{2n}\right) \tag{3.13}$$

$$N_{n,\infty}(k) = O\left(2^{2n}n^{2n}k^{2n}\right). \tag{3.14}$$

All three of these are $O\left(k^{2n}\right)$ with respect to k, so the distance permutation in L_1, L_2, or L_∞ space can be stored in $O(n \log k)$ bits with an appropriate encoding.

Proof The case of the L_2 metric is already covered by Corollary 3.3. 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

$$\begin{aligned} d(\mathbf{x},\mathbf{z}) &= |x_1 - z_1| + |x_2 - z_2| + \cdots + |x_n - z_n| \\ &= \pm(x_1 - z_1) \pm (x_2 - z_2) \pm \cdots \pm (x_n - z_n) \end{aligned}$$

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

$$\begin{aligned} d(\mathbf{x},\mathbf{z}) &= \max\{|x_1 - z_1|, |x_2 - z_2|, \ldots, |x_n - z_n|\} \\ &= \pm(x_i - z_i) \end{aligned}$$

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 $\Theta(m^n)$ [171]. Combining those with the upper bound on number of hyperplanes per bisector given above, the theorem follows. ■

It seems intuitive that in L_1 and L_∞ space, there should be more distance permutations possible than in Euclidean space. Consider the bisectors $A|B$ and $C|D$ in Figure 3.4. They define five pieces between them, whereas two Euclidean bisectors could define at most four. The function of n in the upper bounds of Theorem 3.4 is superexponential for L_1 and L_∞, but for L_2, Corollary 3.3 gives a leading term coefficient of $(2^n n!)^{-1}$. So it seems non-Euclidean L_p metrics should give many more distance permutations.

But there are only 18 pieces in Figure 3.4, the same as in Figure 3.3. The Euclidean result of 18 applies in all non-degenerate cases; but in L_1 space, even achieving that many is not easy. We made some informal experiments with interactive computer graphics and found that choosing four sites without careful thought often produces *fewer* than 18 cells. We found no examples where it gave more than 18.

It appears that for every pair of bisectors like $A|B$ and $C|D$, there must be a pair like $A|D$ and $B|C$; the first pair creates extra pieces, the second pair removes them, and the total never seems to exceed the Euclidean limit of given by Theorem 3.2. That raises the question of whether the Euclidean bound may actually apply to all L_p spaces. Does $N_{n,p}(k) = N_{n,2}(k)$ for all p? As described in the next section, the answer is "no."

3.5 Experimental results on L_p distance permutations

Our main investigation of distance permutations concerns the maximum possible number of them over all choices of sites and assuming a database in which every permutation that can occur, does occur, with uniform probability. That is the worst case for index storage space, because it maximises the number of bits necessary to store a distance permutation. It is also the best case for database search time, because it maximises the amount of information in a distance permutation. However, in a practical implementation, the sites may be chosen at random, and some distance permutations may occur much less often than others, or not at all even though they could, and so the actual number of distance permutations in the database may be significantly less than the theoretical maximum. In this section we present some experimental results on the number of distance permutations actually occurring in randomly-generated vector databases. These experiments were conducted for a paper in SISAP'08 [191], although space considerations limited the presentation in that paper to a summary of the detailed results given here.

We implemented distance permutation counting by extending the SISAP C-language metric space library of Figueroa, Navarro and Chávez [71] to include a new index type called `distperm`, as a minor modification of the library's `pivots` index type. Our index-generation program, while generating an index file which is discarded, produces a line of ASCII for each point in the input describing its distance permutation with respect to the pivots. The ASCII lines can be processed with standard Unix utilities (`sort | uniq | wc`) to find the number of distinct permutations. Site selection, inherited from the original library, simply chooses the first k points from the database; since the database points are generated independently and identically at random, this choice is equivalent to choosing sites uniformly at random.

This implementation approach was chosen because of the workshop requirement to support theoretical results with experiments on the SISAP library. The library's variable names and internal documentation are sparse, and mostly written in Spanish, and its conceptual design (in particular, the focus on testing index data structure time performance, and the limitation of L_p spaces to L_1, L_2, and L_∞) is not well-suited to some of our work. In the process of doing these experiments we found bugs in the library, which were reported to and acknowledged by one of the library's authors [69]. However, we found no bugs that would be expected to materially affect the experimental results, and the most theoretically significant results, the counterexamples to $N_{n,p}(k) = N_{n,2}(k)$, have been carefully double-checked with our own independent Perl code.

For each vector length n from 1 to 10, number of sites k from 2 to 12, and

metric from $\{L_1, L_2, L_\infty\}$, we did 20 trials of generating a random database of 10^6 points chosen uniformly from the unit hypercube, choosing k of them as sites, and counting how many distinct distance permutations occurred among the database points. Note that this count represents only a lower bound on the number of distance permutations generated by the random sites. There could well be some very small generalised-Voronoi regions which happen not to contain any database points. So the results of this kind of experiment differ from the $N_{n,p}$ values described earlier by two maximisations: $N_{n,p}$ is the maximum for any choice of sites, but also the maximum for any choice of database points.

The sample mean permutations counted over 20 trials are shown in Tables 3.2 and 3.3. The column for $k = 2$ is omitted; it consists entirely of the value 2.00. One observation from these tables is that the mean distance permutations seem to decrease from L_1 to L_2 and from L_2 to L_∞. The pattern is not consistent, however; there seems to be substantial variation from one sample to the next, so that 20 trials may not really be enough to get an accurate picture of how the system behaves. Except for the lowest dimensions and numbers of sites, the mean numbers of distance permutations for all three metrics seem to fall far short of the Euclidean theoretical bound from Table 3.1. Note that for the Euclidean metric, that bound is *always achieved* except in degenerate cases, so it is apparent that very many of the generalised-Voronoi cells, which must exist by the theory, are being missed by the database points. The fact that database points occur only in the unit cube may be relevant, if some significant number of distance permutations would only be possible for points outside the cube.

For the same sample of 20 random databases we also recorded the maximum number of distance permutations observed, in order to test the hypothesis that $N_{n,p}(k) = N_{n,2}(k)$. Those results are shown in Table 3.4. The columns for $k < 5$ are omitted; in those columns, all three metrics simply achieved the Euclidean bounds from Table 3.1.

These results indicate the existence of counterexamples to $N_{n,p}(k) = N_{n,2}(k)$ for several cases of n, p, and k. The initial experiment did not allow for isolating and reproducing such cases because the SISAP library as supplied took its random number seeds from the system clock without saving or reporting them. Since each trial overwrote the multi-megabyte database from the previous one, there was no practical way to go back to an earlier trial for more detailed examination. We have since extended the library to use reproducible seeds, and suggested that feature to the authors. In a new experiment which simply ran trials repeatedly until it found some with sufficiently large permutation counts, we were able to find new counterexamples for each of the labelled cases from Table 3.4, and save the complete database for each one. In all but one of the four cases, our new counterexamples actually surpass the permutation counts from Table 3.4. We

		k					
	n	3	4	5	6	7	8
L_1	1	4.00	7.00	11.00	16.00	22.00	29.00
	2	5.10	14.65	34.75	78.70	149.00	268.45
	3	5.90	20.85	71.45	251.40	591.25	1405.20
	4	6.00	23.95	107.50	440.55	1538.25	4705.35
	5	6.00	24.00	117.40	584.40	2699.25	10390.85
	6	6.00	24.00	119.50	672.85	3697.85	16073.50
	7	6.00	24.00	119.20	700.50	4188.10	20811.65
	8	6.00	24.00	120.00	719.40	4574.75	26999.10
	9	6.00	24.00	120.00	718.50	4755.90	30309.60
	10	6.00	24.00	120.00	719.90	4887.05	30715.55
L_2	1	4.00	7.00	11.00	16.00	22.00	28.95
	2	5.60	15.25	38.65	77.35	155.45	268.05
	3	5.90	19.60	72.75	205.20	577.00	1332.75
	4	6.00	22.70	99.40	380.00	1385.25	4214.20
	5	6.00	23.90	109.60	535.00	2043.25	7515.05
	6	6.00	23.75	115.35	610.80	3210.90	13824.25
	7	6.00	23.95	116.65	655.30	3443.85	17349.30
	8	6.00	23.95	115.95	647.50	4137.95	20244.80
	9	6.00	24.00	119.60	704.90	4258.60	21936.45
	10	6.00	24.00	119.50	697.15	4575.65	25562.25
L_∞	1	4.00	7.00	11.00	16.00	22.00	29.00
	2	5.30	13.15	34.10	69.75	136.85	233.15
	3	5.70	18.05	68.30	206.40	537.70	1219.75
	4	5.80	23.50	84.30	358.95	1263.95	3664.60
	5	6.00	22.80	93.60	459.70	2138.85	6488.30
	6	6.00	23.35	110.00	571.60	2642.25	11314.00
	7	6.00	23.70	115.45	626.20	3133.55	14384.65
	8	6.00	24.00	109.15	585.55	3542.75	15433.55
	9	6.00	24.00	118.50	627.30	3779.65	18494.20
	10	6.00	24.00	118.50	709.00	4084.80	22415.00

Table 3.2: Mean distance permutations in L_p experiment.

		k			
	n	9	10	11	12
L_1	1	37.00	46.00	56.00	67.00
	2	428.20	671.65	1021.85	1398.20
	3	2924.30	5886.80	9880.55	16143.70
	4	11078.20	24077.90	48544.80	82253.85
	5	30851.45	66953.80	126826.65	220231.20
	6	50826.35	123731.65	249919.70	394466.85
	7	76755.45	204455.05	377575.30	569807.35
	8	109849.55	275752.35	518661.45	728040.20
	9	122789.75	326614.40	617099.95	809393.30
	10	146384.95	386111.90	688443.85	884013.40
L_2	1	37.00	46.00	56.00	67.00
	2	443.05	686.90	1026.65	1440.15
	3	2773.30	5133.30	8853.50	14584.20
	4	9912.40	21230.50	38047.15	67179.40
	5	19816.05	51259.90	99545.40	182253.50
	6	36326.25	97967.25	210376.30	348609.25
	7	58430.65	161071.90	306082.00	502957.40
	8	74664.80	233592.90	434457.00	657013.80
	9	102166.35	260769.85	513928.05	730146.10
	10	117556.60	325272.35	621300.75	815217.05
L_∞	1	37.00	46.00	56.00	67.00
	2	400.20	602.75	929.15	1314.10
	3	2529.90	5064.90	8094.35	13152.25
	4	8457.70	18137.00	33338.65	54838.10
	5	19071.75	44301.15	87522.55	150360.55
	6	31869.30	87133.05	145385.35	265706.25
	7	49852.35	132839.80	243956.95	357331.00
	8	61401.00	153040.30	293299.55	496952.75
	9	80897.80	212181.70	385720.85	569572.75
	10	95076.85	226682.85	436309.55	648613.15

Table 3.3: Mean distance permutations in L_p experiment (continued).

		k							
	n	5	6	7	8	9	10	11	12
L_1	1	11	16	22	29	37	46	56	67
	2	43	92	168	290	462	772	1080	1532
	3	†98	†354	838	1804	3769	7250	11747	18239
	4	120	†658	2030	5663	16592	30430	57171	94537
	5	120	697	3505	13573	35769	75298	151101	258874
	6	120	720	4904	20234	65759	155220	290896	471375
	7	120	720	4952	27824	97932	257603	435901	653015
	8	120	720	5035	33637	132672	334432	625364	770929
	9	120	720	5039	37198	169753	395440	659222	845181
	10	120	720	5038	35698	191743	492404	757208	917237
L_2	1	11	16	22	29	37	46	56	67
	2	45	94	180	298	527	735	1095	1539
	3	92	273	706	1568	3145	5859	9906	15929
	4	120	537	1845	5079	11754	24075	45396	75850
	5	120	711	3336	10471	27063	71921	129003	208301
	6	120	720	4814	18693	62457	143879	270655	402685
	7	120	720	4875	23944	91908	208659	393085	613857
	8	120	720	4973	34866	118958	304725	568608	796775
	9	120	720	5004	28635	135767	351849	637530	851775
	10	120	720	5040	33097	148751	473234	732197	905490
L_∞	1	11	16	22	29	37	46	56	67
	2	39	87	163	278	492	780	1076	1485
	3	†100	271	823	1712	3676	6677	11331	16162
	4	119	544	1802	4912	12610	24745	40919	70354
	5	120	712	3266	12566	29275	71306	122876	213951
	6	120	711	4485	17837	50834	128718	200456	352150
	7	120	720	4650	23983	74802	192155	314927	466484
	8	120	720	4446	26906	82902	226039	408183	610841
	9	120	720	5002	27160	124835	328629	484824	714881
	10	120	720	5008	34281	129445	284997	532539	770769

† indicates counterexamples to $N_{n,p}(k) = N_{n,2}(k)$.

Table 3.4: Maximum distance permutations in L_p experiment.

also verified each one with a new and separate implementation.

First, for $\mathbb{R}^3$, L_1, and $k = 5$, these sites give at least 108 permutations:

$$\begin{aligned}
\mathbf{x}_1 &= \langle 0.205281, 0.621547, 0.332507 \rangle, \\
\mathbf{x}_2 &= \langle 0.053421, 0.344351, 0.260859 \rangle, \\
\mathbf{x}_3 &= \langle 0.418166, 0.207143, 0.119789 \rangle, \\
\mathbf{x}_4 &= \langle 0.735218, 0.653301, 0.650154 \rangle, \\
\mathbf{x}_5 &= \langle 0.527133, 0.814207, 0.704307 \rangle .
\end{aligned} \tag{3.15}$$

For $\mathbb{R}^3$, L_1, and $k = 6$, these sites give at least 369 permutations:

$$\begin{aligned}
\mathbf{x}_1 &= \langle 0.723033, 0.528501, 0.114338 \rangle, \\
\mathbf{x}_2 &= \langle 0.537893, 0.456652, 0.887229 \rangle, \\
\mathbf{x}_3 &= \langle 0.933624, 0.852988, 0.287647 \rangle, \\
\mathbf{x}_4 &= \langle 0.989538, 0.902709, 0.683371 \rangle, \\
\mathbf{x}_5 &= \langle 0.271674, 0.328448, 0.779628 \rangle, \\
\mathbf{x}_6 &= \langle 0.644903, 0.690669, 0.969264 \rangle .
\end{aligned} \tag{3.16}$$

For $\mathbb{R}^4$, L_1, and $k = 6$, these sites give at least 665 permutations:

$$\begin{aligned}
\mathbf{x}_1 &= \langle 0.594251, 0.972026, 0.340673, 0.046086 \rangle, \\
\mathbf{x}_2 &= \langle 0.995420, 0.894291, 0.679227, 0.073585 \rangle, \\
\mathbf{x}_3 &= \langle 0.609278, 0.466341, 0.729449, 0.924971 \rangle, \\
\mathbf{x}_4 &= \langle 0.330296, 0.724184, 0.379641, 0.275436 \rangle, \\
\mathbf{x}_5 &= \langle 0.378577, 0.690847, 0.570682, 0.657335 \rangle, \\
\mathbf{x}_6 &= \langle 0.928232, 0.417448, 0.127706, 0.183649 \rangle .
\end{aligned} \tag{3.17}$$

Finally, for $\mathbb{R}^3$, L_∞, and $k = 5$, these sites give at least 100 permutations:

$$\begin{aligned}
\mathbf{x}_1 &= \langle 0.902333, 0.530606, 0.513334 \rangle, \\
\mathbf{x}_2 &= \langle 0.202670, 0.267084, 0.123475 \rangle, \\
\mathbf{x}_3 &= \langle 0.364960, 0.539546, 0.615330 \rangle, \\
\mathbf{x}_4 &= \langle 0.063169, 0.494804, 0.465638 \rangle, \\
\mathbf{x}_5 &= \langle 0.080624, 0.338353, 0.681813 \rangle .
\end{aligned} \tag{3.18}$$

These counterexamples prove that $N_{n,p}(k) = N_{n,2}(k)$ is not true in general. It appears that the counterexamples must also hold for values of p slightly more than 1, or finite but sufficiently large, because those metrics could be chosen to have values sufficiently close to the values of L_1 and L_∞ for the generalised-Voronoi cells to remain nonempty. Many questions remain open on how far the limits actually extend.

Chapter 4

Tree metrics

Tree metric spaces are of interest for several reasons. They have a simple definition and allow easy demonstration of our techniques. They also have applications in approximating other metrics, and in software obfuscation.

Definition 4.1

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*. Note that by setting all weights equal to 1, every tree metric is a weighted tree metric.

tree metric space

tree metric

weighted tree metric

Example 4.2

Figure 4.1 shows the flights offered by an airline, with their values in frequent flyer points. A passenger wishing to travel between two cities for which there is no direct flight must make connections in one or both of the hub cities of Thunder Bay and Waterloo. For instance, a trip between Armstrong and Ottawa would route through both hubs and earn $441 + 1203 + 391 = 2035$ frequent flyer points. Because the route map is a weighted tree, the number of frequent flyer points earned between two cities is a weighted tree metric on the set of cities.

Terminology used to describe tree metrics varies, and many authors assume the definition [4, 144]. Gupta does not define "tree metric" as such, but refers implicitly to the distance in weighted trees [93], while Indyk and Matoušek define tree metrics as a special case of graph metrics, and always weighted [109, pages 183–184]. Applications of tree metrics, such as to numerical taxonomy in biology, often assume that the metric applies to a finite number of points [3].

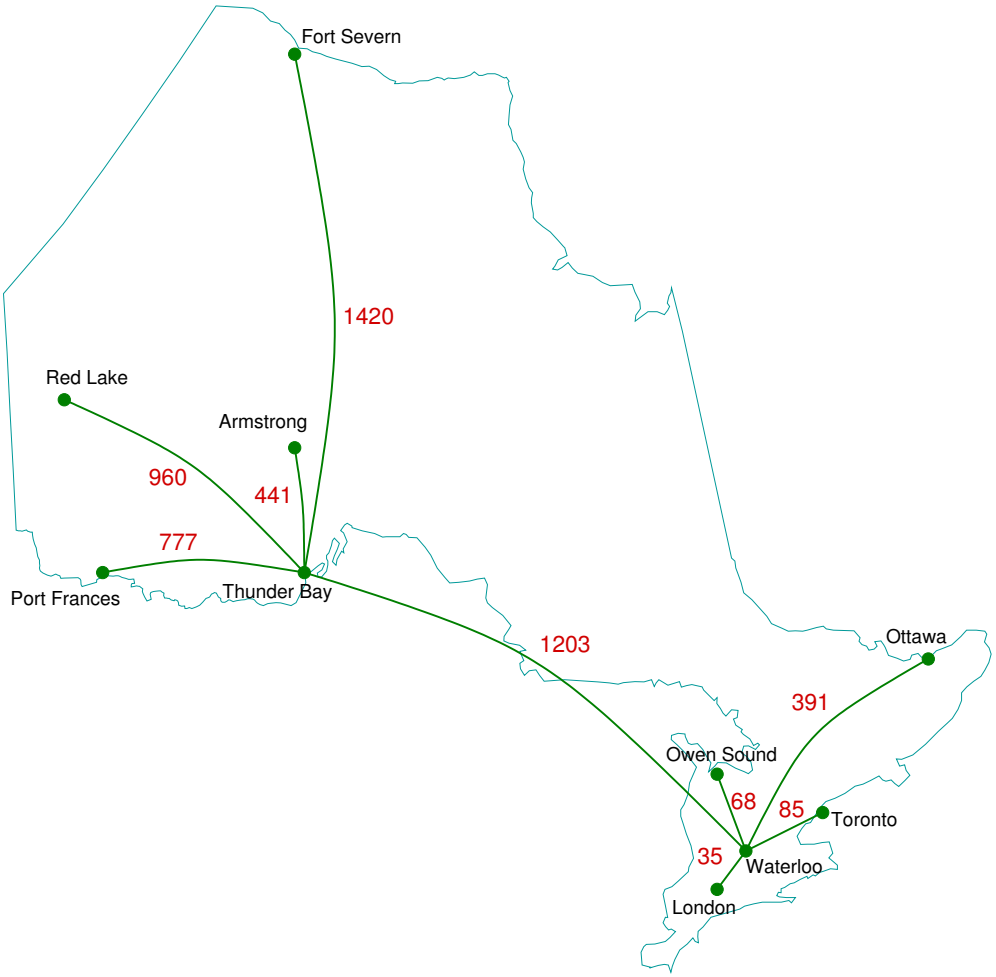


Figure 4.1: Route map for a small airline.

However, to include such metrics as the prefix distance of Definition 4.5, we permit infinite trees.

It is easy to verify that tree metrics as defined above do have the properties required by Definition 1.1. Strong statements can be made about tree metrics that might not apply to more general classes of metrics. For instance, with a tree metric, the triangle inequality $d(x,z) \leq d(x,y) + d(y,z)$ holds as an equality if and only if y is on the unique path between x and z. Tree metrics also have the following property.

Definition 4.3

A metric space $\langle S, d \rangle$ satisfies the *four-point condition* if for every set of four distinct points $\{x, y, z, t\} \subseteq S$, we have [34] four-point condition

$$d(x,y) + d(z,t) \leq \max \begin{cases} d(x,z) + d(y,t) \\ d(x,t) + d(y,z) \end{cases} . \tag{4.1}$$

Some authors use the four-point condition as the definition of tree metrics, calling any metric space a tree metric space if it obeys Definition 4.3 [64, 149]. However, the four-point condition applies to metric spaces that lack trees or paths, such as discrete spaces (including finite ones) and the following infinite example.

Example 4.4

Let S be the set of rational numbers in the closed interval $[0,1]$, with the metric $d(x,y) = |x - y|$. This space obeys Definition 4.3 but does not correspond to distances among the vertices of any tree; no point is adjacent to any other point.

We reserve the term tree metric space for the spaces satisfying Definition 4.1, which requires that the points are exactly the vertices of some tree. There we follow 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 [144]. As Buneman shows, any finite metric space satisfying the four-point condition must also be a subset of a tree metric space satisfying Definition 4.1 [34].

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 4.5

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. prefix distance

The distance between two strings in the prefix metric is the sum of their lengths, minus twice the length of their longest common prefix. 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.

Because tree metrics are simple and strong statements can be made about them, they are frequently used as approximations of less convenient metrics, often in a randomised context. Alon and others displayed a randomised embedding of an arbitrary finite metric space into a set of tree metrics, such that the distortion from the original metric to a randomly selected one of the tree metrics was at most $\exp\left(O(\sqrt{\log n \log\log n})\right)$ [4]. Their main application was the k-server problem, an online problem in which k servers move from point to point serving requests and attempting to keep the total distance moved as small as possible. Bartal improved the distortion bound first to $O(\log^2 n)$ [19] and then to $O(\log n \log\log n)$ [20], with the introduction of additional points to the space, and described multiple applications. Applications of these approximate embeddings include that by Peleg and Reshef to the distributed directory problem [166]; by Kleinberg and Tardos to classification [125]; and by Laoutaris, Zissimopoulos, and Stavrakakis to allocating Internet bandwidth and storage [132, page 412].

Tree metrics are also of interest for software obfuscation. Many security applications require disclosure of a piece of software that computes a function without directly disclosing some secret parameter of the function. We considered algorithms for *blind substring search*, in which a program to search for a substring is published without publishing the substring, in 1998 [189]; and we cryptanalysed a similar system used by a commercial Internet content filtering package, in 2000 joint work with Eddy Jansson [112].

blind substring search

Barak and others showed in 2001 that obfuscation is, in general, impossible [18]. However, users still demand it; and Lynn, Prabhakaran, and Sahai give some very limited positive results for obfuscation. In particular, they show that neighbourhood testing in a tree metric can be obfuscated [144]. As they describe, some security functions of practical interest for access control can then be obfuscated despite the general impossibility results.

4.1 Intrinsic dimensionality

Recall that the intrinsic dimensionality ρ of a space is defined as the square of the mean, divided by twice the variance, of the distance between two random points selected from the native distribution of the space (Definition 1.23). For tree metric spaces this value depends on both the shape of the tree and the native

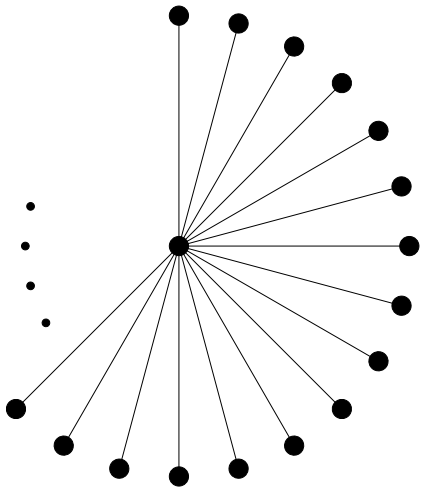


Figure 4.2: A star graph.

distribution for choosing points, and in general it may be difficult to calculate. However, we solve a few simple examples here.

First of all, consider an unweighted tree metric space with a finite number of points and a uniform distribution among them. If the tree is a path, $\rho \to 1$. That case can be analysed as approaching a one-component real vector with the L_1 metric and the lone component uniformly distributed in $[0,1]$, at which point Corollary 2.2 applies. That seems to be the minimum possible value if the distribution is uniform. If the distribution can be non-uniform, we can achieve ρ arbitrarily small with a distribution that chooses one point with almost certain probability, and applying Theorem 1.1.

As for an upper limit, if the tree is a star graph with n vertices, as shown in Figure 4.2, then in the limit for large n with a uniform distribution, the chance of choosing the middle vertex is negligible, the space approximates a discrete space, and $\rho \to n/2$, which is the maximum for any finite metric space with any distribution by Corollary 1.2, and thus maximal for any finite tree metric space (including weighted ones).

For strings of length n with prefix distance, the intrinsic dimensionality is quadratic. Note that this result does not contradict the previous claim that linear intrinsic dimensionality is the maximum, because the value of n in this case is the length of the strings, not the number of points in the space; the number of points in this space of strings is exponential in n.

Theorem 4.1
For the space of strings of length n chosen uniformly at random with the prefix distance, the intrinsic dimensionality ρ obeys

$$\rho \to \frac{(2n+1-|\Sigma|)^2}{|\Sigma|(|\Sigma|-1)} \tag{4.2}$$

where Σ is the alphabet.

Proof The length of the longest common prefix between two infinite strings has a geometric distribution [114] with parameter $p = 1/|\Sigma|$. Our finite strings differ only in that the longest common prefix is limited to n letters; but if n is large in comparison to $\log|\Sigma|$, the chance of the prefix being as long as the strings becomes negligible, so we can approximate with the infinite case.

The mean and variance of a geometric distribution are $(1-p)/p$ and $(1-p)/p^2$ respectively. Then the prefix distance (equal to the total length of the two strings minus twice the length of the longest common prefix) has mean $2(np+p-1)/p$ and variance $4(1-p)/p^2$. The result follows by substitution into the intrinsic dimensionality definition $\rho = \mathrm{E}^2[D]/2\mathrm{V}[D]$ (Definition 1.23):

$$\begin{aligned}\rho &\to \frac{4(np+p-1)^2p^2}{p^2 4(1-p)} \\ &= \frac{(2n+1-|\Sigma|)^2}{|\Sigma|(|\Sigma|-1)}.\end{aligned}$$

4.2 Distance permutations

site
distance permutation

Recall from Definition 1.25 that given k points $x_1, x_2, \ldots, x_k$ called the *sites*, the *distance permutation* for a point y is the unique permutation that sorts the site indices into order of increasing distance from y, using order of increasing index to break ties. Depending on the space and the choice of the k sites, some permutations might not occur. That is, there may be some permutation π such that there is no point y with π as its permutation.

If we store distance permutations as part of a database index, then the number of bits required depends on how many distinct values really do occur; so we consider the question of how many distinct distance permutations can occur. We are interested in a worst-case maximum and so assume that the k sites are chosen

to maximise the number of distinct distance permutations. It turns out that the maximum is quadratic in k. This result appeared, with a sketch of the proof, in SISAP'08 [191].

Theorem 4.2
With 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 $d(x,z) \leq d(y,z)$. 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 4.2 is easily achievable in spaces like the prefix distance space, where long paths are abundant.

Corollary 4.3
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 $2, 4, \ldots, 2^{k-1}$. Now the midpoint of the vertices 0 and 2^i for any $i \geq 1$ 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, and the edges from them to their higher-numbered neighbours are the distinct splitting edges of Theorem 4.2. 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 in Definition 1.25, which considers lower-indexed sites, which are the lower-labelled sites by our choice, to be closer in case of ties. ∎

4.3 Reverse similarity search

The VPREVERSE and GHREVERSE problems (Definitions 1.27 and 1.29 on page 37) are constraint satisfaction problems originating from distance-based binary tree data structures. Recall that a VPREVERSE instance consists of ordered triples of a point which serves as centre, a real radius, and a bit specifying inside or outside. The solution point must be inside or outside each of the spheres as directed by the bits; that is, a point z such that for all triples (x, r, b) in the instance, $d(x,z) \leq r$ if and only if $b = 1$. A GHREVERSE instance specifies the constraints as pairs of points; the solution z must be a point such that for all pairs (x, y) in the instance, $d(z,x) \leq d(z,y)$.

Note 4.6
As mentioned in Note 1.11, tree metric spaces are completely different from distance-based tree data structures even though the data structures could be used to index points that happen to be in a tree metric space. The tree metric spaces discussed in this section, on which we do VPREVERSE and GHREVERSE, are not VP- or GH-trees.

It is obvious that in a finite tree metric space, with the tree given explicitly in the input, VPREVERSE and GHREVERSE are polynomial-time problems. A naive algorithm for either of them in an n-point space might consist of finding all the pairwise distances among points in $O(n^2)$ time by doing n depth-first searches, then testing each of the n points against all the constraints, finding all solutions in quadratic time overall. But in the case of VPREVERSE, it is possible to solve the problem faster.

Theorem 4.4
There exists an algorithm to decide VPREVERSE on a weighted finite tree metric space in time $\Theta(n+m)$ where n is the number of points in the space and m is the number of spheres in the instance.

Proof First, we split vertices as necessary to reduce the maximum degree of the tree to three, giving each new edge a weight of zero and recording for each new vertex which original vertex it came from. Note that each new vertex created by a split is a solution to the original instance if and only if the original vertex was a solution. With the tree expressed by an array of vertices each having a doubly-linked list of outgoing edges and each directed edge linked to its partner in the other direction, we can do this splitting in linear time. Each directed edge

in the tree will be labelled with an interval of distances, initialised to $[0, \infty)$, as well as its weight. The interval on an outgoing edge represents the range of distances from the vertex within which any solution must lie if it is on that branch of the tree; the initial values signify no restriction on where solutions could be.

For each sphere of the form $(x, r, 0)$, which requires solutions to be more than distance r from the centre x, we set all the intervals on edges leading out of x to their intersection with the interval (r, ∞). Similarly, for each sphere of the form $(x, r, 1)$, requiring solutions to be at most distance r from x, we intersect all the intervals on edges leading out of x with the interval $[0, r]$. Now the intervals contain all the information from the input about which points can and cannot be solutions; it remains only to propagate that information around the tree until we have tested all the points.

We choose an arbitrary vertex to be the root and do two depth-first searches starting from it. Each time we visit a vertex we intersect the intervals on its incoming edges, adjusted for edge weight, with the intervals on its outgoing edges to propagate the constraints. For instance, if $(a, b]$ is the interval on an edge from x to y with weight w when we visit y, then we set the intervals on all outgoing edges of y, except the one leading back to x, to their intersections with $(a - w, b - w]$.

After doing our two depth-first searches, any vertex is a solution if and only if all its incoming edges are labelled with intervals containing their weights and all its outgoing edges are labelled with intervals containing the value zero. We can test that for all vertices in linear time.

To establish correctness of the algorithm, note that a vertex y is a solution to the instance if and only if there is no vertex x which is the centre of a sphere such that the distance from x to y violates the constraint given by the sphere. So if we test all paths for all pairs of x and y, against all spheres centred on x, we can establish whether y is a solution. Each y must receive the information from each x as to whether it is at an acceptable radius. That information starts at each sphere centre when we initialise the intervals and the propagates to all neighbours each time we visit a vertex.

The path from x to y may be monotonic downward if x is an ancestor of y, monotonic upward if x is a descendant of y, or bitonic, proceeding first from x up to its common ancestor with y and then down to y. The first depth-first search propagates all constraints from ancestors down to descendants on the way down, and from descendants back up to ancestors on the way up. That handles all monotonic paths. It might handle some bitonic paths as well if they happen to run left to right in the tree, but they might not. However, the first search propagates the information at least up to the common ancestor in a bitonic path,

and then the second depth-first search propagates it the rest of the way. The requirement for outgoing edges from a solution to include zero in their intervals handles the case $x = y$, where a vertex can be excluded because it is the centre of a sphere even without a path to any other distinct vertex. ■

The reason we cannot immediately use a similar algorithm for GHREVERSE is that the information to be propagated along each edge is more complicated for GHREVERSE. There could be an edge with one point from every input pair on one side of the edge and the other point on the other side. Instead of labelling each directed edge with an interval of allowable distances for paths containing that edge, we would have to store the identities and distances of all the vantage points on that side. It is not clear that we can process that much information for every edge fast enough to improve on the quadratic algorithm.

Some tree metric spaces, notably strings with the prefix metric, have too many points for examination of all points to be a useful strategy. Examining all points is especially difficult in spaces where there are an infinite number of points. Nonetheless it remains possible to solve these problems in many typical large spaces, provided that it is reasonably easy to find paths among points. We express this requirement by defining an operation which algorithms can use to find points and relations between them in the space.

Definition 4.7

PATH(x, y)

For any distinct points x and y in a tree metric space, the function PATH(x, y) returns the path with weights from x to y. That is a list of distinct points starting with x and ending with y, in which any two consecutive points are adjacent to each other in the tree; and for each pair of consecutive points, the distance between them, which is the weight of the edge.

Given PATH(x, y) it is easy to compute distances, or test adjacency, between points. For our main results on reverse similarity search in tree metric spaces, we require that PATH(x, y) can be executed in time polynomial to its input. Note that also implies a limit on the number of edges in the path between x and y, because it must have time to write its output. This requirement is not particularly onerous, and we expect it to be satisfied by any space one might use in practice; it exists to exclude obscure special cases where spaces might be defined to have very long paths among points with short names.

For spaces on which PATH is polynomial-time, GHREVERSE is a polynomial-time problem. VPREVERSE places additional requirements on the space in order to be polynomial-time, but we still expect it to be polynomial-time in all practical cases. These results were presented (with the proofs given as sketches) in SISAP'08 [192]. First we give the proof for GHREVERSE.

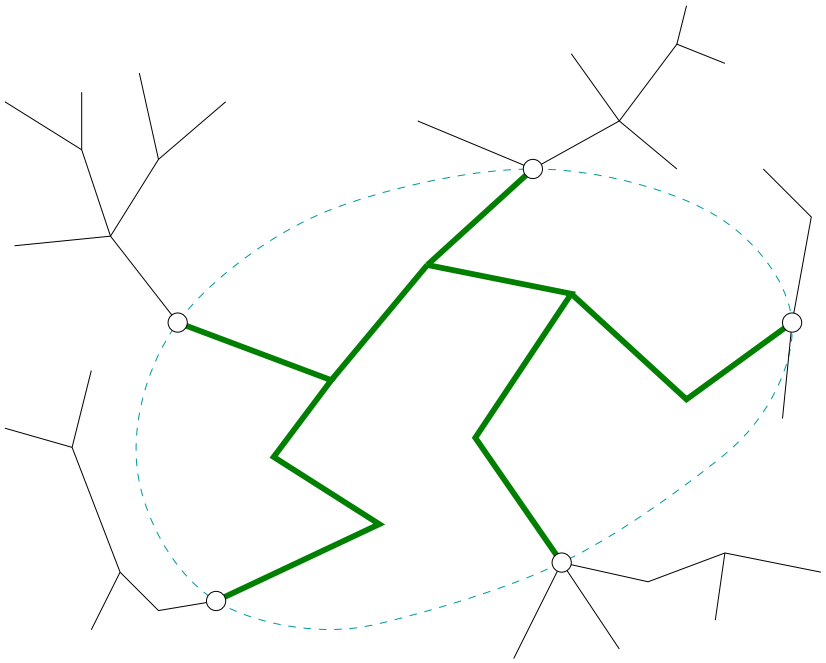


Figure 4.3: The central subtree.

Theorem 4.5
If S is a space with a (possibly weighted) tree metric, and PATH(x, y) runs in polynomial time for this space, then GHREVERSE on S is a polynomial-time problem.

Proof Since finding the path between two points is polynomial-time, the length (number of edges) of the path must also be polynomial. Choosing one of the points in a GHREVERSE instance, say x_1, we find the path from it to all the other points x_i and y_i in the instance. The union of all those paths, which we call the *central subtree*, is the minimum spanning tree of the points, and it is of polynomial size. See Figure 4.3. In polynomial time we can check all the vertices of the central subtree tree as possible solutions to the GHREVERSE instance.

central subtree

Now consider a point u that is not in the central subtree. Because all the points in the space form a tree, there must be some point v in the central subtree such that all paths from u to points in the central subtree pass through v, including all paths from u to any x_i or y_i. Then for any x_i, $d(u, x_i) = d(u, v) + d(v, x_i)$. The same holds for any y_i. All the inequalities that define whether v is a solution to

the GHREVERSE instance also apply to u with the addition of $d(u,v)$ on both sides. So u is a solution to the instance if and only if v is; testing every v also gives us the answer for all u.

Therefore we can solve an instance of GHREVERSE on this space in polynomial time, and the problem is in $\mathcal{P}$. ■

For VPREVERSE, even subtler distinctions can be made among spaces, and to resolve them we introduce the following simplified VPREVERSE problem which can be used as a bellwether for a space's difficulty.

Definition 4.8 (The Simplified VPREVERSE (SVPREV) Problem)
Given a point x in some tree metric space, a subset Y of points adjacent to x in the tree, and an interval of reals I (which may be unbounded), accept if and only if there exists a point z such that $d(x,z) \in I$ and the path from x to z does not pass through any element of Y.

Theorem 4.6
If S is a space with a (possibly weighted) tree metric, and $\textsc{Path}(x,y)$ runs in polynomial time for this space, then VPREVERSE on S is polynomial-time reducible to SVPREV.

Proof As in Theorem 4.5, we begin by finding all paths between points mentioned in the VPREVERSE instance to form the central subtree (Figure 4.3). In polynomial time we can test all the points in the central subtree as possible solutions to the instance.

However, unlike the GHREVERSE case where points outside the central subtree each had the same status as some point in the central subtree, with VPREVERSE there could be some satisfying points outside the central subtree without any in it. We detect such points by invoking SVPREV.

For each point y in the central subtree and each point x outside it but joined through y (that is, all the paths from x to points in the central subtree pass through y), then whether x satisfies the instance can be determined by examining its distance to y and the status of y. The distance $d(x,y)$ is added to $d(y,z)$ to get $d(x,z)$ for all z in the central subtree, and $d(x,y)$ does not change for different z. So for each y in the central subtree, we construct an SVPREV instance centred on y. Let Y (the excluded neighbours) be all the neighbours of y that are in the central subtree, because we wish to consider only points outside the central subtree and joined to it through y. Determine the interval of $d(x,y)$ values which would allow a point joined through y to be a solution to

the VPREVERSE instance and let I be that interval. Evaluating this instance for each y in the central subtree allows us to find any remaining solutions for the VPREVERSE instance. ■

The restriction to spaces where PATH(x, y) runs in polynomial time serves to limit the problem to cases where it is reasonable to do computations on paths at all. Theorems 4.5 and 4.6 should apply to any practical tree space not designed specifically to fall outside the criterion; see Example 4.12 for a carefully-designed space in which Theorem 4.5 may fail.

Given a space where computing paths is easy and Theorem 4.5 applies, the SVPREV problem still may or may not be hard, which is why we separate it out into a defined problem. We would normally expect it to be easy in any practical space, such as the prefix-metric space. But a carefully-designed space can make it non-trivial, as in the following example. The reduction in Theorem 4.6 may not always go in the other direction because of the difficulty in some spaces of expressing the "no path through Y" constraint in terms of spheres, so hardness of SVPREV does not necessarily prove hardness of VPREVERSE; but in the example, the VPREVERSE problem is $\mathcal{NP}$-hard anyway.

Example 4.9

For an arbitrary instance of 3SAT with variables $v_1, v_2, \ldots, v_n$, construct a space where the points are all binary strings $b_1 b_2 \ldots b_k$ where $0 \leq k \leq n$ such that no constraints are violated by assigning each v_i for $1 \leq i \leq k$ to true if and only if b_i is 1. Note that the empty string λ is always included. Use the prefix metric of Definition 4.5, which is equivalent to building a trie of the strings and using unit weights.

It is trivial to test whether any given string is included in the space, and to find the distance and path between any two strings if they are included in the space. The GHREVERSE problem in this space is easy by Theorem 4.5. However, the question of whether there exists a point at distance at least n from λ is the question of whether there exists a satisfying assignment for the original 3SAT instance. That is an instance both of VPREVERSE (with just one sphere) and SVPREV (with the set Y empty), so those problems in this space are as hard as 3SAT.

4.4 Badly-behaved tree metrics

Tree metrics may seem so simple as to render our results on them trivial. That appearance is misleading. Tree metrics can in fact be defined in devious ways that render them hard to handle. We already mentioned Example 4.4, which

is not a tree metric under our definition but meets the definition used by some other authors. In this section we present some of the other difficult cases hinted at by the limitations in the theorem statements of previous sections.

Theorem 4.2 gives the maximum number of distance permutations that can occur in any tree metric space, but Corollary 4.3 only gives sufficient, not necessary, conditions for that many permutations to actually occur. Any ordinary infinite space, such as the prefix metric space of Definition 4.5, would be expected to contain the maximum number of permutations for some choice of sites; if edge weights are not all the same we can still find $\binom{k}{2}$ distinct cuts by using a longer path, provided the weights are reasonably close to uniform.

In a finite space we might expect to find fewer than $\binom{k}{2}+1$ distance permutations. Since the number of permutations can never exceed the number of points in the space, but the upper bound grows quadratically with the number of sites, and there may be as many sites as there are points, then it is clear that the bound cannot always be achieved in a finite space: we could run out of points to label with distinct distance permutations. Note that our proof in Corollary 4.3 that the bound can *ever* be achieved at all, relies on a number of points exponentially larger than the number of sites.

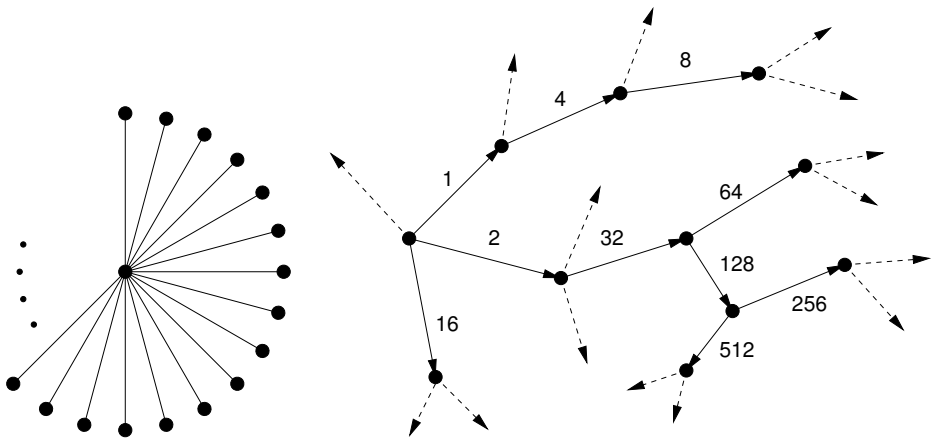
The question remains of whether Theorem 4.2 could fail in some more interesting way, with fewer than $\binom{k}{2}+1$ permutations even though the space is infinite. We give two examples of ways it can fail in infinite tree metric spaces. Both are illustrated by Figure 4.4. The upper bound can be achieved in spaces with long paths having reasonably uniform weights; it fails if there are no long paths, or if the weights are not reasonably uniform, and we give one example of each. Note that in each example there are k distance permutations; that is a trivial lower bound because each of the sites must have a distance permutation starting with itself, so with k distinct sites there must be at least k distinct distance permutations for the entire space.

Example 4.10 (No long paths)

Let the points be the nonnegative integers, and let there be an edge (with weight 1) from zero to each of the others. Note that this is distinct from the star graph space of Section 4.1 because that was a finite space and this one is countably infinite. Now if k points are chosen as sites, each of them has its own distance permutation; the zero point has the same distance permutation as the lowest-indexed site (if zero is not a site itself); and all other points have the same distance permutation as zero. So there are k distinct distance permutations.

Example 4.11 (Unreasonable weights)

Build a tree by starting with one vertex and adding vertices one at a time, each with an edge to one existing vertex called its parent. Let the weight



(a) No long paths (b) Unreasonable weights

Figure 4.4: Infinite tree spaces with only k distance permutations.

of each new edge be greater than the total weight of all previously-existing edges, which could be accomplished by giving them power-of-two weights in order $(1, 2, 4, \ldots)$. Note that any tree with only finite-degree vertices can be given an ordering and edge weights like this, by a breadth-first search.

The midpoint between any two vertices x and y, where y was added after x, must cut the tree on the edge from y to its parent, because that edge must be on the path, the other edges must have been added earlier, and the sum of all earlier-added edges is still less than the weight of the edge from y to its parent. Therefore among any k sites, every pair's midpoint must fall on the edge from the later-added site to its parent. One of the sites is the earliest-added; each of the remaining $k-1$ sites has an associated edge that cuts the tree; and so the tree is cut into k components corresponding to k distinct distance permutations.

The statements of Theorems 4.5 and 4.6 also have a subtle condition: they are limited to tree metric spaces where it is a polynomial-time problem to find the path, with weights, between two points. That raises the question of how common such spaces are. Are they easy to find? Yes—we expect all practical spaces to be of this type.

However, the theorems need this limitation because it may be possible to deliberately construct a very badly-behaved space in which they could fail. Our example encodes a problem from $\mathcal{UP}$ into a tree space in such a way that distances are easy to calculate but paths depend on solving the problem; if $\mathcal{P} \neq \mathcal{UP}$, then we can choose a problem to make path-finding non-polynomial. This example

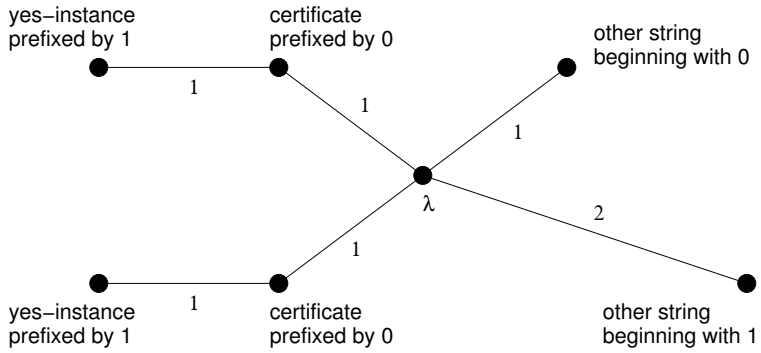


Figure 4.5: A space with easy distances and hard paths.

$\mathcal{UP}$ appeared in SISAP'08 [192]. The class $\mathcal{UP}$ is the class of decision problems in which yes-instances have unique polynomial-time certificates. It is known that $\mathcal{P} \neq \mathcal{UP}$ if and only if worst-case one-way functions exist, which is a necessary condition for security of cryptographic hashes [90].

Example 4.12

For any given problem in $\mathcal{UP}$, create a weighted tree metric space where the points are the set of all binary strings and the edges are defined as follows.

Let there be an edge with weight 1 from every string that begins with 0 to the empty string λ. For every string of the form $1x$, let there be an edge from $1x$ to λ with weight 2 if x is not the encoding of a yes-instance for the problem. If x is the encoding of a yes-instance for the problem, let there be an edge with weight 1 from $1x$ to $0x1z0x^{-1}$ where z is the unique certificate for x and x^{-1} is x with the order of bits reversed. See Figure 4.5.

In this space, the distance between any two given strings is easy to compute. For strings x and y with $x \neq y$, we have

$$\begin{aligned} d(\lambda, 0x) &= 1 \\ d(\lambda, 0x) &= 2 \\ d(0x, 0y) &= 2 \\ d(1x, 1y) &= 4\,. \end{aligned}$$

If z is the unique certificate for x, which we can test in polynomial time by examining $0x1z0x^{-1}$, then $d(0x1z0x^{-1}, 1x) = 1$. For any other cases, $d(0x, 1y) = 3$.

However, finding the path between $1x$ and λ where x is the encoding of an instance of the problem in $\mathcal{UP}$, requires solving the problem in order

to write out the certificate that might be on the path; and that may not be polynomial-time if $\mathcal{P} \neq \mathcal{UP}$.

Chapter 5

Hamming distance

An *edit distance* is a distance function that counts the minimum number of edits made on one object to turn it into another object. Provided that the individual edits are symmetric—that is, making and reversing any given edit count the same toward the distance—edit distances are always metrics. The Hamming distance considered in this chapter is a simple metric on strings and an example of an edit distance.

edit distance

Definition 5.1

Where x and y are strings with the same length, the *Hamming distance* $d(x, y)$ is the number of locations at which they differ. That is, if $x_1 x_2 \ldots x_n$ and $y_1 y_2 \ldots y_n$ are the letters in x and y, then

Hamming distance

$$d(x, y) = |\{i \in \{1, 2, \ldots, n\} | x_i \neq y_i\}|. \tag{5.1}$$

The Hamming distance is usually applied to binary strings, and that is the case we primarily consider here, but it can be applied to strings on any alphabet, and even to vectors of reals or more general elements.

The space of n-bit binary strings with Hamming distance is central to the field of coding theory, and widely studied as a result [169]. Movement through this space provides a model of what happens to a signal subjected to bit errors, so many problems in communications are stated in terms of Hamming distance. For instance, Hamming codes, which are basic to coding theory, are described in terms of their *minimum distance*: the smallest Hamming distance between any two code words, which determines how many errors the code can guarantee to correct. These codes, like the metric, are named for Richard V. Hamming, a pioneer in the field.

minimum distance

Many computer architectures provide an instruction, generally named "population count," for computing the *Hamming weight* of a machine word, which is the same as the number of 1 bits in the word or its Hamming distance from

Hamming weight

the all-zero word. Used with a bitwise exclusive-or instruction, population count can measure Hamming distance between any two words. Popular legend among computer programmers holds that population count must be included in computers sold to the US National Security Agency, as a contractual requirement. For that reason, the population count instruction is often called the "NSA instruction" [184, pages 379–380]. Warren comments that "No one (outside of NSA) seems to know just what they use it for." [212, page 160]

Indexing systems may attempt to reduce a high-dimensional database to something more computationally tractable by describing each object with the answers to a list of yes-or-no questions. The Nilsimsa spam filter is one example of such a system, in which email messages are reduced to 256-bit digests representing the answers to 256 questions about their content [53, 159]. The questions express whether individual hash buckets (filled with hashed trigram counts) have more or less than the median count. Testing two messages for similarity then means examining the Hamming distance between their digests. This kind of match counting applies any time that objects are described in terms of a fixed list of features that may or may not be present; thus, Hamming distance on binary strings can become relevant to indexing objects even when the objects themselves are more naturally thought of as existing in some other space.

An attempt to solve our VPREVERSE and GHREVERSE problems (Definitions 1.27 and 1.29) approximately instead of exactly would also implicate Hamming distance: the constraints in an instance form a list of bits describing a point by its answers to yes-or-no questions, much like a Nilsimsa digest. A point that almost, but not necessarily exactly, solves the instance would be a point whose digest is within a small Hamming distance of that specified by the instance.

5.1 Intrinsic dimensionality

intrinsic dimensionality

The difficulty of indexing strings with Hamming distance can be measured by the *intrinsic dimensionality* ρ of the space, defined as the square of the mean divided by twice the variance of the distance D between two random points drawn from the native distribution (Definition 1.23). Where μ'_1 and μ'_2 are the first two raw moments of the distance, intrinsic dimensionality is given by

$$\rho = \frac{\mathrm{E}^2[D]}{2\mathrm{V}[D]} = \frac{\mu'^2_1}{2(\mu'_2 - \mu'^2_1)}. \tag{5.2}$$

In this section we consider the intrinsic dimensionality of binary strings with Hamming distance. The first result (on Bernoulli-distributed bits) appeared previously in SPIRE'05 [190].

The most obvious native distribution for n-bit binary strings is a simple uniform choice from the 2^n possible strings. Consider a slightly more general case: let the bits be independent and identically distributed Bernoulli random variables, equal to 1 with probability q and 0 otherwise. We use q for the probability to avoid conflict with p from L_p distance. If we treat these strings as vectors, their Hamming distance is equal to their L_1 distance (the sum of per-component differences, Definition 2.1) and then by Corollary 2.2, the intrinsic dimensionality is given by $\rho = nq(1-q)/(1-2q+2q^2)$. By simple calculus, the maximum intrinsic dimensionality of $n/2$ is achieved with $q = 1/2$, which is the uniform-distribution case.

As a more complicated example, consider a ball of radius r in Hamming-distance space with the uniform distribution. In other words, the native distribution is, for some centre string c and radius r, to select a string x such that $d(x,c) \leq r$ uniformly from the set of all such strings. The string length n is assumed to grow large in comparison to the radius r. In that case, the intrinsic dimensionality is linear in n with a constant that depends on r and increases with r to a limit of $1/2$, agreeing with the previous result.

Theorem 5.1
In the space of n-bit binary binary strings chosen uniformly from a ball of constant radius r, the intrinsic dimensionality ρ obeys

$$\rho \to \frac{r}{2r+1} n \,. \tag{5.3}$$

Proof Consider how many ways we could choose i of the n bits, then j of the remaining $n-i$ bits, then k of the remaining $n-i-j$ bits. This number is given by the *multinomial coefficient* [85, 88, page 168]

multinomial coefficient

$$\binom{n}{i,j,k,n-i-j-k} = \frac{n!}{i!j!k!(n-i-j-k)!} \,. \tag{5.4}$$

We can find the first two terms of the expansion of (5.4) into powers of n as follows:

$$\begin{aligned} &\binom{n}{i,j,k,n-i-j-k} \\ &\qquad = \frac{1}{i!j!k!} \left[(n)(n-1)(n-2)\cdots(n-i-j-k+1) \right] \end{aligned}$$

$$= \frac{n^{i+j+k}}{i!j!k!}\left[1 - \left(\sum_{s=0}^{i+j+k-1} s\right) n^{-1} + o\left(n^{-1}\right)\right]$$
$$= \frac{n^{i+j+k}}{i!j!k!}\left[1 - \frac{1}{2}(i+j+k)(i+j+k-1)n^{-1} + o\left(n^{-1}\right)\right].$$

If we choose two strings x and y from the ball, let i be the number of bit positions where x is different from c and y is equal, let j be the number of bit positions where y is different from c and x is equal, and then let k (which must be from zero to $r - \max\{i, j\}$) be the number of bit positions where x and y are both different from c and thus equal to each other. We can count the number of ways to choose these two strings as

$$\begin{aligned}
N &= \sum_{i=0}^{r}\sum_{j=0}^{r}\sum_{k=0}^{r-\max\{i,j\}} \binom{n}{i, j, k, n-i-j-k} \\
&= \binom{n}{r, r, 0, n-2k} + \binom{n}{r-1, r-1, 1, n-2r+1} \\
&\qquad + \binom{n}{r-1, r, 0, n-2r+1} + \binom{n}{r, r-1, 0, n-2r+1} + o\left(n^{2r-1}\right) \\
&= \frac{1}{r!^2}n^{2r} + \left[\frac{1}{(r-1)!^2} + \frac{2}{r!(r-1)!} - \frac{2r-1}{r!^2}\right] n^{2r-1} + o\left(n^{2r-1}\right) \\
&= \frac{n^{2r}}{r!(r-1)!}\left[\frac{1}{r} - (r-3)n^{-1} + o\left(n^{-1}\right)\right]. \qquad (5.5)
\end{aligned}$$

To compute the first two raw moments of the distance between two strings chosen uniformly from the ball, we note that that distance is $i + j$ and so its expected value can be computed in the same way:

$$\begin{aligned}
\mu_1' &= \frac{1}{N}\sum_{i=0}^{r}\sum_{j=0}^{r}\sum_{k=0}^{r-\max\{i,j\}} (i+j)\binom{n}{i, j, k, n-i-j-k} \\
&= \frac{1}{N}\left[2r\binom{n}{r, r, 0, n-2r} + 2(2r-1)\binom{n}{r, r-1, 0, n-2r+1}\right. \\
&\qquad \left. + (2r-2)\binom{n}{r-1, r-1, 1, n-2r+1} + o\left(n^{2r-1}\right)\right] \\
&= \frac{1}{N}2n^{2r-1}\left[\frac{r}{r!^2}n + \frac{-r^2(2r-1)}{r!^2} + \frac{2r-1}{r!(r-1)!} + \frac{r-1}{(r-1)!^2} + o(1)\right] \\
&= \frac{1}{N}\cdot\frac{2n^{2r}}{r!(r-1)!}\left[1 - (r-1)^2 n^{-1} + o\left(n^{-1}\right)\right]. \qquad (5.6)
\end{aligned}$$

When we substitute the value of N from (5.5) into (5.6), the $n^{2r}/r!(r-1)!$

factors cancel out, and we can find μ_1' by long division:

$$\begin{array}{r} 2r - 2r(r+1)n^{-1} + o\left(n^{-1}\right) \\ \hline \frac{1}{r} - (r-3)n^{-1} + o\left(n^{-1}\right) \;\Big)\; 2 - 2(r-1)^2 n^{-1} + o\left(n^{-1}\right) \\ 2 - 2r(r-3)n^{-1} + o\left(n^{-1}\right) \\ \hline -\ 2(r+1)n^{-1} + o\left(n^{-1}\right) \\ -\ 2(r+1)n^{-1} + o\left(n^{-1}\right) \\ \hline \ddots \end{array}$$

$$\mu_1' = 2r - 2r(r+1)n^{-1} + o(n^{-1}). \tag{5.7}$$

Note 5.2

There is an intuition for why this should be the value of μ_1', at least as far as the leading term. As discussed in Section 1.3, points in high-dimensional spaces tend to all be equally distant from each other, at the maximum possible distance, and most (in the limit, all) of the volume of an object tends to be on its surface. Applying those heuristics to two points chosen uniformly from a ball of radius r, we should expect the points to be on the surface of the ball (at distance r from the centre). We should also expect k to go to zero, because the chance of choosing r bits for one string to collide with the r bits from the other, will decrease with increasing n; thus all $2r$ differing bits will tend to count. Sure enough, $\mu_1' \to 2r$: the expected distance approaches $2r$ for large n.

Since the intrinsic dimensionality formula (5.2) uses the square of μ_1', we take the opportunity to calculate it:

$$\mu_1'^2 = \left[2r - 2r(r+1)n^{-1} + o\left(n^{-1}\right)\right]^2 = 4r^2 - 8r^2(r+1)n^{-1} + o\left(n^{-1}\right) \tag{5.8}$$

For the second raw moment, we put $(i+j)^2$ inside the summation and find the first two terms of its expansion:

$$\begin{aligned} \mu_2' &= \frac{1}{N}\sum_{i=0}^{r}\sum_{j=0}^{r}\sum_{k=0}^{r-\max\{i,j\}} (i+j)^2 \binom{n}{i,j,k,n-i-j-k} \\ &= \frac{1}{N}\left[4r^2\binom{n}{r,r,0,n-2r} + 2(2r-1)^2\binom{n}{r,r-1,0,n-2r+1}\right. \\ &\qquad \left. +4(r-1)^2\binom{n}{r-1,r-1,1,n-2r+1} + o\left(n^{2r-1}\right)\right] \\ &= \frac{1}{N}2n^{2r-1}\left[\frac{2r^2}{r!^2}n + \frac{-2r^2(2r-1)}{r!(r-1)!} + \frac{(2r-1)^2}{r!(r-1)!} + \frac{2(r-1)^2}{(r-1)!^2} + o(1)\right] \end{aligned}$$

$$= \frac{1}{N} \cdot \frac{2n^{2r-1}}{r!(r-1)!} \left[2rn - 4r^3 + 2r^2 + 4r^2 - 4r + 1 + 2r^3 - 4r^2 + 2r + o(1) \right]$$
$$= \frac{1}{N} \cdot \frac{2n^{2r}}{r!(r-1)!} \left[2r - (2r^3 - 2r^2 + 2r - 1)n^{-1} + o\left(n^{-1}\right) \right] \tag{5.9}$$

Substituting (5.5) into (5.9), the $n^{2r}/r!(r-1)!$ factors again cancel out, and μ_2' follows:

$$\begin{array}{r|l}
 & 4r^2 - 2r(4r^2+2r-1)n^{-1} + o\left(n^{-1}\right) \\
\hline
\frac{1}{r} - (r-3)n^{-1} + o\left(n^{-1}\right) & 4r - (4r^3-4r^2+4r-2)n^{-1} + o\left(n^{-1}\right) \\
 & 4r - (4r^3-12r^2)n^{-1} + o\left(n^{-1}\right) \\
\hline
 & -(8r^2+4r-2)n^{-1} + o\left(n^{-1}\right) \\
 & -(8r^2+4r-2)n^{-1} + o\left(n^{-1}\right) \\
\hline
 & \ddots
\end{array}$$

$$\mu_2' = 4r^2 - 2r(4r^2+2r-1)n^{-1} + o\left(n^{-1}\right). \tag{5.10}$$

The we can use (5.8) and (5.10) to evaluate the intrinsic dimensionality formula (5.2) and get (5.3).

$$\begin{aligned}
\rho &= \frac{1}{2} \cdot \frac{\mu_1'^2}{\mu_2' - \mu_1'^2} \\
&= \frac{1}{2} \cdot \frac{4r^2 - 8r^2(r+1)n^{-1} + o\left(n^{-1}\right)}{4r^2 - 2r(4r^2+2r-1)n^{-1} - 4r^2 + 8r^2(r+1)n^{-1} + o\left(n^{-1}\right)} \\
&= \frac{r - 2r(r+1)n^{-1} + o\left(n^{-1}\right)}{(2r+1)n^{-1} + o\left(n^{-1}\right)} \\
&\to \frac{r}{2r+1}n
\end{aligned}$$

5.2 Distance permutations

site

Suppose that given k fixed strings $x_1, x_2, \ldots, x_k$, called the *sites*, for any string y we find the closest site to y, the second-closest site to y, and so on, to form a permutation of the sites. If two sites are equidistant from y we choose the lowest-index one first, to make the permutation unique. Such a permutation is called the *distance permutation* of y (Definition 1.25). On strings with Hamming distance, it represents a generalisation of the nearest-neighbour decoding concept from coding theory [169, page 19].

distance permutation

We are interested in how many distinct distance permutations can occur among the 2^n binary strings of length n, if the k sites are chosen to maximise the number of distinct distance permutations. Let $N_{n,\mathrm{H}}(k)$ represent that number. The H for "Hamming" in the subscript is to distinguish this from the similar notation used in Chapter 3 for the maximum count of distance permutations in L_p space. An exact solution for this question seems difficult, partly because of the complexity introduced by tiebreaking among equally distant sites, but we offer some bounds.

Trivial bounds on $N_{n,\mathrm{H}}(k)$ follow naturally from the definitions:

$$N_{n,\mathrm{H}}(k) \leq 2^n \tag{5.11}$$

$$N_{n,\mathrm{H}}(k) \leq k! \tag{5.12}$$

$$N_{n,\mathrm{H}}(k) \geq k\,. \tag{5.13}$$

Because there are only 2^n points in the space and each one has only one distance permutation, there cannot be more than 2^n distance permutations (5.11). Similarly, with k sites there cannot be more than $k!$ permutations of any description, so there can be at most $k!$ distance permutations (5.12). An even stronger statement than (5.13) can actually be made: as described in Section 4.4, there are always at least as many distinct distance permutations as there are distinct sites (not just in the maximum case implied by $N_{n,\mathrm{H}}(k)$).

A less-obvious bound follows from the work on vectors with L_p metrics in Chapter 3. Consider each n-bit binary string as an n-component vector all of whose components happen to be equal to 0 or 1; that is, a *zero-one vector*. Then the Hamming distance between two strings is just the L_1 distance between the corresponding vectors. Restricting vectors to be of this form cannot create any additional distance permutations over the ones that would exist for the same number of unrestricted vectors, so where $N_{n,p}(k)$ is the maximum number of distance permutations of k sites for n-component vectors with the L_p metric, we have $N_{n,\mathrm{H}}(k) \leq N_{n,1}(k)$.

zero-one vector

Furthermore, the L_p distance between two zero-one vectors for any finite p is a strictly increasing function of the Hamming distance, and so the distance permutation of a given point with a given list of sites will be the same regardless of which finite-p L_p metric we use. Thus $N_{n,\mathrm{H}}(k) \leq N_{n,p}(k)$ for all finite p. Combining that statement with Theorem 3.4 in the case $p = 2$ gives an asymptotic bound on the maximum number of distance permutations:

$$N_{n,\mathrm{H}}(k) \leq N_{n,2}(k) = O\left(k^{2n}\right)\,. \tag{5.14}$$

Note 5.3

This argument does not apply to the L_∞ metric because it is not a strictly increasing function of Hamming distance. All unequal zero-one vectors

have L_∞ distance exactly 1. (It degenerates to the equality metric of Example 1.3.) This breakdown of strict increase is discussed at length in Chapter 8, where it necessitates different proof techniques for L_∞ from the other L_p metrics.

It is natural to ask under what circumstances we might be able to have a distinct distance permutation for each of the 2^n points. As the following theorem shows, that can be achieved with $n+1$ sites, and nearly achieved with n sites.

Theorem 5.2
In the space of n-bit binary strings with Hamming distance, the maximum number of distance permutations for k sites $N_{n,\mathrm{H}}(k)$ obeys

$$N_{n,\mathrm{H}}(n+1) = 2^n\,, \tag{5.15}$$

$$N_{n,\mathrm{H}}(n) \geq 2^n - n\,. \tag{5.16}$$

Proof Let u_i be the n-bit string consisting entirely of zero bits except for a one bit in bit position i counted from 1; that is,

$$u_i = 0^{i-1}10^{n-i}\,. \tag{5.17}$$

Consider (5.15). The sites used are $u_1, u_2, \ldots, u_n, 0^n$ in that order. For any arbitrary n-bit binary string y, let h be its Hamming weight (the number of 1 bits in the string). The distance from y to u_i is $h+1$ if y contains a 0 in bit position i, and $h-1$ if y contains a 1 in bit position i. The distance from y to 0^n is always h.

Then from any string y, the distance permutation will consist of the indices of all the bit positions where it contains 1, in order of increasing index; then n (the index of the all-zero site); then the indices of all remaining sites, which correspond to the bit positions where y contains 0. Furthermore, given any permutation of that form, we can find a unique y to generate that distance permutation by placing 1 in all the positions whose indices appear before n and 0 in all the positions whose indices appear after n. For instance, with $n=6$, $k=7$, the permutation $\langle 2,4,5,7,1,3,6\rangle$ corresponds to the bit string 010110. Therefore, there is a bijection between distance permutations and points in the space, and (5.15) holds.

Consider (5.16). The sites are the same except without the all-zero string: $u_1, u_2, \ldots, u_n$. As in the previous case, the distance permutation of a string y will consist of all the indices of its 1 bits in ascending order followed by all the

indices of its 0 bits in ascending order. Each pair of successive indices will be increasing except the pair representing the last 1 and first 0. If the first 0 bit comes before the first 1 bit, that pair will be decreasing, so we can decode such a permutation unambiguously to find the value of y by searching for the one decreasing pair of indices. Indices before that correspond to 1 bits and indices after that correspond to 0 bits. For example, with $k = n = 6$, the permutation $\langle 1,4,2,3,5,6\rangle$ corresponds to the bit string 100100.

However, if it happens that all 1 bits (if any) appear before all 0 bits (if any), then the distance permutation of y will be the identity permutation, and there may be many such bit strings. For instance, with $k = n = 6$, the strings 100000 and 111100 both give the distance permutation $\langle 1,2,3,4,5,6\rangle$. All strings with this property must be of the form $1^i 0^{n-i}$ for some integer $0 \le i \le n$. There are $n+1$ strings, together they share one distance permutation, and among the 2^n points in the space these are the only ones with non-unique distance permutations. Therefore the total number of distinct distance permutations is $2^n - n$, and that is a lower bound on $N_{n,\mathrm{H}}(n)$ (5.16). ■

The bound given by (5.16) is not perfectly tight, as shown by the following example.

Example 5.4
Let $k = n = 6$ and let the sites be $\langle$000000, 000011, 001101, 011110, 110100, 111001$\rangle$. Then each point in the space has a unique distance permutation except for these three pairs of points, each of which describes two points with the same distance permutation:

000010	000110	$\langle 1,2,4,3,5,6\rangle$
010001	100001	$\langle 1,2,6,3,5,4\rangle$
011110	101110	$\langle 4,3,5,1,2,6\rangle$

.

That makes a total of 61 distance permutations, where the lower bound is only 58.

From the other direction, we can ask how many dimensions are necessary to provide a point for each of the $k!$ distance permutations. The following result gives a loose upper bound on the answer.

Theorem 5.3
We can choose k binary strings of length $k(k-1)$ such that for each of the $k!$ permutations of sites there is a string having that as its distance permutation.

That is,

$$N_{k(k-1),\mathrm{H}}(k) = k!\,. \tag{5.18}$$

Proof We divide each string into k blocks of $k-1$ bits each. The sites x_i are the strings with one block consisting of all ones and all other blocks zero:

$$x_i = 0^{(i-1)\cdot(k-1)}1^{k-1}0^{(n-i)\cdot(k-1)} \text{ for integers } 1 \le i \le n\,.$$

Then where $\pi : \{1,2,\ldots,n\} \to \{1,2,\ldots,n\}$ is the distance permutation, consider this string:

$$y = 1^{k-\pi(1)}0^{\pi(1)-1}1^{k-\pi(2)}0^{\pi(2)-1}\ldots 1^{k-\pi(k)}0^{\pi(k)-1}\,.$$

The number of ones in y is $\pi(1)-1+\pi(2)-1+\cdots+\pi(k)-1$, which is a constant relative to k because π is a permutation. Call this number h. (It happens to be equal to $k(k-1)/2$.) Now the distance from y to a site x_i is h, because ones in y generally differ from zeros in x_i; minus the number of ones in the i-th block of y, because within that block they match; plus the number of zeros in the i-th block, which differ from the ones in x_i. Therefore the distance is $h-k-1+2\pi(i)$. That is a strictly increasing function of $\pi(i)$, so the closest site is x_i where $\pi(i)=1$, the second-closest site is x_i where $\pi(i)=2$, and so on; the distance permutation is exactly π. We can find such a y for any π, so $N_{k(k-1),\mathrm{H}}(k) = k!$. ∎

The encoding used in Theorem 5.3 is more powerful than necessary in that it allows all the inequalities defining the distance permutations to be strict, for all $k!$ permutations. The space includes many points from which two or more sites are equidistant, and none of those are used by the construction. Making use of the tiebreaking rule in the definition of distance permutations can allow for all $k!$ permutations with fewer dimensions. For instance, with $n=4$ and $k=3$, we have all $k! = 6$ permutations with the sites $\langle 0001, 0010, 1100\rangle$.

5.3 Reverse similarity search

The difficulty of reverse similarity search on binary strings with Hamming distance follows from work by Frances and Litman on two closely-related problems: the "minimum radius" (MR) and "maximum covering radius" (MCR) problems [76]. These results were presented, with the proofs given as sketches, at SISAP'08 [192].

The formal definitions we give below are generalised to all metric spaces, and written in a way that emphasises their similarity. We mention numeric precision for cases where it may be important, but since we only use the case of binary strings with Hamming distance, where distances must be integers, the issue vanishes.

Definition 5.5 (The MR Problem)
In some metric space (S, d), given a set $C \subseteq S$ and real r given to some precision, accept if and only if there exists a point $z \in S$ such that $d(x, z) \leq r$ for all $x \in C$.

Definition 5.6 (The MCR Problem)
In some metric space (S, d), given a set $C \subseteq S$ and real r given to some precision, accept if and only if there exists a point $z \in S$ such that $d(x, z) \geq r$ for all $x \in C$.

The MR problem asks whether there exists a sphere of a given radius such that all the points in a set are inside the sphere, and the MCR problem asks whether, for a list of spheres, there exists a point anywhere in the space that is outside all of them. Recall that VPREVERSE is a constraint satisfaction problem in which we list spheres, specify inside or outside for each one, and seek a point meeting all the constraints (Definition 1.27). Then MCR is a special case of VPREVERSE where the spheres all have the same radius and the solution must be outside all of them.

Frances and Litman briefly present an argument due to Karpovsky that the MR and MCR problems are equivalent [76, 118]. The underlying insight is simple: in the space of binary strings with Hamming distance, every string x has a *complement* $\bar{x}$ formed by inverting all its bits and the total distance from anywhere to x and $\bar{x}$ must always be n, the length of the strings. If flipping some bits gives one string, flipping all the others must give the complementary string. Then the complement of a ball around one centre c must be another ball centred on $\bar{c}$. A point within a radius of all the points on a list (which would solve MR) must be the complement of a point outside a radius of all the points on the list (thus a solution to MCR). Solving either problem then also solves the other.

complement

Then they proceed to show the $\mathcal{NP}$-completeness of MCR by a reduction from 3SAT [76]. Each variable is represented by a pair of bits, 00 for false and 11 for true. One additional pair is used as a slack variable, so the total length of the strings is $2(n+1)$ for n variables. It is possible to place constraints on subsets of the variables by using 01 as a don't care value; it is equidistant from 00 and 11. Clauses from 3SAT are translated into statements about whether the solution is or is not within distance $n+1$ of a string consisting mostly of repetitions of 01, with other values in the pairs corresponding to the literals in the clause. The

result is an instance of MR satisfiable if and only if the original 3SAT instance was satisfiable. Then MCR is $\mathcal{NP}$-complete also, because the two problems are equivalent, and since an MCR instance in this space is also a VPREVERSE instance, we have $\mathcal{NP}$-completeness of VPREVERSE.

Theorem 5.4
In the space of binary strings with Hamming distance, VPREVERSE is $\mathcal{NP}$-complete, even when the spheres are constrained to all have the same radius.

Proof As described above, the MCR problem (Definition 5.6) on this space was shown $\mathcal{NP}$-complete by Frances and Litman [76]. An instance of MCR is an instance of VPREVERSE, so the existence of hard MCR instances implies the existence of hard VPREVERSE instances, and VPREVERSE on this space is $\mathcal{NP}$-hard. Moreover, all the spheres in the hard instances happen to have the same radius of $n+1$ where n is the number of variables in the corresponding 3SAT instance. The VPREVERSE problem is also in $\mathcal{NP}$ because we can test in polynomial time that the solution string does meet the constraints. ■

Recall from Definition 1.29 that GHREVERSE is a constraint satisfaction problem where the constraints are pairs of points (x, y); the solution z must satisfy $d(z, x) \leq d(z, y)$ for every constraint in the problem. Imagine what happens if the two points in a pair are complementary, $y = \bar{x}$. Then the constraint is that the distance to x must be less than or equal to half the number of bits in the strings. That is exactly the same as a sphere constraint from a VPREVERSE instance, with the radius equal to half the number of bits. The hard instances of VPREVERSE proved to exist in Theorem 5.4 have only spheres of that radius, so they can be easily converted to GHREVERSE instances.

Corollary 5.5
In the space of binary strings with Hamming distance, GHREVERSE is $\mathcal{NP}$-complete.

Proof The hard instances of MR proved to exist by Frances and Litman happen to involve strings of length $2(n+1)$ and the constraint that a point must be within radius $n+1$ of all the points in the input set [76]. The equivalent MCR instances

require the solution to be outside that radius. Either type of constraint can be rewritten as a constraint that the solution z must be closer to some point x than to $\bar{x}$. Those points are the sphere centre and its complement. The resulting list of constraints forms a GHREVERSE instance, so GHREVERSE is $\mathcal{NP}$-hard; and it is $\mathcal{NP}$-complete because the solution string is a polynomial-time certificate. ■

Chapter 6

Levenshtein edit distance

The previous chapter described the Hamming distance as an edit distance where edits consist of substituting letters in a fixed-length string. If we also allow insertions or deletions, which change the length of the string, the resulting edit distance is one called the Levenshtein distance, described in 1965 by its namesake Vladimir Levenshtein [138]. The term "edit distance" without qualification is often assumed to refer to Levenshtein distance, though we avoid that usage in this work because we also discuss several other kinds of edit distance.

Definition 6.1

The *Levenshtein distance* between two strings is the minimal number of edits required to transform one into the other, where each edit consists of inserting, deleting, or substituting a letter.

Levenshtein distance

Example 6.2

The Levenshtein distance from DEFENDER to BEFRIEND is 5. A minimal sequence of edits is to delete the two letters ER to give DEFEND, substitute one letter B for D to give BEFEND, and insert the two letters RI to give BEFRIEND.

Note that the Levenshtein distance is bounded below by the difference in length between the two strings, because that many edits must be used just to change the lengths regardless of the content of the strings, and bounded above by the sum of the lengths, because we could just delete all of one string and then insert all of the other. For equal-length strings, the Hamming distance is also an upper bound on the Levenshtein distance.

Levenshtein considered this metric in the context of coding theory, but it has since become popular in biological applications because it models mutations in DNA or other sequences. Common types of DNA mutations substitute one base for another, insert a base between two existing ones, or delete an existing base.

Rawn describes the chemistry behind these processes [173]. The metric can be generalised further by assigning different weights to the different types of edits. Sequence comparison techniques used in bioinformatics may attempt to choose weights that accurately model the probabilities of each mutation. Then the distance between two sequences expressed as strings can be taken as a measure of how far apart the sequences are in evolutionary terms, and used to infer phylogeny. The details of how bioinformatics applications generalise edit distance, attempt to approximate it with tractable algorithms, and apply it to multiple strings at once to form structures called alignments, are beyond the scope of the present work. Lesk gives a practical overview [137] while Dardel and Képès give more theoretical detail [55]. The alignment algorithms used in bioinformatics generally trace back to the work of Needleman and Wunsch on global alignments [158] and Smith and Waterman [193] on local alignments.

It is easy to compute Levenshtein distance exactly for two strings in $\Theta(n^2)$ time by a dynamic programming algorithm essentially the same as the one for longest common subsequence [208]. Masek and Paterson give a $O(n^2/\log n)$ algorithm [148] based on what has become known as the Four Russians technique [12]. Ukkonen gives a $O(dn)$ algorithm, where d is the distance; it can also test whether the distance is less than k in $O(kn)$ time [206]. Many of the results for longest common subsequence discussed in Section 6.1 are also applicable to computing Levenshtein distance.

But although computing Levenshtein distance between two strings is easy in the sense that it is in $\mathcal{P}$, many related problems remain too difficult to be practical. Edit distance questions involving more than two strings tend to be $\mathcal{NP}$-complete and hard to approximate [113], and quadratic time is too slow for strings the size of genome databases, necessitating approximate and compromise techniques like the ubiquitous BLAST [5].

In this chapter, we first consider the intrinsic dimensionality of strings with Levenshtein distance. It appears to be somewhere between linear and quadratic in string length. We characterise the number of neighbours of a string with Levenshtein distance; this is the first space considered in which number of neighbours is not trivial. Finally, we analyse the reverse similarity search problems VPREVERSE and GHREVERSE in this space, showing that both are $\mathcal{NP}$-complete.

6.1 Intrinsic dimensionality

Let x and y be two n-bit binary strings selected independently and uniformly at random. What is the distribution of the Levenshtein distance between them? Previous work on this question has primarily focused on the closely-connected question of the length of the longest common subsequence. If we disallow

substitutions, or make them cost at least twice as much as additions or deletions, then the resulting edit distance between two n-letter strings will be $2(n-l)$ where l is the length of the longest common subsequence. Most of the work relevant to this topic is aimed at that length l. However, the distribution of l is difficult to evaluate, even approximately. For instance, Dančík spends most of a PhD thesis making a small improvement in the bounds on the mean value [54]. In this section we discuss known results and their implications for the intrinsic dimensionality of strings with Levenshtein distance.

Finding the longest common subsequence given two strings is a classic example in teaching dynamic programming [50]. The usual $\Theta(n^2)$ algorithm for it has been independently rediscovered many times, but is often credited to Wagner and Fischer [208]. Many other algorithms for the exact problem have been proposed. Dančík surveys them through the early 1990s [54, pages 10–14]. Batu, Ergun, and Sahinalp in 2006 give a $\tilde{O}(n)$-time (that is, $O(n)$ up to logarithmic factors) algorithm to approximate the edit distance d within a factor of $\min\{n^{1/3+o(1)}, d^{1/2+o(1)}\}$, by means of a low-distortion embedding into a space of strings with larger alphabet and shorter length [22]. For the mutual longest common subsequence of a variable number of strings, the problem is $\mathcal{NP}$-hard; Jiang and Li give approximations and non-approximability results [113].

Chvátal and Sankoff introduce the problem of the mean length of the longest common subsequence between two random strings [45]. They give exact formulas for constant string length n up to five, and prove that in the limit for large n, the length of the longest common subsequence is linear in n with a constant of proportionality c_k determined by the alphabet size k. Notation for the constant used by various authors includes a_c [30], c_k [45], and γ_k [54]. The most popular one seems to be γ_k, but we follow Chvátal and Sankoff in using c_k, to avoid confusion with γ (the Euler-Mascheroni constant) and Γ (the generalised factorial) used elsewhere in this work.

The constants c_k are not known. For large alphabets, Kiwi, Loebl, and Matoušek show that $\lim_{k\to\infty} c_k = 2/\sqrt{k}$, which prove a conjecture given by Sankoff and Mainville [124, 180]. For c_2, the one relevant to binary strings, Chvátal and Sankoff show that $0.727273 \le c_2 \le 0.866595$ [45]. Deken improves the bounds to $0.7615 \le c_2 \le 0.8665$ [60, 61]. Dančík further improves them to $0.77391 \le c_2 \le 0.837623$ [54]. The current best bounds we know are $0.788071 \le c_2 \le 0.826280$ from Lueker in 2003, which rule out the conjecture of Arratia (reported by Steele) that $c_2 = 2/(1+\sqrt{2}) \approx 0.828427$ [143, 199].

An intuitive reason for the difficulty of computing the distribution is that the optimal sequence of edits between two strings must be evaluated globally. The best sequence of edits could include deleting an arbitrarily large perfectly-matched segment of the string if that allows a larger segment to be shifted to

match elsewhere. So there are many dependencies among the random variables involved in the solution.

To derive intrinsic dimensionality we need not only the mean of the distance but also its variance. Less work has been done on the variance than on the mean. Chvátal and Sankoff conjecture (apparently from their experimental results) that the variance is $o(n^{2/3})$ [45]. Steele gives an upper bound of $(1 - 1/k)n$ for the variance, where k is the alphabet size, so this is $n/2$ for binary strings [199]. Booth and others mention higher moments as a topic for future work in their paper of 2004 on longest common subsequence [30]; unfortunately, Booth is now deceased, and our reference seems to be the last publication from her group on this subject. Lember and Matzinger show, in a 2006 preprint, that for *sufficiently biased* binary strings the variance is $\Theta(n)$, but their result does not apply to the uniform case [136]. The variance question for the uniform binary case, let alone characterisation of the distribution beyond mean and variance, remains open.

All those results are for longest common subsequence. An edit distance allowing only insertions and deletions on n-bit strings has an obvious relation to longest common subsequence in that that edit distance must be $2(n-l)$ where l is the length of the longest common subsequence. Levenshtein actually considered such a distance in his 1965 paper, along with the distance we now call Levenshtein distance, which permits substitutions [138]. If we permit substitutions, it is not obvious how much of the work on longest common subsequence can still be applied. Navarro discusses this issue briefly, and shows that for a k-letter alphabet the Levenshtein distance between two random strings of length n is at least $n(1 - e/\sqrt{k})$, but for $k < 8$ (for instance, the binary-alphabet case) that is a negative number, and not useful [156].

We conjecture that even if the constants change, the same asymptotic behaviour known for the longest common subsequence between two random strings should also apply to Levenshtein edit distance, for three reasons. None of these are proofs, but they suggest intuitions for how future work on the problem could proceed.

First of all, much of the theory already cited for longest common subsequence can also be applied to the case where substitutions are permitted; for instance, the automata used by Dančík for lower bounds could be modified to count substitutions as cheaper than the equivalent insertion-deletion pairs, and although a new computation would be needed, and other complications would no doubt arise, the technique still appears workable [54].

Second, Arratia and Waterman prove phase transition behaviour for this problem depending on the values of two penalty parameters that express the cost of insertions, deletions, and substitutions [14]. The topography of the phase space is only partly established, but for the case of a binary alphabet with only

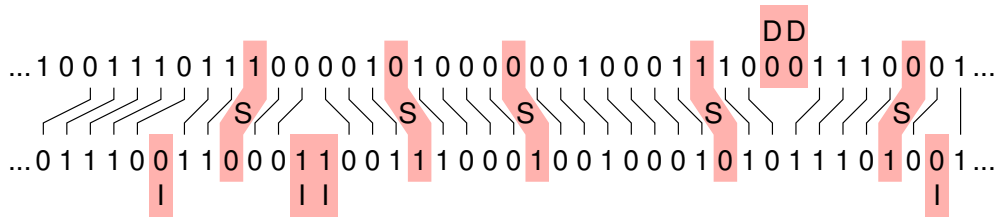


Figure 6.1: Edits between two long strings.

insertions and deletions permitted, the behaviour is already clearly into the linear phase. Permitting substitutions at the same cost as insertions or deletions only drives the problem further away from the phase boundary.

Third, consider Figure 6.1, which shows two 40-bit substrings from somewhere in the middle of two long random strings, with the edits between them marked. As can be seen, most of the bits match well, even though the substrings for the figure were chosen randomly. That should be no surprise, since we know about $4/5$ of bits should match, that being the approximate value of c_2. The important consequence is that non-matching bits are relatively rare, and tend to occur in small clusters (of size roughly exponentially distributed). Despite the fact that there may be arbitrarily large gaps in the optimal matching, those occur in highly unlikely strings, and for random strings, things tend to happen locally. The important consequence is that if we take an optimal edit sequence permitting only insertions and deletions, and then say that substitutions will be permitted too, very little will really change. Any pairs of consecutive insertions and deletions will be switched to the cheaper substitutions, and there may be local rearrangement where two bits that formerly paired will no longer pair in the optimal matching, but we will not see large segments of the string changing allegiance to dramatically different locations. The optimal matching will be substantially the same with either metric. As a result, we should expect that the constant of proportionality may change, but the general asymptotic linear behaviour, and even the order of magnitude of the variance, should not change.

Now we can make a reasonable guess about intrinsic dimensionality in the Levenshtein space. Recall that considering the distance between two randomly chosen points from the native distribution of the space, the *intrinsic dimensionality* of the space is defined to be the mean squared divided by twice the variance (Definition 1.23). We denote intrinsic dimensionality by ρ. Assuming the mean of Levenshtein distance to be linear (as it is known to be for longest common subsequence), then the conjecture of Chvátal and Sankoff for longest common subsequence [45] implies that

intrinsic dimensionality

$$\rho = \omega(n^{4/3}). \tag{6.1}$$

The little-omega notation is standard, but infrequently used; recall that $f(n) = \omega(g(n))$, read "$f(n)$ is little-omega $g(n)$," if $f(n)$ is greater than any constant multiple of $g(n)$ for sufficiently large n, making ω a strict version of Ω. As a mnemonic: ω is to Ω as o is to O.

To test this conjecture, we ran an experiment on the Levenshtein distance between randomly chosen binary strings for each power-of-two string length up to 2^{18}. For lengths 1 through 16, the spaces are small enough that we could exhaustively test all pairs of strings and compute a population mean μ and standard deviation σ, allowing the calculation of the intrinsic dimensionality as $\rho = \mu^2/2\sigma^2$. For longer strings, we used random sampling of point pairs, starting with 10^7 pairs for length 32 and decreasing the sample size for longer strings as the computation load increased, to determine the sample mean Levenshtein distance $\bar{d}$, the sample standard deviation s, and the sample intrinsic dimensionality $\rho = \bar{d}^2/2s^2$. Note that because the distribution of distances becomes tightly concentrated around its mean as the length of the strings increases, large sample sizes are less necessary for long strings anyway. The results are given numerically in Table 6.1.

The sample mean and standard deviation are plotted in Figure 6.2 along with these functions, found by least-squares fitting a linear function of $\log n$ to the logarithm of the data:

$$\begin{aligned} \hat{d} &= 0.448945 n^{0.953767} \\ \hat{s} &= 0.517177 n^{0.370306} \\ \hat{\rho} &= \hat{d}^2/2\hat{s}^2 = \Theta(n^{1.166923}). \end{aligned} \tag{6.2}$$

These curves seem to match the data well. However, the curve for mean distance is $o(n)$ where theory predicted it should be $\Theta(n)$, and the curve for ρ is considerably shallower than the $\omega(n^{4/3})$ of (6.1) though still definitely steeper than $\Theta(n)$.

The short strings may be hiding the true asymptotic behaviour (much as short vectors gave misleading results in Chapter 2). Note that the data points for distance and intrinsic dimensionality in Figures 6.2 and 6.3 both seem to be concave upward, suggesting that their true asymptotes should be steeper than the fit lines. If we restrict the curve fitting to exclude the shortest strings, say by considering only the results for $n \geq 512$, we get a best-fit exponent of 0.993, supporting the hypothesised linear model.

If we assume that mean distance is linear and fit new curves to the distance and standard deviation using only the experimental data for $n \geq 512$, we get

length	pairs	μ	σ	ρ
1	2^2	0.500000	0.500000	0.500000
2	2^4	1.000000	0.707107	1.000000
4	2^8	1.898438	0.908638	2.182635
8	2^{16}	3.433899	1.136159	4.567375
16	2^{32}	6.212229	1.443025	9.266537

length	sample	$\bar{d}$	s	ρ
32	1×10^7	11.449	1.863	18.890
64	1×10^7	21.482	2.401	40.015
128	1×10^7	40.945	3.081	88.304
256	1×10^7	79.090	3.952	200.217
512	2×10^6	154.383	5.073	463.119
1024	5×10^5	303.672	6.518	1085.293
2048	2×10^5	600.602	8.392	2560.960
4096	5×10^4	1192.340	10.938	5941.517
8192	1×10^4	2373.177	14.236	13895.427
16384	2000	4731.293	18.750	31836.957
32768	500	9443.710	24.002	77405.880
65536	100	18859.490	32.309	170369.744
131072	50	37694.720	42.722	389245.933
262144	50	75349.540	54.272	963787.869

Table 6.1: Experimental results on random strings with Levenshtein distance.

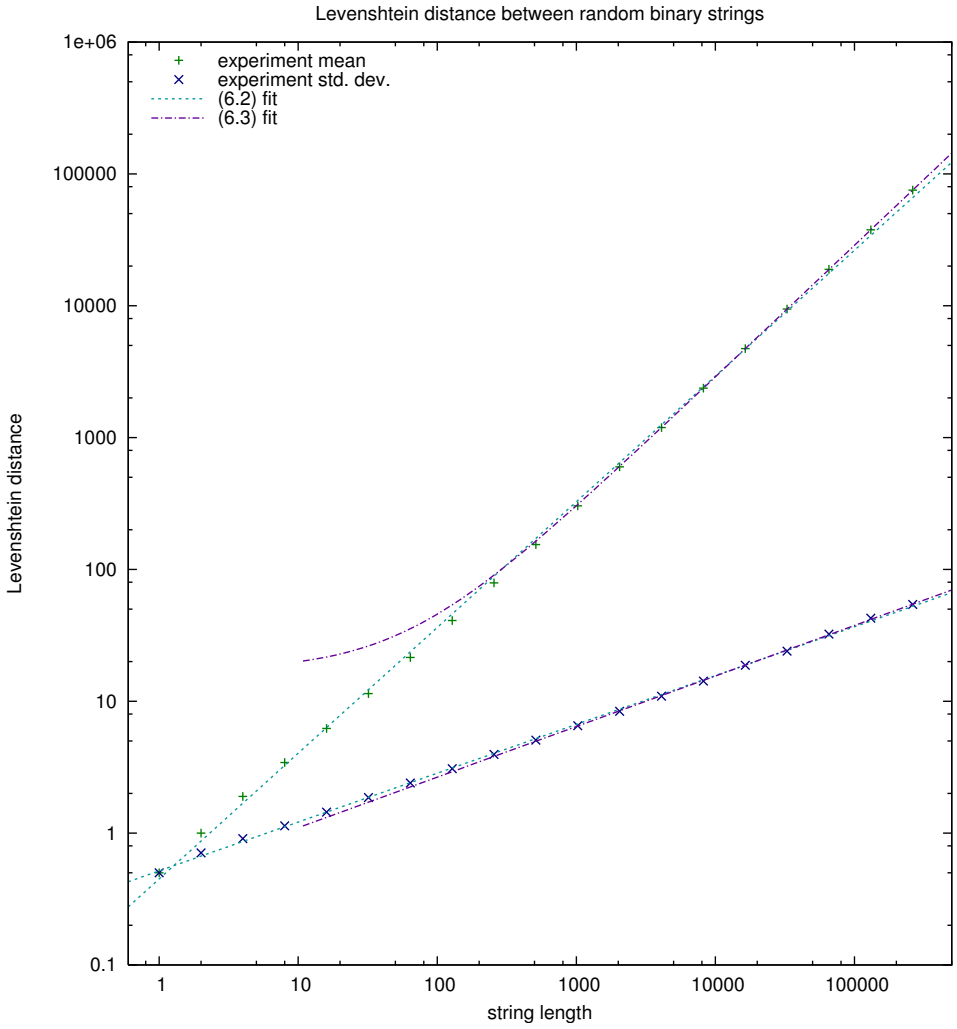


Figure 6.2: Levenshtein distance from the experiment.

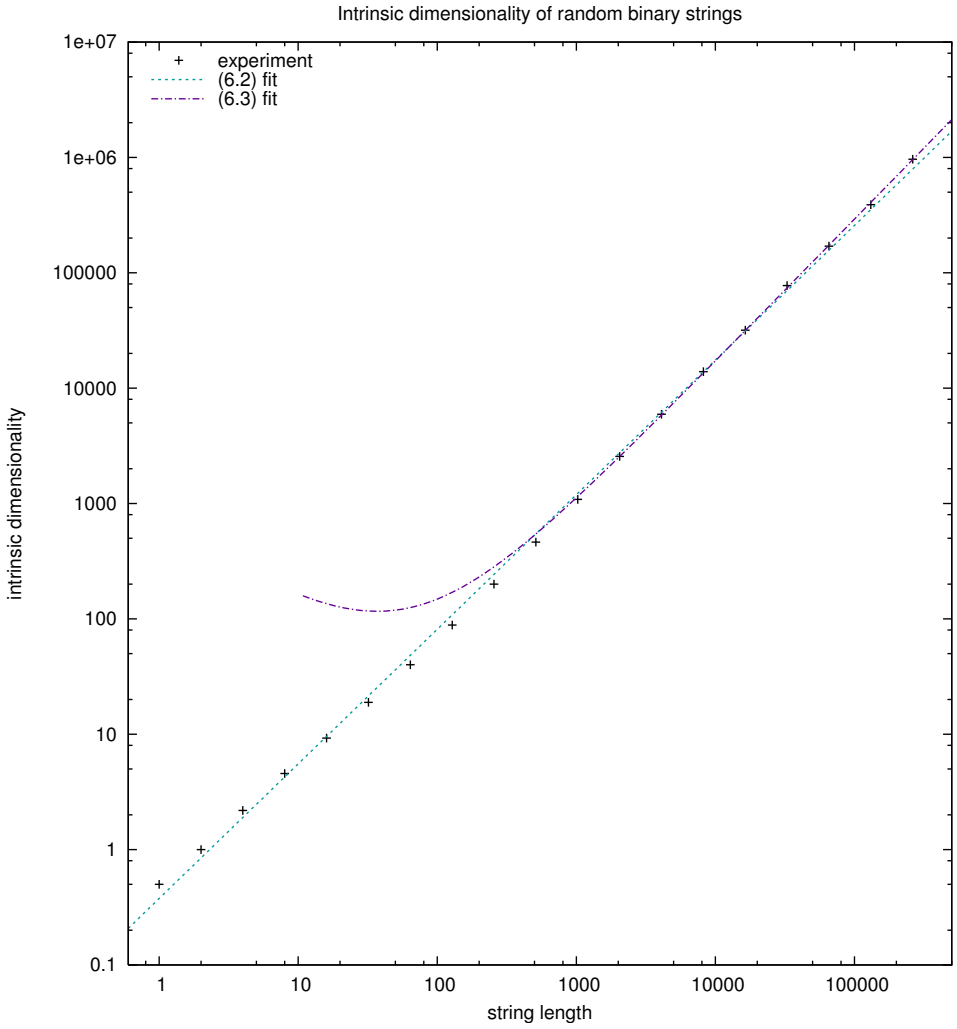


Figure 6.3: Intrinsic dimensionality from the experiment.

these functions, which are also plotted in Figures 6.2 and 6.3:

$$\begin{aligned} \hat{d} &= 0.287398n + 17.072985 \\ \hat{s} &= 0.454622n^{0.383543} \\ \hat{\rho} &= \hat{d}^2/2\hat{s}^2 = \Theta(n^{1.232914}). \end{aligned} \tag{6.3}$$

The most important result here seems to be that ρ is super-linear in string length for this space, but less than $\Theta(n^2)$. That is a distinct new behaviour compared to other spaces we have considered. Strings with Hamming distance and most of the distributions we studied in vector spaces turned out to have intrinsic dimensionality linear in the size of the objects. In the case of vectors with L_∞ distance and the normal distribution per component, we found the intrinsic dimensionality to be $\Theta(\log^2)$. Strings with prefix distance had intrinsic dimensionality $\Theta(n^2)$. Levenshtein distance, with its apparent exponent of approximately $5/4$, fits somewhere between the L_p metrics and prefix distance. The reason for the unusual behaviour may be that as strings get longer, a substring in one can match substrings at a greater distance in the other, so the individual bits in the strings count more towards dimensionality when they are in longer strings.

6.2 Number of neighbours

Levenshtein distance differs from previously-considered metrics in another significant way: different points in Levenshtein distance have meaningfully different numbers of neighbours. First, we must define what we mean by a neighbour. This definition may seem strange in the context of spaces like real vectors, but we use it for spaces like strings with Levenshtein distance, where distance is the number of edges in the shortest path through some graph. For such spaces the definition below corresponds to the graph-theoretic definition of neighbour.

neighbour

Definition 6.3

A point x in a metric space with metric d is a *neighbour* of a point y if $d(x, y) = 1$.

In discrete spaces, all distinct points are neighbours of each other, so with n points in the space, each one has $n - 1$ neighbours. In real vector spaces (Chapters 2, 3, and 8), each point has an infinite number of neighbours; we could perhaps define a measure and use it to compare sets of neighbours for different points, but for now we restrict ourselves to spaces where points have finite numbers of neighbours. In an unweighted tree metric space (Chapter 4), the neighbours of a point are simply its neighbours in the underlying tree; for

the prefix metric in particular, with alphabet size $|\Sigma|$ the number of neighbours is $|\Sigma|+1$ (add any of $|\Sigma|$ letters, or remove one) except for the empty string which only has $|\Sigma|$ neighbours (we cannot remove a letter from it). With the Hamming distance (Chapter 5), each of the letters in a string of length n can be changed to any of the $|\Sigma|-1$ other letters, so each string has $n(|\Sigma|-1)$ neighbours. These are all straightforward, and consistent throughout the space.

But for strings with Levenshtein distance, the number of neighbours varies depending on the content of the string, in particular the number of single-letter runs it contains.

Definition 6.4

A *single-letter run* is a string of the form α^k for some letter α and integer $k \geq 1$. We say that a string x consists of r single-letter runs if r is the minimal integer such that

single-letter run

$$x = \alpha_1^{k_1}\alpha_2^{k_2}\ldots\alpha_r^{k_r} \tag{6.4}$$

for some letters $\alpha_1, \alpha_2, \ldots, \alpha_r$ and integers $k_1, k_2, \ldots, k_r \geq 1$.

Example 6.5

The string 000110010000 consists of five single-letter runs. The unique string consisting of zero single-letter runs is the empty string λ.

Note that some authors define more general "runs" as repeating sequences of possibly more than a single letter. Here we are concerned exclusively with the single-letter variety. The number of single-letter runs in a string determines its number of neighbours under Levenshtein distance, as shown by the following theorem.

Theorem 6.1

If a string of length n on alphabet Σ consists of r single-letter runs, then it has $(2n+1)|\Sigma|+r+1$ neighbours under Levenshtein distance.

Proof Substitution, deletion, and insertion always produce distinct results from each other because of their effects on the length of the string, so we can enumerate separately the distinct results from each type of edit.

Substitution. The substitution can occur at any of the n positions in the string, and replace the letter there with any of the $|\Sigma|-1$ other letters in the alphabet, and all of these produce distinct results, so there are $n(|\Sigma|-1)$ neighbours by substitution.

Deletion. A letter can be deleted from any of the r runs. By comparing the result with the original string we can determine which run was shortened, but not which letter within the run was deleted because they are all the same, so there are r distinct neighbours by deletion.

Insertion. In general, any of the $|\Sigma|$ letters can be inserted in any of $n+1$ locations (before any existing letter or at the very end). That gives $(n+1)|\Sigma|$ possible insertions. However, some of these produce identical results. Inserting any letter adjacent to a copy of itself has the effect of extending whichever run contained that letter; and all the ways of extending a given run will produce the same result. To account for these identical neighbours, we will only count extending a run if we do it at the left-hand end. Subtract n for the insertions where the inserted letter is identical to the one on its right, and r for the insertions where the inserted letter is identical to the one on its left and not its right; that is, inserting a letter between two runs or at the very end of the string. That leaves all the insertions that do not extend runs (but instead create new runs). Then add back r for the one extension we permit per run. The result is a total of $(n+1)(|\Sigma|-1)+1$ distinct neighbours by insertion.

The overall total number of distinct neighbours is $(2n+1)|\Sigma|+r+1$. ■

6.3 Reverse similarity search

The VPREVERSE and GHREVERSE problems in this space combine Levenshtein distance among multiple strings with constraint satisfaction, both of which are often associated with $\mathcal{NP}$-complete problems. As we might expect, these problems turn out to be $\mathcal{NP}$-complete. Recall from Definition 1.27 that VPREVERSE is a set of triples (x, r, b) where x is a point in the space, r is is a real number greater than zero called the radius, and b is 0 or 1. The instance is satisfied if there exists a point z such that for every (x, r, b) in the instance, $(x, r, b) \in P$, $d(x, z) \leq r$ if and only if $b = 1$. In other words, VPREVERSE is a constraint satisfaction problem where the constraints each require z to be inside or outside a sphere. The GHREVERSE problem of Definition 1.29 is similar: an instance consists of a set of ordered pairs of points (x, y), and the solution z must satisfy $d(z, x) \leq d(z, y)$ for every (x, y) in the instance.

To prove $\mathcal{NP}$-completeness, we will reduce from VPREVERSE and GHREVERSE on n-bit binary strings with Hamming distance, to the same problems on $(2n^2+n)$-bit binary strings with Levenshtein distance. The encoding used is that 0 in the Hamming-distance string translates to 0^n10^n in the Levenshtein-distance string, and 1 to $0^{2n}1$. Here is a summary of the steps in the proof:

- Characterisation of the distance from an arbitrary string to a string of the form α^k (Lemma 6.2).
- Placing a lower limit on string length (Lemma 6.3).
- Placing an upper limit on the number of occurrences of a letter (Lemma 6.4).
- Constraining the solution to a string of the form $\{0^n, 0^n1\}^{2n}$ (Lemma 6.5).
- Equivalence between Levenshtein and Hamming distances (Lemma 6.6).
- Assembling the gadgets into a complete instance (Theorem 6.8).

Lemma 6.2 and Theorem 6.8 were presented, without the detailed proofs we give here, at SISAP'08 [192]. In the present work we also prove Corollary 6.7 along the way; it gives a loose bound on distance permutations in Levenshtein space.

Proving anything about Levenshtein distance is complicated by the fact that in general there is no simple expression for Levenshtein distance from an arbitrary string to a given constant string. It is clear that the Levenshtein distance from an arbitrary string x to the empty string λ is $|x|$, the length of x: we can transform x to λ by deleting every letter, and any other edit could be eliminated because we could delete the letter involved instead. The next most general case for the distance from x to y is where x might not be λ, but consists of only one letter repeated some number of times.

Lemma 6.2
If x is a string in which the letter α occurs m times, then the Levenshtein distance from x to a string of k repetitions of α is given by

$$d(x, \alpha^k) = \begin{cases} |x| - k & \text{if } m \geq k, \\ |x| - m & \text{if } m \leq k \text{ but } |x| \geq m, \\ k - m & \text{if } |x| \leq k; \end{cases} \tag{6.5}$$

$$= \max\{|x| - k, |x| - m, k - m\}\,. \tag{6.6}$$

Proof At least $|x| - k$ edits are always necessary to change x into α^k. If $|x| > k$ then we must remove all the extra letters to reduce the length to k, and if not, then $|x| - k$ is zero or less and the Levenshtein distance is always at least zero. Similarly, at least $|x| - m$ edits are always necessary, because that is the number of letters

that are not α, and each of those must be removed with an edit. And $k-m$ edits are always necessary, because α^k contains k copies of α, and we must use at least one edit to introduce each copy beyond the n that already exist in s. So, noting that this is a maximum and not a sum (because one edit might well serve more than one of those three purposes), we have $d(x,\alpha^k) \geq \max\{|x|-k, |x|-m, k-m\}$, a lower bound on the Levenshtein distance.

If $m \geq k$, then we have k copies of α in x and can transform x into α^k by removing everything except those, in exactly $|x|-k$ edits. Then $|x|-m \leq |x|-k$ and $k-m \leq 0$, so we have achieved the lower bound.

If $m \leq k$ but $|x| \geq k$, then we can remove letters other than α from x in $|x|-k$ edits to get a string of length k containing n copies of α, and we can convert the remaining non-α letters to α in $k-m$ further edits for a total of $|x|-m$ edits. In this case $|x|-k \leq |x|-m$ and $k-m \leq |x|-m$, so again we have achieved the lower bound.

Finally, if $|x| \leq k$, then we can convert the non-α letters to α in $|x|-m$ edits and then add $k-|x|$ additional copies of α for a total of $k-m$ edits. In this case, $|x|-k \leq 0$, and $|x|-m \leq k-m$, so we have achieved the lower bound.

In all cases $\max\{|x|-k, |x|-m, k-m\}$ edits are necessary and sufficient, so the result holds. ■

The distance proved by Lemma 6.2 does not seem to be an especially convenient function of x, but we nonetheless can form a comparison between two strings differing by one letter to make a useful function of x. The next result shows that we can create a lower limit on the length of x.

Lemma 6.3

For any string x, these statements are equivalent:

$$|x| > k, \tag{6.7}$$

$$d(\lambda, x) > k, \tag{6.8}$$

$$d(\alpha^{k+1}, x) \leq d(\alpha^k, x). \tag{6.9}$$

Proof The result for (6.8) is obvious: distance to the empty string is just the length of x. For (6.9), the proof follows the three cases of Lemma 6.2, making use of the fact that they overlap. Let m be the number of times α occurs in x. In the first case, suppose $m > k$, which implies $|x| > k$ because $|x| \geq m$. Then $m \geq k+1$, $d(\alpha^{k+1}, x) = |x| - (k+1)$, and $d(\alpha^k, x) = |x| - k$. We have (6.9).

If $m \leq k$, consider $|x|$. If $|x| > k$, then we have $|x| \geq k+1$. Both distances are equal to $|x| - m$, and so (6.9) holds.

Finally, if $|x| \leq k$, then $|x| \leq k+1$. The distances are given by $d(\alpha^{k+1}, x) = (k+1) - m$ and $d(\alpha^k, x) = k - m$. Then (6.9) does not hold. Therefore, (6.9) is true if and only if $|x| > k$. ∎

With that result, we are ready to begin the reduction. Starting with an arbitrary instance of VPREVERSE or GHREVERSE on n-bit binary strings with Hamming distance, we will create an equivalent instance on strings of some alphabet with at least two letters (without loss of generality, say 0 and 1) using Levenshtein distance. In a VPREVERSE instance, it is convenient to bound the length in both directions, so the first two spheres (centre, radius, and bit triples) will be

$$\left(\lambda, 2n^2 + n, 1\right), \tag{6.10}$$

$$\left(\lambda, 2n^2 + n - 1, 0\right). \tag{6.11}$$

In a GHREVERSE instance, the first pair will be

$$\left(0^{2n^2+n}, 0^{2n^2+n-1}\right). \tag{6.12}$$

By Lemma 6.3, inclusion of this gadget means that any solution x to the edit-distance instance must be a string of length at least $2n^2 + n$ for GHREVERSE or exactly that for VPREVERSE.

Next, we need to constrain the composition of the string in terms of how many times each letter of the alphabet can appear. For GHREVERSE we use the same kind of gadget with the order of points reversed, and by restricting all the letters in the alphabet we form an upper bound on the total string length to match the lower bound already set. For VPREVERSE, we already have a fixed total length from the previous gadget, and we depend on that in the present gadget.

Lemma 6.4

If x is a string in which the letter α occurs m times, with $|x| > k$, then these statements are equivalent:

$$m \leq k, \tag{6.13}$$

$$d(\alpha^{k+1}, x) > |x| - k - 1, \tag{6.14}$$

$$d(\alpha^k, x) \leq d(\alpha^{k+1}, x). \tag{6.15}$$

Proof First, consider (6.14). If $m \leq k$, then $d(\alpha^{k+1}, x) = |x| - m > |x| - k - 1$ by Lemma 6.2, and (6.14) holds. If $m > k$, then $d(\alpha^{k+1}, x) = |x| - k - 1$ and (6.14) does not hold. Therefore it is equivalent to $m \leq k$.

For (6.15) we again follow the three cases of Lemma 6.2. If $m > k$, then $d(\alpha^k, x) = |x| - k$ and $d(\alpha^{k+1}, x) = |x| - (k+1)$, so (6.15) does not hold. If $m \leq k$ but $|x| > k$, then both distances are equal to $|x| - m$, and (6.15) holds. If $|x| \leq k$, which implies $n \leq k$, then $d(\alpha^k, x) = k - m$ and $d(\alpha^{k+1}, x) = (k+1) - m$, so (6.15) holds. In summary, it holds if and only if $m \leq k$. ■

Lemma 6.4, in conjunction with the limits already placed on total length of the string, lets us place an upper limit on how many times each letter α can occur in the solution. For VPREVERSE, we add a triple of the form

$$\left(\alpha^{k+1}, 2n^2 + n - k - 1, 0\right) \tag{6.16}$$

and for GHREVERSE we add a pair of the form

$$\left(\alpha^k, \alpha^{k+1}\right). \tag{6.17}$$

In either case, the gadget forces there to be at most k occurrences of α in the solution. We use this gadget to limit the solutions to at most $2n^2$ occurrences of 0, n of 1, and no occurrences of any other letter if the alphabet contains any other letters. Since the solution is already also constrained to a total length of at least $2n^2 + n$, all these constraints must hold with equality.

Now the Levenshtein distance solution string must contain the same number of each letter as the string that would be produced by encoding an n-bit Hamming-distance string, but at this point there is no constraint on the order in which those letters can appear. We build up the constraint that x must actually be an encoded n-bit string in two steps. The first one is to require that x be of the form $\{0^n, 0^n1\}^{2n}$; that is, every 1 follows a block of n copies of 0.

Lemma 6.5
If x is a string consisting of $2n^2$ occurrences of 0 and n occurrences of 1 in some order, then these statements are equivalent:

$$x \text{ is of the form } \{0^n, 0^n1\}^{2n}, \tag{6.18}$$

$$d\left((0^n1)^{2n}, x\right) \leq n, \tag{6.19}$$

$$d\left((0^n1)^{2n}, x\right) \leq d\left(0^{2n^2}, x\right). \tag{6.20}$$

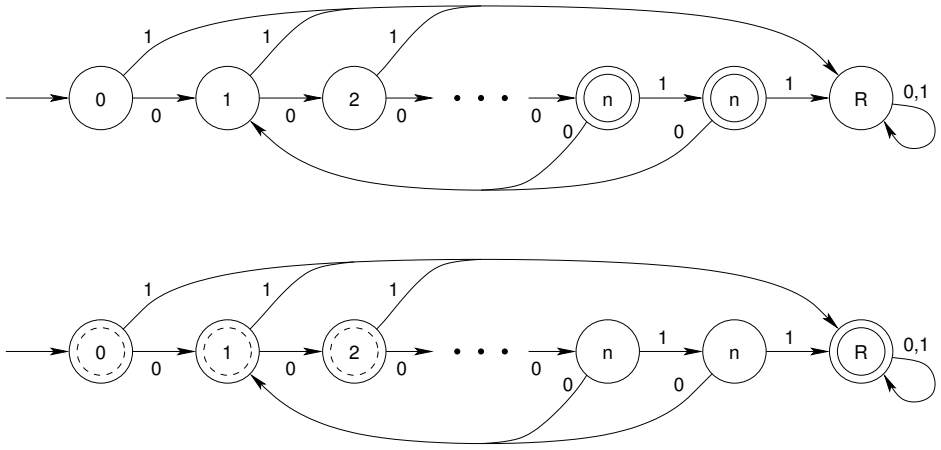


Figure 6.4: Automata accepting strings of the form $\{0^n, 0^n1\}^{2n}$ and strings not of that form.

Proof By Lemma 6.2, $d = (0^{2n^2}, x) = n$, so (6.19) is equivalent to (6.20). There are n letters other than 0, and n edits are necessary and sufficient to remove them and leave 0^{2n^2}. If x is of the specified form, then n edits are sufficient to transform it into $(0^n1)^2\, n$: it must consist of n copies of 0^n and n copies of 0^n1 for the total counts to be correct, and then we can transform the n copies of 0^n1 into n copies of 0^n with n edits, leaving $(0^n1)^2\, n$. Furthermore, n edits are necessary because that is the difference in length between x and $(0^n1)^2\, n$, and each edit adds at most one to the length. Therefore (6.19) holds if x is of the form $\{0^n, 0^n1\}^{2n}$.

There are two ways that x could fail to be of that form: it could contain a 1 at some position i such that the number of occurrences of 0 before position i is not a multiple of n, or it could contain a 1 that is not immediately preceded by a 0. These possibilities are illustrated by automata in Figure 6.4; the upper automaton accepts strings of the form $\{0^n, 0^n1\}^{2n}$, and the lower one is simply its inverse. The state labels (other than R) refer to the number of occurrences of 0 seen so far. The lower automaton accepts strings in which the number of occurrences of 0 is not divisible by n, but x cannot be such a string (and so the accept states are shown with dashed circles) because we know it contains exactly $2n^2$ occurrences of 0. The only other way for the lower automaton to accept is in the state R, which is entered (and never left) if a 1 is seen after a non-multiple of n occurrences of 0, if a 1 is seen at the start of the string, or if 11 occurs.

If x contains a 1 at some position i such that the number of occurrences of 0 before position i is not a multiple of n, then in transforming x to $(0^n1)^{2n}$ we must add or remove a 0 before position i. That changes the number of occurrences of

0 in the string, but that number is the same for the two strings, so we must make one more edit to change it back. Of those two edits, at most one can introduce a 1 (because one must introduce a 0); so we need to make at least $n-1$ more edits to introduce the remaining occurrences of 1. Then the total Levenshtein distance is at least $n+1$, and (6.19) does not hold.

If x contains a 1 not immediately preceded by a 0, then when we edit x to obtain $(0^n1)^{2n}$ we must eliminate that. We could do so by inserting a 0 (or some letter other than 1) before the 1; but then we would still need to make n edits to add the needed n copies of 1, so (6.19) would not hold. Similarly, if we deleted or changed to some other letter the 1 not immediately preceded by a 0, we would need $n+1$ other edits to achieve a total of $2n$ occurrences of 1. The only remaining possibility is to delete the letter immediately preceding the offending 1; but in that case, we again would need to make $n+1$ additional edits to bring the string up to length $2n^2+n$.

Therefore, when x consists of $2n^2$ occurrences of 0 and n occurrences of 1, (6.19) holds if and only if x is of the form $\{0^n, 0^n1\}^{2n}$. ■

So we can force the solution string into a form whose Levenshtein distance models Hamming distance by inserting the gadget suggested by Lemma 6.5; for VPREVERSE we add a triple

$$\left((0^n1)^{2n}, n, 1\right) \tag{6.21}$$

and for GHREVERSE we add a pair

$$\left((0^n1)^{2n}, 0^{2n^2}\right). \tag{6.22}$$

We were forced to constrain the number of occurrences of 1 to exactly n, and that complicates the encoding; but before dealing with that, we prove that Levenshtein distances within the current constrained set of strings really do reflect Hamming distances.

Lemma 6.6
Let x and y be bit strings of the same length, with the Hamming distance between x and y less than or equal to $2n$. Encode x and y by replacing each 0 with 0^n and each 1 with 0^n1 to produce new strings x' and y'. Then the Levenshtein distance between x' and y' is equal to the Hamming distance between x and y.

Proof First of all, the Levenshtein distance is at most the Hamming distance because we could transform x' to y' by inserting a 1 for each bit that is 0 in x and 1 in y and deleting a 1 for each bit that is 1 in x and 0 in y; the total number of edits to do that is the Hamming distance.

Now consider a minimal sequence of edits from x' to t'. In case our alphabet includes letters other than 0 and 1, none of the edits in the minimal sequence can introduce such a letter α because we would be forced to delete it or change it to a 0 or 1 in some future edit, and we could, in fewer edits, either not introduce it at all (and never have to delete it) or introduce instead the 0 or 1 that we would eventually change it to, saving one edit. Thus the sequence of strings formed by our minimal sequence of edits, consists entirely of strings on the alphabet $\{0,1\}$.

Now, x' and y' each contain the same number of length-n blocks of 0. In editing x' to y' in a minimal number of steps, at least one 0 in each block must be left untouched. If we changed one but changed it back, we could in fewer edits not change it at all; so all changed copies of 0 must be changed to 1 or deleted. If we changed or deleted them all, in n edits, we would be missing a block in y' and would have to spend an additional n edits to re-create it. That would use up our maximum of $2n$ edits, and we could instead follow the Hamming-distance procedure without touching any occurrences of 0. So if the minimal number of edits is less, choose one unchanged 0 from each block except the first, and split up x' and y' into fragments by cutting immediately after each of the chosen letters.

Each fragment of x' and y' corresponds to a bit from x or y respectively. It contains some number of copies of 0, and exactly one 1 if the bit is 1, or no 1 if the bit is 0. Each fragment from x' is transformed to the corresponding fragment in y', with (by the construction) no letters from other fragments involved.

If the bits are the same, the number of edits between the fragments must still be at least zero. If the bits differ, the number of edits must be at least one, to create or destroy the 1, whether by insertion, deletion, or changing a letter to or from 0. So, either way, the number of edits between the fragments must be at least the Hamming distance between those two bits. Then the number of edits between x' and y' must be at least the Hamming distance between x and y. We already have that it is at most the Hamming distance, so the two distances must be equal. ∎

Now we have a Levenshtein edit-distance model for Hamming distance on $2n$-bit strings which contain n 1 bits each. To model arbitrary strings, we use a trick similar to the dimension-doubling of Frances and Litman [76]. Each bit from the Hamming-distance instance will give rise to two blocks in the edit-distance instance, with one containing a 1 and the other not. That way, the count of each

letter is unchanged regardless of whether we encode a 0 or a 1.

To limit x to the desired form, we need to make sure that consecutive pairs are constrained to 01 or 10. For VPREVERSE, with $1 \leq i \leq n$, we insert triples

$$\left(0^{(2i-1)n}10^n10^{(2n-2i)n}, n, 1\right) . \tag{6.23}$$

The string at the centre of the sphere consists of the encoding of a string consisting entirely of 0 except for one pair 11. Knowing that 1 must occur exactly n times in the solution, there must be at least one 1 in the pair of positions under consideration for the Hamming distance to be at most n. Since there must be at least one 1 in every one of the n pairs and a total of n of them overall, there must be exactly one in every pair, and the string is of the desired form.

Similarly, for GHREVERSE, with $1 \leq i \leq n$, insert pairs

$$\left(0^{(2i-1)n}10^n10^{(2n-2i)n}, 0^{2n^2}\right) , \tag{6.24}$$

$$\left(0^{2n^2}, 0^{(2i-1)n}10^n10^{(2n-2i)n}\right) . \tag{6.25}$$

These pairs split the $2n$ bits encoded in the form described by Lemma 6.6 into pairs. For any pair other than the i-th one, the bit values contribute equally to the distance from both strings and so can be ignored. Thus a string x is allowed by these pairs as a function of its bit values in that one pair of bit positions. Since the same two strings are used in both directions, it becomes an equality constraint: the bit values must be equidistant from 00 and 11. As a result, they must be 01 or 10. Then x must be the encoding of an m-bit Hamming distance string, and we can prove the $\mathcal{NP}$-completeness result.

site

distance permutation

First, however, we digress briefly to consider distance permutations. Recall that given k fixed strings called the *sites*, $x_1, x_2, \ldots, x_k$, if for another string y we find the closest site to y, the second-closest site to y, and so on, we can form a permutation that sorts the site indices into order of increasing distance from y. If two sites are equidistant we break the tie by choosing the lower-index site first. The resulting permutation is called the *distance permutation* of y (Definition 1.25). Depending on the space and the choice of sites, a given permutation may not be the distance permutation of any point. In general, we are interested in the maximum number of distance permutations that occur in the space, as a function of the number of sites and whatever concept of size (such as string length) applies to the particular space.

For the space of binary strings with Levenshtein distance, this appears to be a very difficult problem because of the complex relationships that can occur among points. It is not even clear how best to pose the question. There is only one space of strings with Levenshtein distance, containing arbitrarily long strings, and it seems reasonable that we could always get all $k!$ permutations from k sites if we

had no restriction on string length; so to get an interesting answer, we should have some restriction on string length. The following result allows us to apply the Hamming distance lower bounds of Section 5.2 to Levenshtein distance, with the size of the problem taken as the length of the longest site.

Corollary 6.7
If $N_{n,\mathrm{H}}(k)$ is the maximum number of distinct distance permutations of k sites among n-bit binary strings with Hamming distance and $N_{n,\mathrm{L}}(k)$ is the maximum number of distinct distance permutations of k sites among binary strings with Levenshtein distance where n is the length of the longest site, then we have

$$N_{2n(n+1),\mathrm{L}}(k) \geq N_{2n,\mathrm{H}}(k)\,. \tag{6.26}$$

Proof If we encode strings of length $2n$ by replacing each 0 with 0^n and each 1 with 0^n1, we get strings of length at most $2n(n+1)$ whose pairwise Levenshtein distances match the Hamming distances of the original strings. The maximum Hamming distance between strings is just their length, so Lemma 6.6 applies. Then we can take a set of k sites that achieves $N_{2n,\mathrm{H}}(k)$ and encode them to get a set of sites that achieves at least as many distance permutations with Levenshtein distance. ∎

Now we proceed to the main result of this section, which follows naturally from the gadgets already introduced.

Theorem 6.8
In the space of strings on an alphabet Σ of at least two letters with Levenshtein distance, VPREVERSE and GHREVERSE are $\mathcal{NP}$-complete.

Proof Without loss of generality, say that two of the letters in Σ are 0 and 1. Starting with an arbitrary instance of VPREVERSE or GHREVERSE on n-bit strings with the alphabet $\{0,1\}$, which are $\mathcal{NP}$-complete problems by Theorem 5.4 and Corollary 5.5 respectively, we will create a polynomial-sized equivalent instance of the same problem on strings with alphabet Σ and edit distance.

For clarity, we repeat the pairs and triples making up the gadgets as they are used. First, introduce the gadget described by (6.10) and (6.11) for VPREVERSE,

or (6.12) for GHREVERSE:

$$\left(\lambda, 2n^2+n, 1\right) \tag{6.10}$$
$$\left(\lambda, 2n^2+n-1, 0\right) \tag{6.11}$$
$$\left(0^{2n^2+n}, 0^{2n^2+n-1}\right). \tag{6.12}$$

By Lemma 6.3, these bound the length of the solution string.

Then, use the gadget described by (6.16) for VPREVERSE or (6.17) for GHREVERSE and proved by Lemma 6.4, to bound the number of occurrences of each letter in the solution string:

$$\left(\alpha^{k+1}, 2n^2+n-k-1, 0\right) \tag{6.16}$$
$$\left(\alpha^{k}, \alpha^{k+1}\right). \tag{6.17}$$

Add the gadget described by (6.21) for VPREVERSE or (6.22) for GHREVERSE,

$$\left((0^n1)^{2n}, n, 1\right) \tag{6.21}$$
$$\left((0^n1)^{2n}, 0^{2n^2}\right), \tag{6.22}$$

and the one described by (6.23) for VPREVERSE or (6.24) and (6.25) GHREVERSE:

$$\left(0^{(2i-1)n}10^n10^{(2n-2i)n}, n, 1\right) \tag{6.23}$$
$$\left(0^{(2i-1)n}10^n10^{(2n-2i)n}, 0^{2n^2}\right) \tag{6.24}$$
$$\left(0^{2n^2}, 0^{(2i-1)n}10^n10^{(2n-2i)n}\right). \tag{6.25}$$

At this point, by Lemmata 6.5 and 6.6 and the discussion above, we have a situation where the solution string can be any n-bit string encoded by replacing 0 with 0^n10^n and 1 with $0^{2n}1$, and the Levenshtein distance between any two such encoded strings is twice the Hamming distance of the corresponding unencoded strings.

Now all the constraints from the original VPREVERSE or GHREVERSE instance can be translated to constraints in the new instance by encoding the centres or vantage points, and doubling the radii in the case of VPREVERSE triples. A solution to the new instance must be an encoded solution to the original instance, and a solution to the original instance can be encoded to find a solution to the new instance, so the two instances are equivalent. The string that satisfies an instance is trivially a polynomial-time certificate for the decision problem, placing the decision problem in $\mathcal{NP}$. Therefore VPREVERSE and GHREVERSE on strings with Levenshtein edit distance are $\mathcal{NP}$-complete problems. ∎

Chapter 7

Superghost distance

Ghosts

In the game of *Ghosts*, players take turns adding letters to the end of a string subject to the constraint that it must always be a prefix of some dictionary word. The player who is forced to spell a complete word loses the round and is said to have become one third of a ghost. Players who have lost three rounds are complete ghosts and drop out of the game, until only the winner remains. The dictionary for the game normally excludes words of three letters or less, because they tend to make the game boring. Ghosts has been part of children's folklore for a long time; Morris and Morris describe it in 1959 as already more than a century old [155].

Example 7.1

Alice:	I	
Bob:	IN	
Alice:	INS	
Bob:	INSO	(hoping to make Alice spell INSOFAR)
Alice:	INSOL	(hoping for INSOLENT)
Bob:	INSOLU	
Alice:	INSOLUB	(she doesn't have much choice)
Bob:	INSOLUBL	
Alice:	INSOLUBLE	👻

Ghosts has some interest in the context of the combinatorial game theory introduced by Conway [49] and developed by Berlekamp, Conway, and Guy [28] because its game tree is exactly the trie of dictionary words, after removing words that have shorter prefixes in the dictionary and so could never be reached. If we use that tree to define a tree metric, we get the familiar prefix distance studied in Chapter 4; it could reasonably be called the "Ghost distance."

As an amusement, Ghosts only has limited appeal. The last few moves of each round tend to be forced, as in Example 7.1, and not much fun. If we allow letters to be added at the start as well, the result is a more difficult and interesting game called *Superghosts* [155, 204]. The more difficult game also naturally yields a distance function.

Superghosts

Definition 7.2

Superghost distance

The *Superghost distance* between two strings x and y is the minimal number of edits to transform x to y where an edit consists of adding or removing a letter at either end of the string.

Example 7.3

The Superghost distance from TYPOGRAPHER to BIOGRAPHY is 8. Remove TYP and ER (five edits), leaving OGRAPH, and then add BI and Y (three more edits).

It is plausible that Superghost distance could arise in some application beyond the parlour game. For instance, in bioinformatics we might describe distance among linear RNA molecules in equilibrium with polymerases such that they can add and remove bases at either end but only at the ends. However, the main reason for studying it is theoretical: the Superghost distance is a compromise between the prefix distance and the Levenshtein and Hamming distances. The prefix distance allows edits at one end only, and with it, VPREVERSE and GHREVERSE are easy. The Levenshtein and Hamming distances allow edits anywhere, and with them, VPREVERSE and GHREVERSE are $\mathcal{NP}$-hard. So by choosing a metric which allows edits at more places than just one end, but less than everywhere in the string, we can explore the boundary between easy and $\mathcal{NP}$-hard problems.

Thurber writes that "Starting words in the middle and spelling them in both directions lifts the pallid pastime of Ghosts out of the realm of children's parties and ladies' sewing circles and makes it a game to test the mettle of the mature adult mind." [204] This modification similarly tests the mettle of reverse similarity search algorithms. It turns out that our reverse similarity problems are $\mathcal{NP}$-complete in Superghost space (Theorem 7.9). That is the main result of this chapter. The $\mathcal{NP}$-completeness result was presented, with a sketch of the proof, at SISAP'08 [192].

Although Superghost distance is primarily introduced for the purpose of studying VPREVERSE and GHREVERSE, we also discuss our other metric space problems in relation to this metric. In particular, we give an asymptotic lower bound on intrinsic dimensionality for fixed-length uniformly distributed binary strings; fully characterise the number of neighbours of an arbitrary string; and

give a construction (not necessarily tight in terms of length) for a set of sites that achieves all possible distance permutations.

Just as the prefix distance between two strings could be characterised by the distance from each string to their longest common prefix, the Superghost distance is the total distance from two strings to their longest common substring. The following lemma formalises that.

Lemma 7.1
Where x and y are strings and $\text{lcs}(x, y)$ is the longest common substring of x and y, then the Superghost distance between x and y denoted by $d(x, y)$ is $|x| + |y| - 2|\text{lcs}(x, y)|$.

Proof Find a minimal sequence of edits, each consisting of adding or deleting a letter at either end of the string, to transform x into y. Arrange them to place all deletions before all additions. This must be possible because an addition and deletion must either be at different ends of the string, in which case they may be freely rearranged, or else at the same end with the deletion before the addition. If we ever did a deletion after an addition on the same end of the string, then we could omit that deletion and the corresponding addition to get an equivalent but shorter sequence of edits, contradicting minimality.

Consider the string z that results from applying all the deletions and none of the additions. It is obviously a common substring of x and y. The length of the sequence of edits is the number of deletions to get from x to z, which is $|x| - |z|$, plus the number of additions to get from z to y, which is $|y| - |z|$, for a total of $|x| + |y| - 2|z|$. But we can construct a sequence of edits of length $|x| + |y| - 2|z|$ any common substring z, so the minimal-length sequence of edits must be for some z that minimises that expression; namely, the longest possible z. Then $d(x, y) = |x| + |y| - 2|\text{lcs}(x, y)|$. ∎

Note 7.4
We always use the word *substring* to refer to a contiguous substring; a possibly-discontiguous selection of the letters in a string is a *subsequence*.

substring

subsequence

7.1 Intrinsic dimensionality and neighbour count

Recall that we measure intrinsic dimensionality by measuring the tendency noted in high-dimensional spaces for points to usually be equidistant from each other. The intrinsic dimensionality ρ is defined as the mean squared divided by twice

the variance of the distance between two random points drawn from the space's native distribution (Definition 1.23). For Superghost distance on n-bit strings chosen uniformly at random, we start by bounding the length of the longest common substring.

Lemma 7.2
The longest common substring between two uniformly chosen random binary strings of length n has length $\Theta(\log n)$ with probability $1 - o(n^{-c})$ for any constant $c > 0$.

Proof Let x and y be the two random strings of length n. With high probability, they contain a common substring of length $\frac{1}{2}\lg n$. Divide the strings into nonoverlapping substrings of that length. Each of x and y contains $2n/\lg n$ such substrings, each of which is chosen independently and uniformly at random. There are $\sqrt{n}$ strings of length $\frac{1}{2}\lg n$, so each nonoverlapping substring in x or y has $1/\sqrt{n}$ probability of being that string. For any given string of length $\frac{1}{2}\lg n$, the chance of it failing to occur in x is at most $(1 - 1/\sqrt{n})^{2n/\lg n}$. By the well-known formula $(1 + t/n)^n \leq e^t$, the probability of any given string of length $\frac{1}{2}\lg n$ failing to occur in one of x or y is less than or equal to $e^{-2\sqrt{n}/\lg n}$. Then with probability $1 - o(n^{-c})$ it occurs in both of them and so they have a common substring of length $\Omega(\log n)$.

In the other direction, consider substrings of length $(3 + c)\lg n$. There are n^{3+c} strings of that length, but each of x and y can contain at most n of them, with the conservative assumption that all the substrings in x or y are distinct from each other. So there are n^2 ways to choose a pair of substrings of length $(3+c)\lg n$ by choosing one from x and one from y. Each of those pairs, considered independently, has $1/n^{3+c}$ probability of matching. With n^2 pairs the chance of any of them matching is at most $1/n^{1+c}$, so the probability of x and y having a common substring of length $3\lg n$ is $o(n^{-c})$, and with probability $1 - o(n^{-c})$, the longest common substring has length $O(\log n)$. ∎

Note that although the longest common substring between two random strings is potentially a very difficult problem because of the dependencies among overlapping substrings, we can at least get an asymptotic lower bound on intrinsic dimensionality with the very loose result above. Difficulties like those seen in Section 6.1 would only arise if we tried to push it further, to the variance and the exact values of the constants.

Theorem 7.3
The intrinsic dimensionality of uniformly chosen random binary strings of length n with Superghost distance is $\Omega(n^2/\log^2 n)$.

Proof The longest common substring of two uniformly chosen random strings of length n is of $O(\log n)$ length with probability $1 - o(n^{-c})$ for any constant $c > 1$ by Lemma 7.2, so because its length must always be between 0 and n its expected value and variance must be $\Theta(\log n)$ and $O(\log^2 n)$ respectively. Then the Superghost distance must have mean $\Theta(n)$ and variance $O(\log^2 n)$, and from the intrinsic dimensionality definition (mean squared divided by twice variance) the intrinsic dimensionality is $\Omega(n^2/\log^2 n)$. ∎

As with the spaces discussed in Section 6.2, we can ask how many neighbours strings have under Superghost distance. Recall that a *neighbour* of a point x is a point y such that the distance between x and y is 1. At first glance it may appear that the number of neighbours in Superghost space is the same for all strings of sufficient length: we can always delete either the first or last letter, and we can always add any letter from the alphabet at the start or end. Some exceptions must be made if the string is empty or a single letter, at which point deletion may be impossible or operations on the two ends may be the same, but in fact, an exception can also apply to an arbitrarily long string if it happens to consist of a single letter repeated. In such a case, deleting at either end will produce an identical result. So the complete answer to the number of neighbours is that a string x on alphabet Σ with Superghost distance has $|\Sigma|$ neighbours if $x = \lambda$ (add any alphabet letter, the two ends are indistinguishable), $2|\Sigma|$ neighbours if x consists of a single letter repeated one or more times (add an alphabet letter or delete, at either end, but the ends are indistinguishable for deletions and the one letter that makes up the string), and $2|\Sigma| + 2$ in any other case (all moves produce distinct results).

neighbour

7.2 Distance permutations

As with Levenshtein distance, the question of distance permutations under the Superghost metric is complicated by the dependence between overlapping substrings. Recall that with k fixed strings called the *sites*, $x_1, x_2, \ldots, x_k$, if for another string y we find the closest site to y, the second-closest site to y, and so on, we can form a permutation that sorts the site indices into order of increasing distance

site

distance permutation

from y, breaking ties by placing the lower-index site first. This permutation is called the *distance permutation* of y (Definition 1.25). The main question we ask is how many distinct distance permutations can occur among all the points in a space, if k sites are chosen to maximise the number of distance permutations. The maximum number of distance permutations determines the storage space required to represent a distance permutation, if we build an index data structure around distance permutations.

For the Superghost metric, it seems natural that if we allow the k sites to be strings of unlimited length, we can easily achieve all $k!$ permutations of the sites as distance permutations. As a first step toward understanding the limits of distance permutations in this space, we give the following result that achieves all $k!$ permutations with limited-length strings. It introduces the encoding technique also used in the next section.

Theorem 7.4
In the space of strings on an alphabet of at least two letters with Levenshtein distance, there exist k sites of length $O(k \log k)$ such that for every permutation of the k sites, there is a string of length $O(k^2 \log k)$ with that as its distance permutation.

enc(i)

Proof Say without loss of generality that two of the letters in the alphabet are 0 and 1. Let enc(i) be the encoding of an integer $0 < i \leq k$, which is the value of $i-1$ written as a ($\lg k$)-bit number with the digit 0 replaced by 001 and 1 replaced by 011. This encoding has the property of not containing more than two consecutive identical bits. The k sites are $x_i = (\text{enc}(i))^{k+2}$, that is, each one consists of $k+2$ repetitions of its own encoded index.

Let $\pi : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$ be an arbitrary permutation of the sites and let

$$y = 00(\text{enc}(1))^{k-\pi(1)+3}1100(\text{enc}(2))^{k-\pi(2)+3}11\ldots00(\text{enc}(k))^{k-\pi(k)+3}11 \qquad (7.1)$$

The string y consists of k blocks corresponding to the k sites. Now, since the sites are all the same length, we can compare their distances to y just by comparing the lengths of their longest common substrings with y. The site x_i has a common substring of length at least $6\lceil \lg k \rceil$ with the i-th block of y, because that block contains at least two copies of enc(i). Each block starts and ends with 000 and 111, neither of which occurs in a site, so the longest common substring between y and a site cannot cross a boundary between blocks. Furthermore, the longest

common substring with any block other than the i-th block can be at most the length of one encoded index, which is less than the known common substring with the i-th block. Therefore the longest common substring between y and x_i is of length $3\lceil \lg k \rceil (k - \pi(i) + 3)$, which is monotonically decreasing with $\pi(i)$. Therefore the distance from y to x_i is monotonically increasing with $\pi(i)$, and so π is the distance permutation of y. ■

7.3 Reverse similarity search

The main purpose for which we introduce the Superghost distance is to test the boundary of $\mathcal{NP}$-completeness of the VPREVERSE and GHREVERSE problems. Recall that VPREVERSE is constraint satisfaction based on spheres: an instance consists of a set of balls and complements of balls in some metric space, and the instance is satisfiable if there exists a point in the intersection of all those sets (Definition 1.27). Similarly, a GHREVERSE instance is a set of points (x, y), and a point z satisfies the instance if $d(z, x) \leq d(z, y)$ for every (x, y) in the set (Definition 1.29). It turns out that even with such a simple metric as the Superghost distance, these problems remain $\mathcal{NP}$-complete.

The proof for this metric follows substantially the same outline as the proof for Levenshtein distance given in Section 6.3:

- Characterisation of the distance between two strings (Lemma 7.1).
- Forcing a substring to appear, forbidding it from appearing, or constraining the length of the solution, all in the VPREVERSE case (trivial).
- Forcing a substring to appear in GHREVERSE (Lemma 7.5).
- Forbidding a substring from appearing in GHREVERSE (Lemma 7.6).
- Setting up the encoded form of the problem (Lemma 7.7).
- Requiring each clause to be satisfied (Lemma 7.8).
- $\mathcal{NP}$-completeness of the problems (Theorem 7.9).

In this proof we use an encoding function enc(i), which is the value of the integer i written as a k-bit binary number (k to be defined later) with the digit 0 replaced by 001 and 1 replaced by 011. The purpose of that encoding is to express integers in a fixed-length form that does not contain any letter repeated more than twice consecutively. enc(i)

The proof of Theorem 7.9 is by reduction from 3SAT, with variable assignments encoded into strings by adding four dummy variables and then concatenating a value for each variable v_i in the form 0001 enc i0b1, where b is 0 if the variable is false and 1 if the variable is true. The dummy variables are all given the value of false, and the length k is chosen to provide enough bits to encode the $n+4$ distinct indices. A series of gadgets will be introduced to force the solution to be of this form and restrict it to encode satisfying solutions for an original 3SAT instance.

Recall that in Lemma 7.1 we showed that the Superghost distance between two strings was the sum of their lengths minus twice the length of their longest common substring. Then we can easily fix the length of the solution in a VPREVERSE problem by fixing its distance from the empty string; and with a known fixed length of the solution string, it becomes easy to require that a substring appear, or forbid it from appearing, by bounding the distance from solution to substring to include, or not, the case of the entire substring appearing in the solution. The detailed proof for the VPREVERSE gadgets is omitted; the equivalent gadgets for the GHREVERSE case are given in more detail because they are more complicated and depend on having a known substring included in the solution.

Lemma 7.5
Suppose x and y are strings, with $y = \alpha_1\alpha_2\ldots\alpha_n$; that is, $y_1, y_2, \ldots, y_n$ are the letters of y. Then $d(x,y) \leq d(x, y_1y_2\ldots y_{n-1}) \wedge d(x,y) \leq d(x, y_2y_3\ldots y_n)$ if and only if y is a substring of x.

Proof If y is a substring of x, then the longest common substring of x and y' where y' is any substring of y, must be y'. Then

$$d(x,y) = |x| - |y| < |x| - |y| + 1 = d(x, y_1y_2\ldots y_{n-1}) = d(x, y_2y_3\ldots y_n).$$

Suppose y is not a substring of x. Then let y' be the longest common substring of x and y; $|y'| < |y|$. It must be the case that y' is also the longest common substring of x and at least one of $y_1y_2\ldots y_{n-1}$ and $y_2y_3\ldots y_n$; without loss of generality, say $y_1y_2\ldots y_{n-1}$. Then

$$d(x,y) = |x| + |y| - 2|y'| > |x| + |y| - 1 - 2|y'| = d(x, y_1y_2\ldots y_{n-1}).$$ ■

Lemma 7.5 allows us to force a given substring to appear in the solution. Once we have forced a sufficiently long substring, it then becomes possible to

forbid substrings as well, using the next result. It is for this purpose, to provide a sufficiently long known substring, that we add dummy variables to the problem.

Lemma 7.6
Suppose x, y, and z are strings with $|y| \leq |z| + 1$ and z a substring of x with $z = z_1 z_2 \ldots z_n$; that is, $z_1, z_2, \ldots, z_n$ are the letters of z. Then $d(x, z_1 z_2 \ldots z_{|y|-1}) \leq d(x, y)$ if and only if y is not a substring of x.

Proof Since z is a substring of x, $d(x, z_1 z_2 \ldots z_{|y|-1}) = |x| - |y| + 1$. If y is a substring of x, we have

$$d(x, z_1 z_2 \ldots z_{|y|-1}) = |x| - |y| + 1 > |x| - |y| = d(x, y).$$

If y is not a substring of x, then $|\operatorname{lcs}(x, y)| < |y|$ and we have

$$d(x, z_1 z_2 \ldots z_{|y|-1}) = |x| - |y| + 1 \leq |x| + |y| - 2|\operatorname{lcs}(x, y)| = d(x, y). \qquad \blacksquare$$

With the ability to force and forbid substrings, we can then constrain the solution into a form that encodes a 3SAT truth assignment. In the GHREVERSE case, it is at this point that the length of the solution becomes fixed implicitly by the strings required to appear, their fixed relationship to each other, and the exclusion of other strings that could make the encoding longer.

Lemma 7.7
We can construct a polynomial-sized Superghost distance VPREVERSE or GHREVERSE instance whose solutions are exactly the encodings of n-variable truth assignments.

Proof Choose the smallest k such that $2^k \geq n+4$. Then the encoding of a variable v_i with $0 \leq i < n$ is given by $\mathtt{0001}\,\mathrm{enc}(i)\mathtt{0}b\mathtt{1}$ where $\mathrm{enc}(i)$ is the index i written in binary and then transformed by $\{\mathtt{0} \to \mathtt{001}, \mathtt{11} \to \mathtt{011}\}$, and b is $\mathtt{0}$ if v_i is false, $\mathtt{1}$ if w_i is true. The encoding of the complete truth assignment consists of the encodings of the assignments for all the variables in order of index, followed by the encodings for four dummy variables $v_n, v_{n+1}, v_{n+2}, v_{n+3}$, all of which are given the value false. Thus the encoding of the complete truth assignment has length $(3k+7) \cdot (n+4)$.

Subsequent gadgets depend on known properties of the solution string. In a VPREVERSE instance, we force the string to known length by adding two triples $(\lambda, (3k+7)\cdot(n+4)-1, 0)$ and $(\lambda, (3k+7)\cdot(n+4), 1)$. In a GHREVERSE instance, we force the inclusion of the known substring consisting of the encoded truth assignments for the dummy variables using the gadget of Lemma 7.5; that substring is of length $12k+28$.

Consider all strings of length $6k+14$, which is the length of two encoded variable assignments. The number of such strings is polynomial in n because k is logarithmic in n. For any string of that length we can easily compute whether it could ever appear in a valid encoded set of assignments: it must not contain any substrings forbidden by the encoding rules (such as 0000, 111, or any letter other than 0 and 1), any encoded variable indices it contains must be consistent with the ordering of the variables, and if it contains the encoded value of a dummy variable, the encoded value must be false.

Examine all the strings of length $6k+14$ and forbid the ones that cannot occur, Forbidding a substring x in a VPREVERSE instance consists of adding a triple

$$(x, (3k+7)\cdot(n+4)-|x|, 0)\ ;$$

in the GHREVERSE case, we use the gadget of Lemma 7.6, using the known forced substring of length $12k+28$ introduced previously.

We require every variable to appear in the solution by forcing a substring of the form 0001 enc(i)0 (that is, the encoded variable index without the variable's value) to appear for every variable. In combination with the forbidden substrings, the net effect is that the solution string must be the encoding of some truth assignment for the n variables, but could be the encoding of any truth assignment. ■

With the solution encoding any truth assignment, it remains to constrain it to truth assignments that satisfy the original 3SAT instance. Each 3SAT clause is translated to a constraint in the VPREVERSE or GHREVERSE instance as described in the following lemma.

Lemma 7.8
Given that the solution of a Superghost distance VPREVERSE or GHREVERSE instance is constrained to the encoding of a truth assignment, we can add a gadget to require that a 3-clause must be satisfied by the solution.

Proof Let v_a, v_b, and v_c be the variables that appear in the clause, and let b_a, b_b, and b_c be bits equal to 0 if the corresponding variables are negated in the clause,

1 if they are not. Let

$$x = 0001\,\mathrm{enc}(i)0b_a1100001\,\mathrm{enc}(j)0b_b1100001\,\mathrm{enc}(k)0b_c1\,.$$

Note that x consists of an encoding of each of the three variable assignments that could satisfy the clause, separated by additional copies of 0 and 1 to form forbidden substrings (specifically, 0000, which cannot occur at all, and 11, which cannot occur immediately after an encoded variable). The longest common substring between x and any solution string cannot contain more than one of the encoded variables because of the forbidden substrings in between, and cannot be of length greater than $3k+7$; it will be of that length if the clause is satisfied, but can be of length at most $3k+5$ if the clause is not satisfied.

So the gadget must require the longest common substring to have length $3k+7$. For the VPREVERSE problem, that is accomplished with the triple

$$(x, (3k+7)\cdot(n+5)+1, 1)\,,$$

and for the GHREVERSE problem, the gadget consists of the pair

$$\left(x, 1^{3k+10}\right)\,.$$ ■

Now we can prove Theorem 7.9: that VPREVERSE and GHREVERSE on strings with Superghost distance are each $\mathcal{NP}$-complete problems.

Theorem 7.9
In the space of strings on an alphabet of at least two letters with Superghost distance, VPREVERSE and GHREVERSE are $\mathcal{NP}$-complete.

Proof The proof is by reduction from 3SAT. Starting from any instance of 3SAT with n variables, let $k = \lceil \log(n+4) \rceil$. In the VPREVERSE case, force the solution to be of length $(3k+7)\cdot(n+4)$ by bounding on both sides its distance from the empty string, and then force it to contain or not contain any arbitrary string by bounding the distance from that string, as discussed above. In the GHREVERSE case, we use Lemmata 7.5 and 7.6.

Use Lemma 7.7 to constrain the solution to an encoded truth assignment for the 3SAT instance, and Lemma 7.8 for each clause to require satisfaction of the clause. Then the VPREVERSE or GHREVERSE instance will be satisfied if and only if there is a satisfying variable assignment for the 3SAT instance, and the solution string that satisfies all the constraints is a polynomial-time certificate, so the problems are $\mathcal{NP}$-complete. ■

Chapter 8

Real vectors: reverse similarity search

We have seen that the difficulty of the VPREVERSE and GHREVERSE problems depends on the space. Intuitively, some spaces with complicated metrics seem intrinsically difficult and lead to apparently difficult reverse similarity search problems, whereas others with simpler metrics allow for polynomial-time solutions. We even designed the Superghost distance to narrow down the location of the boundary between $\mathcal{P}$ and $\mathcal{NPC}$ among edit distances: assuming such a boundary exists at all, it comes between the Superghost and prefix distances.

In the present chapter, we consider the same question for real vectors with L_p metrics. As p varies, what happens to the difficulty of reverse similarity search? The overall result we prove is that VPREVERSE and GHREVERSE are $\mathcal{NP}$-complete for real vectors to some specified precision with any L_p metric except that GHREVERSE is polynomial-time for Euclidean (L_2) vectors.

Note 8.1

For all the results of this chapter, it is assumed that vectors must be given explicitly, component by component, in the input. Real numbers in the input must be given to some specified precision, unless we state otherwise; since all the metrics in this class are independent of scaling the entire problem, it would be convenient to simply require that all the numbers in the input be integers.

These considerations are significant because n, the number of components, is the quantity in which the hard problems appear not to be polynomial-time. If n were fixed, the problems might no longer be hard; and if vectors could be given in some abbreviated form instead of explicitly, then the hardness of the problems would be less surprising. The numeric precision issues will be discussed in more detail as they come up.

The complexity results were announced, with brief sketches of the proof techniques involved, in SISAP'08 [192]. In the present work, the proofs are broken down as follows.

- Theorem 8.3: VPREVERSE in $\mathcal{NPC}$ for L_p with finite p;
- Theorem 8.4: VPREVERSE in $\mathcal{NPC}$ for L_∞;
- Theorem 8.5: GHREVERSE in $\mathcal{P}$ for L_2;
- Theorem 8.9: GHREVERSE in $\mathcal{NPC}$ for L_p with finite $p \neq 2$; and
- Theorem 8.10: GHREVERSE in $\mathcal{NPC}$ for L_∞.

The exception for GHREVERSE in L_2 is interesting in two ways. It shows that Euclidean space is special in having an easy GHREVERSE problem even though in many other ways all L_p metrics are interchangeable; and it shows a boundary between VPREVERSE and GHREVERSE. In most spaces the two problems are basically equivalent. The strongest division between them so far was the one in Theorem 4.6, which only becomes relevant for strange, badly-behaved spaces. The results of this chapter show a situation where, with ordinary well-behaved spaces actually used in practice, VPREVERSE and GHREVERSE are really fundamentally different.

Throughout this chapter, we will use n to refer to the length (number of components) of a vector, or the number of letters in a string. Table 8.1 is a quick reference guide to that and other variable names used in this chapter; where possible, we keep the names consistent among different proofs. Similar symbols like $\mathbf{x}_i$ and x_i are differentiated by typography, but the difference between them should be clear from context also. We define the following notation for unit vectors.

unit vector ($\mathbf{u}_i$)

Definition 8.2

Let $\mathbf{u}_i$ represent the *unit vector* along the positive i-th coordinate axis; that is, the vector with the i-th component equal to 1 and all other components zero.

We use the same general approach for all the $\mathcal{NP}$-complete cases. Enough details differ between VPREVERSE and GHREVERSE, and between L_p for finite p and L_∞, to require separate proofs, but in all four cases, we begin by mapping binary strings to the corners of a hypercube. Then it becomes possible to reduce either from the same problem on strings with Hamming distance, or directly from 3SAT, to establish $\mathcal{NP}$-hardness.

The polynomial-time case (Theorem 8.5) is a reduction to linear programming. Some issues arise from questions of which model of computation we use; those

δ, ϵ	arbitrarily small positive reals
i, j, k	indices
n	number of components in vector or letters in string
p	type of L_p metric
r, r_1, r_2	sphere radii
$\mathbf{u}_i$	unit vector along positive i-th axis
v_i	i-th variable in 3SAT instance
$\mathbf{v}$	normal vector for linear programming constraint
$\mathbf{x}$	sphere centre (VP) or nearer point in pair (GH)
x_i	i-th component of $\mathbf{x}$
$\mathbf{x}_i$	centre of i-th sphere or nearer point in i-th pair
$x_{i,j}$	j-th component of $\mathbf{x}_i$
$\mathbf{y}$	sphere centre or farther point in pair
y_i	i-th component of $\mathbf{y}$
$\mathbf{y}_i$	farther point in i-th pair
$y_{i,j}$	j-th component of $\mathbf{y}_i$
$\mathbf{z}$	solution to VPREVERSE or GHREVERSE instance
z_i	i-th component of $\mathbf{z}$
$\mathbf{z}_1, \mathbf{z}_2$	two solution vectors
z'_1, z'_2	strings corresponding to solution vectors

Table 8.1: Some variable names used in vector reverse-similarity proofs.

are discussed in detail in Section 8.3. The intuition for why that case is easy comes from the fact that it is, uniquely, the question of finding a point in the intersection of a collection of convex sets. All the other cases allow some or all the sets to be nonconvex, and the $\mathcal{NP}$-hardness proofs involve arranging the sets to make their nonconvexity encode an $\mathcal{NP}$-hard problem.

We conclude the chapter with some comments on what happens to the VPREVERSE question when all the spheres are constrained to the same diameter.

8.1 VPREVERSE with the L_p metric for finite p

For the VPREVERSE problem in a finite-p L_p space, we will reduce from the same problem in Hamming distance space, known to be $\mathcal{NP}$-complete by Theorem 5.4. The basic encoding is almost trivial: n-bit binary strings reduce to n-component vectors, with `0` encoded by 0 and `1` encoded by 1. The complication is that the definition of VPREVERSE does not allow for placing a strict equality constraint on distances: we can write a triple $(\mathbf{x}, r, 1)$ to require the solution to be within or on the surface of a sphere, or $(\mathbf{y}, r, 0)$ to require that it be strictly outside, but we cannot combine them to require an exact distance from the solution to the centre, as we might with integer-valued metrics. Instead, we get around the limitation in the obvious way by adding an arbitrarily small offset ϵ; then the solution has a nonempty topological interior, we can expect to find a solution with rational coordinates if there is a solution at all, and some of the thornier numeric precision issues vanish.

The approach used to limit the solution to the corners is illustrated in Figure 8.1. We place two spheres to limit the solutions to be close to the surface of the sphere that circumscribes the hypercube, then use another sphere offset in the plus or minus direction along each of the n dimensions ($2n$ of these in total) to remove the edges, faces, and higher-dimensional analogues, leaving the solution restricted to a small region around each corner. We can make the regions arbitrarily small, and when they are small enough, an equivalence to the VPREVERSE problem on Hamming strings follows.

The following lemma is part of the proof that the corner-limiting gadget in Figure 8.1 really works. It may seem obvious, but we state it explicitly to emphasise that it really does work for all $p \geq 1$. The figure only shows the intuitive case of two-dimensional Euclidean space; and excessive reliance on easy cases is dangerous, as we have seen with such things as intrinsic dimensionality in Chapter 2 and will see again later in this chapter when we consider other metrics.

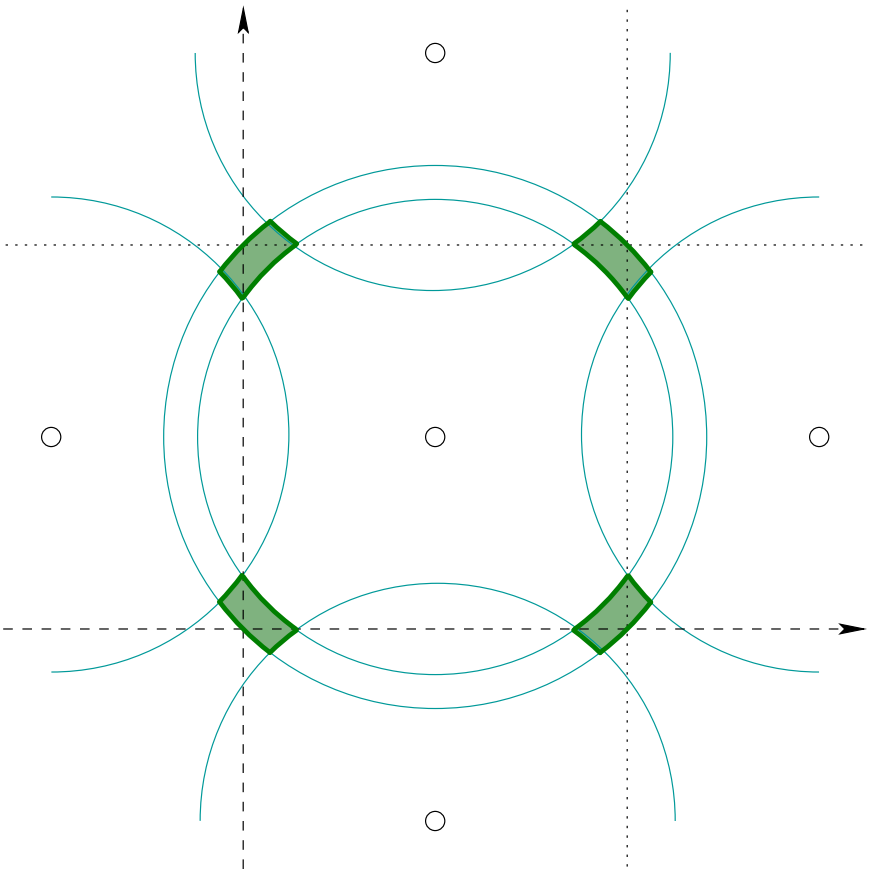


Figure 8.1: Limiting the solution to the corners for VPREVERSE in L_p space.

Lemma 8.1
Let $f(x) = |x|^p - |1-x|^p$. Then for any $\epsilon > 0$, $p \geq 1$, there is a $\delta > 0$ such that if $f(x) < \delta$ then $x < {}^1\!/_2 + \epsilon$. Also, for fixed p and sufficiently small ϵ, δ is only polynomially small in ϵ.

Proof The function $f(x)$ is everywhere continuous. Consider its derivative:

$$f'(x) = \begin{cases} -p(-x)^{p-1} + p(1-x)^{p-1} & \text{if } x \leq 0 \\ px^{p-1} + p(1-x)^{p-1} & \text{if } 0 \leq x \leq 1 \\ px^{p-1} - p(x-1)^{p-1} & \text{if } x \geq 1 \end{cases}$$

The derivative is defined and nonnegative for all x, so $f(x)$ is monotonically increasing. Furthermore, $f({}^1\!/_2) = 0$. Therefore by requiring $f(x)$ to be less than some positive δ, we can require x to be less than ${}^1\!/_2 + \epsilon$ for arbitrarily small ϵ.

Note that $f(x)$ is also a polynomial-sized function of x; actually only piecewise polynomial-sized, but for x close to ${}^1\!/_2$ only the middle piece is relevant. Therefore as ϵ decreases, δ can only decrease polynomially. ■

The point about δ being polynomial in ϵ matters because it means we will not find it impossible to write the sphere radii in polynomial space during the $\mathcal{NP}$-completeness reduction. The present case is not a difficult or mysterious one. It should be intuitive that we can make the spheres nearly touch each other, but not quite, and get them close enough to use in the reduction, without running out of bits in which to name the radii. But use of real numbers in these kinds of problems always calls for some caution, as will be discussed for the case of linear programming (where it does become difficult and mysterious) in Section 8.3. For now, we proceed to apply the lemma.

Lemma 8.2
For any L_p metric and arbitrarily small $\epsilon > 0$, there exists a polynomial-sized VPREVERSE instance on n-component vectors such that every solution is within a distance less than ϵ from an n-bit string (that is, a vector in which every component is 0 or 1), and every n-bit string is a solution.

Proof Let $\mathbf{x} = \sum_{i=1}^{n} {}^1\!/_2 \mathbf{u}_i$. That represents the centre of the hypercube. Let δ be a small positive real number we will choose later. Let $r_1 = {}^1\!/_2 \sqrt[p]{n}$; that is the radius

of the L_p sphere circumscribing the hypercube. Let $r_2 = (n2^{-p} - \delta)^{1/p}$; that is a slightly smaller radius, used for all the other spheres. These are the spheres in the VPREVERSE instance:

$$(\mathbf{x}, r_1, 1) \tag{8.1}$$

$$(\mathbf{x} \pm \mathbf{u}_i, r_2, 0) \text{ for all } i \in \{1, 2, \ldots, n\} \tag{8.2}$$

$$(\mathbf{x}, r_2, 0)\,. \tag{8.3}$$

It is easy to verify that all corners of the hypercube, which are exactly the set of n-bit strings written as vectors, are included as solutions. Consider a solution $\mathbf{z}$ and the spheres (8.1) and (8.2) for some vector component z_i, without loss of generality z_1. By the definition of L_p spheres, they impose the following constraints on the components:

$$|1/2 - z_1|^p + \sum_{j=2}^{n} \left|1/2 - z_j\right|^p \le n2^{-p}$$

$$|-1/2 - z_1|^p + \sum_{j=2}^{n} \left|1/2 - z_j\right|^p > n2^{-p} - \delta$$

$$|3/2 - z_1|^p + \sum_{j=2}^{n} \left|1/2 - z_j\right|^p > n2^{-p} - \delta\,.$$

Then we can subtract the inequalities and apply Lemma 8.1 to place bounds on z_1, using $f(x) = |x|^p - |1-x|^p$. By the lemma, we can choose δ to make ϵ as small as desired. We have

$$\begin{aligned} |-1/2 - z_1|^p - |1/2 - z_1|^p &> -\delta \\ |1/2 - z_1|^p - |1 - (1/2 - z_1)|^p &< \delta \\ f(1/2 - z_1) &< \delta \\ 1/2 - z_1 &< 1/2 + \epsilon \\ z_1 &> -\epsilon \end{aligned} \qquad \begin{aligned} |3/2 - z_1|^p - |1/2 - z_1|^p &> -\delta \\ |z_1 - 1/2|^p - |1 - (z_1 - 1/2)|^p &< \delta \\ f(z_1 - 1/2) &< \delta \\ z_1 - 1/2 &< 1/2 + \epsilon \\ z_1 &< 1 + \epsilon \end{aligned}$$

and therefore for all components z_i:

$$\left|1/2 - z_i\right| < 1/2 + \epsilon\,. \tag{8.4}$$

Now consider the sphere described by (8.3). It is equivalent to this constraint on the components of $\mathbf{x}$:

$$\sum_{j=1}^{n} \left|1/2 - z_j\right|^p > n2^{-p} - \delta\,.$$

Combining that constraint with (8.4) gives, for each component z_i:

$$(n-1)(\tfrac{1}{2}+\epsilon)^p + \left|\tfrac{1}{2} - z_i\right|^p > n2^{-p} - \delta$$

and by making δ sufficiently small, we can force $\left|\tfrac{1}{2} - z_i\right|$ arbitrarily close to $\tfrac{1}{2}$. Then each component of $\mathbf{z}$ must be arbitrarily close to 0 or 1, and so by abuse of notation the distance from $\mathbf{z}$ to the nearest n-bit string can be made less than any chosen ϵ. Moreover, we can write down the VPREVERSE instance in a number of bits polynomial in n, for fixed p, by increasing r_1 and r_2 to conveniently-representable numbers polynomially close to their preferred values, while introducing distortion of less than ϵ in the distances. ■

Limiting the solutions to the corners is the difficult part of the result; with that out of the way, the $\mathcal{NP}$-completeness of the problem follows naturally.

Theorem 8.3
VPREVERSE is in $\mathcal{NPC}$ for real vectors with an L_p metric and finite p.

Proof By Lemma 8.2, we can construct a polynomial-sized VPREVERSE instance for this space such that every solution is within ϵ distance of an n-bit string. Then by the triangle inequality, the distance between two solutions $\mathbf{z}_1$ and $\mathbf{z}_2$ is within 2ϵ of $\sqrt[p]{d_{\mathrm{H}}(z_1', z_2')}$ where z_1' and z_2' are the strings corresponding to the nearest corners to $\mathbf{z}_1$ and $\mathbf{z}_2$, and d_{H} is Hamming distance. Then each possible Hamming distance (they are all nonnegative integers) corresponds to an interval of width 4ϵ of L_p distances, and by choosing sufficiently small ϵ we can make these intervals non-overlapping.

Note that $\sqrt[p]{\cdot}$ is a strictly increasing function. For any sphere in Hamming-distance space we can find a sphere in L_p space such that a solution in L_p space is in the L_p sphere if and only if it corresponds to a string in the Hamming sphere. The VPREVERSE problem on n-bit strings with Hamming distance is $\mathcal{NP}$-complete by Theorem 5.4. Then we can take a Hamming-distance VPREVERSE instance and convert it to an equivalent L_p VPREVERSE instance, and so L_p VPREVERSE is also $\mathcal{NP}$-hard.

This problem is also in $\mathcal{NP}$, making it $\mathcal{NP}$-complete, because the solution vector $\mathbf{z}$ is a polynomial-time-verifiable certificate. Note that although we give less detail on this point here than in the previous proof, there is still nothing subtle going on with precision: the solution regions near the hypercube corners are of polynomial size, any solution will do, and so we can always pick a solution that can be written conveniently in a polynomial number of bits. ■

8.2 VPREVERSE with the L_∞ metric

The VPREVERSE problem is also $\mathcal{NP}$-hard with the L_∞ metric, but the proof in the previous section cannot be applied because of problems with enforcing the encoding. The L_p construction shown in Figure 8.1 depends on using the inner sphere (8.3) to eliminate all the interior of the hypercube except the corners. If we imagine the sphere being inflated like a bubble from a single point at the centre of a three-dimensional cube, it first intersects the cube's surface at the centres of the faces, then circular bulges grow in each face until they meet along the edges, then larger and larger segments of the edges are engulfed until finally the vertices disappear.

In the L_∞ case, the sphere is a hypercube, and it does not intersect the surface of the hypercube representing truth assignments at all until it suddenly coincides with the entire surface all at once. There is no possibility to exclude all but the corners by a careful selection of radius. Similarly, we would not find a strictly increasing function from Hamming distance to L_∞ distance; any two distinct corners of a hypercube have equal L_∞ distance.

Instead, we use a different encoding and reduce directly from 3SAT. Vector components will represent variables. Each component will be nonnegative for a variable set to true, and negative for a variable set to false. As in the previous section the overall goal is to restrict solutions to be near the corners of a hypercube; but we count components with the correct signs as close enough instead of requiring an arbitrarily small error. We also exploit the convenient form of L_∞ spheres: they are simply products of intervals.

Theorem 8.4
VPREVERSE is in $\mathcal{NPC}$ for real vectors with L_∞.

Proof The reduction is from 3SAT. Let $v_1, v_2, \ldots, v_n$ be the variables of any arbitrary 3SAT instance. We will create a polynomial-sized equivalent instance of VPREVERSE on n-component real vectors with the L_∞ metric. Let the VPREVERSE instance contain the sphere $(\mathbf{0}, 1, 1)$; that restricts solutions to have each component in the closed interval $[-1, 1]$.

For each clause in the input 3SAT instance, where v_i, v_j, v_k are the variables in the clause, add a sphere $(\pm\mathbf{u}_i \pm \mathbf{u}_j \pm \mathbf{u}_\mathbf{k}, {}^3\!/_2, 0)$ where the signs on the unit vectors are chosen positive if the corresponding variables are negated in the clause, negative if not negated. Now, bearing in mind that every component in a solution vector must be in the range $[-1, 1]$, consider a solution $\mathbf{z}$ to the VPREVERSE

problem. To be outside this sphere there must be at least one component that differs from the sphere's centre by more than $^3/_2$. No component but z_i, z_j, and z_k can be that far from the sphere's centre because the sphere's centre is zero in all but those components and solution components must be in $[-1, 1]$. But the solution vector can differ from the sphere centre by more than $^3/_2$ if one of the three components corresponding to variables in the clause is less than $-^1/_2$ for a negated literal or greater than $^1/_2$ for a non-negated literal. Those values are exactly the ones equivalent to assignments that satisfy the clause in the original 3SAT problem.

Therefore, when spheres have been added corresponding to all the clauses, then for any solution to the 3SAT instance there will be a corresponding vector consisting of -1 or 1 in each component, corresponding to F or T respectively, and for any vector that solves the VPREVERSE instance, we can find a satisfying assignment for the 3SAT problem by converting negative components to F assignments and nonnegative to T. Note that this VPREVERSE instance is made up entirely of easy-to-represent constants and is polynomial-sized. Therefore VPREVERSE in this space is $\mathcal{NP}$-hard; and it is $\mathcal{NP}$-complete because the satisfying vector is a polynomial-time certificate. ■

8.3 GHREVERSE in Euclidean space

There is a special case for the GHREVERSE problem: Euclidean space. The proof for finite p given in the next section does not apply to $p = 2$, and that does not just represent a gap in the proof, but an actual and fundamental difference in the underlying spaces. For finite p in general, GHREVERSE is $\mathcal{NP}$-hard; but for $p = 2$, it is easy, as shown by the following theorem.

Theorem 8.5
GHREVERSE is in $\mathcal{P}$ for real vectors with the Euclidean metric, by a polynomial-time equivalence with linear programming in the same space.

Proof Each pair of points in the input problem corresponds to a hyperplane that bisects the two points, and the solution $\mathbf{z}$ is any point that is on the specified sides of all the hyperplanes. For each pair of vectors $(\mathbf{x}, \mathbf{y})$ in a GHREVERSE instance, the vector $(\mathbf{x}+\mathbf{y})/2$ is on the hyperplane and the vector $(\mathbf{y}-\mathbf{x})$ is normal to it, pointing in the direction of the $\mathbf{y}$ (undesired) half-space, so the corresponding inequality is $(\mathbf{y}-\mathbf{x}) \cdot \mathbf{z} \leq (\mathbf{y}-\mathbf{x}) \cdot (\mathbf{x}+\mathbf{y})/2$. The points that satisfy all those

inequalities are exactly the points that are solutions to the GHREVERSE instance. Solving the system of inequalities is an instance of linear constraint feasibility, or linear programming if we supply any linear objective function (such as the constant zero). Linear programming is in $\mathcal{P}$, so GHREVERSE is also.

In the other direction, given a linear programming instance stated as a decision problem (the decision being whether it is possible to achieve a given value for the objective), the constraint on the objective function can be written as just another constraint on a linear function. Then each constraint on the solution vector $\mathbf{z}$ is of the form $\mathbf{z} \cdot \mathbf{v} \leq c$ for some vector $\mathbf{v}$ and scalar c, and this GHREVERSE pair is equivalent:

$$\left(\left(\frac{c}{\mathbf{v} \cdot \mathbf{v}} - 1 \right) \mathbf{v}, \left(\frac{c}{\mathbf{v} \cdot \mathbf{v}} + 1 \right) \mathbf{v} \right) .$$

Therefore we can convert any linear programming instance into an equivalent GHREVERSE instance. ∎

That is a simple result, but it has subtle consequences because the status of linear programming is not fully known. Throughout this work we assume that real numbers in the input to a decision problem must be stated in binary, with their sizes counted in the size of the input. That implies they must actually be rationals; and in fact, because of the scaling-independence of both linear programming and L_p reverse similarity problems, it is convenient to require that all the input numbers be integers. In that model, linear programming is known to be in $\mathcal{P}$; one well-known algorithm for it is due to Karmarkar [117].

But the known polynomial-time algorithms for linear programming, including Karmarkar's, are interior-point methods that work iteratively, improving the precision of an estimate in each successive iteration until they reach an exact solution. The time to complete the process is not limited by a polynomial in the number of constraints and dimensions. Suppose instead of counting bits we use a model of computation that allows for arbitrary real numbers and charges a cost of one unit (in space) for storing a real number or (in time) for doing an arithmetic operation on two real numbers—hence called the *unit cost model*. Then we can construct an ill-conditioned linear programming instance for which existing methods will need more than polynomial time. The real numbers used in such an instance require more than polynomial bits to write down, which is how this result does not contradict the polynomial bound for the standard model.

unit cost model

Megiddo describes an algorithm for linear programming in the unit cost model with time linear for any fixed dimension, but exponential in number of dimensions [151]. There is also a randomised algorithm due to Matoušek, Sharir, and Welzl with expected time polynomial in the unit cost model, but worst case time exponential [150]. The expected polynomial time of that algorithm applies

to any input; worst-case behaviour is determined by bad random choices in the algorithm, not bad input. These results seem to come very close to establishing linear programming as worst-case polynomial time in the unit cost model, but it remains open.

For our purposes, however, the main question was in the Turing-machine model where inputs must be given in binary, and in that model, GHREVERSE in Euclidean space is special for being in $\mathcal{P}$ while all the other cases of VPREVERSE or GHREVERSE on L_p real vector spaces are $\mathcal{NP}$-complete.

8.4 GHREVERSE with the L_p metric for finite $p \neq 2$

Intuitively, the reason for GHREVERSE with the Euclidean metric to be easy is that the set of solutions is convex. The bisector of two points in Euclidean space is a flat hyperplane. It divides the space into two sets which are each convex. The intersection of any number of such sets remains convex. That convexity seems to be what makes the problem easy.

With any other L_p metric, the set described by a pair of points and their bisector is no longer convex; and the non-convex sets can have arbitrarily complicated intersections. As a result, with any L_p metric other than L_2, the GHREVERSE problem becomes $\mathcal{NP}$-complete. As in the VPREVERSE case, we will restrict the solutions to the corners of a hypercube and then reduce from the same problem on binary strings. The gadget to do that uses a dimension-doubling trick similar to that used by Frances and Litman in their work on Hamming distance strings [76]. Each letter in the binary strings corresponds to two components in the vectors, and we use one more component as a slack variable, for a total of $2n + 1$ vector components.

The proof is complicated, so we will introduce it in several pieces. First, assume we start with an instance of GHREVERSE on binary strings of length n with Hamming distance. We will construct an equivalent instance of GHREVERSE on real vectors of length $2n + 1$, using the L_p metric for some finite constant $p \geq 1$ with $p \neq 2$. A string is encoded to a vector consisting of a zero component followed by two repetitions of the letters of the string split into components, so that for instance the string 011 corresponds to the vector $\langle 0, 0, 1, 1, 0, 1, 1 \rangle$. The first step is to restrict the vector components to 0 for the first component and the interval $[0, 1]$ for the other components; the latter restriction is illustrated in Figure 8.2.

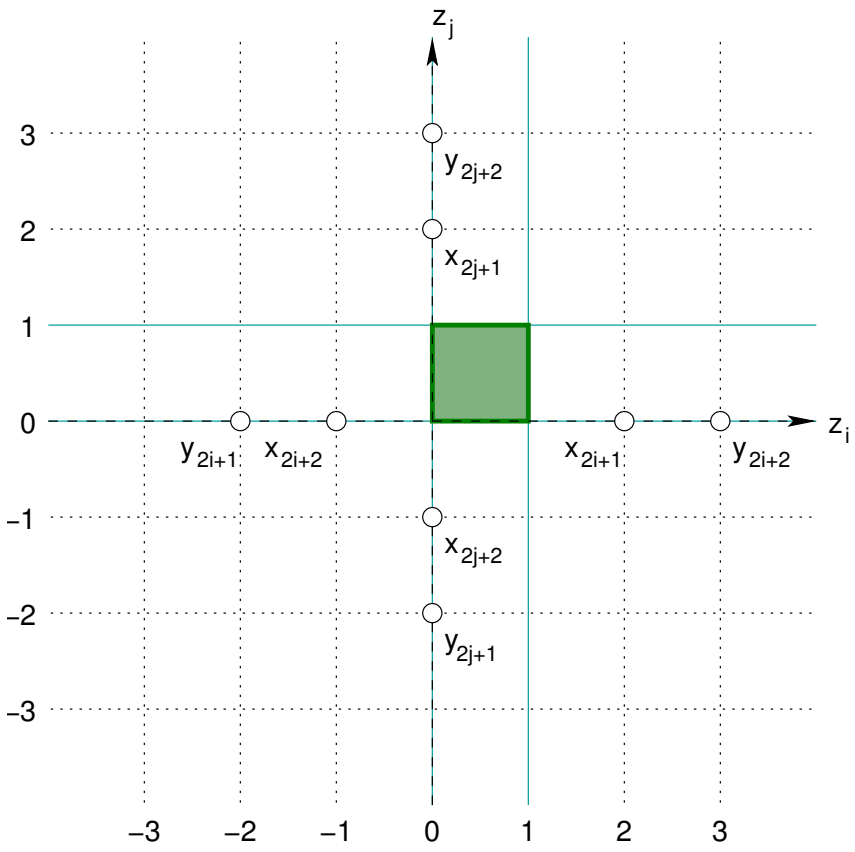


Figure 8.2: Limiting vector components to $[0, 1]$.

Lemma 8.6
There exists a GHREVERSE instance on $2n+1$-component real vectors with the L_p metric such that the following hold for any solution $\langle z_1, z_2, \ldots, z_n \rangle$:

$$z_1 = 0 \tag{8.5}$$
$$z_i \geq 0 \quad 1 \leq i \leq 2n+1 \tag{8.6}$$
$$z_i \leq 1 \quad 1 \leq i \leq 2n+1\,. \tag{8.7}$$

Proof Introduce two pairs of points for each of the $2n+1$ vector components:

$$\mathbf{x}_1 = 2\mathbf{u}_1 \qquad \mathbf{y}_1 = -2\mathbf{u}_1 \tag{8.8}$$
$$\mathbf{x}_2 = -2\mathbf{u}_1 \qquad \mathbf{y}_2 = 2\mathbf{u}_1 \tag{8.9}$$
$$\mathbf{x}_{2i+1} = 2\mathbf{u}_i \qquad \mathbf{y}_{2i+1} = -2\mathbf{u}_i \quad 1 < i \leq 2n+1 \tag{8.10}$$
$$\mathbf{x}_{2i+2} = -\mathbf{u}_i \qquad \mathbf{y}_{2i+2} = 3\mathbf{u}_i \quad 1 < i \leq 2n+1\,. \tag{8.11}$$

For finite p and each pair $(\mathbf{x}_i, \mathbf{y}_i)$, a vector $\mathbf{z}$ is a solution to the instance only if

$$d(\mathbf{x}_i, \mathbf{z}) \leq d(\mathbf{y}_i, \mathbf{z}) \tag{8.12}$$
$$\left[\sum_{j=1}^{2n+1} |x_{i,j} - z_j|^p\right]^{1/p} \leq \left[\sum_{j=1}^{2n+1} |y_{i,j} - z_j|^p\right]^{1/p}. \tag{8.13}$$

Since p and all the absolute values are nonnegative, and $\mathbf{x}_i$ and $\mathbf{y}_i$ differ in only one component, we can simplify (8.13) to refer only to that component. Where j is the component in which the vectors differ, we have (8.12) if and only if $|x_{i,j} - z_j| \leq |y_{i,j} - z_j|$. For $x_{i,j} < y_{i,j}$, we can remove the absolute value signs as follows:

$$|x_{i,j} - z_j| \leq |y_{i,j} - z_j| \Leftrightarrow \begin{cases} x_{i,j} - z_j \leq y_{i,j} - z_j & \text{(true) if } z_j \leq x_{i,j}; \\ z_j - x_{i,j} \leq y_{i,j} - z_j & \text{if } x_{i,j} < z_j \leq y_{i,j}; \\ z_j - x_{i,j} \leq z_j - y_{i,j} & \text{(false) if } y_{i,j} < z_j\,. \end{cases} \tag{8.14}$$

The inequality (8.14) is therefore equivalent to $z_j - x_{i,j} \leq y_{i,j} - z_j$, which is equivalent to $z_j \leq (x_{i,j} + y_{i,j})/2$, when $x_{i,j} < y_{i,j}$. The same kind of argument gives $z_j \geq (x_{i,j} + y_{i,j})/2$ when $x_{i,j} > y_{i,j}$. For the pathological case of $x_{i,j} = y_{i,j}$ and therefore $\mathbf{x}_i = \mathbf{y}_i$, there is no restriction on $\mathbf{z}$. In summary, a pair can be used as a gadget to place a "less than or equal" or "greater than or equal" condition on any component of $\mathbf{z}$.

The pairs we have introduced then use that technique to create the required constraints on the value of $\mathbf{z}$. The pairs (8.8) and (8.9) enforce the constraints $z_1 \geq 0$ and $z_1 \leq 0$ respectively, which combine to enforce $z_1 = 0$, (8.5). The pairs (8.10) and (8.11) enforce the constraints $z_i \leq 1$ and $z_i \geq 0$, making up (8.6) and (8.7) for each z_i except the first, to which those constraints already apply because of the stronger constraint that $z_1 = 0$. ■

Now we must constrain the values of the components even further, so that except for the first component, which is used by the construction, the remaining components each have only two values. It was in preparation for this step that we introduced two components for each letter in the string instance; although we only need one component to model a letter, the second is required by the gadget shown in Figure 8.3. The case $p = 1.7$ is shown. The open circles are the vantage points used in GHREVERSE pairs; the one in the middle is displaced a distance in a third, unshown dimension ($\mathbf{u}_1$), and its bisectors with the other two intersect to limit the values of $z_i + 1$ and $z_i + n + 1$.

Here is where we make use of the non-convexity of Voronoi cells in L_p metrics other than L_2. The line segment from $(0,0)$ to $(1,1)$ is a counterexample to the convexity of Voronoi cells: both endpoints are in a cell but intermediate points between them are not. As a result we can intersect two cells, and the existing square constrained region from previous pairs, to give only two pairs of values for the two components that satisfy all constraints. As p increases from 1 to 2, the spindle-shaped region in the centre of the figure becomes thinner and thinner. For $p > 2$, the curves pass each other, but we can invert the sense of the pairs and again have two concave-sided triangles with an empty spindle between them. But for p exactly 2, no such trick applies: the triangular regions intersect everywhere along the segment from $(0,0)$ to $(1,1)$ and the gadget fails. That is the case shown to be in $\mathcal{P}$ by Theorem 8.5.

Although it is easy to explain intuitively why this gadget works, proving that it works for general p is surprisingly difficult because the spindle-shaped region is described only implicitly, by functions involving general powers of binomials. Trivial algebra does not suffice; the following lemma, resorting to techniques from calculus, makes the proof possible.

Lemma 8.7
Given reals $p \geq 1$ and $0 \leq x \leq 1$, the following hold:

$$x^p + |1-x|^p = 1 - 2^{1-p} + 2\left|\tfrac{1}{2} - x\right|^p \quad \text{if } x = 0, x = 1, \text{ or } p = 2; \tag{8.15}$$

$$x^p + |1-x|^p > 1 - 2^{1-p} + 2\left|\tfrac{1}{2} - x\right|^p \quad \text{if } 0 < x < 1 \text{ and } 1 \leq p < 2; \tag{8.16}$$

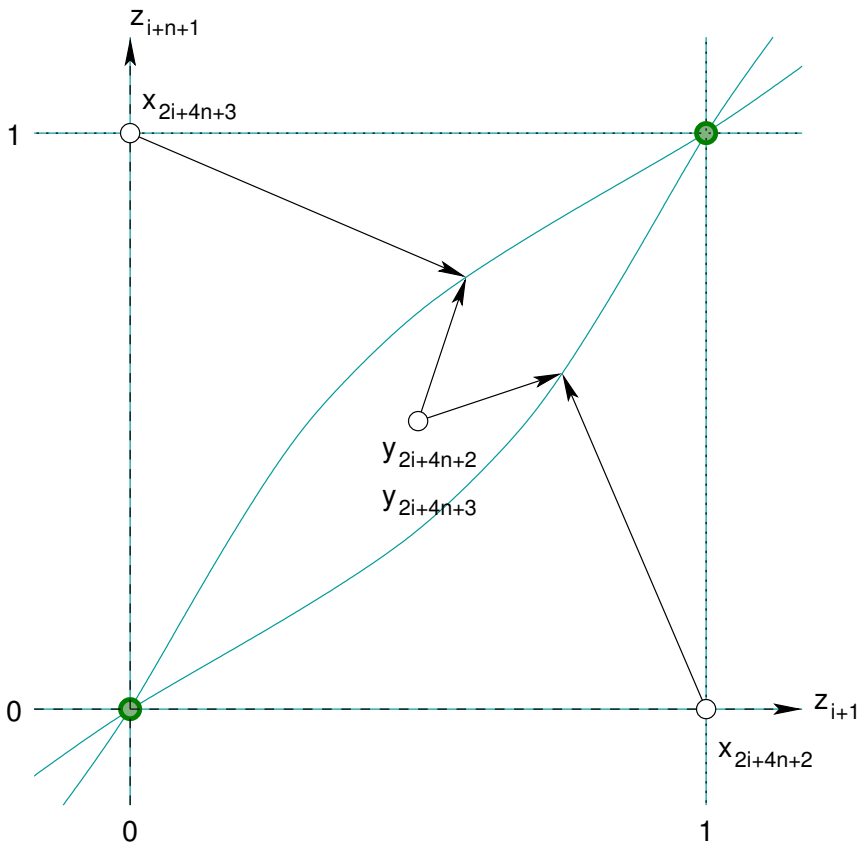


Figure 8.3: Limiting a pair of components to $\{0, 1\}$.

$$x^p + |1-x|^p < 1 - 2^{1-p} + 2|{}^1\!/_2 - x|^p \quad \text{if } 0 < x < 1 \text{ and } p > 2. \tag{8.17}$$

Proof The equality (8.15) follows by substituting $x = 0$, $x = 1$, and $p = 2$ in turn. Then for the remaining cases, we can assume $0 < x < 1$. Let $f_x(p) = x^p + |1-x|^p$ and $g_x(p) = 1 - 2^{1-p} + 2|{}^1\!/_2 - x|^p$; that is, the left-hand and right-hand sides of the statements we are proving, with notation that lends itself to considering them as functions of p. Both functions are continuous with respect to p.

Two special cases will be relevant:

$$f_x(1) = 1 > 2|{}^1\!/_2 - x| = g_x(1) \tag{8.18}$$
$$f_x(2) = 2x^2 - 2x + 1 = g_x(2). \tag{8.19}$$

The concept of the proof is that as we increase p starting from $p = 1$, the curve $f_x(p)$ begins above $g_x(p)$, and they both move towards each other to coincide when $p = 2$. After that, $f_x(p)$ remains below $g_x(p)$ as p goes to infinity. The curves, for representative values of p, are illustrated in Figures 8.4 and 8.5. In (8.18) we have the starting point of the process, with $f_x(p)$ above $g_x(p)$; then (8.19) shows that they do coincide at $p = 2$. The increase from $g_x(1)$ and $g_x(2)$ is not monotonic for all values of x, but the non-monotonicity is numerically small and almost invisible in Figure 8.5. Figure 8.6 shows a magnified section of $g_{0.15}(p)$ that makes it more obvious.

Consider the derivative of $f_x(p)$:

$$f_x'(p) = x^p \log x + |1-x|^p \log|1-x|.$$

Given $0 < x < 1$, $p \geq 1$, both logarithms must be negative and both p-th powers must be positive, so the derivative is always negative. The function $f_x(p)$ is maximised for $p = 1$ and strictly decreasing with increasing p.

Now, consider the first two derivatives of $g_x(p)$:

$$g_x'(p) = 2^{1-p} \log 2 + 2|{}^1\!/_2 - x|^p \log|{}^1\!/_2 - x|$$
$$g_x''(p) = -2^{1-p} \log^2 2 + 2|{}^1\!/_2 - x|^p \log^2|{}^1\!/_2 - x|.$$

Although $\log|{}^1\!/_2 - x|$ is undefined at $x = {}^1\!/_2$, the derivative is by definition a limit, and since $p \geq 1$ we can compute the limit $0 \log 0 = 0$ to define a value for these derivatives even at $x = {}^1\!/_2$; the logarithmic term vanishes. With these derivatives, we can describe the behaviour of $g_x(p)$ on intervals of p values with enough precision to prove the desired results.

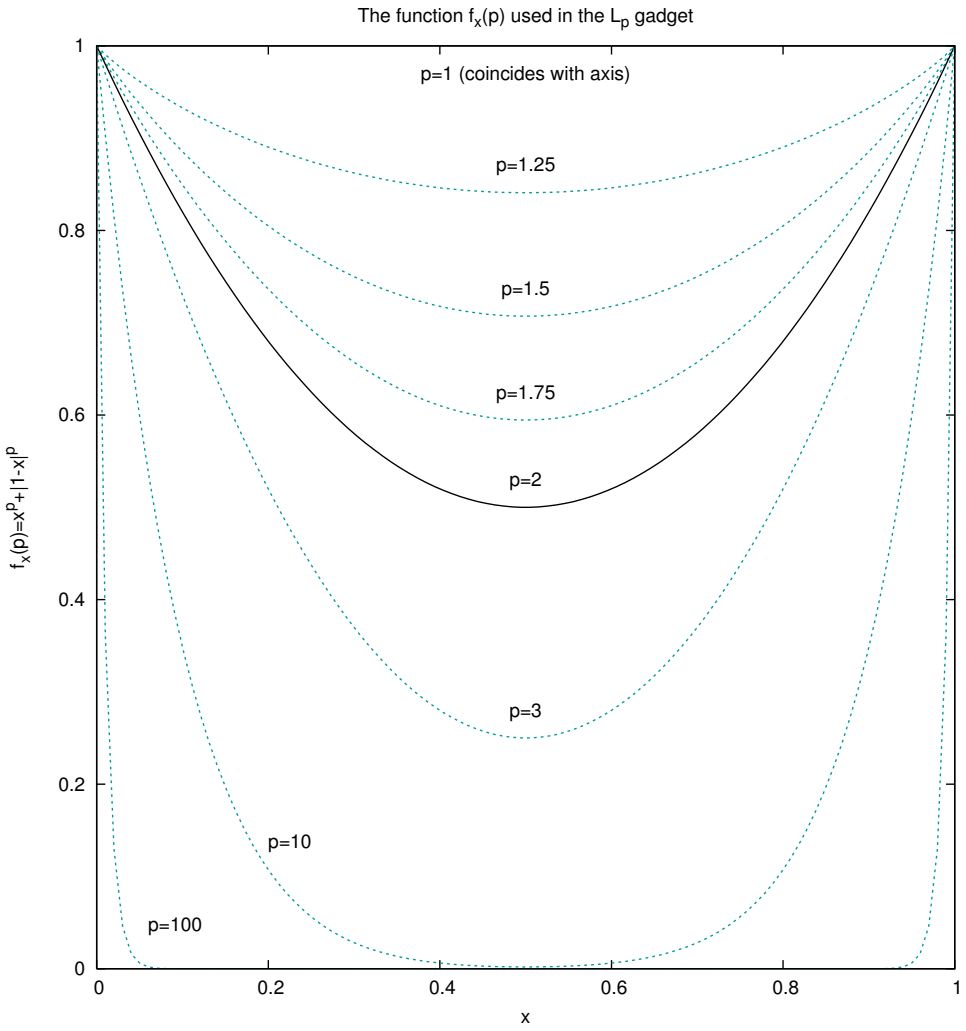


Figure 8.4: The function $f_x(p)$ for some representative values of p.

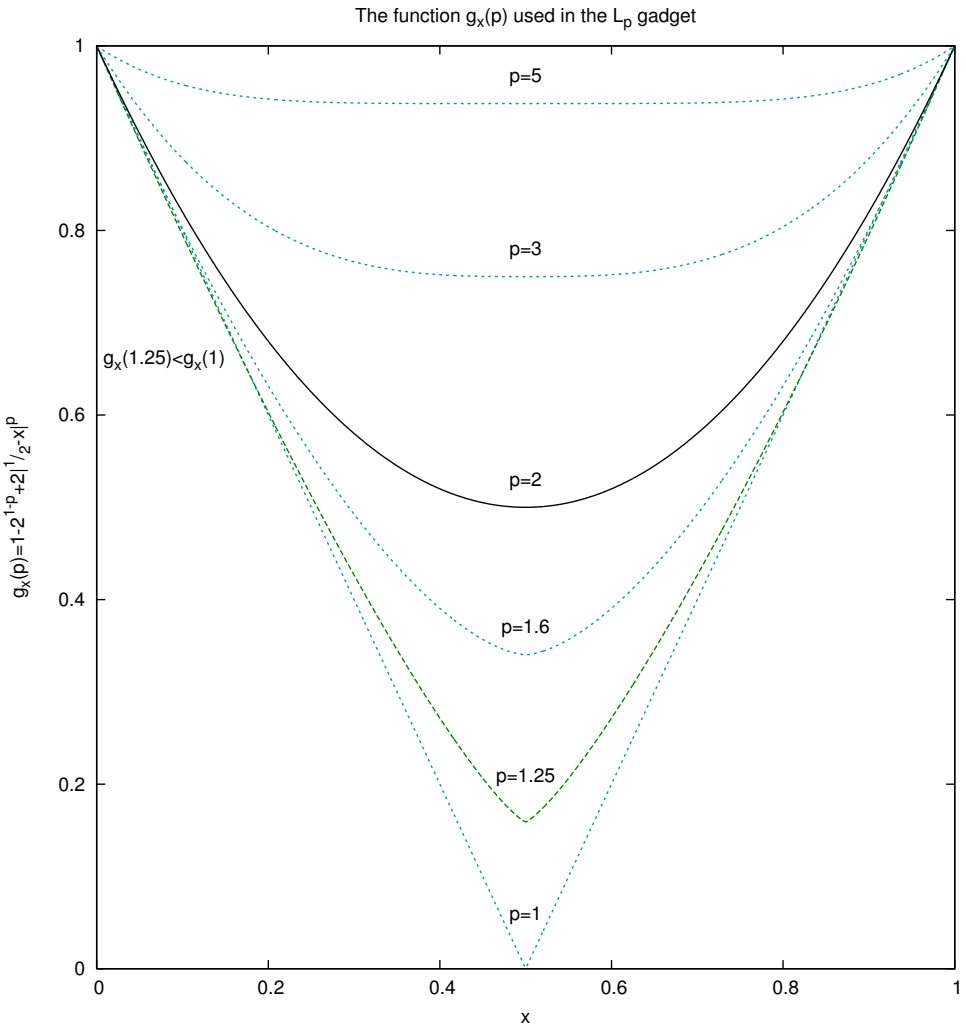


Figure 8.5: The function $g_x(p)$ for some representative values of p.

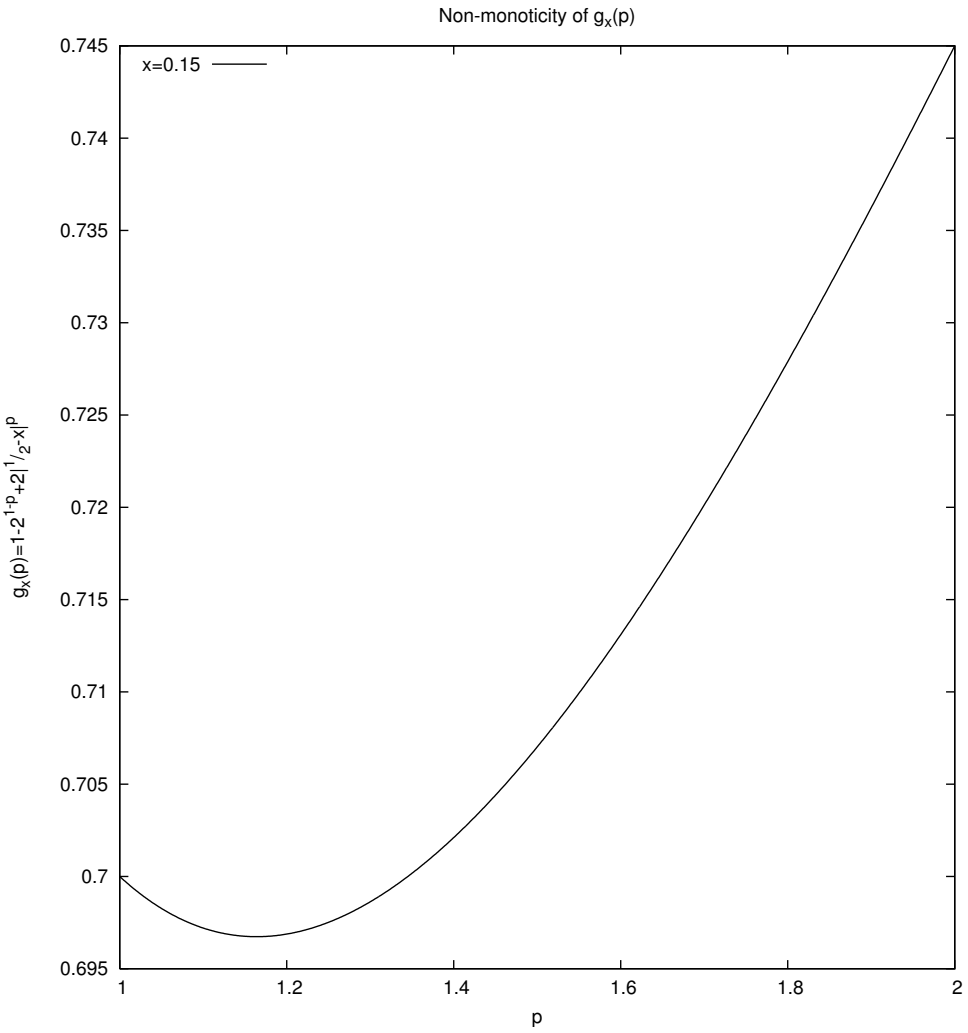


Figure 8.6: Curve showing the non-monotonicity of $g_x(p)$.

Consider the case $1 \leq p < 2$. We already have that $f_x(p) > f_x(2) = g_x(2)$ for all p in this interval, so it remains to show that $g_x(p) < g_x(2)$. We do that by determining the maximum value of $g_x(p)$ in the (closed) interval from $p = 1$ to $p = 2$. The values at the endpoints were calculated in (8.18) and (8.19). We split (8.18) into two cases depending on the sign inside the absolute value function, and find that the difference $g_x(2) - g_x(1)$ is positive in both cases, using the assumption that $0 < x < 1$:

$$\begin{aligned} 2x^2 - 2x + 1 - 2(\tfrac{1}{2} - x) &= 2x^2 > 0 \\ 2x^2 - 2x + 1 - 2(x - \tfrac{1}{2}) &= 2x^2 - 4x + 2 = 2(x-1)^2 > 0\,. \end{aligned}$$

Therefore $g_x(2) > g_x(1)$.

But the maximum could still occur at some intermediate value between $p = 1$ and $p = 2$. To address that case, we observe that the first two (indeed, all) derivatives are continuous over the entire interval, so we can find the relative extrema by setting the first derivative zero:

$$\begin{aligned} g_x'(p) &= 0 \\ -2^{1-p}\log 2 &= 2|\tfrac{1}{2} - x|^p \log|\tfrac{1}{2} - x|\,. \end{aligned}$$

That allows a substitution into the second derivative:

$$\begin{aligned} g_x''(p) &= -2^{1-p}\log^2 2 + 2|\tfrac{1}{2} - x|^p \log^2|\tfrac{1}{2} - x| \\ &= -(2^{1-p}\log 2)(\log 2) + (2|\tfrac{1}{2} - x|^p \log|\tfrac{1}{2} - x|)(\log|\tfrac{1}{2} - x|) \\ &= -(2^{1-p}\log 2)(\log 2 + \log|\tfrac{1}{2} - x|)\,. \end{aligned}$$

Now, $(2^{1-p}\log 2)$ must be positive. Since $0 < x < 1$, then $|\frac{1}{2} - x| < \frac{1}{2}$ and its logarithm is less than $-\log 2$, so the second term is negative. Then $g_x''(p)$ is the negative of a product of a positive and a negative, and so it is positive; therefore the graph is concave upward at this value of p, and so this is a relative minimum, not a maximum. The maximum on the closed interval is achieved at $p = 2$; $g_x(p)$ is less than $g_x(2)$ for all $1 \leq p < 2$. Then since $f_x(p) > f_x(2)$ and $g_x(p) < g_x(2)$, and $f_x(2) = g_x(2)$, we have (8.16).

Consider the case $p > 2$. We already have that $f_x(p) < f_x(2) = g_x(2)$ for all $p > 2$. The equation $g_x'(p) = 0$ has a unique solution and so $g_x'(p)$ changes sign at most once as p increases; there is only one p such that $g_x'(p) = 0$. And because we know $g_x''(p)$ is positive at that point, the change must be from negative to positive. We must determine the sign of $g_x'(p)$ at $p = 2$.

We have $g_x'(2) = \frac{1}{2}\log 2 + 2|\frac{1}{2} - x|^2 \log|\frac{1}{2} - x|$. The first term is an obviously positive constant, and the second term is of the form $z^2 \log z$. That expression is minimised when its derivative $2z\log z = 0$, at $z = e^{-1/2} \approx 0.6065$, which is

outside the range of values for $|^1/_2 - x|$. Therefore the minimum possible value of the second term is that achieved at the endpoint $|^1/_2 - x| = {}^1/_2$, namely $-{}^1/_2 \log 2$, just enough to make the derivative zero, not negative; and since $|^1/_2 - x| = {}^1/_2$ cannot actually occur given the strict inequalities (open interval) $0 < x < 1$, then the derivative $g'_x(2)$ is always positive. The value of p minimising $g_x(p)$ must be less than $p = 2$; so for increasing $p > 2$, $g_x(p)$ is strictly increasing.

So for $p > 2$, $g_x(p) > g_x(2) = f_x(2) > f_x(p)$ and we have (8.17). ∎

Now we can precisely define the gadget and prove its correctness.

Lemma 8.8
There exists a GHREVERSE instance on $(2n+1)$-component real vectors with the L_p metric such that for any solution $\langle z_1, z_2, \ldots, z_n \rangle$, $z_1 = 0$ and for any integer $1 < i \leq n+1$, $z_i \in \{0, 1\}$ and $z_{i+n} = z_i$.

Proof Start with the instance of Lemma 8.6, which restricts the first component to zero and all others to $[0, 1]$. For each $1 \leq i \leq n$ we add two pairs ($\mathbf{x}_{2i+4n+2}$, $\mathbf{x}_{2i+4n+2}$) and ($\mathbf{x}_{2i+4n+3}$, $\mathbf{x}_{2i+4n+3}$), with the following values for $p < 2$. Note that the indices are chosen to continue the numbering used in Lemma 8.6.

$$\begin{aligned} \mathbf{x}_{2i+4n+2} &= \mathbf{u}_{i+1} & \mathbf{y}_{2i+4n+2} &= (1 - 2^{1-p})^{1/p} \mathbf{u}_1 + {}^1/_2(\mathbf{u}_{i+1} + \mathbf{u}_{i+n+1}) \\ \mathbf{x}_{2i+4n+3} &= \mathbf{u}_{i+n+1} & \mathbf{y}_{2i+4n+3} &= (1 - 2^{1-p})^{1/p} \mathbf{u}_1 + {}^1/_2(\mathbf{u}_{i+1} + \mathbf{u}_{i+n+1}) . \end{aligned} \tag{8.20}$$

For $p > 2$ the pairs are reversed:

$$\begin{aligned} \mathbf{x}_{2i+4n+2} &= (1 - 2^{1-p})^{1/p} \mathbf{u}_1 + {}^1/_2(\mathbf{u}_{i+1} + \mathbf{u}_{i+n+1}) & \mathbf{y}_{2i+4n+2} &= \mathbf{u}_{i+1} \\ \mathbf{x}_{2i+4n+3} &= (1 - 2^{1-p})^{1/p} \mathbf{u}_1 + {}^1/_2(\mathbf{u}_{i+1} + \mathbf{u}_{i+n+1}) & \mathbf{y}_{2i+4n+3} &= \mathbf{u}_{i+n+1} . \end{aligned} \tag{8.21}$$

Consider a pair of components z_i and z_{i+n}. From Lemma 8.6, they are both constrained to the interval $[0, 1]$.

Each pair of vectors $(\mathbf{x}, \mathbf{y})$ corresponds to a constraint $d(\mathbf{x}, \mathbf{z}) \leq d(\mathbf{y}, \mathbf{z})$. As before, we can expand that to (8.13). Since $x^{1/p}$ is a strictly increasing function we can eliminate it without changing the truth of the inequality; and since all the vectors $\mathbf{x}$ and $\mathbf{y}$ under consideration here are zero, and more importantly, equal to each other, in components other than those numbered 1, i, and $i+n$, then they contribute equally to both sides of the inequality and can be ignored. With $z_1 = 0$ from (8.5), we are left with

$$x_1 + |x_i - z_i|^p + |x_{i+n} - z_{i+n}|^p \leq y_1 + |y_i - z_i|^p + |y_{i+n} - z_{i+n}|^p . \tag{8.22}$$

Suppose $1 \leq p < 2$. Substituting (8.20) into (8.22) gives

$$\begin{aligned}
|1-z_i|^p + z_{i+n}^p &\leq 1 - 2^{1-p} + \left|\tfrac{1}{2} - z_i\right|^p + \left|\tfrac{1}{2} - z_{i+n}\right|^p \\
z_i^p + |1-z_{i+n}|^p &\leq 1 - 2^{1-p} + \left|\tfrac{1}{2} - z_i\right|^p + \left|\tfrac{1}{2} - z_{i+n}\right|^p \\
z_i^p + |1-z_i|^p + z_{i+n}^p + |1-z_{i+n}|^p &\leq 2\left[1 - 2^{1-p} + \left|\tfrac{1}{2} - z_i\right|^p + \left|\tfrac{1}{2} - z_{i+n}\right|^p\right].
\end{aligned}$$

From Lemma 8.7 we have

$$\begin{aligned}
z_i^p + |1-z_i|^p &\geq 1 - 2^{1-p} + \left|\tfrac{1}{2} - z_i\right|^p \\
z_{i+n}^p + |1-z_{i+n}|^p &\geq 1 - 2^{1-p} + \left|\tfrac{1}{2} - z_{i+n}\right|^p \\
z_i^p + |1-z_i|^p + z_{i+n}^p + |1-z_{i+n}|^p &\geq 2\left[1 - 2^{1-p} + \left|\tfrac{1}{2} - z_i\right|^p + \left|\tfrac{1}{2} - z_{i+n}\right|^p\right],
\end{aligned}$$

which is a contradiction unless z_i and z_{i+n} are both in $\{0,1\}$. Substituting each of the four combinations gives $z_i \in \{0,1\}$, $z_{i+n} = z_i$.

Suppose $p > 2$. As above, substituting (8.21) into (8.22) gives

$$\begin{aligned}
|1-z_i|^p + z_{i+n}^p &\geq 1 - 2^{1-p} + \left|\tfrac{1}{2} - z_i\right|^p + \left|\tfrac{1}{2} - z_{i+n}\right|^p \\
z_i^p + |1-z_{i+n}|^p &\geq 1 - 2^{1-p} + \left|\tfrac{1}{2} - z_i\right|^p + \left|\tfrac{1}{2} - z_{i+n}\right|^p \\
z_i^p + |1-z_i|^p + z_{i+n}^p + |1-z_{i+n}|^p &\geq 2\left[1 - 2^{1-p} + \left|\tfrac{1}{2} - z_i\right|^p + \left|\tfrac{1}{2} - z_{i+n}\right|^p\right],
\end{aligned}$$

and from Lemma 8.7 we have

$$\begin{aligned}
z_i^p + |1-z_i|^p &\leq 1 - 2^{1-p} + \left|\tfrac{1}{2} - z_i\right|^p \\
z_{i+n}^p + |1-z_{i+n}|^p &\leq 1 - 2^{1-p} + \left|\tfrac{1}{2} - z_{i+n}\right|^p \\
z_i^p + |1-z_i|^p + z_{i+n}^p + |1-z_{i+n}|^p &\leq 2\left[1 - 2^{1-p} + \left|\tfrac{1}{2} - z_i\right|^p + \left|\tfrac{1}{2} - z_{i+n}\right|^p\right],
\end{aligned}$$

which is a contradiction unless z_i and z_{i+n} are both in $\{0,1\}$. Substituting each of the four combinations gives $z_i \in \{0,1\}$, $z_{i+n} = z_i$. ■

At this point we have constrained $\mathbf{z}$ to a set of 2^n points representing truth assignments. The complexity result then follows easily.

Theorem 8.9
GHREVERSE is in $\mathcal{NPC}$ for real vectors with an L_p metric and finite $p \neq 2$.

Proof Starting with any instance of GHREVERSE on binary strings of length n with Hamming distance, we will construct an instance of GHREVERSE on $(2n+1)$-component vectors with the L_p metric. By Lemma 8.6 we can constrain

the components of a solution $\mathbf{z}$ to zero for the first component and the interval $[0,1]$ for the others. By Lemma 8.8 we can further constrain them to the set $\{0,1\}$, as well as forcing pairs of components z_i and $z_i + n$ to be equal to each other. The remaining possibilities are exactly the vectors that encode n-bit strings.

Furthermore, the L_p distance between any two of these points is a strictly increasing function (namely, the $1/p$ power) of the number of components that differ, which is twice the Hamming distance. Therefore the inequality $d(x,z) \leq d(y,z)$ on strings is unchanged if we replace all the strings with the vectors that encode them, and the Hamming distance with the L_p distance. So by encoding the strings in a pair from the input space, we get an equivalent pair for the output space. We encode all the pairs in the input instance and add them to the output instance, and the resulting instance of GHREVERSE on vectors with L_p is equivalent to the original instance of GHREVERSE on binary strings.

By inspection of the proofs, all these steps involve polynomial numbers of objects, processed a polynomial number of times, and can be conducted in time polynomial to the size of the initial instance. Only a small constant number of distinct real numbers are used as vector components, and they are well-behaved in the sense that we can write each on in a constant number of bits for fixed p. Therefore GHREVERSE on strings with Hamming distance, which is $\mathcal{NP}$-hard by Corollary 5.5, can be reduced in polynomial time to a polynomial-sized instance of GHREVERSE on vectors with the L_p metric, making GHREVERSE on vectors with the L_p metric $\mathcal{NP}$-hard, and the satisfying vector $\mathbf{z}$ is a polynomial-time certificate, so it is in $\mathcal{NP}$ and therefore $\mathcal{NP}$-complete. ∎

8.5 GHREVERSE with the L_∞ metric

For the L_∞ metric, the approach above does not work without modification. One problem is that the gadget used in Lemma 8.6 for limiting the values of an individual component becomes pathological in the L_∞ case and requires a more elaborate proof to show that the constraints still hold; the details are below. The gadget of Lemma 8.8 suffers a similar problem. The most serious problem is that distance between corners of a cube is no longer a strictly increasing function of L_1 distance; in L_∞, all the corners of a cube are equidistant from each other, and so the proof of Theorem 8.9 fails. So, as with VPREVERSE, we instead reduce directly to GHREVERSE from 3SAT, using an encoding based on the signs of the components instead of forcing the components to zero and one.

Theorem 8.10
GHREVERSE is in $\mathcal{NPC}$ for real vectors with L_∞.

Proof Starting with any arbitrary instance of 3SAT with n variables, we will construct an equivalent instance of GHREVERSE on $(n+1)$-component real vectors. Vectors will encode truth assignments as follows: the first component z_1 of a solution vector $\mathbf{z}$ will be zero, and the remaining components will correspond to the n variables in sequence with negative values corresponding to false and nonnegative values to true.

As in the finite-p case, we introduce pairs of points to limit the values of the components:

$$\begin{aligned}
\mathbf{x}_1 &= 2\mathbf{u}_1 & \mathbf{y}_1 &= -2\mathbf{u}_1 & \\
\mathbf{x}_2 &= -2\mathbf{u}_1 & \mathbf{y}_2 &= 2\mathbf{u}_1 & \\
\mathbf{x}_{2i+1} &= 1\mathbf{u}_i & \mathbf{y}_{2i+1} &= -3\mathbf{u}_i & 1 < i \leq m+1 \\
\mathbf{x}_{2i+2} &= -\mathbf{u}_i & \mathbf{y}_{2i+2} &= 3\mathbf{u}_i & 1 < i \leq m+1\,.
\end{aligned}$$

From the proof of Lemma 8.6 it should be clear that these pairs are intended to limit the component values to zero for z_1 and the interval $[-1,1]$ for all other z_i. However, in the case of the L_∞ metric, the "less than or equal" in the definition of GHREVERSE causes trouble; when the points in a pair are axis-aligned, as here, then the set of $\mathbf{z}$ equidistant from $\mathbf{x}$ and $\mathbf{y}$ is more than just a flat plane, as shown in Figure 8.7. For large enough radius, specifically radius $r \geq d(\mathbf{x},\mathbf{y})/2$, L_∞ spheres of radius r centred on $\mathbf{x}$ and $\mathbf{y}$ will partially coincide instead of merely intersecting.

The desired behaviour of the gadget is to limit one component of $\mathbf{z}$, say the j-th. As shown in the figure, we do have that behaviour, provided the maximum of the other components is not too large. As long as $\mathbf{z}$ remains in the regions labelled (a) and (b) in Figure 8.7, or on the hyperplane (c), the argument from Lemma 8.6 applicable to finite p also applies to L_∞. The conditions could fail if $\mathbf{z}$ is in the region labelled (d).

When $\mathbf{x}_i$ and $\mathbf{y}_i$ are both scalar multiples of some $\mathbf{u}_j$, that is, they differ in one component and are zero in all others, then the restriction actually enforced by the gadget is

$$z_j \leq \max\left\{\frac{x_{i,j}+y_{i,j}}{2}, x_{i,j} + \max_{1\leq k\leq 2n+1} |z_k|\right\} \text{ for } x_{i,j} < y_{i,j}, \text{ and} \tag{8.23}$$

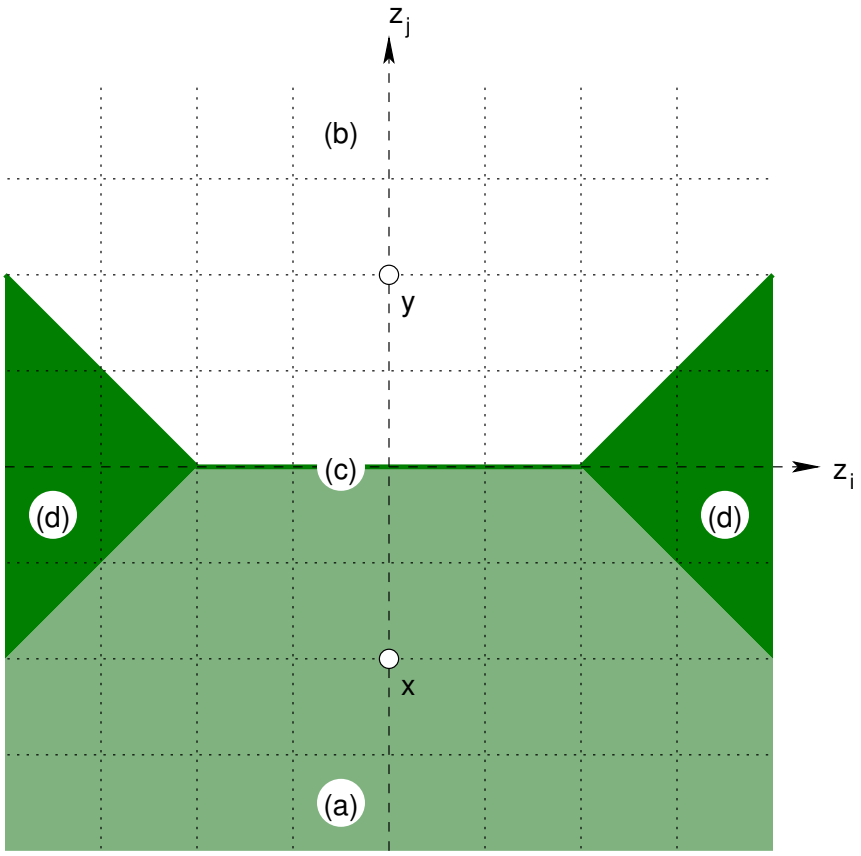


Figure 8.7: The gadget for limiting one component in L_∞ fails if some other component is too large.

$$z_j \geq \min\left\{\frac{x_{i,j}+y_{i,j}}{2}, x_{i,j} - \max_{1\leq k\leq 2n+1}|z_k|\right\} \text{ for } x_{i,j} > y_{i,j}\,. \tag{8.24}$$

As already mentioned, when $x_{i,j} = y_{i,j}$ there is no restriction.

Now, because we are using these pairs of points two at a time, symmetrically about the origin, we can turn (8.23) and (8.24) into a single constraint on the absolute value of z_j. For each z_j we have two pairs with $x_{i,j} = \pm 1$ and $y_{i,j} = \mp 3$ except z_1, which is restricted even further, so for all z_j we have

$$|z_j| \leq \max\left\{1, -1 + \max_{1\leq k\leq 2n+1}|z_k|\right\}.$$

In other words: if the absolute value of a component of $\mathbf{z}$ is more than 1, then it must be smaller by at least 1 than the greatest absolute value of any component. That would be a contradiction if any component had absolute value more than 1; there would have to be a maximum but none of them could be it. Therefore none of the components do have absolute value greater than 1. Figure 8.8 shows how this step works: the limitations on each component, despite being conditional on other components also being limited, combine to limit $\mathbf{z}$ to the desired values.

Now, for each clause in the 3SAT instance, let i, j, and k be the indices of the variables v_i, v_j, and v_k in the clause, and let l_i, l_j, and l_k be indicator variables, each equal to 1 if the corresponding variable is negated in the clause and -1 if it is not negated. Insert a pair $(\mathbf{x}, \mathbf{y})$ in the GHREVERSE instance with the vectors below. This gadget is shown in Figure 8.9, limited to two components for clarity.

$$\begin{aligned}\mathbf{x} &= 2\mathbf{u}_1\\ \mathbf{y} &= {}^3\!/_2\left(l_i\mathbf{u}_{i+1} + l_j\mathbf{u}_{j+1} + l_k\mathbf{u}_{k+1}\right).\end{aligned} \tag{8.25}$$

Since the components of $\mathbf{z}$ are all limited to the interval $[-1, 1]$, they can differ from the zero components of $\mathbf{x}$ by at most 1, and so the distance $d(\mathbf{x}, \mathbf{z})$ is solely determined by the first component, which is constant 0 for $\mathbf{z}$ and constant 2 for $\mathbf{x}$. Therefore $d(\mathbf{x}, \mathbf{z}) = 2$.

Any $\mathbf{z}$ corresponding to an assignment of truth values that failed to satisfy the clause, would have the same sign as $\mathbf{y}$ in each of the three components i, j, and k. So its distance from $\mathbf{y}$ along those axes could be at most ${}^3\!/_2$. Its distance from $\mathbf{y}$ along any other axis could be at most one; and so in all cases it would be closer to $\mathbf{y}$ than $\mathbf{x}$ with its constant distance of two, and $\mathbf{z}$ could not be a solution to the GHREVERSE instance.

Conversely, for any assignment of truth values that satisfied the clause, we could find a corresponding vector $\mathbf{z}$ by setting components to 1 for true variables and -1 for false variables, and the resulting $\mathbf{z}$ would differ from $\mathbf{y}$ by ${}^5\!/_2$ in at least one component (the component corresponding to the literal that satisfied the clause), making $d(\mathbf{y}, \mathbf{z}) > d(\mathbf{x}, \mathbf{z})$.

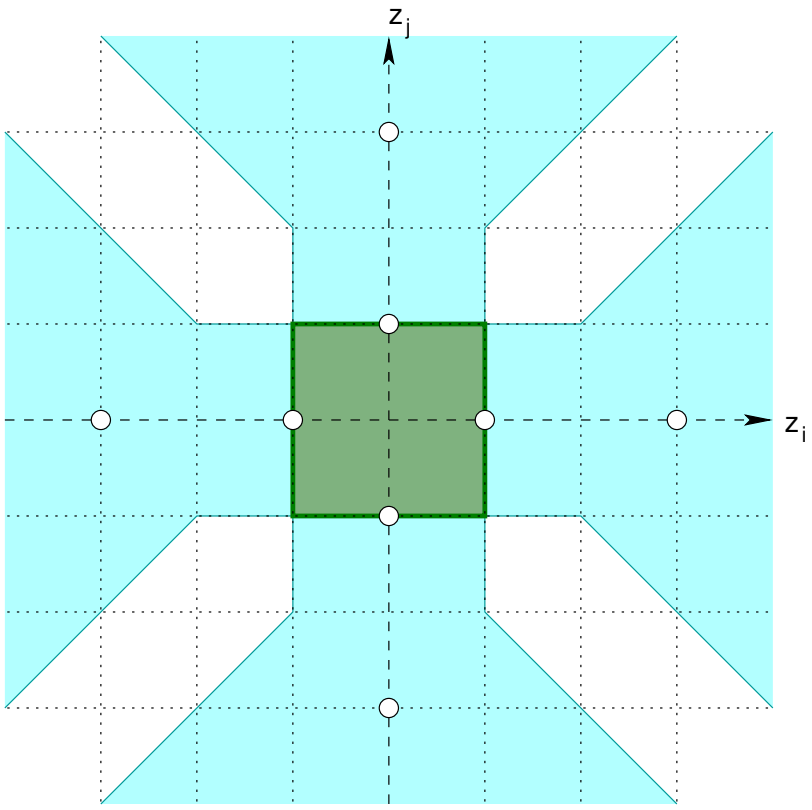


Figure 8.8: Multiple limiting gadgets support each other.

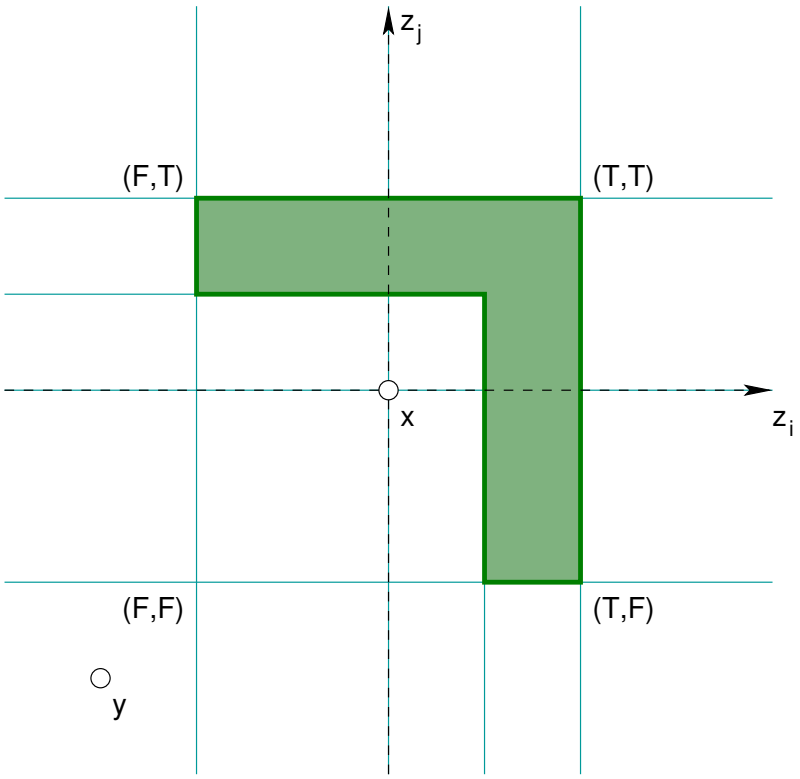


Figure 8.9: The gadget for clause satisfiability in L_∞.

Therefore with such a pair included for every clause in the 3SAT instance, the GHREVERSE instance has a solution if and only if the 3SAT instance has a solution, and the solution to the GHREVERSE instance is a polynomial-time certificate for the decision problem. Therefore GHREVERSE on real vectors with L_∞ is $\mathcal{NP}$-complete. ■

8.6 VPREVERSE with equal radii

Our results on VPREVERSE for finite-p L_p spaces involve reduction from the Hamming distance case studied by Frances and Litman [76]. But Frances and Litman did not study the fully general VPREVERSE problem described by Definition 1.27: their MCR problem (Definitions 5.5 and 5.6) is actually a special case of VPREVERSE, where all spheres are of the same type (solutions constrained to be either inside, or outside, all the spheres) and all radii are equal. In the case of Hamming distance, those special cases are as hard as the fully general problem. We might ask about similar special cases for VPREVERSE on L_p vectors.

The case where the solution must be outside all the spheres is trivially satisfiable because a solution vector's components may be arbitrarily large; more detail on that is given in the proof below. We do not study it in detail, but the case where a point must be inside all the spheres also seems likely to be easily decidable, because the set of solutions remains convex no matter how many spheres we add to the instance. The restriction that all spheres must have the same radius, with the solution constrained to be inside some of them and outside others, seems most interesting; the problem remains $\mathcal{NP}$-complete in that case by the following theorem.

Theorem 8.11
VPREVERSE with all radii equal is in $\mathcal{NPC}$ for real vectors with any L_p metric (including L_∞).

Proof The finite-p case. We will reduce from an arbitrary VPREVERSE instance on n-dimensional vectors with L_p to one on $n+1$-dimensional vectors with L_p and all spheres having the same radius. Note that if all the constraints in the input instance require a solution to be outside a sphere, then the instance is trivially satisfiable by choosing a sufficiently large vector; for example, one with any component equal to the sum of the absolute values of all the real numbers in the instance, plus one. In that case any satisfiable output instance will do.

Assuming there are some "inside" spheres, add a component which we will call z_{n+1} to all the vectors. Choose a fixed radius r larger than any radius in the input instance by a factor of more than $\sqrt[p]{n+1}$. Then replace each sphere in the input with a pair of spheres of the same type and of radius r, arranged symmetrically about the $z_{n+1} = 0$ hyperplane as shown by the top two gadgets in Figure 8.10, such that their intersection with each other and the $z_{n+1} = 0$ hyperplane is exactly the sphere from the input.

Any solution to the input instance corresponds to a solution to the output instance identical except for the inclusion of the extra component with value zero. In the other direction, because of the choice of r, the distance between the centres of the two spheres in each pair must be more than r. Then the presence of at least one inside sphere pair creates the constraint $|z_{n+1}| < r$ on any solution $\mathbf{z}$. With that constraint, for each pair the set of solutions permitted with $z_{n+1} \neq 0$ is a subset of the set of solutions permitted with $z_{n+1} = 0$. So no additional solutions to the underlying $\mathcal{NP}$-hard problem are permitted by this modified encoding. Therefore the theorem holds for finite p.

The infinite-p case. No changes to the encoding or vector lengths are needed. Start with the reduction proved in Theorem 8.4; note that it only uses two distinct radii, namely 1 and $3/2$, and the radius 1 is only used for the one sphere $(\mathbf{0}, 1, 1)$. Replace it with two spheres of radius $3/2$:

$$\left(\pm \frac{1}{2} \sum_{i=1}^{n} \mathbf{u}_i, \frac{3}{2}, 1 \right)$$

As shown for two dimensions by the bottom gadget in Figure 8.10, the two L_∞ spheres intersect to create the same effect as one L_∞ sphere, which is exactly the one we replaced. Therefore the theorem holds for L_∞. ∎

A further relaxation of the problem is possible: we can require all the radii to be equal and then leave the radius unspecified. That variation is considered for general metric spaces in the following chapter.

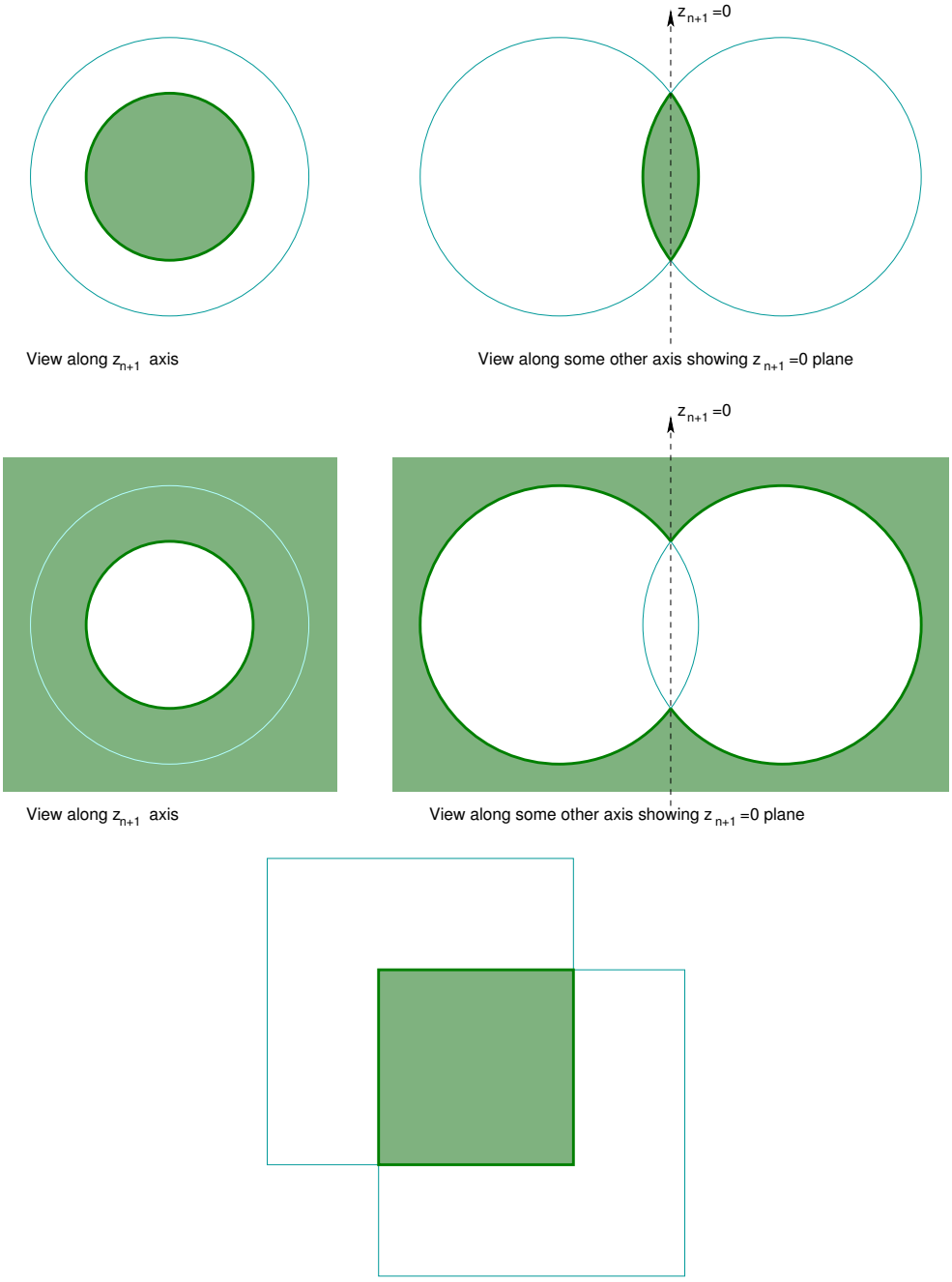


Figure 8.10: Gadgets used in proof of Theorem 8.11.

Chapter 9

Additional results

In this chapter we present a few additional results of interest that fall outside the structure of the main body of this work. In particular, we show that D_q dimensions are independent of the intrinsic dimensionality ρ; we comment on some other problems that reduce easily to GHREVERSE; we give distance permutation counts for some practical databases; and we discuss the question of distance permutations in hyperbolic spaces.

9.1 Independence of dimensionality measures

Recall that the intrinsic dimensionality ρ and the D_q dimensions both measure the dimensionality of a space by describing the probability distribution of distances among randomly chosen points. Intrinsic dimensionality measures the tendency for points in high-dimensional spaces to all be equidistant; it is defined as the mean squared divided by twice the variance of the distance between two randomly chosen points, a quantity which (as we have described in previous chapters) tends to increase as difficulty of distance-based indexing increases. That statistic is primarily concerned with the behaviour of typical points; the width of the main body of the probability distribution.

The D_q dimensions are based on the lower tail of the distribution—the behaviour of distances in the limit of small distance. In an ordinary Euclidean space, with locally uniform probability density, the density in a small neighbourhood will increase as some power (determined by the number of dimensions) of the radius of the neighbourhood. Attempting to find the power by which it increases, and using that as a measure of the number of dimensions, gives a spectrum of D_q

dimensions defined by

$$D_q = \frac{-1}{1-q} \lim_{\epsilon \to 0^+} \frac{\log \sum_{i=1}^{n} \Pr[x \in B_i]^q}{\log \epsilon} \tag{9.1}$$

where $\{B_1, B_2, \ldots, B_n\}$ is a minimal cover of the space by open balls of radius at most ϵ (Definition 1.22). For $q = 0$ we use the convention $0^0 = 0$, and for $q = 1$ we take the limit as q approaches 1. The D_q dimension is only meaningful for metrics that admit arbitrarily small values, though something less formal can be derived from attempts to fit a power-law curve to the discrete values of, for instance, Hamming distance among strings.

We can ask whether there is always a connection between these two measures of dimensionality, and the answer is that there is not. Although spaces where one is large seem to usually have the other large as well, it is possible to construct spaces with both measures chosen arbitrarily and independently of each other. First we give a construction for a generalised Cantor dust with arbitrary D_q dimension. Distributions like this one are standard examples in introductory works on fractals and dynamical systems, but usually from the point of view of calculating the dimensionality of a given distribution [162, 168]. We give a detailed description here to emphasise that we can go in the other direction, from any desired dimensionality to a distribution with that dimensionality.

Lemma 9.1
For any real $\delta > 0$, there exists a probability distribution on vectors with $\lceil \delta \rceil$ components, all in the interval $[0, 1]$, and L_1 distance, such that all the D_q dimensions of the distribution are equal to δ.

Proof Let $d = \lceil \delta \rceil$ and let F_0 (mnemonic: F for "fractal") be the distribution of d-component vectors with each component chosen uniformly from $[0, 1]$. Let λ be a real number, $0 < \lambda \leq 1/2$.

Define F_k as the distribution formed by choosing a vector from F_{k-1}, multiplying each component by λ, and then for each component independently, subtracting the component from 1 with $1/2$ probability. The first few steps of the construction are shown in Figure 9.1. We will choose a value of λ so that the limiting distribution of F_k as k goes to infinity, has all D_q dimensions equal to δ. Call that distribution F_∞. We could select uniformly from it by choosing a real number uniformly at random from $[0, 1)$ and using successive binary digits of its expansion to choose between the "high" and "low" halves of the distribution along each coordinate.

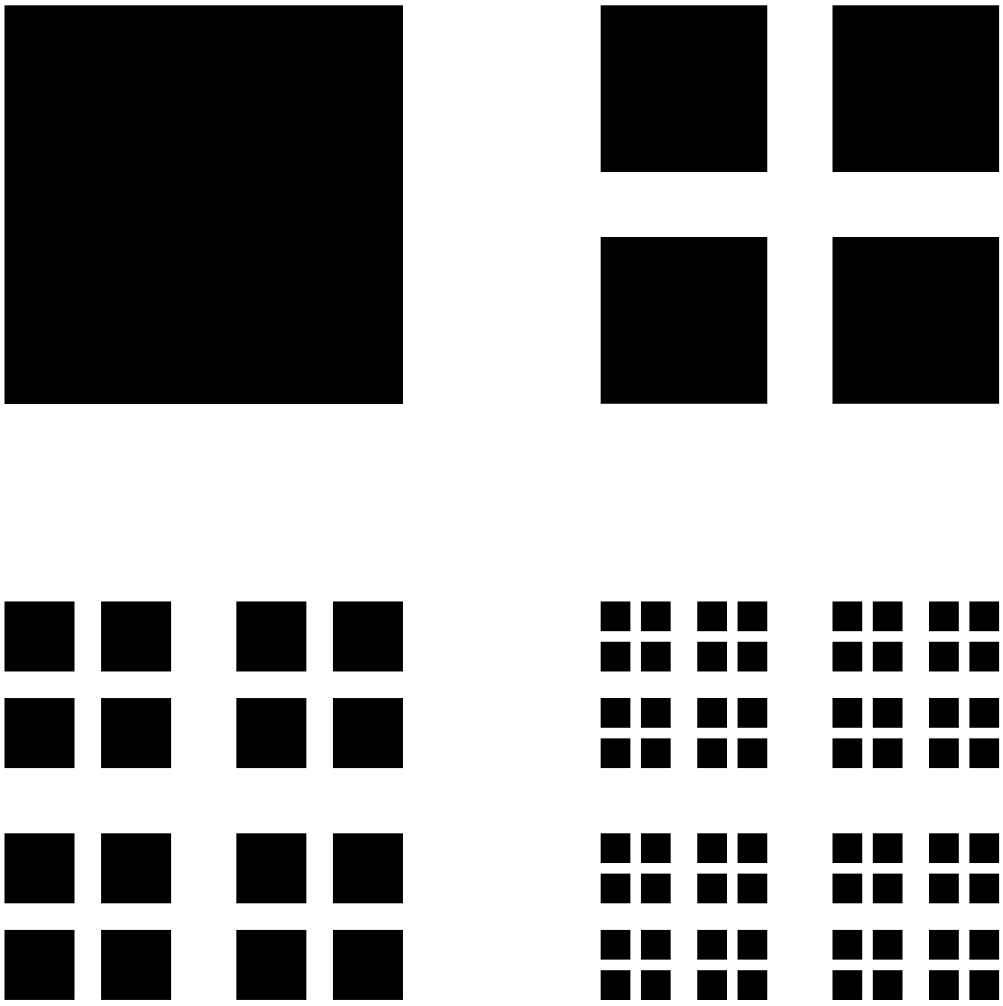


Figure 9.1: Constructing a distribution of arbitrary dimension: $d = 2$, $\lambda = 0.4$, $\delta \approx 1.513$.

Now, consider covering the distribution with balls of radius ϵ. For convenience, scale the vectors so that a ball of radius 1 exactly covers the unit cube F_0; we can do this because D_q dimensions are invariant under scaling (the constant multiplier disappears in the limit). If we limit consideration to values of ϵ that are integer powers of λ, which is also possible because of constant factors disappearing in the limit, then we have a situation where the distribution can be covered by 2^{kd} balls of radius λ^k, each of which contains 2^{-kd} probability. Then we can evaluate the D_q dimensionality as follows:

$$\begin{aligned} D_q &= \frac{-1}{1-q} \lim_{\epsilon \to 0^+} \frac{\log \sum_{i=1}^{n} \Pr[x \in B_i]^q}{\log \epsilon} \\ &= \frac{-1}{1-q} \lim_{\epsilon \to 0^+} \frac{\log \sum_{i=1}^{2^{kd}} 2^{-kdq}}{\log \lambda^k} \\ &= \frac{-1}{1-q} \lim_{\epsilon \to 0^+} \frac{\log 2^{kd(1-q)}}{\log \lambda^k} \\ &= -d \frac{1}{\lg \lambda}. \end{aligned}$$

Note that the value q cancels out; this value applies to all q. By choosing $\lambda = 2^{-d/\delta}$ we can force all the D_q dimensions equal to δ. ■

We can also construct a distribution of any arbitrary intrinsic dimensionality ρ; as described in Section 5.1, the intrinsic dimensionality of binary strings of length n where each bit is 1 with probability q (not the same q as in D_q dimensions), is $nq(1-q)/(1-2q+2q^2)$ by Corollary 2.2. We can choose n and q to achieve any desired value for that. The result follows that we can choose the two dimensionality measures independently.

Theorem 9.2
For any real $\delta, \rho > 0$ we can construct a space such that all the D_q dimensions are equal to δ and the intrinsic dimensionality is arbitrarily close to ρ.

Proof The space will be a space of real vectors with L_1 distance. Create a distribution with D_q dimensions all equal to δ by the construction of Lemma 9.1. Create a distribution with intrinsic dimensionality ρ by choosing a vector length n and making all components equal to 0 or 1 with probability q such that $\rho = nq(1-q)/(1-2q+2q^2)$ (as described above and in Section 5.1). If these

two distributions use vectors of different lengths, extend the shorter ones by appending zero components.

Now, choose a small constant c. Random points from our space will be points $c\mathbf{x} + \mathbf{y}$ where $\mathbf{x}$ is chosen from the distribution with D_q dimensions all equal to δ and $\mathbf{y}$ is chosen from the distribution with intrinsic dimensionality equal to ρ.

For small c, the effect of adding $c\mathbf{x}$ on the distance among randomly chosen pairs of points will be negligible; thus the intrinsic dimensionality of the combined distribution can be made arbitrarily close to ρ by choosing a small enough c. On the other hand, for sufficiently small ϵ, the effect of $\mathbf{y}$ is invisible to the limit that defines D_q dimension; at small enough scales, this space looks exactly like our space with all D_q dimensions equal to δ. Thus we can construct a space for arbitrary, independent, D_q and intrinsic dimensionality. ■

9.2 Other problems that reduce to GHREVERSE

In Section 8.6 we considered the special case of VPREVERSE where all the radii are equal: that is, a decision problem in which an instance consists of a radius r and two sets of points X and Y, and a solution z must be a point within radius r of all the points in X and not within radius r of any of the points in Y. The original definition of VPREVERSE, with the added condition of all radii being equal, is equivalent to this definition; the sets X and Y correspond to the values of the bit b in Definition 1.27. By treating z as the centre of a sphere, that question is also equivalent to asking whether there exists of a sphere of radius r containing all the points in X and none of the points in Y.

A further relaxation is possible: we can require all the radii to be equal and then leave the radius unspecified. The instance then consists only of a list of centres and bits specifying inside or outside, and it is satisfiable if there exists any positive radius r for which the corresponding VPREVERSE instance, with all radii set to r, would be satisfied. This relaxation is drastic enough to warrant formal definition as a new problem in itself, below.

Definition 9.1 (The VPRU (VPREVERSE, unspecified radius) Problem)
In some metric space (S, d), given a set P of ordered triples (x, b) with $x \in S$ and $b \in \{0, 1\}$, accept if and only if there exists a point $z \in S$ and real $r > 0$ such that for every $(x, b) \in P$, $d(x, z) \leq r$ if and only if $b = 1$.

An equivalent definition is that a VPRU instance consists of two sets of points X and Y and is satisfiable if there exists a sphere (of any radius, which is the difference from equal-raidus VPREVERSE) containing all of X and none of Y. The VPRU problem can be reduced to GHREVERSE. So can two other problems we define now.

Definition 9.2 (The Generalised GHREVERSE Problem)
In some metric space (S, d), given a set P of ordered pairs (x_i, Y_i) where $x_i \in S$ and $Y_i \subseteq S$, with $n = |P|$, accept if and only if there exists a point $z \in S$ such that $d(z, x_i) \leq d(z, y)$ for every $y \in Y_i, (x_i, Y_i) \in P$.

Definition 9.3 (The DPREVERSE Problem)
In some metric space (S, d), given a sequence of points $\langle x_1, x_2, \ldots, x_n \rangle$ and a permutation $\pi : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$, accept if and only if there exists a point $z \in S$ such that π is the distance permutation of z with respect to $\langle x_1, x_2, \ldots, x_n \rangle$; that is, if whenever $i < j$ then $d(x_{\pi(i)}, z) < d(x_{\pi(j)}, z)$ or $d(x_{\pi(i)}, z) = d(x_{\pi(j)}, z)$ and $\pi(i) < \pi(j)$.

These two problems are motivated by further examination of indexing structures, much in the same way as the original VPREVERSE and GHREVERSE problems. Generalised GHREVERSE corresponds to a generalised GH-tree, with possibly more than two vantage points in each internal node; the ordinary GH-tree is then the special case where each internal node contains exactly two vantage points. Such trees are used, for instance, by the `ght` index type in the SISAP library of Figueroa, Navarro, and Chávez [71]. The DPREVERSE problem corresponds to a distance permutation-based index, forming a link between our work on distance permutations and on reverse similarity search. Just as VPREVERSE is the problem of generating a point that would be stored in a particular leaf in a VP-tree, DPREVERSE is the problem of finding a point that would be given a particular distance permutation in a distance permutation index.

All three of Definitions 9.1–9.3 can be reduced to the original GHREVERSE problem of Definition 1.29. The concept is very simple: in all three problems, the conditions for a satisfying point are inequalities on the distance to points from the problem, and GHREVERSE can express any such set of inequalities.

Theorem 9.3
For any metric space, the following polynomial-time reductions hold among decision problems in the space:

$$\text{DPREVERSE} \Rightarrow \text{generalized GHREVERSE} \Longleftrightarrow \text{GHREVERSE} \Leftarrow \text{VPRU}. \tag{9.2}$$

Proof Any arbitrary instance of DPREVERSE can be expressed as a single constraint on the solution z: its distance permutation Π_z must be equal to π. That can be split into k constraints $\Pi_z(1) = \pi(1), \Pi_z(2) = \pi(2), \ldots, \Pi_z(k) = \pi(k)$. But

each of those is equivalent to a constraint of the form used in a generalised GHREVERSE instance: the closest site among all k sites must be $x_{\pi(1)}$, the closest among the remaining $k-1$ sites must be $x_{\pi(2)}$, and so on. Thus we can replace the DPREVERSE instance with $k-1$ generalised GHREVERSE constraints (no constraint is needed for the last equality, which involves the closest among a singleton set of one vantage point) and obtain an equivalent generalised GHREVERSE instance of polynomial size.

Any arbitrary instance of generalised GHREVERSE consists of constraints of the form "$d(x,z) \leq d(y_i,z)$ for all $y_i \in Y$." Each such constraint can be split into $|Y|$ number of constraints of the form $d(x,z) \leq d(y,z)$; by doing so, we can convert a generalised GHREVERSE instance to a polynomial-sized equivalent instance of the basic GHREVERSE problem. The reduction in the other direction is trivial, by changing all the points y into singleton sets $\{y\}$.

Consider an arbitrary instance of the VPRU problem. For a point z to be a solution means that there is some radius r such that z is within distance r of all the points in one set (call it N for "near"; $N = \{x|(x,1) \in P\}$), and not within distance r of all the points in another set (call it F for "far"; $F = \{x|(x,0) \in P\}$). In other words, for each pair of points (x,y) such that $x \in N, y \in F$, z is nearer to x than to y. Those pairs of points are exactly the pairs of points in an equivalent GHREVERSE instance. Given a satisfying point z for the GHREVERSE instance, we can find a solution to the VPRU instance just by using that point and choosing an r greater than the distance from z to any point in N but less than its distance to any point in F. ■

It is not necessarily possible, and at least not trivial, to reduce in the other direction for the reductions where Theorem 9.3 does not show equivalence. The difficulty can be demonstrated by imagining an instance carried all the way from one end of (9.2) to the other. If we start with a distance permutation, it naturally converts to a set of less-than-or-equal constraints on distance restricting the distances to vantage points to a total order. But a single VPRU instance is limited to two sets of distances, and enforces constraints pairwise between elements of the two sets but no constraints within a set. Short of transforming the problem into a completely different space, no equivalence seems possible. In the other direction, it seems impossible to express the permissiveness of a VPRU instance in the form of a distance permutation: VPRU should accept any of the possible permutations among (for instance) vantage points that appear only as inside centres, but DPREVERSE would require a separate instance for each permutation, apparently blowing up to factorially many instances.

9.3 Distance permutations in practical databases

In Section 3.5 we presented some results on the number of distance permutations actually occurring in randomly-generated databases of vectors. As part of the same series of experiments, which were also presented at SISAP'08 [191], we counted the distance permutations in the sample databases provided with the SISAP library [71]. The results are shown in Tables 9.1 and 9.2. Note that the columns labelled n report the number of points in each database; not number of components in a vector, as we have usually used n in this work.

The language databases (`Dutch`, `English`, `French`, `German`, `Italian`, `Norwegian`, and `Spanish`) are dictionaries "of unknown source" [71] with the Levenshtein distance as described in Chapter 6. The `listeria` database is similarly a database of strings with Levenshtein distance, but here the strings represent DNA sequences of genes from the Gram-positive bacterium *Listeria monocytogenes*, on the alphabet $\{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$ [32]. The `long` and `short` databases represent a collection of news articles as vectors with cosine distance, as described by Figueroa, Navarro, and Chávez in the library documentation [71]. Finally, the `colors` and `nasa` databases are vectors with Euclidean distance, having 112 and 20 components respectively. The `colors` database represents "color histograms... from an image database" provided by Seidl [71, 185]. The `nasa` database represents feature vectors from images in the NASA archives, provided by Katayama and Satoh as part of the Sixth DIMACS Implementation Challenge [120].

For each database we tested each number of sites k from 2 to 12. The site selection code was inherited from the library's `pivots` index type, which chooses the first k sites of the database as the sites. Since each database was supplied in randomised order, this selection is equivalent to choosing sites uniformly at random from the points in the database. It does mean that a choice of $k+1$ sites will always start with the same sites chosen for a choice of k sites; as a result, we can expect some correlation among columns of Tables 9.1 and 9.2 in that if one column (because of a lucky or unlucky choice of sites) happens to produce a count significantly more or less than the mean, then all subsequent columns probably will also.

Even in this very simple experiment, however, some interesting results are evident. First, all the numbers are surprisingly small. For instance, the `long` sample database has only 261 distinct distance permutations for $k = 12$, compared to $12! = 479001600$ general permutations of 12 options. The small number of points in that database may be a major cause, but similar effects show up in other databases too. For instance, the `colors` database only has 4408 distance permutations for $k = 12$, even though with 112544 points there are on average

			k:					
database	n	ρ	3	4	5	6	7	8
Dutch	229328	7.159	6	24	119	577	2693	11566
English	69069	8.492	6	24	120	645	2211	7140
French	138257	10.510	6	24	118	475	2163	8118
German	75086	7.383	6	24	119	517	1639	4839
Italian	116879	10.436	6	24	120	653	3103	10872
Norwegian	85637	5.503	6	24	118	632	2530	7594
Spanish	86061	8.722	6	24	118	598	2048	5428
listeria	20660	0.894	4	11	19	29	49	85
long	1265	2.603	5	10	22	47	51	98
short	25276	808.739	6	24	111	508	2104	6993
colors	112544	2.745	6	18	44	96	200	365
nasa	40150	5.186	6	24	115	530	1820	3792

Table 9.1: Distance permutations for sample databases.

			k:			
database	n	ρ	9	10	11	12
Dutch	229328	7.159	34954	74954	116817	163129
English	69069	8.492	16212	28271	38289	45744
French	138257	10.510	19785	35903	58453	81006
German	75086	7.383	10154	19489	30347	43208
Italian	116879	10.436	27843	45754	71921	90316
Norwegian	85637	5.503	15147	25872	42992	57988
Spanish	86061	8.722	13357	23157	39443	54628
listeria	20660	0.894	206	510	952	1145
long	1265	2.603	114	163	252	261
short	25276	808.739	13792	20223	23102	23940
colors	112544	2.745	796	1563	2800	4408
nasa	40150	5.186	7577	13243	19066	24154

Table 9.2: Distance permutations for sample databases (continued).

more than 25 points per permutation. That can be compared with the last column of Table 3.3, showing distance permutations for vectors: we did not go as far as 20-dimensional vectors, and even at the 10-dimension level the count was being limited by the database size of 10^6, but it is clear that a 20-dimensional uniform distribution would give far more than 4408 distance permutations. So the low distance permutation count for `colors` reflects some fact about the arrangement of points in that database, which is less complicated (in some sense, lower-dimensional) than a similar-sized database chosen from a uniform distribution.

We added intrinsic dimensionality results to the experiment in response to comments from the anonymous SISAP referees. For each sample database, we chose 10^7 pairs of points uniformly at random from each database and computed the intrinsic dimensionality as the mean squared divided by twice the variance of the distance between the points. The results are shown in Tables 9.1 and 9.2 in the columns labelled ρ.

Intrinsic dimensionality (Subsection 1.3.2) is primarily a function of the native distribution of the space. Different kinds of distribution can produce qualitatively different behaviour for ρ; for instance, as described in Section 2.1, the intrinsic dimensionality of vectors with L_∞ can be linear or log-squared for uniform or normal components respectively. The number of distance permutations, on the other hand, is determined solely by the sites and the set of points that can exist (essentially, the support of the native distribution). Thus a comparison between distance permutations and intrinsic dimensionality may not be entirely meaningful: we can change intrinsic dimensionality arbitrarily by changing the distribution, without changing the distance permutations.

Nonetheless, we can form some sort of comparison by assuming a distribution. Note that there are two distributions relevant here: the one that generated the database, and the one for selecting points from the database. It seems reasonable to select points uniformly from the the database, so that both distributions are the same relative to the space. That is the assumption used for the ρ columns in our tables, and it leads to an evident relationship between distance permutations and ρ. Spaces with large ρ seem to have more distance permutations in general. Also, even without a comparison to ρ, it seems clear that higher-dimensional spaces tend to have more distance permutations; so distance permutations may be useful as another way of expressing the dimensionality of a space. We can compare the results from Tables 9.1 and 9.2 with the Euclidean theoretical bounds from Table 3.1 to estimate the dimensionality of an Euclidean space equivalent to each of our databases.

In particular, `colors` has a few more distance permutations than we would expect from a Euclidean space with two dimensions. Similarly, `nasa` has a few

more distance permutations than a Euclidean space with three dimensions, if we ignore the values for $k > 10$ where the number of points in the database seem to limit the permutations. The results from the dictionary databases are variable, but in general they seem comparable to Euclidean spaces of up to four dimensions. The document databases (`long` and `short`) are problematic: with `long` there are so few points that the number of points limits the number of distance permutations before we get to a reasonable number of sites, and with `short`, it appears that very many pairs of points are at maximum distance (leading to the huge measured value of ρ). That may impair the counting of distance permutations. The `listeria` database, despite having plenty of points, seems equivalent to a Euclidean space with between one and two dimensions.

Note that because distance permutations depend only on the ordering of distances, not their magnitudes, the number of distance permutations is invariant not only under scaling (like intrinsic dimensionality) but also under monotonic functions of distance. For instance, if we take the square of a metric (as we might do for instance with Euclidean distance, to avoid expensive computation of square roots), the resulting distance function will produce a different intrinsic dimensionality; but distance permutations will be unchanged because the square function is monotonic. This effect seems to make distance permutations especially robust to changes in representation.

9.4 Distance permutations in hyperbolic space

Hyperbolic spaces occur in the study of non-Euclidean geometry. Whereas in the Euclidean plane a line and a point not on the line define exactly one line parallel to the first and passing through the line, and in the projective plane all lines intersect, in the hyperbolic plane a given point and line define an infinite number of distinct lines passing through the point and failing to intersect the given line. Higher-dimensional hyperbolic spaces can be defined analogously. Anderson gives a detailed introduction to hyperbolic space and its properties [8].

One important intuitive description is that at greater distances in hyperbolic space, the space expands. The circumference of a circle in the hyperbolic plane, for instance, does not increase linearly with radius as in the Euclidean plane, but exponentially. One demonstration of the expansion uses crochet, building up a physical model of the space in expanding rings of stitches [214].

Hyperbolic spaces are metric spaces and just as we did in other metric spaces, we can define a bisector as the set of points equidistant from two sites (Definition 3.3), and the distance permutation of a point y with respect to some sites as the permutation that sorts them into increasing order of distance from y (Definition 1.25). Then we can ask how many distinct distance permutations can

be generated by k sites in n-dimensional hyperbolic space.

One convenient model for hyperbolic space is the Poincaré disc (or, generalised to higher dimensions, Poincaré ball) model, where points are described by vectors of (Euclidean) length less than 1 with this metric:

$$d(\mathbf{x},\mathbf{y}) = \operatorname{arc\,cosh}\left(1 + \frac{2|\mathbf{x}-\mathbf{y}|^2}{(1-|\mathbf{x}|^2)(1-|\mathbf{y}|^2)}\right).$$

In the Poincaré model, straight lines in the two-dimensional version take the form of diameters, or Euclidean circles meeting the edge of the disc (which corresponds to infinity in the hyperbolic space) at right angles; and in higher-dimensional versions, flat hyperplanes in the hyperbolic space take the form of Euclidean hyperplanes through the centre, or spherical caps meeting the boundary at right angles. It is intuitively clear (we omit a formal proof, but it follows from the more detailed presentation given by Anderson [8]) that a bisector in this model must be one of those flat hyperplanes. If the two sites happen to be opposite vectors, then their Euclidean and hyperbolic bisectors coincide as the hyperplane passing through the centre and normal to both vectors, and by symmetry, any other pair of sites must also have a flat hyperbolic hyperplane as their bisector because we should be able to choose a centre for the model arbitrarily.

With k sites in general position there must always be at least $\binom{k}{2}+1$ permutations; that is the number created by $\binom{k}{2}$ bisectors that fail to intersect each other at all. In Euclidean space that can only happen if all the bisectors are parallel; a degenerate case for $n > 1$, corresponding to all the sites being on a line. However, in hyperbolic space this lower bound is achievable with points in general position; we can put the points on a line so that their bisectors do not intersect, then perturb the points within small neighbourhoods and keep the bisectors non-intersecting. Such a case (for $n = 2$ and $k = 4$) is shown in Figure 9.2 at upper left.

As shown in the figure, many other cases are also possible. Recall that in the Euclidean plane, four sites can generate up to 18 distance permutations and four sites in general position always generate exactly 18. In Figure 9.2 we display numbers of distance permutations ranging from 7 to 18, all achieved with four points in general position.

It appears that we can always achieve at least as many distance permutations in hyperbolic space as in Euclidean space by starting with a set of sites from Euclidean space and scaling it down to bring the points sufficiently close together. Such a case is shown at lower right in Figure 9.2. In a small enough neighbourhood, the hyperbolic metric becomes arbitrarily close to the Euclidean metric, so by making all the bounded cells of the generalised Voronoi diagram small enough, we can have as many cells in hyperbolic space as we would in Euclidean

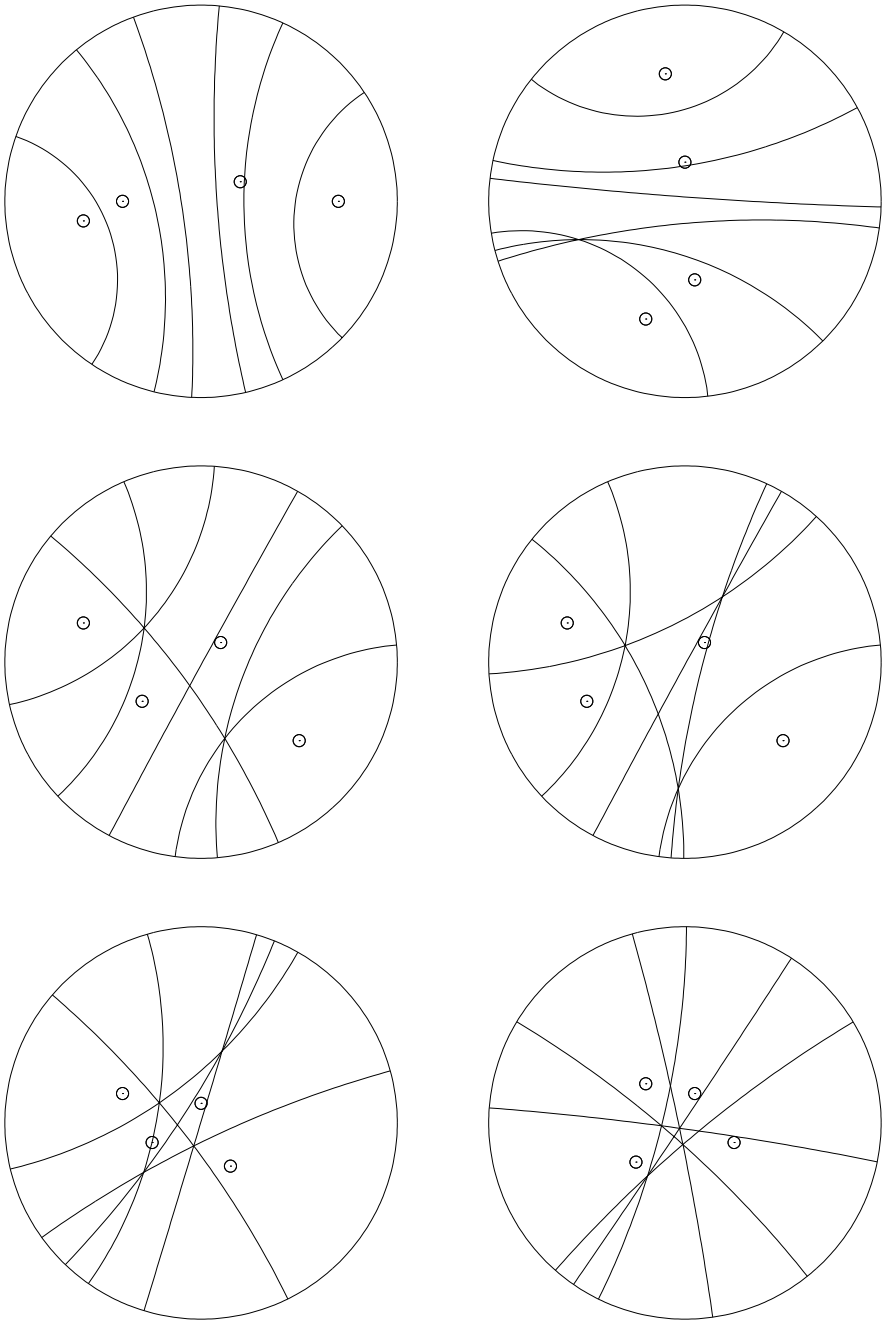


Figure 9.2: Four-point bisector systems with between 7 and 18 regions on the Poincaré disc.

space. As the example of nearly-collinear points shows, however, we can also have many fewer. We have found no examples of more distance permutations in hyperbolic space than would be possible with the same number of sites and dimensions in Euclidean space, suggesting that perhaps the Euclidean count given by Theorem 3.2 is also an upper bound for hyperbolic space.

Chapter 10

Conclusion

We have described three computer science questions relating to metric spaces: the questions of intrinsic dimensionality; number of distance permutations; and complexity of similarity search. We have explored each of these questions with theoretical and experimental work on a variety of spaces, including real vectors with the Minkowski L_p metrics; weighted and unweighted tree metrics including the prefix distance on strings; strings with Hamming distance; strings with Levenshtein distance; and strings with a new distance we introduce called Superghost distance.

For intrinsic dimensionality, we have discussed its definition and relationship to other measures of dimensionality, as well as its application to measuring the difficulty of building distance-based index data structures. We have analysed its asymptotic behaviour in a variety of spaces and distributions, both with theoretical and experimental work. For vectors uniform in the unit cube, we have the counterintuitive result that intrinsic dimensionality for the L_∞ metric approaches an asymptotic line approximately equivalent to the line for $L_{5.778}$, much different from the lines for L_p with large p. For strings with different kinds of edit distance, we have found intrinsic dimensionality varying from linear to approximately $\Theta(n^{5/4})$ to $\Omega(n^2/\log^2 n)$.

We have given exact results for the number of distance permutations that occur in Euclidean space, asymptotic bounds for many other spaces, and experimental results on real-life and randomly-generated databases. These bounds improve the best previous limits on storage space for an index data structure based on distance permutations. Where the concept can be meaningfully defined, we have also given results on the number of neighbours for each point. In addition to their application to index data structures, these distance permutation results implicate questions in combinatorial geometry and reveal theoretical differences among spaces.

The reverse similarity search problems VPREVERSE and GHREVERSE also reveal theoretical differences among spaces, in addition to having practical application to the security of robust hash schemes. After describing these problems in detail, we analyse their complexity in all our study spaces, showing that they are $\mathcal{NP}$-complete for most spaces but with polynomial-time cases showing that some spaces are special. For instance, in Euclidean space GHREVERSE is equivalent to linear programming. In tree spaces the problems are generally easy, but can be made equivalent to apparently difficult problems by special construction of the space. We also show equivalences among these problems, variations on them, and problems connected with distance permutations.

Future work on intrinsic dimensionality could include theoretical results on other spaces and the cases not addressed here; connections between intrinsic dimensionality and the practical considerations of indexing difficulty; practical studies of the intrinsic dimensionality occurring in real-life databases; and other ways to measure dimensionality of spaces and their native distributions. For distance permutations, many theoretical questions about the number of permutations and their structure (possibly as an oriented matroid) remain open. There is also room for more experimental work on how many occur in real databases, and how best to apply distance permutations to indexing. The reverse similarity search problems, as novel $\mathcal{NP}$-complete problems, admit to the same kinds of studies made for problems like 3SAT, including difficulty and phase transition behaviour for random instances; good heuristic algorithms for real cases; the possibility of approximation algorithms; and the original application to robust hash. Since it may be difficult to randomly generate known hard instances of $\mathcal{NP}$-complete problems in general, another area for further work is the question of what problems would be well-suited to the original application to robust hash in general metric spaces.

Bibliography

The numbers at the end of each entry list pages where the reference was cited. In the electronic version, they are clickable links to the pages.

[1] Milton Abramowitz and Irene A. Stegun, editors. *Handbook of Mathematical Functions*. [U.S.] National Bureau of Standards, 1964. Tenth printing, 1972. 13, 218, 220, 226

[2] George W. Adamson and Jillian Boreham. The use of an association measure based on character structure to identify semantically related pairs of words and document titles. *Information Storage and Retrieval*, 10(7–8):253–260, July–August 1974. 6

[3] Richa Agarwala, Vineet Bafna, Martin Farach, Babu Narayanan, Mike Paterson, and Mikkel Thorup. On the approximability of numerical taxonomy (fitting distances by tree metrics). In SODA'96 [195], pages 365–372. 99

[4] Noga Alon, Richard M. Karp, David Peleg, and Douglas B. West. A graph-theoretic game and its application to the k-server problem. *SIAM Journal on Computing*, 24(1):78–100, 1995. 99, 102

[5] Stephen F. Altschul, Warren Gish, Webb Miller, Eugene W. Myers, and David J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990. 132

[6] Carme Àlvarez and Maria J. Serna, editors. *Experimental and Efficient Algorithms: 5th International Workshop (WEA'06)*, volume 4007 of *Lecture Notes in Computer Science*, Cala Galdana, Menorca, Spain, May 24–27, 2006. Springer. 219

[7] Laurent Amsaleg and Patrick Gros. Content-based retrieval using local descriptors: Problems and issues from a database perspective. *Pattern Analysis Applications*, 4(2–3):108–124, 2001. 5

[8] James W. Anderson. *Hyperbolic Geometry*. Springer, 1999. 207, 208

[9] George Arfken. *Mathematical Methods for Physicists*. Academic Press, London, third edition, 1985. 13

[10] A. V. Arkhangel'skiĭ and V. V. Fedorchuk. The basic concepts and constructions of general topology. In Arkhangel'skiĭ and Pontryagin [11], part I. 2, 4, 21

[11] A. V. Arkhangel'skiĭ and L. S. Pontryagin, editors. *General Topology I*, volume 17 of *Encyclopaedia of Mathematical Sciences*. Springer, 1990. 214, 219

[12] V. L. Arlazarov, E. A. Dinic, M. A. Kronrod, and I. A. Faradžev. On economical construction of the transitive closure of a directed graph. *Soviet Mathematics—Doklady*, 11(5):1209–1210, 1970. English translation of *Doklady Akademii Nauk SSSR*, 194(3):487–488, 1970. 132

[13] Barry C. Arnold, N. Balakrishnan, and H. N. Nagaraja. *A First Course in Order Statistics*. Wiley, 1992. 15, 42, 46, 47, 48, 53, 56, 57

[14] Richard Arratia and Michael S. Waterman. A phase transition for the score in matching random sequences allowing deletions. *Annals of Applied Probability*, 4(1):200–225, February 1994. 134

[15] Franz Aurenhammer. Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Computing Surveys, Sept 1991*, 23(3), 1991. 78

[16] David A. Bader and Ashfaq A. Khokhar, editors. *Proceedings of the ISCA 17th International Conference on Parallel and Distributed Computing Systems (ISCA'04)*, San Francisco, California, USA, September 15–17, 2004. 217

[17] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, October15, 1999. 28

[18] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In CRYPTO'01 [122], pages 1–18. 102

[19] Yair Bartal. Probabilistic approximations of metric spaces and its algorithmic applications. In FOCS'96 [73], pages 184–193. 102

[20] Yair Bartal. On approximating arbitrary metrics by tree metrics. In STOC'98 [202], pages 161–168. 102

[21] Alexander Basilevsky. *Statistical Factor Analysis and Related Methods: Theory and Applications*. Wiley, 1994. 59

[22] Tuğkan Batu, Funda Ergun, and Cenk Sahinalp. Oblivious string embeddings and edit distance approximations. In SODA'06 [197], pages 792–801. 133

[23] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The R*-tree: An efficient and robust access method for points and rectangles. In SIGMOD'90 [81], pages 322–331. 8

[24] Richard Bellman. *Adaptive Control Processes: a guided tour*, page 94. Princeton University Press, Princeton, New Jersey, USA, 1961. 19

[25] Charles H. Bennett, Péter Gács, Ming Li, Paul M. B. Vitányi, and Wojciech H. Zurek. Information distance. *IEEE Transactions on Information Theory*, 44(4):1407–1423, 1998. 6

[26] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, September 1975. 7

[27] Stefan Berchtold, Christian Böhm, and Hans-Peter Kriegel. The pyramid-technique: Towards breaking the curse of dimensionality. In SIGMOD'98 [96], pages 142–153. 8, 12, 71

[28] Elwyn R. Berlekamp, John H. Conway, and Richard K. Guy. *Winning Ways for Your Mathematical Plays*. Academic Press, London, 1982. 3, 153

[29] Anders Björner, Michel Las Vergnas, Bernd Sturmfels, Neil White, and Günter M. Ziegler. *Oriented Matroids*. Cambridge University Press, second edition, 1999. 81

[30] Hilary S. Booth, Shevarl F. MacNamara, Ole M. Nielsen, and Susan R. Wilson. An iterative approach to determining the length of the longest common subsequence of two strings. *Methodology and Computing in Applied Probability*, 6(4):401–421, December 2004. 133, 134

[31] Sergey Brin. Near neighbor search in large metric spaces. In VLDB'95 [58], pages 574–584. 12

[32] Broad Institute of Harvard and MIT. *Listeria monocytogenes Sequencing Project*. Online http://www.broad.mit.edu/. 204

[33] Adam L. Buchsbaum and Jack Snoeyink, editors. *Algorithm Engineering and Experimentation: Third International Workshop (ALENEX'01)*, volume 2153 of *Lecture Notes in Computer Science*, Washington, D.C., USA, January 5–6, 2001. Springer. 216

[34] Peter Buneman. A note on the metric properties of trees. *Journal of Combinatorial Theory, Series B*, 17:48–50, 1974. 101

[35] W. A. Burkhard and R. M. Keller. Some approaches to best-match file searching. *Communications of the ACM*, 16(4):230–236, April 1973. 8

[36] Christian Cachin and Jan Camenisch, editors. *Advances in Cryptology: International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT'04)*, volume 3027 of *Lecture Notes in Computer Science*, Interlaken, Switzerland, May 2–6, 2004. Springer. 225

[37] *Proceedings of the 11th Canadian Conference on Computational Geometry (CCCG'99)*, Vancouver, B.C., August 15–18, 1999. UBC. 222

[38] Kaushik Chakrabarti and Sharad Mehrotra. The hybrid tree: An index structure for high dimensional feature spaces. In ICDE'99 [105], pages 440–447. 8

[39] Edgar Chávez, Karina Figueroa, and Gonzalo Navarro. Proximity searching in high dimensional spaces with a proximity preserving order. In MICAI'05 [84], pages 405–414. 33, 34

[40] Edgar Chávez and Gonzalo Navarro. Measuring the dimensionality of general metric spaces. Technical Report TR/DCC-00-1, Department of Computer Science, University of Chile, 2000. Submitted. Online ftp://ftp.dcc.uchile.cl/pub/users/gnavarro/metricmodel.ps.gz. 26, 27, 28, 41, 44, 45, 51, 68

[41] Edgar Chávez and Gonzalo Navarro. A probabilistic spell for the curse of dimensionality. In ALENEX'01 [33], pages 147–160. 19

[42] Edgar Chávez, Gonzalo Navarro, Ricardo Baeza-Yates, and José Luis Marroquín. Searching in metric spaces. *ACM Computing Surveys*, 33(3):273–321, September 2001. 4, 7

[43] Weidong Chen, Jeffrey F. Naughton, and Philip A. Bernstein, editors. *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data (SIGMOD'00)*, Dallas, Texas, USA, May 16–18, 2000. ACM. 224

[44] L. Paul Chew and Robert L. (Scot) Drysdale, III. Voronoi diagrams based on convex distance functions. In SCG'85 [182], pages 235–244. 44

[45] Vacláv Chvátal and David Sankoff. Longest common subsequences of two random sequences. *Journal of Applied Probability*, 12(2):306–315, June 1975. 133, 134, 135

[46] *Proceedings of the 40th Annual Conference on Information Sciences and Systems (CISS'06)*, Princeton University, New Jersey, USA, March 2006. IEEE. 217

[47] Communications Security Establishment. *Proceedings of the 10th Annual Canadian Information Technology Security Symposium (CITSS'98)*, Ottawa, Ontario, June 1–8, 1998. 229

[48] Mariano P. Consens and Gonzalo Navarro, editors. *String Processing and Information Retrieval: 12th International Conference (SPIRE'05)*, volume 3772 of *Lecture Notes in Computer Science*, Buenos Aires, Argentina, November 2–4, 2005. Springer. 224, 227, 229

[49] J. H. Conway. *On Numbers and Games*. Number 6 in L.M.S. Monographs. Academic Press, London, 1976. 153

[50] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*, pages 314–320. MIT Press, Cambridge, Massachusetts, USA, 1990. 8, 133

[51] Baris Coskun and Nasir Memon. Confusion/diffusion capabilities of some robust hash functions. In CISS'06 [46]. 18, 38

[52] *Fourth International IEEE Computer Society Computational Systems Bioinformatics Conference (CSB'05)*, Stanford, California, USA, August 8–11, 2005. IEEE Computer Society. 225

[53] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, and P. Samarai. An open digest-based technique for spam detection. In ISCS'04 [16]. 39, 118

[54] Vladimír Dančík. *Expected Length of Longest Common Subsequences*. PhD thesis, University of Warwick, September 1994. 133, 134

[55] Frédéric Dardel and François Képès. *Bioinformatics: Genomics and Post-Genomics*, chapter 2, pages 26–59. Wiley, 2006. Translated by Noah Hardy. 132

[56] Sanjoy Dasgupta and Anupam Gupta. An elementary proof of the Johnson-Lindenstrauss Lemma. Technical Report TR-99-006, Berkeley, CA, 1999. 4

[57] Philip J. Davis. Gamma function and related functions. In Abramowitz and Stegun [1], chapter 6. 59, 61

[58] Umeshwar Dayal, Peter M. D. Gray, and Shojiro Nishio, editors. *Proceedings of the 21st International Conference on Very Large Data Bases (VLDB'95)*, Zurich, Switzerland, September 11–15, 1995. 215

[59] Umeshwar Dayal, Krithi Ramamritham, and T. M. Vijayaraman, editors. *Proceedings of the 19th International Conference on Data Engineering (ICDE'03)*, Bangalore, India, March 5–8, 2003. IEEE Computer Society. 228

[60] Joseph Deken. Probabilistic behaviour of longest-common-subsequence length. In Sankoff and Kruskal [179], chapter 16, pages 359–362. 133

[61] Joseph G. Deken. Some limit results for longest common subsequences. *Discrete Mathematics*, 26:17–31, 1979. 133

[62] Jay L. Devore. *Probability and Statistics for Engineering and the Sciences*. Wadsworth, Belmont, CA, USA, fourth edition, 1995. 6, 15

[63] Lee R. Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, July 1945. 6

[64] Andreas 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):321–402, September 1984. 101

[65] Patrice Enjalbert, Ernst W. Mayr, and Klaus W. Wagner, editors. *Proceedings of the 11th Annual Symposium on Theoretical Aspects of Computer Science (STACS'94)*, volume 775 of *Lecture Notes in Computer Science*, Caen, France, February 24–26, 1994. Springer. 224

[66] Christos Faloutsos and Ibrahim Kamel. Beyond uniformity and independence: Analysis of R-trees using the concept of fractal dimension. In PODS'94 [170], pages 4–13. 25, 26

[67] V. V. Fedorchuk. The fundamentals of dimension theory. In Arkhangel'skiĭ and Pontryagin [11], part II. 20

[68] Joan Feigenbaum and Lance Fortnow. Random-self-reducibility of complete sets. *SIAM Journal on Computing*, 22(5):994–1005, October 1993. 40

[69] Karina Figueroa. personal communication, January 11, 2008. 93

[70] Karina Figueroa, Edgar Chávez, Gonzalo Navarro, and Rodrigo Paredes. On the least cost for proximity searching in metric spaces. In WEA'06 [6], pages 279–290. 34

[71] Karina Figueroa, Gonzalo Navarro, and Edgar Chávez. *Metric Spaces Library*. Online `http://www.sisap.org/?f=library`. Accessed November 24, 2007. 4, 34, 93, 202, 204

[72] R. A. Fisher and L. H. C. Tippett. Limiting forms of the frequency distribution of the largest or smallest member of a sample. *Proceedings of the Cambridge Philosophical Society*, 24:180–190, 1928. 47

[73] *37th Annual Symposium on Foundations of Computer Science (FOCS'96)*, Burlington, Vermont, USA, October 14–16, 1996. IEEE. 214

[74] *40th Annual Symposium on Foundations of Computer Science (FOCS'99)*, New York, New York, USA, October 17–18, 1999. IEEE Computer Society. 223

[75] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics: Principles and Practice*, pages 550–555. Addison-Wesley, 1996. Second Edition in C. 7

[76] M. Frances and A. Litman. On covering problems of codes. *Theory of Computing Systems*, 30(2):113–119, 1997. 126, 127, 128, 149, 176, 194

[77] Andrew U. Frank, Irene Campari, and Ubaldo Formentini, editors. *GIS—From Space to Territory: Theories and Methods of Spatio-Temporal Reasoning*, volume 639 of *Lecture Notes in Computer Science*, Pisa, Italy, September 21–23, 1992. Springer. 4

[78] Maurice Fréchet. Sur quelques points du calcul functionnel [On some points of functional calculus]. *Rendiconti del Circolo Matematico di Palermo*, 22:1–74, 1906. 4

[79] Jiri [Jessica] Fridrich. Visual hash for oblivious watermarking. In Ping W. Wong and Edward J. Delp III, editors, *Security and Watermarking of Multimedia Contents II*, volume 3971 of *Proceedings of SPIE*, pages 286–294, San Jose, California, January 24–26, 2000. SPIE. 18

[80] Janos Galambos. *The Asymptotic Theory of Extreme Order Statistics*. Robert E. Krieger Publishing Company, Malabar, Florida, U.S.A., second edition, 1987. 46, 48, 56, 57

[81] Hector Garcia-Molina and H. V. Jagadish, editors. *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data (SIGMOD'90)*, Atlantic City, New Jersey, USA, May 23–25, 1990. ACM Press. 215

[82] Martin Gardner. Crafty cheese cuts. In *aha! Insight*, pages 32–33. W. H. Freeman and Company, 1978. 85

[83] Martin Gardner. Piet Hein's superellipse. In *Mathematical Carnival*, chapter 18, pages 240–254. Mathematical Association of America, Washington, D.C., USA, 1989. 43

[84] Alexander F. Gelbukh, Alvaro de Albornoz, and Hugo Terashima-Marín, editors. *Advances in Artificial Intelligence: 4th Mexican International Conference on Artificial Intelligence (MICAI'05)*, volume 3789 of *Lecture Notes in Computer Science*, Monterrey, Mexico, November 14–18, 2005. Springer. 216

[85] K. Goldberg, M. Newman, and E. Haynsworth. Multinomial coefficients. In Abramowitz and Stegun [1], subsection 24.1.2, pages 823–824. 13, 119

[86] Michael T. Goodrich and Catherine C. McGeoch, editors. *Algorithm Engineering and Experimentation: International Workshop (ALENEX'99)*, volume 1619 of *Lecture Notes in Computer Science*, Baltimore, Maryland, USA, January 15–16, 1999. Springer. 231

[87] I. S. Gradshteyn and I. M. Ryzhik. *Table of Integrals, Series and Products*. Academic Press, Boston, sixth edition, 2000. 50, 60, 61, 65

[88] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics*. Addison-Wesley, Reading, Massachusetts, USA, second edition, 1994. 13, 59, 119

[89] Peter Grassberger and Itamar Procaccia. Measuring the strangeness of strange attractors. *Physica D*, 9:189–208, 1983. 25, 26

[90] Joachim Grollmann and Alan L. Selman. Complexity measures for public-key cryptosystems. *SIAM Journal on Computing*, 17(2):309–335, April 1988. 114

[91] S. Grumbach and F. Tahi. A new challenge for compression algorithms: Genetic sequences. *Journal of Information Processing and Management*, 30(6):875–886, 1994. 5

[92] Branko Grünbaum. *Arrangements and Spreads*. Number 10 in Conference Board of the Mathematical Sciences Regional Conference Series in Mathematics. American Mathematical Society, Providence, June 1971. 81

[93] Anupam Gupta. Embedding tree metrics into low-dimensional euclidean spaces. *Discrete & Computational Geometry*, 24(1):105–116, 2000. 99

[94] Allan Gut. *Probability: A Graduate Course*. Springer, 2005. 45, 46

[95] Antonin Guttman. *R*-trees: a dynamic index structure for spatial searching. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 14(2):47–57, 1984. 8

[96] Laura M. Haas and Ashutosh Tiwary, editors. *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data (SIGMOD'98)*, Seattle, Washington, USA, June 2–4, 1998. ACM Press. 215

[97] Alon Y. Halevy, Zachary G. Ives, and AnHai Doan, editors. *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data (SIGMOD'03)*, San Diego, California, USA, June 9–12, 2003. ACM. 228

[98] Peter Hall. On the rate of convergence of normal extremes. *Journal of Applied Probability*, 16:433–439, 1979. 56, 57, 69

[99] Felix Hausdorff. *Grundzüge der Mengenlehre*. Chelsea Publishing Company, New York, reprint of first edition, 1965. First published 1914. 4, 221

[100] Felix Hausdorff. *Set Theory*. Chelsea Publishing Company, New York, second edition, 1962. Translation by John R. Aumann et al. of the 1935 third edition of *Grundzüge der Mengenlehre* [99]. 4

[101] Gisli R. Hjaltason and Hanan Samet. Index-driven similarity search in metric spaces. *ACM Transactions on Database Systems*, 28(4):517–580, December 2003. 4, 7, 9

[102] Kenneth Hoffman and Ray Kunze. *Linear Algebra*. Prentice Hall, 1971. 4

[103] Douglas R. Hofstadter. *Metamagical Themas: Questing for the essence of mind and pattern*. Basic Books, New York, 1985. 3, 43

[104] Gregory M. Hunter and Kenneth Steiglitz. Operations on images using quad trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(2):145–153, April 1979. 7

[105] *Proceedings of the 15th International Conference on Data Engineering (ICDE'99)*, Sydney, Australia, March 23–26, 1999. IEEE Computer Society. 216

[106] C. Icking, R. Klein, N.-M. Lê, L. Ma, and F. Santos. On bisectors for convex distance functions in 3-space. In CCCG'99 [37], pages 119–123. 85

[107] Christian Icking, Rolf Klein, Ngọc-Minh Lê, and Lihong Ma. Convex distance functions in 3-space are different. *Fundamenta Informaticae*, 22(4):331–352, 1995. 81, 82

[108] James Pickands III. Moment convergence of sample extremes. *The Annals of Mathematics Statistics*, 39(3):881–889, 1968. 49

[109] Piotr Indyk and Jiří Matoušek. Low-distortion embeddings of finite metric spaces. In Jacob E. Goodman and Joseph O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 8. Chapman and Hall, second edition, 2004. 99

[110] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In STOC'98 [202], pages 604–613. 19

[111] Chris L. Jackins and Steven L. Tanimoto. Oct-trees and their use in representing three-dimensional objects. *Computer Graphics and Image Processing*, 14(3):249–270, November 1980. 7

[112] Eddy L. O. Jansson and Matthew Skala. The breaking of Cyber Patrol® 4. Electronic white paper., March 11, 2000. 102

[113] Tao Jiang and Ming Li. On the approximation of shortest common supersequences and longest common subsequences. *SIAM Journal on Computing*, 24(5):1122–1139, October 1995. 132, 133

[114] Norman L. Johnson, Adrienne W. Kemp, and Samuel Kotz. *Univariate Discrete Distributions*. Wiley, third edition, 2005. Effectively the first volume of, and has chapter numbering continuous with, Johnson, Kotz, and Balakrishnan [115, 116]. 27, 104, 223

[115] Norman L. Johnson, Samuel Kotz, and N. Balakrishnan. *Continuous Univariate Distributions*, volume 1. Wiley, 1994. References in this volume to "Volume 1" actually refer to Johnson, Kemp, and Kotz [114]. 27, 54, 60, 72, 222

[116] Norman L. Johnson, Samuel Kotz, and N. Balakrishnan. *Continuous Univariate Distributions*, volume 2. Wiley, 1994. 27, 47, 48, 222

[117] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–396, 1984. 175

[118] Mark G. Karpovsky. Weight distribution of translates, covering radius, and perfect codes correcting errors of given weights. *IEEE Transactions on Information Theory*, 27(4):462–471, 1981. 127

[119] Norio Katayama and Shin'ichi Satoh. The SR-tree: an index structure for high-dimensional nearest neighbor queries. In SIGMOD'97 [165], pages 369–380. 8

[120] Norio Katayama and Shin'ichi Satoh. Experimental evaluation of disk-based data structures for nearest neighbor searching. In *Data Structures, Near Neighbor Searches, and Methodology: Fifth and Sixth DIMACS Implementation Challenges*, volume 59 of *AMS DIMACS Series*, pages 87–104. AMS, 2002. 204

[121] Mohamed A. Khamsi and William A. Kirk. *An Introduction to Metric Spaces and Fixed Point Theory*. Wiley, 2001. 2

[122] Joe Kilian, editor. *Advances in Cryptology: 21st Annual International Cryptology Conference (CRYPTO'01)*, volume 2139 of *Lecture Notes in Computer Science*, Santa Barbara, California, USA, August 19–23, 2001. Springer. 214

[123] L. Christine Kinsey. *Topology of Surfaces*. Undergraduate Texts in Mathematics. Springer, 1993. 20

[124] Marcos Kiwi, Martin Loebl, and Jiří Matoušek. Expected length of the longest common subsequence for large alphabets. In LATIN'04 [133], pages 302–311. 133

[125] Jon M. Kleinberg and Éva Tardos. Approximation algorithms for classification problems with pairwise relationships: Metric labeling and markov random fields. In FOCS'99 [74], pages 14–23. 102

[126] Keith Knight. *Mathematical Statistics*. Chapman and Hall/CRC, Boca Raton, 2000. 27, 46

[127] Donald E. Knuth. *Computer Modern Typefaces*, volume E of *Computers and Typesetting*, pages 15, 23. Addison-Wesley, 1986. 43

[128] Grzegorz Kondrak. N-gram similarity and distance. In SPIRE'05 [48], pages 115–126. 6

[129] Flip Korn and S. Muthukrishnan. Influence sets based on reverse nearest neighbor queries. In SIGMOD'00 [43], pages 201–212. 35

[130] Daniel Kunkle and Gene Cooperman. Twenty-six moves suffice for Rubik's cube. In ISSAC'07 [210], pages 235–242. 3

[131] George Lakoff and Mark Johnson. *Metaphors We Live By*. University of Chicago Press, 1980. 1

[132] Nikolaos Laoutaris, Vassilis Zissimopoulos, and Ioannis Stavrakakis. On the optimization of storage capacity allocation for content distribution. *Computer Networks*, 47(3):409–428, 2005. 102

[133] *Theoretical Informatics: 6th Latin American Symposium (LATIN'04)*, volume 2976 of *Lecture Notes in Computer Science*, Buenos Aires, Argentina, April 5–8, 2004. Springer. 223

[134] Ngọc-Minh Lê. On voronoi diagrams in the L_p-metric in higher dimensions. In STACS'94 [65], pages 711–722. 81

[135] G. Lejeune Dirichlet. Über die reduction der positiven qudratischen formen mit drei unbestimmten ganzen zahlen [On the reduction of positive quadratic forms of three integer variables]. *Journal für die reine und angwandte Mathematik*, 40:209–227, 1850. 78

[136] J. Lember and H. Matzinger. Standard deviation of the longest common subsequence. Preprint 06-035, Bielefeld University CRC 701, 2006. Online http://www.math.uni-bielefeld.de/sfb701/preprints/sfb06035.pdf. 134

[137] Arthur M. Lesk. *Introduction to Bioinformatics*, chapter 4, pages 160–215. Oxford University Press, 2002. 132

[138] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics—Doklady*, 10(8):707–710, 1966. English translation of *Doklady Akademii Nauk SSSR*, 163(4):845–848, 1965. 131, 134

[139] Lun Li, David Alderson, John Doyle, and Walter Willinger. Towards a theory of scale-free graphs: Definition, properties, and implications. *Internet Mathematics*, 2(4), 2005. 28

[140] Ming Li, Xin Chen, Xin Li, Bin Ma, and Paul M. B. Vitányi. The similarity metric. *IEEE Transactions on Information Theory*, 50(12):3250–3264, 2004. 6, 7

[141] Ming Li and Paul Vitányi. *An Introduction to Kolmogorov Complexity and its Applications*. Springer, 1993. 5, 20

[142] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004. 4, 5

[143] George S. Lueker. Improved bounds on the average length of longest common subsequences. In SODA'03 [196], pages 130–131. Extended version online http://www.ics.uci.edu/~lueker/papers/lcs/. 133

[144] Ben Lynn, Manoj Prabhakaran, and Amit Sahai. Positive results and techniques for obfuscation. In EUROCRYPT'04 [36], pages 20–39. 99, 101, 102

[145] Arnaldo Mandel. *Topology of Oriented Matroids*. PhD thesis, University of Waterloo, 1982. 81

[146] Benoit B. Mandelbrot. *The Fractal Geometry of Nature*. W. H. Freeman, 1982. 18, 25

[147] Rui Mao, Weijia Xu, Smriti Ramakrishnan, Glen Nuckolls, and Daniel P. Miranker. On optimizing distance-based similarity search for biological databases. In CSB'05 [52], pages 351–361. 27

[148] William J. Masek and Michael S. Paterson. A faster algorithm computing string edit distances. *Journal of Computer and System Sciences*, 20(1):18–31, February 1980. 132

[149] Jiří Matoušek. On embedding trees into uniformly convex Banach spaces. *Israel Journal of Mathematics*, 114(1):221–237, December 1999. 101

[150] Jiří Matoušek, Micha Sharir, and Emo Welzl. A subexponential bound for linear programming. *Algorithmica*, 16(4–5):498–516, 1996. 175

[151] Nimrod Megiddo. Linear programming in linear time when the dimension is fixed. *Journal of the ACM*, 31(1):114–127, January 1984. 175

[152] Luisa Micó, José Oncina, and Enrique 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. 33, 34

[153] R. D. Milne. *Applied Functional Analysis: An Introductory Treatment*. Pitman, Boston, 1980. 42

[154] H. Minkowski. Space and time. In *The Principle of Relativity: A collection of original memoirs on the special and general theory of relativity*, pages 73–91. Dover, 1952. Translation by W. Perrett and G. B. Jeffery of an address to the 80th Assembly of German Natural Scientists and Physicians, Cologne, September 21, 1908. 4

[155] William [Morris] and Mary Morris. *The Word Game Book*, pages 25–27. Harper and Brothers, 1959. 153, 154

[156] Gonzalo Navarro. A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1):31–88, 2001. 134

[157] Joseph Needham and Wang Ling. *Mathematics and the Sciences of the Heavens and the Earth*, volume 3 of *Science and Civilisation in China*, page 101. Cambridge University Press, 1959. 54

[158] S. B. Needleman and C. D. Wunsch. A general method applicable to the search of similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453, 1970. 132

[159] Chad Norwood and cmeclax. Digest::Nilsimsa 0.06. Computer software, June 13, 2002. Online http://search.cpan.org/~vipul/Digest-Nilsimsa-0.06/. 118

[160] Fritz Oberhettinger. Hypergeometric functions. In Abramowitz and Stegun [1], chapter 15. 13

[161] F. W. J. Olver. Bessel functions of integer order. In Abramowitz and Stegun [1], chapter 9. 61

[162] Edward Ott. *Chaos in dynamical systems*. Cambridge University Press, Cambridge, U.K., 2nd edition, 2002. 24, 25, 26, 198

[163] Allan Paivio. *Mental Representations: A dual coding approach*, volume 9 of *Oxford Psychology Series*. Oxford University Press, 1986. 1

[164] Rodrigo Paredes and Edgar Chávez. Using the k-nearest neighbor graph for proximity searching in metric spaces. In SPIRE'05 [48], pages 127–138. 12

[165] Joan Peckham, editor. *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data (SIGMOD'97)*, Tucson, Arizona, USA, May 13–15, 1997. ACM Press. 223

[166] David Peleg and Eilon Reshef. A variant of the arrow distributed directory with low average complexity. In ICALP'99 [215], pages 615–624. 102

[167] Persistence of Vision Raytracer Pty. Ltd. *POV-Ray 3.6.1 Documentation*, August 3, 2004. Online http://www.povray.org/documentation/. 43, 83

[168] Yakov B. Pesin. *Dimension Theory in Dynamical Systems: Contemporary Views and Applications*. University of Chicago Press, 1997. 25, 198

[169] Vera S. Pless, W. Cary Huffman, and Richard A. Brualdi. An introduction to algebraic codes. In *Handbook of Coding Theory*, volume 1, chapter 1, pages 3–139. Elsevier, Amsterdam, The Netherlands, 1998. 4, 117, 122

[170] *Proceedings of the Thirteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'94)*, volume 13, Minneapolis, Minnesota, May 24–26, 1994. ACM Press. 218

[171] Derek J. Price. Some unusual series occurring in n-dimensional geometry. *The Mathematical Gazette*, 30:149–150, 1946. 86, 88, 92

[172] Singiresu S. Rao. *The Finite Element Method in Engineering*, pages 57–59. Butterworth Heinemann, Boston, third edition, 1999. 7

[173] J. David Rawn. *Biochemistry*, pages 826–829. Neil Patterson Publishers, Carolina Biological Supply Company, Burlington, North Carolina, USA, 1989. 132

[174] Michael Reid. Superflip requires 20 face turns. Electronic mailing list message, January 18, 1995. Online http://www.math.rwth-aachen.de/~Martin.Schoenert/Cube-Lovers/michael_reid__superflip_requires_20_face_turns.html. 3

[175] Arun Ross, Jidnya Shah, and Anil K. Jain. From template to image: Reconstructing fingerprints from minutiae points. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(4):544–560, 2007. 38

[176] E. Vidal Ruiz. An algorithm for finding nearest neighbors in (approximately) constant time. *Pattern Recognition Letters*, 4:145–157, 1986. 11, 34

[177] S. Cenk Sahinalp, Murat Tasan, Jai Macker, and Z. Meral Ozsoyoglu. Distance based indexing for string proximity search. In ICDE'03 [59]. 6

[178] Magnus Sahlgren. *The Word-Space Model*. PhD thesis, Stockholm University, 2006. Online http://www.sics.se/~mange/TheWordSpaceModel.pdf. 4, 21

[179] David Sankoff and Joseph B. Kruskal, editors. *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*. Addison-Wesley, 1983. 218, 228

[180] David Sankoff and Sylvie Mainville. Common subsequences and monotone subsequences. In Sankoff and Kruskal [179], chapter 17, pages 363–365. 133

[181] Francisco Santos. On Delaunay oriented matroids for convex distance functions. *Discrete & Computational Geometry*, 16(2):197–210, 1996. 81

[182] *Proceedings of the Symposium on Computational Geometry (SCG'85)*, Baltimore, Maryland, USA, June 5–7, 1985. ACM. 217

[183] Saul Schleimer, Daniel S. Wilkerson, and Alex Aiken. Winnowing: local algorithms for document fingerprinting. In SIGMOD'03 [97], pages 76–85. 7

[184] Bruce Schneier. *Applied Cryptography*. Wiley, second edition, 1996. 38, 118

[185] Thomas Seidl. Untitled computer data file. Online http://www.dbs.informatik.uni-muenchen.de/~seidl/DATA/histo112.112682.gz. Accessed January 25, 2008. 204

[186] Timos K. Sellis, Nick Roussopoulos, and Christos Faloutsos. The R+-tree: A dynamic index for multi-dimensional objects. In VLDB'87 [203], pages 507–518. 8

[187] Dennis Shasha and Tsong-Li Wang. New techniques for best-match retrieval. *ACM Transactions on Information Systems*, 8(2):140–158, April 1990. 12

[188] *Proceedings of the 1st International Workshop on Similarity Search and Applications (SISAP'08)*, Cancún, Mexico, April 11–12, 2008. IEEE Computer Society. 229

[189] Matthew Skala. A limited-diffusion algorithm for blind substring search. In CITSS'98 [47], pages 397–410. 102

[190] Matthew Skala. Measuring the difficulty of distance-based indexing. In SPIRE'05 [48], pages 103–114. 28, 44, 68, 71, 118

[191] Matthew Skala. Counting distance permutations. In SISAP'08 [188], pages 69–76. 75, 93, 105, 204

[192] Matthew Skala. On the complexity of reverse similarity search. In SISAP'08 [188], pages 149–156. 108, 114, 126, 143, 154, 166

[193] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197, 1981. 132

[194] *Proceedings of the Fourth Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms (SODA'93)*, Austin, Texas, USA, January 25–27, 1993. ACM/SIAM. 231

[195] *Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'96)*, Atlanta, Georgia, USA, January 28–30, 1996. ACM/SIAM. 213

[196] *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'03)*, Baltimore, Maryland, USA, January 12–14, 2003. ACM/SIAM. 225

[197] *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'06)*, Miami, Florida, USA, January 22–26, 2006. ACM Press. 215

[198] Robert R. Sokal and Peter H. Sneath. *Principles of Numerical Taxonomy*, pages 125–141. W. H. Freeman, San Francisco, 1963. 6

[199] J. Michael Steele. An Efron-Stein inequality for nonsymmetric statistics. *The Annals of Statistics*, 14(2):753–758, June 1986. 133, 134

[200] Lynn Arthur Steen and J. Arthur Seeback, Jr. *Counterexamples in Topology*. Springer, second edition, 1978. 2, 4

[201] Mark Steyvers and Joshua B. Tenenbaum. The large-scale structure of semantic networks: Statistical analyses and a model of semantic growth. *Cognitive Science*, 29(1):41–78, 2005. 28

[202] *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing (STOC'98)*, Dallas, Texas, USA, May 23–26, 1998. ACM. 215, 222

[203] Peter M. Stocker, William Kent, and Peter Hammersley, editors. *Proceedings of the 13th International Conference on Very Large Data Bases (VLDB'87)*, Brighton, England, September 1–4, 1987. 228

[204] James Thurber. Do you want to make something out of it? In *Thurber Country*, chapter 13. Simon and Schuster, New York, 1953. 154

[205] Jeffrey K. Uhlmann. Satisfying general proximity/similarity queries with metric trees. *Information Processing Letters*, 40:175–179, November 25, 1991. 10

[206] Esko Ukkonen. Algorithms for approximate string matching. *Information and Control*, 64(1-3):100–118, January/February/March 1985. 132

[207] Georges Voronoï. Nouvelles applications des paramètres continus à la théorie des formes qudratiques: Premier mémoire, Sur quelques propriétés des formes quadratiques positives parfaites [New applications of continuous parameters to the theory of qudratic forms: First report, On some properties of perfect positive quadratic forms]. *Journal für die reine und angwandte Mathematik*, 133:97–178, 1908. 78

[208] Robert A. Wagner and Michael J. Fischer. The string-to-string correction problem. *Journal of the ACM*, 21(1):168–173, January 1974. 132, 133

[209] Gerard Walschap. *Metric Structures in Differential Geometry*. Springer, 2004. 4

[210] Dongming Wang, editor. *Proceedings of the 2007 International Symposium on Symbolic and Algebraic Computation (ISSAC'07)*, Waterloo, Ontario, July 28–August 1, 2007. ACM. 224

[211] John Edward Warnock. *A hidden surface algorithm for computer generated halftone pictures*. PhD thesis, University of Utah, 1969. 7

[212] Henry S. Warren, Jr. The quest for an accelerated population count. In Andy Oram and Greg Wilson, editors, *Beautiful Code*, chapter 10. O'Reilly, Sebastopol, California, USA, 2007. 118

[213] Eric W. Weisstein. Rising factorial. In *The CRC Concise Encyclopedia of Mathematics*, page 2581. Chapman and Hall, Boca Raton, Florida, USA, second edition, 2003. 13

[214] Margaret Wertheim. Crocheting the hyperbolic plane: An interview with David Henderson and Daina Taimina. *Cabinet*, (16), Winter 2004. Online http://www.cabinetmagazine.org/issues/16/crocheting.php. 207

[215] Jirí Wiedermann, Peter van Emde Boas, and Mogens Nielsen, editors. *Automata, Languages and Programming: 26th International Colloquium (ICALP'99)*, volume 1644 of *Lecture Notes in Computer Science*, Prague, Czech Republic, July 11-15, 1999. Springer. 227

[216] Maurice Vincent Wilkes. *Time-Sharing Computer Systems*. Elsevier, New York, 1968. 38

[217] Peter N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In SODA'93 [194], pages 311–321. 8

[218] Peter N. Yianilos. Excluded middle vantage point forests for nearest neighbour search. In ALENEX'99 [86]. 28, 44, 56

[219] L.-S. Young. Dimension, entropy, and Lyapunov exponents. *Ergodic Theory & Dynamical Systems*, 2:109–124, 1982. 25

[220] Pavel Zezula, Giuseppe Amato, Vlastislav Dohnal, and Michal Batko. *Similarity Search: The Metric Space Approach*. Springer, 2006. 4

[221] Barton Zweibach. *A First Course in String Theory*, chapter 12. Cambridge University Press, 2004. 17

Index

Page numbers printed like **123** indicate definitions. Other relevant page numbers are printed like 123.