Indexing schemes for similarity search: an illustrated paradigm

Vladimir Pestov*
University of Ottawa, Ontario, Canada
vpest283@science.uottawa.ca
www.mathstat.uottawa.ca/~vpest283

Aleksandar Stojmirović
Victoria University of Wellington, New Zealand
aleksand@mcs.vuw.ac.nz
www.mcs.vuw.ac.nz/~aleksand

Abstract

We suggest a variation of the Hellerstein—Koutsoupias—Papadimitriou indexability model for datasets equipped with a similarity measure, with the aim of better understanding the structure of indexing schemes for similarity-based search and the geometry of similarity workloads. This in particular provides a unified approach to a great variety of schemes used to index into metric spaces and facilitates their transfer to more general similarity measures such as quasi-metrics. We discuss links between performance of indexing schemes and high-dimensional geometry. The concepts and results are illustrated on a very large concrete dataset of peptide fragments equipped with a quasi-metric tree indexing scheme.

1. Introduction

Indexing into very large datasets with the aim of fast similarity search still remains a challenging and largely elusive problem of data engineering. The main motivation for the present work comes from sequence-based biology, where high-speed access methods for biological sequence databases will be vital both for developing large-scale datamining projects [8] and for testing the nascent mathematical conceptual models [5].

What is needed, is a fully developed mathematical paradigm of indexability for similarity search that would incorporate the existing structures of database theory and possess a predictive power. While the fundamental building blocks - similarity measures, data distributions, hierarchical tree index structures, and so forth - are in plain view, the only way they can be assembled together is by examining concrete datasets of importance and taking one step at a time. Theoretical developments and massive amounts of computational work must proceed in parallel; generally, we share the philosophy espoused in [16].

The master concept was introduced in the influential paper [11] (cf. also [10]): a *workload*, W, is a triple consisting of a search domain Ω, a dataset X, and a set of queries, $\mathcal{Q}$. An *indexing scheme* according to [11] is just a collection of blocks covering X. While this concept is fully adequate for many aspects of theory, we believe that analysis of indexing schemes for similarity search, with its strong geometric flavour, requires a more structured approach, and so we put forward a concept of an indexing scheme as a system of blocks equipped with a tree-like search structure and decision functions at each step. We also suggest the notion of a *reduction* of one workload to another, allowing one to create new access methods from the existing ones. One example is the new concept of a quasi-metric tree, proposed here. We discuss how geometry of high dimensions (asymptotic geometric analysis) may offer a constructive insight into the nature of the curse of dimensionality.

Our concepts and results are illustrated throughout on a concrete dataset of short peptide fragments, containing nearly 24 million data points and equipped with a biologically significant similarity measure. In particular, we construct a quasi-metric tree index structure into our dataset, using on a well-known idea in molecular biology. Even if intended as a mere illustration and a building block for more sophisticated approaches, this scheme outputs 100 nearest neighbours

*Corresponding Author. Full Address: Department of Mathematics and Statistics, University of Ottawa, 585 King Edward Ave., Ottawa, ON K1N 6N5, Canada. Tel: (613) 562-5800, Ext. 3523, Fax: (613) 562-577 from the actual dataset to any one of the 20^{10} virtual peptide fragments through scanning on average 0.53 %, and at most 3.5 %, of data.

2. Workloads

2.1. Defintion and basic examples

A *workload* [11] is a triple $W = (\Omega, X, \mathcal{Q})$, where Ω is the *domain*, X is a finite subset of the domain (*dataset*, or *instance*), and $\mathcal{Q} \subseteq 2^{\Omega}$ is the set of *queries*, that is, some specified subsets of Ω. *Answering a query* $Q \in \mathcal{Q}$ means listing all datapoints $x \in X \cap Q$.

Example 2.1. The *trivial workload*: $\Omega = X = \{*\}$ is a one-element set, with a sole possible query, $Q = \{*\}$.

Example 2.2. Let $X \subseteq \Omega$ be a dataset. *Exact match queries* for X are singletons, that is, sets $Q = \{\omega\}$, $\omega \in \Omega$.

Example 2.3. Let $W_i = (\Omega_i, X_i, \mathcal{Q}_i), i = 1, 2, \ldots, n$ be a finite collection of workloads. Their *disjoint sum* is a workload $W = \sqcup_{i=1}^{n} W_i$, whose domain is the disjoint union $\Omega = \Omega_1 \sqcup \Omega_2 \sqcup \ldots \sqcup \Omega_n$, the dataset is the disjoint union $X = X_1 \sqcup X_2 \sqcup \ldots \sqcup X_n$, and the queries are of the form $Q_1 \sqcup Q_2 \sqcup \ldots \sqcup Q_n$, where $Q_i \in \mathcal{Q}_i$, $i = 1, 2, \ldots, n$.

2.2. Similarity queries

A (*dis*)*similarity measure* on a set Ω is a function of two variables $s \colon \Omega \times \Omega \to \mathbb{R}$, possibly subject to additional properties. A *range similarity query centred at* $x^* \in \Omega$ consists of all $x \in \Omega$ determined by the inequality $s(x^*, x) < K$ or $> K$, depending on the type of similarity measure.

A *similarity workload* is a workload whose queries are generated by a similarity measure. Different similarity measures, S_1 and S_2, on the same domain Ω can result in the same set of queries, $\mathcal{Q}$, in which case we will call S_1 and S_2 *equivalent*.

Metrics are among the best known similarity measures. A similarity measure $d(x, y) \geq 0$ is called a *quasi-metric* if it satisfies $d(x, y) = 0 \Leftrightarrow x = y$ and the triangle inequality, but is not necessarily symmetric.

2.3. Illustration: short protein fragments

The domain Ω consists of strings of length $m = 10$ from the alphabet Σ of 20 standard amino acids: $\Omega = \Sigma^{10}$.

	T	S	A	N	I	V	L	M	K	R	D	E	Q	W	F	Y	H	G	P	C
T	0	3	4	6	5	4	5	6	6	6	7	6	6	13	8	9	10	8	8	10
S	4	0	3	5	6	6	6	6	5	6	6	5	5	14	8	9	9	6	8	10
A	5	3	0	8	5	4	5	6	6	6	8	6	6	14	8	9	10	6	8	9
N	5	3	6	0	7	7	7	7	5	5	5	5	5	15	9	9	7	6	9	12
I	6	6	5	9	0	1	2	4	8	8	9	8	8	14	6	8	11	10	10	10
V	5	6	4	9	1	0	3	4	7	8	9	7	7	14	7	8	11	9	9	10
L	6	6	5	9	2	3	0	3	7	7	10	8	7	13	6	8	11	10	10	10
M	6	5	5	8	3	3	2	0	6	6	9	7	5	12	6	8	10	9	9	10
K	6	4	5	6	7	6	6	6	0	3	7	4	4	14	9	9	9	8	8	12
R	6	5	5	6	7	7	6	6	3	0	8	5	4	14	9	9	8	8	9	12
D	6	4	6	5	7	7	8	8	6	7	0	3	5	15	9	10	9	7	8	12
E	6	4	5	6	7	6	7	7	4	5	4	0	3	14	9	9	8	8	8	13
Q	6	4	5	6	7	6	6	5	4	4	6	3	0	13	9	8	8	8	8	12
W	7	7	7	10	7	7	6	6	8	8	10	8	7	0	5	5	10	8	11	11
F	7	6	6	9	4	5	4	5	8	8	9	8	8	10	0	4	9	9	11	11
Y	7	6	6	8	5	5	5	6	7	7	9	7	6	9	3	0	6	9	10	11
H	7	5	6	5	7	7	7	7	6	5	7	5	5	13	7	5	0	8	9	12
G	7	4	4	6	8	7	8	8	7	7	7	7	7	13	9	10	10	0	9	12
P	6	5	5	8	7	6	7	7	6	7	7	6	6	15	10	10	10	8	0	12
C	6	5	4	9	5	5	5	6	8	8	9	9	8	13	8	9	11	9	10	0

Figure 1: BLOSUM62 asymmetric distances. Distances within members of the alphabet partition used for indexing are greyed.

The dataset X is formed by all peptide fragments of length 10 contained in the SwissProt database [2] of protein sequences of a variety of biological species (the release 40.30 of 19-Oct-2002). The fragments containing parts of low-complexity segments masked by the SEG program [21], as well as the fragments containing non-standard letters, were removed. The size of the filtered set is $|X| = 23,817,598$ unique fragments (31,380,596 total fragments).

The most commonly used scoring matrix in sequence comparison, BLOSUM62 [12], serves as the similarity measure on the alphabet Σ, and is extended over the domain Σ^m via $S(a, b) = \sum_{i=1}^{m} S(a_i, b_i)$ (the *ungapped* score).

The formula $d(a, b) = s(a, a) - s(a, b)$, $a, b \in \Sigma$, applied to the similarity measure given by BLOSUM62, as well as of most other matrices from the BLOSUM family, is a quasi-metric on Σ (Figure 1). One can now prove that the quasi-metric $\tilde{d}$ on the domain given by $\tilde{d}(a, b) = \sum_{i=1}^{m} d(a_i, b_i)$ is equivalent to the similarity measure S.

2.4. Inner and outer workloads

We call a workload W *inner* if $X = \Omega$, otherwise W is *outer*. Typically, for outer workloads $|X| \ll |\Omega|$.

Example 2.4. Our illustrative workload is outer, with the ratio $|X|/|\Omega| = 23,817,598/20^{10} \approx 0.0000023$.

Moreover, Fig. 2 shows that an overwhelming number of points $\omega \in \Omega$ have neighbours $x \in X$ within the distance of $\epsilon = 25$, which on average indicates

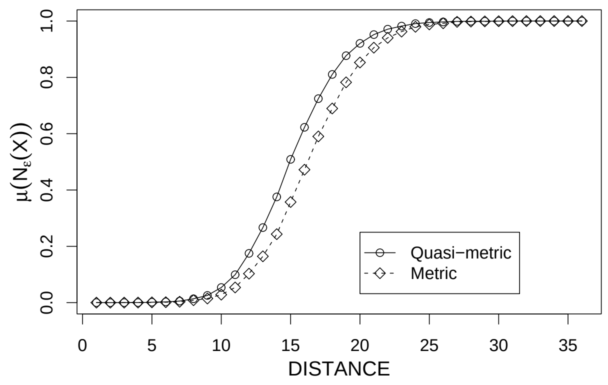

Figure 2: Growth with regard to the product measure of ϵ-neighbourhoods of our illustrative dataset X in $\Omega = \Sigma^{10}$.

high biological relevance. For this reason, most of the possible queries $Q = B_\epsilon(\omega)$ are meaningful, and our illustrative workload is inherently outer indeed.

The difference between inner and outer searches is particularly significant for similarity searches.

3. Indexing schemes

3.1. Basic concepts and examples

An *access method* for a workload W is an algorithm that on an input $Q \in \mathcal{Q}$ outputs all elements of $Q \cap X$. Typical access methods come from indexing schemes.

For a rooted finite tree T by $L(T)$ we will denote the set of leaf nodes and by $I(T)$ the set of inner nodes of T. The notation $t \in T$ will mean that t is a node of T, and C_t will denote the set of all children of a $t \in I(T)$, while the parent of t will be denoted $p(t)$.

Definition 3.1. Let $W = (\Omega, X, \mathcal{Q})$ be a workload. An *indexing scheme* on W is a triple $\mathcal{I} = (T, \mathcal{B}, \mathcal{F})$, where

- T is a rooted finite tree, with root node $*$,
- $\mathcal{B}$ is a collection of subsets $B_t \subseteq \Omega$ (*blocks*, or *bins*), where $t \in L(T)$.
- $\mathcal{F} = \{F_t \colon t \in I(T)\}$ is a collection of set-valued *decision functions,* $F_t \colon \mathcal{Q} \to 2^{C_t}$, where each value $F_t(Q) \subseteq C_t$ is a subset of children of the node t.

Definition 3.2. An indexing scheme $\mathcal{I} = (T, \mathcal{B}, \mathcal{F})$ for a workload $W = (\Omega, X, \mathcal{Q})$ will be called *consistent* if the following is an access method.

Algorithm 3.3.

```
on input $Q$ do
  set $A_0 = \{*\}$
  for each $i = 0, 1, \ldots$ do
    if $A_i \neq \emptyset$
    then for each $t \in A_i$ do
      if $t$ is not a leaf node
      then $A_{i+1} \leftarrow A_{i+1} \cup F_t(Q)$
      else for each $x \in B_t$ do
        if $x \in Q$
        then $A \leftarrow A \cup \{x\}$
return $A$
```

The following is an obvious sufficient condition for consistency. While it is not necessary, it is in common use, being easy to verify.

Proposition 3.4. *An indexing scheme $\mathcal{I} = (T, \mathcal{B}, \mathcal{F})$ for a workload $W = (\Omega, X, \mathcal{Q})$ is consistent if for every $Q \in \mathcal{Q}$ and each node $t \neq *$, having a descendant s with the property $B_s \cap Q \neq \emptyset$, one has $t \in f_{p(t)}(Q)$.*

In the future we will be considering consistent indexing schemes only.

Example 3.5. A simple linear scan of a dataset X corresponds to the indexing scheme where $T = \{*, \star\}$ has a root and a single child, $\mathcal{B}$ consists of a single block $B_\star = \Omega$, and the decision function F_* always outputs the same value $\{\star\}$.

Example 3.6. *Hashing* can be described in terms of the following indexing scheme. The tree T has depth one, with its leaves corresponding to bins, and the decision function f_* on an input Q outputs the entire family of bins in which elements of $Q \cap X$ are stored.

Example 3.7. If the domain Ω is linearly ordered (for instance, assume $\Omega = \mathbb{R}$) and the set of queries consists of intervals $[a, b]$, $a, b \in \Omega$, then a well-known and efficient indexing structure is constructed using a binary tree. The nodes t of T can be identified with elements of Ω chosen so that the tree is balanced. Each decision function F_t on an input $[a, b]$ outputs the set of all children nodes s of t satisfying

$$((t - a)(s - a) \geq 0) \wedge ((t - b)(s - b) \geq 0).$$

Remark 3.8. The computational complexity of the decision functions $F_t(Q)$, as well as the amount of 'branching' resulting from an application of Algorithm 3.3, become major efficiency factors in case of similarity-based search, which is why we feel they should be brought into the picture.

3.2. Metric trees

Let (Ω, X, ρ) be a similarity workload, where ρ is a metric, that is, each query $Q = B_\epsilon(\omega)$ is a ball of radius $\epsilon > 0$ around the query centre $\omega \in \Omega$.

A *metric tree* is an indexing structure into (Ω, X, ρ) where the decision functions are of the form

$$F_t(B_\epsilon(\omega)) = \{s \in C_t \colon f_s(\omega) \leq \epsilon\} \qquad (1)$$

for suitable 1-Lipschitz functions $f_s \colon \Omega \to \mathbb{R}$, one for each node $s \in T$. (Recall that $f \colon \Omega \to \mathbb{R}$ is 1-*Lipschitz* if $|f(x) - f(y)| \leq \rho(x, y)$ for each $x, y \in \Omega$.) We call those f_t *certification functions.* The set $F_t(B_\epsilon(\omega))$ is output by scanning all children s of t and accepting / rejecting them in accordance with the above criterion.

Theorem 3.9. *Let $W = (\Omega, X, \rho)$ be a metric similarity workload. Let T be a finite rooted tree, and let $B_t, t \in T$ be a collection of subsets of Ω (blocks), covering X and having the property that for every node t, the blocks indexed with the children of t cover B_t. Let $f_t \colon \Omega \to \mathbb{R}$ be 1-Lipschitz functions with the property $(\omega \in B_t) \Rightarrow (f_t(\omega) \leq 0)$. Define decision functions F_t as in Eq. (1). Then the triple $(T, \{B_t\}_{t \in L(T)}, \{F_t\}_{t \in I(T)})$ is a consistent indexing scheme for W.*

If a query $Q = B_\epsilon(\omega)$ meets a block B_t, then it is easy to show, using triangle inequality, that $f_t(\omega) \leq \epsilon$ and thus $F_{p(t)}(\omega)$ contains t and the branch starting at t has to be followed through, while the tree is being traversed. This assures that we won't miss any hits.

1-Lipschitz functions f_t with the required property always exist. Once the collection $B_t, t \in T$ of blocks has been chosen, put

$$f_t(\omega) = \rho(B_t, \omega) := \inf_{x \in B_t} \rho(x, \omega),$$

the distance from a block B_t to an ω. However, such distance functions from sets are typically computationally very expensive. The art of constructing a metric tree consists in choosing computationally inexpensive certification functions that at the same time don't result in an excessive branching.

Example 3.10. The *GNAT* indexing scheme [4] uses certification functions of the form $f_{t_\pm}(\omega) = \pm\left(\rho(\omega, x_t) - M_t\right)$, where x_t is a datapoint chosen for the node t, M_t is the median value for the function $\omega \mapsto \rho(\omega, x_t)$, and $t_\pm$ are two children of t.

Example 3.11. The *vp-tree* uses certification functions of the form $f_t(\omega) = (1/2)(\rho(x_{t_+}, \omega) - \rho(x_{t_-}, \omega))$, where again $t_\pm$ are two children of t and $x_{t_\pm}$ are the *vantage points* for the node t.

Example 3.12. The *M-tree* [7] employs, as certification functions, those of the form

$$f_t(\omega) = \rho(x_t, \tau) - \sup_{\tau \in B_t} \rho(x_t, \tau),$$

where B_t is a block corresponding to the node t, x_t is a datapoint chosen for each node t, and the suprema on the r.h.s. are precomputed and stored.

There are many other examples of metric trees, e.g. k-d tree, *gh*-tree, mvp-tree, etc. [19, 20, 6]. They all seem to fit into the concept of a general metric tree equipped with 1-Lipschitz certification functions, first formulated in the present exact form in [18].

Example 3.13. Suppose $\Omega = X = \{0, 1\}^m$, the set of all binary strings of length m. The *Hamming distance* between two strings x and y is the number of terms where x and y differ. A k-neighbourhood of any point with respect to the Hamming distance can be output by a combinatorial generation algorithm such as traversing the binomial tree of order m to depth k.

3.3. Quasi-metric trees

Quasi-metrics often appear as similarity measures on datasets, and even if they are being routinely replaced with metrics by way of what we call a *projective reduction* of a workload (Ex. 4.6), this may result in a loss of performance (cf. Ex. 5.2). It is therefore desirable to develop a theory of indexability for quasi-metric spaces.

The concept of a 1-Lipschitz function is no longer adequate. Indeed, a 1-Lipschitz function $f \colon \Omega \to \mathbb{R}$ remains such with regard to the metric $d(x, y) = \max\{\rho(x, y), \rho(y.x)\}$ on Ω, and so using 1-Lipschitz functions for indexing in effect amounts to replacing ρ with a coarser metric d. A subtler concept becomes necessary.

Definition 3.14. Call a function f on a quasi-metric space (Ω, ρ) *left 1-Lipschitz* if for all $x, y \in \Omega$

$$f(x) - f(y) \leq \rho(x, y),$$

and *right 1-Lipschitz* if $f(y) - f(x) \leq \rho(x, y)$.

Example 3.15. Let A be a subset of a quasi-metric space (Ω, ρ). The distance function from A computed on the left, $d(x, A) = \inf\{\rho(x, a) \colon a \in A\}$, is left 1-Lipschitz, while the function $d(A, x)$ is right 1-Lipschitz.

Now one can establish an analog of Theorem 3.9.

Theorem 3.16. *Let $W = (\Omega, X, \rho)$ be a quasi-metric similarity workload. Let T be a finite rooted tree, and let $B_t, t \in T$ be blocks covering X in such a way that for every $t \in T$, $B_t \subseteq \cup_{s \in C(t)} B_s$. Let $f_t \colon \Omega \to \mathbb{R}$ be left 1-Lipschitz functions such that $(\omega \in B_t) \Rightarrow (f_t(\omega) \leq 0)$. Define decision functions F_t as in Eq. (1). Then the triple $(T, \{B_t\}_{t \in L(T)}, \{F_t\}_{t \in I(T)})$ is a consistent indexing scheme for W.*

Example 3.17. Many of the particular types of metric trees generalize to a quasi-metric setting. For instance, M-tree (Ex. 3.12) becomes a 'QM-tree' if the certification functions are chosen as

$$f_t(\omega) = \rho(\omega, x_t) - \sup_{\tau \in B_t} \rho(\tau, x_t),$$

where B_t and x_t are as in Ex. 3.12.

3.4. Illustration: a quasi-metric tree for protein fragments

Here is a simple but rather efficient implementation of a quasi-metric tree on our workload of peptide fragments (Subs. 2.3).

Let Σ, $\Omega = \Sigma^m$, and d be as in Subs. 2.3. Let γ be a partition of Σ, that is, a finite collection of disjoint subsets covering Σ. Denote by T the prefix tree of γ^m, that is, nodes of T are strings of the form $t = A_1 A_2 \dots A_l$, where $A_i \in \gamma$, $i = 1, 2, \dots, l$, $l \leq m$, and the children of t are all strings of length $l+1$ having t as its prefix. To every t as above assign a *cylinder subset* $B_t \subseteq \Omega$, consisting of all strings $\omega \in \Sigma^m$ such that $\omega_i \in A_i$, $i = 1, 2, \dots, l$.

The certification function f_t for the node t is the distance from the cylinder B_t, computed on the left: $f_t(\omega) := d(\omega, B_t)$. The value of f_t at any ω can be computed efficiently using precomputed and stored values of distances from each $a \in \Sigma$ to every $A \in \gamma$. The construction of a quasi-metric tree indexing into Σ^m is accomplished as in Th. 3.16.

In our case, the standard amino acid alphabet is partitioned into five groups (Figure 1) based on some known classification approaches to aminoacids. This partition induces a partition of $\Omega = \Sigma^{10}$ into $5^{10} = 9,765,625$ bins.

Since X contains 23,817,598 datapoints, there are on average 2.4 points per bin. The actual distribution

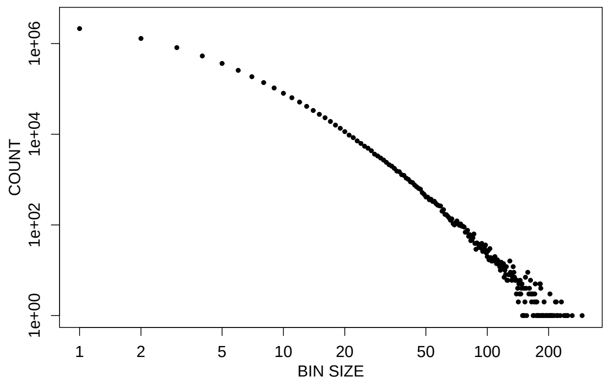

Figure 3: Distribution of bin sizes (3,455,126 empty bins out of 9,765,625 total).

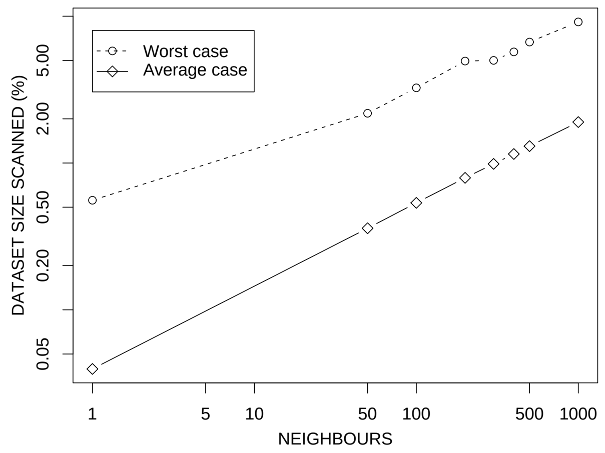

Figure 4: Percentage of dataset points scanned to obtain k nearest neighbours. Based on 20000 searches for each k. Query points were sampled with respect to the product measure based on amino acid frequencies.

of bin sizes is strongly skewed in favour of small sizes (Fig. 3) and appears to follow the DGX distrubition described in [3].

The performance of our indexing scheme is reflected in Fig. 4. Recall that an indexing scheme for similarity search that reduces the fraction of data scanned to below 10 % is already considered successful. Our figures are many times lower.

Remark 3.18. While other partitions of Σ producing different indexing schemes are certainly possible, ours can be used for searches based on other BLOSUM matrices with little loss of efficiency, because most amino acid scoring matrices used in practice reflect chemical and functional properties of amino acids and hence produce very similar collections of queries.

4. New indexing schemes from old

4.1. Disjoint sums

Any collection of access methods for workloads $W_1, W_2, \ldots, W_n$ leads to an access method for the disjoint sum workload $\sqcup_{i=1}^n W_i$: to answer a query $Q = \sqcup_{i=1}^n Q_i$, it suffices to answer each query Q_i, $i = 1, 2, \ldots, n$, and then merge the outputs.

In particular, if each W_i is equipped with an indexing scheme, $\mathcal{I}_i = (T_i, \mathcal{B}_i, \mathcal{F}_i)$, then a new indexing scheme for $\sqcup_{i=1}^n W_i$, denoted $\mathcal{I} = \sqcup_{i=1}^n \mathcal{I}_i$, is constructed as follows: the tree T contains all T_i's as branches beginning at the root node, while the families of bins and of certification functions for $\mathcal{I}$ are unions of the respective collections for all $\mathcal{I}_i$, $i = 1, 2, \ldots, n$.

4.2. Inductive reduction

Let $W_i = (\Omega_i, X_i, \mathcal{Q}_i)$, $i = 1, 2$ be two workloads. An *inductive reduction* of W_1 to W_2 is a pair of mappings $i\colon \Omega_2 \to \Omega_1$, $i^{\leftarrow}\colon \mathcal{Q}_1 \to \mathcal{Q}_2$, such that

- $i(X_2) \supseteq X_1$,
- for each $Q \in \mathcal{Q}_1$, $i^{-1}(Q) \subseteq i^{\leftarrow}(Q)$.

Notation: $W_2 \overset{i}{\rightrightarrows} W_1$.

An access method for W_2 leads to an access method for W_1, where a query $Q \in \mathcal{Q}_1$ is answered as follows:

```
**on input** $Q$ **do**
  answer the query $i^{\leftarrow}(Q)$
  **for** each $y \in X_2 \cap i^{\leftarrow}(Q)$ **do**
    **if** $i(y) \in Q$
    **then** add $x = i(y)$ on the list $A$
**return** $A$
```

If $\mathcal{I}_2 = (T_2, \mathcal{B}_2, \mathcal{F}_2)$ is a consistent indexing scheme for W_2, then a consistent indexing scheme $\mathcal{I}_1 = r_*(\mathcal{I}_1)$ for W_1 is constructed by taking $T_1 = T_2$, $B_t^{(1)} = i(B_t^{(2)})$, and $f_t^{(1)}(Q) = f_t^{(2)}(i^{\leftarrow}(Q))$ (the upper index $i = 1, 2$ refers to the two workloads).

Example 4.1. Let Γ be a finite graph of bounded degree, k. Associate to it a *graph workload*, W_Γ, which is an inner workload with $X = V_\Gamma$, the set of vertices, and a *k-nearest neighbour* query consists in finding N nearest neighbours of a vertex.

A *linear forest* is a graph that is a disjoint union of paths. The *linear arboricity*, $la(\Gamma)$, of a graph Γ is the smallest number of linear forests whose union is Γ. This number is, in fact, fairly small: it does not exceed $\lceil 3d/5 \rceil$, where d is the degree of Γ [1]. This concept leads to an indexing scheme for the graph workload W_Γ, as follows.

Let F_i, $i = 1, \ldots, la(\Gamma)$ be linear forests. Denote $F = \sqcup_{i=1}^{la(\Gamma)} F_i$. let $\phi\colon F \to \Gamma$ be a surjective map preserving the adjacency relation. Every linear forest can be ordered, and indexed into as in Ex. 3.7. At the next step, index into the disjoint sum F as in Subs. 4.1. Finally, index into Γ using the inductive reduction $\phi\colon F \to \Gamma$. This indexing scheme outputs nearest neighbourhs of any vertex of Γ in time $O(d \log n)$, requiring storage space $O(n)$, where n is the number of vertices in Γ.

4.3. Projective reduction

Let $W_i = (\Omega_i, X_i, \mathcal{Q}_i)$, $i = 1, 2$ be two workloads. A *projective reduction* of W_1 to W_2 is a pair of mappings $r\colon \Omega_1 \to \Omega_2$, $r^{\rightarrow}\colon \mathcal{Q}_1 \to \mathcal{Q}_2$, such that

- $r(X_1) \subseteq X_2$,
- for each $Q \in \mathcal{Q}_1$, $r(Q) \subseteq r^{\rightarrow}(Q)$.

Notation: $W_1 \overset{r}{\Rightarrow} W_2$.

An access method for W_2 leads to an access method for W_1, where a query $Q \in \mathcal{Q}_1$ is answered as follows:

```
**on input** $Q$ **do**
  answer the query $r^{\rightarrow}(Q)$
  **for** each $y \in X_2 \cap r^{\rightarrow}(Q)$ **do**
    **for** each $x \in r^{-1}(y)$ **do**
    **if** $x \in Q$
    **then** add $x$ on the list $A$
**return** $A$
```

Let $\mathcal{I}_2 = (T_2, \mathcal{B}_2, \mathcal{F}_2)$ be a consistent indexing scheme for W_2. The projective reduction $W_1 \overset{r}{\Rightarrow} W_2$ canonically determines an indexing scheme $\mathcal{I}_1 = r^*(\mathcal{I}_2)$ as follows: $T_1 = T_2$, $B_t^{(1)} = r^{-1}(B_t^{(2)})$, and $f_t^{(1)}(Q) = f_t^{(2)}(i^{\rightarrow}(Q))$, $i = 1, 2$.

Example 4.2. The linear scan of a dataset is a projective reduction to the trivial workload: $W \Rightarrow \{*\}$.

If $W = (\Omega, X, \mathcal{Q})$ is a workload and Ω' is a domain, then every mapping $r\colon \Omega \to \Omega'$ determines the *direct image workload*, $r_*(W) = (\Omega', r(X), r(\mathcal{Q}))$, where $r(X)$ is the image of X under r and $r(\mathcal{Q})$ is the family of all queries $r(Q), Q \in \mathcal{Q}$.

Example 4.3. Let $\mathcal{B}$ be a finite collection of *blocks* covering Ω. Define the *discrete workload* $(\mathcal{B}, \mathcal{B}, 2^{\mathcal{B}})$, and define the reduction by mapping each $w \in \Omega$ to the corresponding block and defining each $\tilde{r}(Q)$ as the union of all blocks that meet Q. The corresponding reduction forms a basic building block of many indexing schemes.

Example 4.4. Let W_i, $i = 1, 2$ be two metric workloads, that is, their query sets are generated by metrics d_i, $i = 1, 2$. In order for a mapping $f\colon \Omega_1 \to \Omega_2$ with the property $f(X_1) \subseteq X_2$ to determine a projective reduction $f\colon W_1 \overset{r}{\Rrightarrow} W_2$, it is necessary and sufficient that f be 1-Lipschitz: indeed, in this case every ball $B^X_\epsilon(x)$ will be mapped inside of the ball $B^Y_\epsilon(f(x))$ in Y.

Example 4.5. Pre-filtering is an often used instance of projective reduction. In the context of similarity workloads, this normally denotes a procedure whereby a metric ρ is replaced with a coarser distance d which is computationally cheaper. This amounts to the 1-Lipschitz map $(\Omega, X, \rho) \to (\Omega, X, d)$.

Example 4.6. The same applies to quasi-metrics. Moreover, it is routine to have a quasi-metric, ρ, replaced with a metric, d, having the property $\rho(x, y) \leq d(x, y)$, so that one does not miss any hits. The usual choices are $d(x, y) = \max\{\rho(x, y), \rho(y, x)\}$, or else $d(x, y) = \rho(x, y) + \rho(y, x)$, followed by a rescaling.

Example 4.7. A frequently used tool for dimensionality reduction of datasets is the famous Johnson–Lindenstrauss lemma, cf. e.g. [13]. Let $\Omega = \mathbb{R}^N$ be an Euclidean space of high dimension, and let $X \subset \mathbb{R}^N$ be a dataset with n points. If $\epsilon > 0$ and p is a randomly chosen orthogonal projection of $\mathbb{R}^N$ onto a linear subspace of dimension $k = O(\log n)/\epsilon^2$, then with overwhelming probability the mapping $\left(\sqrt{N/k}\right) p$ does not distort distances within X by more than the factor of $1 \pm \epsilon$.

The same is no longer true of the entire domain $\Omega = \mathbb{R}^N$, meaning that the technique can be only applied to indexing for similarity search of the *inner workload* $(X, \mathcal{Q})$, and not the outer workload $(\Omega, X, \mathcal{Q})$.

Example 4.8. A projective reduction of a metric space Ω to one of a smaller cardinality, Ω', which in turn is equipped with a hierarchical tree index structure, is at the core of a general paradigm of indexing into metric spaces developed in [6].

4.4. Illustration: our indexing scheme

Our indexing scheme can be also interpreted in terms of projective reduction as in example 4.3. Denote by γ the alphabet consisting of five groups into which the 20 aminoacids have been partitioned. Let $q\colon \Sigma \to \gamma$ be the map assigning to each amino acid the corresponding group. This map in its turn determines the map $r = q^m\colon \Omega \to \Omega_\gamma$, where $\Omega = \Sigma^m$ and $\Omega_\gamma = \gamma^m$. The direct image workload with domain Ω_γ, determined by the map r, can be indexed into using the binomial tree as in example 3.13 to generate all bins that can intersect the neighbourhood of the query point. Denote this indexing scheme by $\mathcal{I}$. Then the indexing scheme into Ω, described in Subs. 3.4, is just $r^*(\mathcal{I})$ as defined in Subs. 4.3.

5. Performance and geometry

5.1. Access overhead

Let $W_i = (\Omega_i, X_i, \mathcal{Q}_i)$, $i = 1, 2$ be two workloads, and let $W_1 \overset{(r, r^{\to})}{\Rrightarrow} W_2$ be a projective reduction of W_1 to W_2. The *relative access overhead* of the reduction r is the function $\beta_r\colon \mathcal{Q} \to [1, +\infty)$, assuming for each query Q the value $\beta_r(Q) := \left|r^{-1}\left(r^{\to}(Q)\right) \cap X\right| / |Q \cap X|$.

Example 5.1. The values for relative access overhead of our indexing scheme for protein fragments considered in terms of a projective reduction as in Subs. 4.4 can be easily obtained from Fig. 4.

Example 5.2. The access overhead of the projective reduction consisting in replacing a quasi-metric with a metric (Example 4.6) can be very considerable. Fig. 5 shows the overhead in the case of our dataset of fragments, where the quasi-metric ρ is replaced with the metric $d(x, y) = \max\{\rho(x, y), \rho(y, x)\}$. In our view, this in itself justifies the development of theory of quasi-metric trees.

5.2. Concentration

Let now $W = (\Omega, X, \mathcal{Q})$ be a similarity workload generated by a metric, d, on the domain. Denote by μ the normalized counting measure supported on the instance X, that is,

$$\mu(A) = |A \cap X| \,/\, |X| \tag{2}$$

for an $A \subseteq \Omega$. This μ is a probability measure on Ω.

The triples of this kind, (Ω, ρ, μ), where ρ is a metric and d is a probability measure on the metric space (Ω, ρ), are known as *mm-spaces,* or *probability metric spaces,* and they form objects of study of *geometry of*

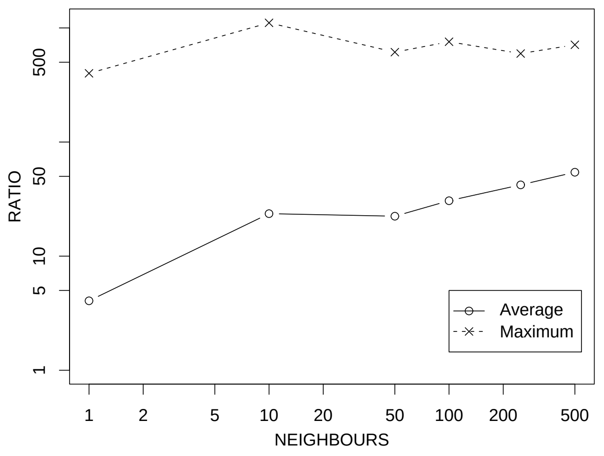

Figure 5: Ratio between the sizes of metric and quasi-metric balls containing k nearest neighbours with respect to quasi-metric. Each point is based on 5,000 samples.

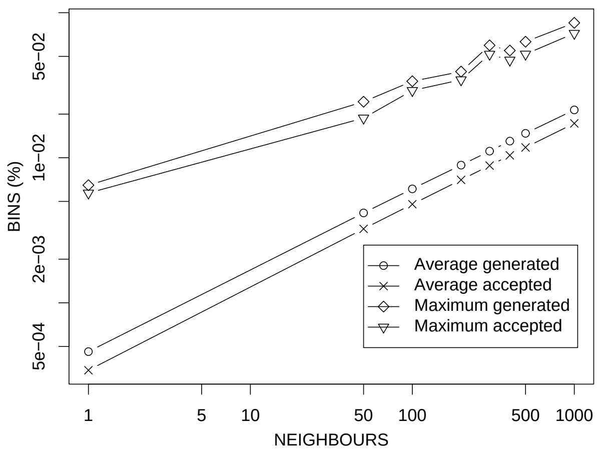

Figure 6: Percentage of bins scanned to obtain k nearest neighbours. Based on 20000 searches for each k. The query points were sampled with respect to the product measure based on amino acid frequencies.

high dimensions, see [9, 14, 15] and many references therein.

The central technical concept is that of the *concentration function* α_Ω of an mm-space Ω: for $\epsilon > 0$,

$$\alpha_\Omega(\epsilon) = 1 - \inf\left\{\mu(A_\epsilon)\colon A \subseteq \Omega,\ \mu(A) \geq \frac{1}{2}\right\},$$

and $\alpha_\Omega(0) = \frac{1}{2}$. If the intrinsic dimension of a triple (Ω, ρ, μ) is high, the concentration function $\alpha_\Omega(\epsilon)$ drops off sharply near zero. Typically, the concentration function of a probability metric space of dimension of order d satisfies $\alpha_\Omega(\epsilon) \leq C_1 \exp(-C_2\epsilon^2 d)$, where C_1, C_2 are suitable constants. This observation is known as the *concentration phenomenon.*

The concentration function α is non-increasing, but need not be strictly monotone. For each $x \geq 0$, denote $\alpha^{\prec}(x) = \inf\{\epsilon \leq 0\colon \alpha(\epsilon) \leq x\}$. The following is proved using similar arguments to Lemma 4.2 in [18].

Theorem 5.3. *Let $\Omega = (W, X, \rho)$ be a metric similarity workload, and let $\mathcal{I}$ be a metric tree indexing scheme into Ω. Denote by α the concentration function of the mm-space (W, ρ, μ), where μ is defined as in Eq. (2). Let $n = |X|$ be the number of datapoints, and let the largest bin contain m of them. The number of bins accessed to process the worst case similarity query is at least*

$$\frac{n}{m}\lfloor \alpha(\alpha^{\prec}(\ell/n) - \epsilon)\rfloor.$$

Example 5.4. In order to apply such estimates to a particular workload, one needs to determine its concentration function. If one equips the dataset of peptide fragments with a metric as in Ex. 4.6, then it is not difficult to derive Gaussian upper estimates for the concentration function $\alpha_W(\epsilon)$ using standard techniques of asymptotic geometric analysis. First, one estimates the concentration function of $\Omega = \Sigma^{10}$ equipped with the product measure using the martingale technique, and then one uses the way X sits inside of Ω (the rate of growth of neighbourhoods of the dataset, cf. Fig. 2). However, the bounds obtained this way are too loose and do not lead to meaningless bounds on performance. One needs to learn how to estimate the concentration function of a workload more precisely.

Fig. 6 shows the actual number of bins accessed by our indexing scheme in order to retrieve k nearest neighbours for various k. Notice that both the number of bins and the number of points of the dataset visited (Fig. 4) appear to follow the power law with exponent approximately $\frac{1}{2}$ with respect to the number of neighbours retrieved. As yet, we are unable to give a conceptual explanation of this fact.

However, one can obtain reasonable lower bounds on the performance of an M-tree.

Theorem 5.5. *Let W be a quasi-metric (or a metric) similarity workload, equipped with an M-tree as in Ex. 3.17. Denote by $\langle\mu(B_\epsilon)\rangle$ the average measure of a ball of radius $\epsilon > 0$. The worst-case number of bins accessed while processing a similarity query $Q = B_\epsilon(\omega)$ is at least $\lfloor\langle\mu(B_{2\epsilon}\rangle/\langle\mu(B_\epsilon\rangle\rfloor$.*

Example 5.6. Fig. 7 shows (on the log-log scale) the rate of growth of measure of balls $B_\epsilon(\omega)$ in the illustrative dataset of peptide fragments for the quasi-metric.

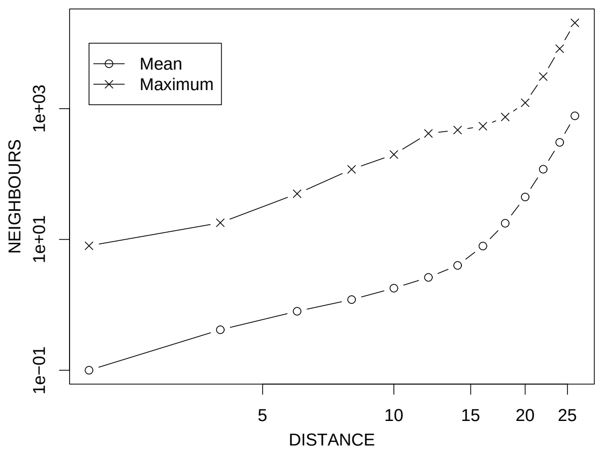

Figure 7: Growth of balls in the illustrative dataset.

The rate of growth in the most meaningful range of ϵ for similarity search can be estimated as being between 10 and 11. (This is, essentially, the Minkowski dimension of the dataset.) Now Th. 5.5 leads to conclude that the worst-case number of bin accesses for any M-tree is at least on the order of $2^{10} \approx 1024$.

5.3. Concentration and certification functions

Let $f\colon \Omega \to \mathbb{R}$ be a 1-Lipschitz function. Denote by M the median value of f. It is known that

$$\mu\{\omega\colon\ |f(\omega) - M| > \epsilon\} \leq 2\alpha_\Omega(\epsilon),$$

that is, if Ω is high-dimensional, the values of f are concentrated near one value. If one sees such functions as random variables respecting the distance, the concentration phenomenon says that on a high-dimensional Ω, the distribution of f peaks out near one value. Using such f as certification functions in indexing scheme leads to a massive amount of branching and the dimensionality curse [18].

Yet, there are reasons to believe that the main reason for the curse of dimensionality is not the inherent high-dimensinality of datasets, but a poor choice of certification functions. Efficient indexing schemes require usage of *dissipating functions*, that is, 1-Lipschitz functions whose spread of values is more broad, and which are still computationally cheap. This interplay between complexity and dissipation is, we believe, at the very heart of the nature of dimensionality curse.

Example 5.7. One reason for a relative efficiency of our quasi-metric tree lies in the good choice of certification functions, which are less concentrated than distances from points (Fig. 8).

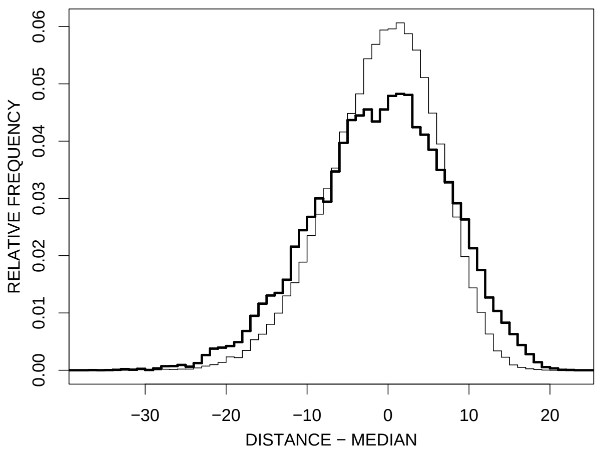

Figure 8: Distributions of distances from 40,000 random points to a typical point (SEDRELLTEQ) in Ω and of distances to a bin (the one containing the above fragment).

6. Conclusions

Our proposed approach to indexing schemes used in similarity search allows for a unifying look at them and facilitates the task of transferring the existing expertise to more general similarity measures than metrics. In particular, we propose the concept of a quasi-metric tree based on a new notion of left 1-Lipschitz functions, and implement it on a very large dataset of peptide fragments to obtain a simple yet efficient indexing scheme.

We hope that our concepts and constructions will meld with methods of geometry of high dimensions and lead to analysis of performance of indexing schemes for similarity search. While we have not yet reached the stage where asymptotic geometric analysis can give accurate predictions of performance, at least it leads to some conceptual understanding of their behaviour.

We suggest using our dataset of protein fragments as a simple benchmark for testing indexing schemes for similarity search.

7. Acknowledgements

The authors are grateful to Bill Jordan for his gentle guidance in the area of proteomics and stimulating discussions. The investigation was supported by the Marsden Fund of the Royal Society of New Zealand and by the University of Ottawa research funds. The second named author (A.S.) was also supported by a Bright Future PhD scholarship awarded by the NZ Foundation for Research, Science and Technology

jointly with the Fonterra Research Centre.

8. References

[1] N. Alon, *The linear arboricity of graphs,* Israel J. Math. **62** (1988), no. 3, 311–325.

[2] A. Bairoch and R. Apweiler. The SWISS-PROT protein sequence database and its supplement TrEMBL in 2000. *Nucleic Acids Res.*, 28:45–48, 2000.

[3] Z. Bi, C. Faloutsos, and F. Korn. The ”DGX” distribution for mining massive, skewed data. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 17–26, San Francisco, California, 2001. ACM Press New York, NY, USA.

[4] S. Brin, *Near neighbor search in large metric spaces,* in: Proc. of the 21st VLDB Conf., Zurich, Switzerland, Sept. 1995, pp. 574–584.

[5] A. Carbone and M. Gromov. Mathematical slices of molecular biology. *Numéro spécial de la Gazette des Mathematiciens, Société Mathématique de France*, 88:11–80, 2001.

[6] E. Chávez, G. Navarro, R. A. Baeza-Yates, and J. L. Marroquín. Searching in metric spaces. *ACM Computing Surveys*, 33(3):273–321, 2001.

[7] P. Ciaccia, M. Patella, and P. Zezula, *A cost model for similarity queries in metric spaces,* in: Proc. 17-th Annual ACM Symposium on Principles of Database Systems (PODS’98), Seattle, WA, June 1998, pp. 59–68.

[8] N. Goodman. Ome sweet ome. *Genome Technology*, pages 56–59, April 2002.

[9] M. Gromov. *Metric Structures for Riemannian and Non-Riemannian Spaces*, volume 152 of *Progress in Mathematics*. Birkhauser, 1999.

[10] J. M. Hellerstein, E. Koutsoupias, D. P. Miranker, C. H. Papadimitriou, and V. Samoladas. On a model of indexability and its bounds for range queries. *Journal of the ACM (JACM)*, 49(1):35–55, 2002.

[11] J. M. Hellerstein, E. Koutsoupias, and C. H. Papadimitriou. On the analysis of indexing schemes. In *Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 249–256, Tucson, Arizona, 12–15 May 1997.

[12] S. Henikoff and J. Henikoff. Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci. U.S.A.*, 89:10915–10919, 1992.

[13] P. Indyk and R. Motwani, *Approximate nearest neighbors: towards removing the curse of dimensionality,* Proc. 30-th Symp. on Theory of Computing, 1998, pp. 604–613.

[14] M. Ledoux. *The Concentration of Measure Phenomenon*, volume 89 of *Mathematical Surveys and Monographs*. American Mathematical Society, 2001.

[15] V.D. Milman, *Topics in asymptotic geometric analysis,* GAFA 2000 (Tel Aviv, 1999), Geom. Funct. Anal. Special Volume, Part I (2000), 792–815.

[16] C.H. Papadimitriou, *Database metatheory: asking the big queries,* in: Proc. 14-th PODS, San Jose, CA, May 1995, pp. 1-10.

[17] V. Pestov. A geometric framework for modelling similarity search. In *Proceedings of the 10th International Conference on Database and Expert Systems Applications (DEXA’99)*, pages 150–154, Florence, Italy, Sept 1999. IEEE Computer Society, Los Alamitos, CA.

[18] V. Pestov. On the geometry of similarity search: dimensionality curse and concentration of measure. *Information Processing Letters*, 73:47–51, 2000.

[19] J.K. Uhlmann, *Satisfying general proximity/similarity queries with metric trees,* Information Processing Lett. **40** (1991), 175–179.

[20] R. Weber, H.-J. Schek, and S. Blott, *A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces,* in: Proc. of the 24st VLDB Conf., New York, USA, Aug. 1998, pp. 194–205.

[21] J. Wootton and S. Federhen. Analysis of compositionally biased regions in sequence databases. *Methods Enzymol.*, 266:554–571, 1996.