Induced Permutations for Approximate Metric Search

Lucia Vadicamo*, Giuseppe Amato, Claudio Gennaro

Institute of Information Science and Technologies (ISTI), Italian National Research Council (CNR), Via G. Moruzzi 1, 56124,Pisa, Italy

Abstract

Permutation-based Indexing (PBI) approaches have been proven to be particularly effective for conducting large-scale approximate metric searching. These methods rely on the idea of transforming the original metric objects into permutation representations, which can be efficiently indexed using data structures such as inverted files.

The standard conceptualization of permutation associated with a metric object involves only the use of object distances and their relative orders from a set of anchors called pivots. In this paper, we generalized this definition in order to enlarge the class of permutation representations that can be used by PBI approaches. In particular, we introduced the concept of permutation induced by a space transformation and a sorting function, and we investigated which properties these transformations should possess to produce permutations that are effective for metric search. Furthermore, as a practical outcome, we defined a new type of permutation representation that is calculated using distances from pairs of pivots. This proposed technique allowed us to produce longer permutations than traditional ones for the same number of object-pivot distance calculations. The advantage lies in the fact that when longer permutations are employed, the use of inverted files built on permutation prefixes leads to greater efficiency in the search phase.

Keywords: Permutation-Based Indexing, Metric Space, Metric Search, Similarity Search, Approximate search, Planar projection.

1. Introduction

The paradigm of metric spaces offers an elegant way to approach the problem of similarity search. The clear advantage of this approach is its flexibility, as it is not tied to a particular data type and thus encompasses a wide variety of applications such as multimedia information retrieval, pattern recognition, data mining, and computational biology. To this end, the *Metric Search* framework [1] allows one to develop index structures by employing only the knowledge of a predefined black-box distance between objects.

A *metric space* is defined as a pair (D, d), where D represent the domain and $d : D \times D \to \mathbb{R}$ is a distance function that satisfies the metric postulates of *non-negativity*, *identity of indiscernibles*, *symmetry*, and *triangle inequality* [1]. The downside of this paradigm is the inability to use any algebraic operations, such as sum, mean, or scalar product of two objects, but we are constrained to use only the distance between pairs of objects. Therefore, it is often desirable to map a metric object $o \in D$ to another (more tractable) space, such as a vector space. This mapping must rely only on algorithms that use the distances of the object o from other metric objects, typically a fixed set of selected reference objects within the space.

Beyond rare exceptions, exact metric search suffers from the problem of the *curse of dimensionality* [2], which causes objects in the space to tend to have the same distance as the dimension grows making any attempt to organize them into indexes useless since their performance may be no better than a sequential scan for spaces with high intrinsic dimensionality [3, 4].

To mitigate this problem, researchers have turned

*Corresponding author

Email addresses: lucia.vadicamo@isti.cnr.it (Lucia Vadicamo), giuseppe.amato@isti.cnr.it (Giuseppe Amato), claudio.gennaro@isti.cnr.it (Claudio Gennaro)

their attention to the study of approximate indexes. Successful examples of approximate methods are the *Permutation-based Indexing* (PBI) techniques that transform the metric data into permutations of a set of integers $\{1, \ldots, N\}$. The advantage of working with permutations arises from the fact that permutations can be efficiently indexed and searched using data structures such as prefix trees [5, 6] and inverted files [7, 8]. The similarity search is then performed in the permutation space and no longer in the original space. Therefore, the effectiveness of a permutation-based access method depends not only on its particular indexing and searching algorithm, but also on the "quality" of the permutations used to represent the objects. In other words, similar objects should be mapped to similar permutations and, vice versa, similar permutations should correspond to similar objects.

The original definition of permutation-based representation of a metric object was derived by computing the distances of the object to a set of *pivots* (reference objects) and then by reordering the pivot identifiers according to these distances. The initial introduction of this idea can be attributed to [9] and it has since been further developed by [10, 11]. Then this characterization has been adopted in several research papers that further investigated the properties of these data representations and ways to efficiently index them, e.g. [3, 5, 6, 12, 13, 14, 15, 16, 17]. Moreover, some alternative permutation-based representations were defined in the literature [18, 19], but only for representing objects of specific metric spaces.

In this work, we generalize the definition of permutation associated with a metric object. We introduce the concept of *permutations induced by a space transformation* $f : (D, d) \rightarrow \mathbb{R}^N$, where f is a function that projects the metric objects of D into an N-dimensional vector space. This function typically relies only on some distance calculations to transform the objects, such as the distances to a set of pivots as done in traditional permutation. However, the way these distances are combined and exploited to represent objects may be different from what is done in the traditional approach. We believe that this generalization can open up new lines of research, on the one hand, to understand theoretically what properties the function f should have in order to generate permutations that have good performance for the approximate search, and on the other hand, to define alternative permutation-based representations. In this paper, whose preliminary version appeared in [20], we start investigating both these aspects. We show that a fundamental property of the function f is producing coordinate values with nearly identical distribution. Moreover, we define a permutation-representation that is derived using both the distances of each object to a set of pivots as well as the distances between pivot pairs. In this way, for a fixed set of n pivots, we can generate permutations from a larger set $N > n$ of permutants, with respect to the traditional permutation-based approach where permutants were n. An expanded set of permutants enhances efficiency during search operations, especially when using an inverted index built upon permutation prefixes (e.g., MI-File [7]). In fact, the inverted index contains as many posting lists as the number N of permutants (i.e., the length of the full permutation) and so, for a fixed permutation prefix length λ, the higher N, the shorter the posting lists, and hence the smaller the fraction of the database accessed to answer a query.

The remainder of this paper is structured as follows: Section 2 provides an overview of basic concepts and related work concerning permutations used in approximate metric search. Section 3 presents our generalization of the concept of permutation-based representation associated to a metric object, namely, permutation induced by a space transformation. Section 4 investigates, both theoretically and experimentally, the properties a space transformation should possess to induce permutations suitable for metric search. Section 5 presents a novel permutation representation, called Pivot Pair permutation, which is built using both object-pivot and pivot-pivot distances. It also reports experiments on both real-world and synthetic datasets. Finally, Section 6 draws the conclusions.

The notations used throughout this manuscript are summarized in Table 1.

2. Background and Related Work

The original definition of a permutation-based representation of a metric object, introduced by Chávez et al . [9, 11] and further adopted by Amato et al. [10, 7], relies on ordering the identifiers of a fixed set of pivots, used as representative data objects, according to their distances to the object to be represented. Formally, these permutations[1] can be defined as follows

[1] In this work, we use the term "permutation" to refer

Table 1: Notation used throughout this paper

Symbol	Definition
$\lvert\cdot\rvert$	size of a set
(D, d)	metric space, where D is a data domain and $d : D \times D \to \mathbb{R}$ is a metric
$(S_N, \tilde{d})$	permutation metric space, where S_N is the symmetric group on the set $\{1, \ldots, N\}$, and $\tilde{d}$ is a distance function on S_N
ℓ_p, ℓ_2, ℓ_∞	Minkowski distance, Euclidean distance, and Chebyshev distance
$\tilde{d}_\rho, \tilde{d}_{\rho,\lambda}$	Spearman's rho and Spearman's rho with location parameter λ
$\{p_1, \ldots, p_n\}$	set of pivots, $p_i \in D$
n	number of pivots
m	number of pivot pairs
N	length of permutations (number of permutants)
λ	permutation prefix length (location parameter)
o, o_i, q	data objects, $o, o_i, q \in D$
$\mathbf{x}, \mathbf{x}', \mathbf{v}$	real-valued vectors
$f = [f_1, \ldots, f_n]$	space transformation $f : (D, d) \to \mathbb{R}^N$, $f(o) = [f_1(o), \ldots, f_N(o)]$ with $f_i : (D, d) \to \mathbb{R}$ and $f_i(o) = (f(o))_i$
ξ	generic sorting function on N-dimensional vectors
$\xi_\uparrow, \xi_\downarrow$	sorting functions in ascending and descending order
ϕ_{p_i,p_j}	planar projection $\phi_{p_i,p_j} : (D, d) \to (\mathbb{R}^2, \ell_2)$ given the pivots p_i, p_j [21]
Π_o, Π_o^{-1}	permutation and inverted permutation of a metric object $o \in D$
$\Pi_o^{f,\xi}$, $(\Pi_o^{f,\xi})^{-1}$	permutation and inverted permutation of $o \in D$ induced by a space transformation f and a sorting function ξ
Π_o^f , $(\Pi_o^f)^{-1}$	permutation and inverted permutation of $o \in D$ induced by a space transformation f and the sorting in ascending order
$\Pi_{o,\lambda}$, $\Pi_{o,\lambda}^{-1}$	permutation prefix of length λ, and its inverse
$\Pi_{o,\lambda}^f$, $(\Pi_{o,\lambda}^f)^{-1}$	prefix of length λ of the permutation induced by f, and its inverse
$\boldsymbol{\mathcal{X}}, \mathcal{X}_i$	multivariate and univariate continuous random variables
$\mathcal{N}(\mu, \sigma)$	Gaussian distribution with mean μ and variance σ^2
k, k'	number of results of a nearest neighbour search

Definition 1 (Permutation of a metric object given a set of pivots)**.** *For a metric space (D, d), the permutation-based representation Π_o (briefly* permutation*) of an object $o \in D$ with respect to the pivot set $\{p_1, \ldots, p_n\} \subset D$ is the sequence $\Pi_o = [\pi_1, \ldots, \pi_N]$, where $N = n$, that lists the pivot identifiers $\{1, \ldots, n\}$ (called* permutants*) in an order such that $\forall i \in \{1, \ldots, n-1\}$*

$$d(o, p_{\pi_i}) < d(o, p_{\pi_{i+1}}) \tag{1}$$

or

$$\left[d(o, p_{\pi_i}) = d(o, p_{\pi_{i+1}})\right] \wedge \left[\pi_i < \pi_{i+1}\right]. \tag{2}$$

This representation is also referred to as the *full-length* permutation to distinguish it from the *permutation prefix* adopted in several PBI methods [7, 5, 6]. In fact, based on the intuition that the most relevant information in the permutation is present in its very first elements, i.e., the identifiers of the closest pivots to an object, several researchers proposed to represent the data by using a fixed-length prefix of the permutation: $\Pi_{o,\lambda} = [\pi_1, \ldots, \pi_\lambda]$ with $\lambda < N$, usually referred to as *permutation prefix* or *truncated permutation of length* λ. The use of permutation prefixes may be dictated by either the employed data structure (e.g., prefix tree [5]), efficiency issues (more compact data encoding and better performance when using inverted files [7]), or even by effectiveness reasons (in certain cases, using prefixes yields superior results compared to full-length permutations [12, 7]).

Several metric functions have been proposed in the literature to assess the similarity of two permutations. Notable examples include Kendall's tau, Spearman's rho, and the Spearman's Footrule distances [22, 23]. The permutation prefixes are usually compared using *top-λ distances* proposed by Fagin et al. [24], including the Spearman's rho and the Spearman's Footrule with location parameter λ. Most of these distances can be easily computed as distances between Cartesian points obtained by considering the so-called *inverted permutations*.

The inverse of a full-length permutation Π_o is another permutation denoted as $\Pi_o^{-1} =$

briefly to a permutation-based representation of a data object. It is essential to clarify that mathematically, a permutation is merely an arrangement or ordering of elements within a finite set. Therefore, when we mention a "new" or "different" permutation of a metric object, we are referring to the methodology employed to derive the final permutation representation.

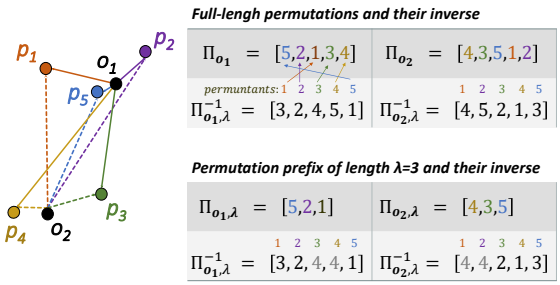


Figure 1: Example of traditional full-length and prefix permutations, along with their inverses, associated to two data objects given five pivots. (Best view in color)

$[\Pi_o^{-1}(1), \ldots, \Pi_o^{-1}(N)]$ where $\Pi_o^{-1}(i)$ represents the position of the permutant i in the permutation Π_o. For example, if $\Pi_o = [5, 2, 1, 3, 4]$ then $\Pi_o^{-1} = [3, 2, 4, 5, 1]$ because the permutant "1" is in third position in the permutation Π_o, the permutant "2" is in the second position, the permutant "3" is in the fourth position, and so on. Note that the value at the i-th coordinate in Π_o corresponds to the *permutant identifier* at rank i, while the value at the j-th coordinate in the inverted permutation Π_o^{-1} represents the *rank* of the permutant "j" in the permutation Π_o.

A unique definition of the inverse of a permutation prefix does not exist because it is impossible to determine the exact rank of the permutants that are not appearing in the given permutation prefix. For instance, the inverse of $\Pi_{o,\lambda} = [5, 2, 1]$ should have the form $[3, 2, *, *, 1]$ where the values "$*$" cannot be uniquely determined. To address this, we follow the convention of assigning the rank $\lambda + 1$ to those permutants. Therefore, we denote the *inverted permutation prefix* $\Pi_{o,\lambda}^{-1}$ as the integer vector $[\Pi_{o,\lambda}^{-1}(1), \ldots, \Pi_{o,\lambda}^{-1}(N)]$ where

$$\Pi_{o,\lambda}^{-1}(i) = \begin{cases} \Pi_o^{-1}(i) & \text{if} \quad \Pi_o^{-1}(i) \leq \lambda \\ \lambda + 1 & \text{otherwise} \end{cases} \tag{3}$$

An example of permutations, inverted permutations, and permutation prefixes for two data objects and five pivots is illustrated in Figure 1.

By using the inverted permutations we can easily compute the Spearman's rho ($\tilde{d}_\rho$) and the Spearman's rho with location parameter λ ($\tilde{d}_{\rho,\lambda}$) as the Euclidean distances between two vectors:

$$\tilde{d}_\rho(\Pi_{o_1}, \Pi_{o_2}) = \ell_2(\Pi_{o_1}^{-1}, \Pi_{o_2}^{-1}) \tag{4}$$

$$\tilde{d}_{\rho,\lambda}(\Pi_{o_1}, \Pi_{o_2}) = \ell_2(\Pi_{o_1,\lambda}^{-1}, \Pi_{o_2,\lambda}^{-1}). \tag{5}$$

Note that if $\lambda = N$ then $\tilde{d}_{\rho,\lambda} = \tilde{d}_\rho$.

Other permutation-based representations. In the literature, some alternative permutation-based representations were defined for specific metric spaces. For example, the *Deep Permutations* [18, 25] are computed by reordering the dimensions of a vector according to the corresponding element values. This approach is limited to vector spaces and has so far only been tested on deep features, i.e., Euclidean data descriptors obtained as the output of an intermediate layer of a deep neural network. The *SPLX-Perms* [19] approach uses the n-Simplex projection [26] to transform a metric object into a Euclidean vector and then computes the permutation by reordering the components of the vector as done in the Deep Permutations. As a difference with respect to the Deep Permutations technique, the SPLX-Perms approach also uses a random rotation to distribute the variance before performing the reordering of the vector components. This method can be used on the large class of spaces meeting the n-point property [26], but it is not applicable to general metric spaces.

It is worth noting that the permutation prefixes and the inverted permutations can be computed as described above for these alternative permutation-based representations as well.

3. Permutations Induced by a Space Transformation

In the previous section, we summarized the permutation-based representations that have been proposed so far in the literature in the context of metric search. Figure 2 provides an overview of the steps involved in calculating each of these representations. Our observation is that all these approaches belong to the same family of transformations, as explained hereafter. As a result, the traditional definition of permutation associated with a metric object (Def. 1) could be generalized to be more inclusive. In this context, the first trivial but useful observation to make is that any sorting function defined on a finite-dimensional Coordinate space implicitly produces a permutation representation of the data. Suppose that $\xi : \mathbb{R}^N \to \mathbb{R}^N$ is a function that sorts the coordinate elements of a N-dimensional real vector with respect to a predefined criterion (e.g., ascending order). For any vector $\mathbf{x} \in \mathbb{R}^N$, the sorting function ξ is described by the permutation $\Pi_{\mathbf{x}}^{\xi}$ of the indices $\{1, \ldots, N\}$

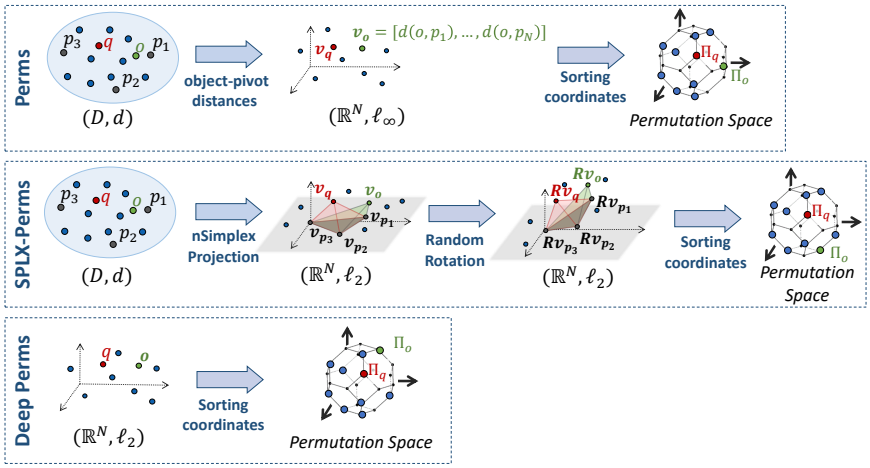


Figure 2: Illustration of the steps needed to compute the traditional permutations (top) [11], the SPLX-Perms (middle) [19], and the Deep Permutations (bottom) [18]. (Best view in color)

that specifies the arrangement of the elements of $\mathbf{x}$ into $\mathbf{x}' = \xi(\mathbf{x})$. Specifically, if $\mathbf{x} = [x_1, \ldots, x_N]$ and $\xi(\mathbf{x}) = [x_{i_1}, \ldots, x_{i_N}]$ then $\Pi^{\xi}_{\mathbf{x}} = [i_1, \ldots, i_N]$. In other words, the j-th element of the permutation $\Pi^{\xi}_{\mathbf{x}}$ corresponds to the index $i \in \{1, \ldots, N\}$ such that the i-th element of $\mathbf{x}$ is equal to the j-th element of $\xi(\mathbf{x})$. For example, if $\mathbf{x} = [8, 10, 6]$ and $\xi(\mathbf{x}) = [6, 8, 10]$ then $\Pi^{\xi}_{\mathbf{x}} = [3, 1, 2]$ because (i) the first element of $\xi(\mathbf{x})$ is 6 which is at the *third* coordinate position in the vector $\mathbf{x}$; (ii) the second element of $\xi(\mathbf{x})$ is equal to the *first* element of the vector $\mathbf{x}$; (iii) the third element of $\xi(\mathbf{x})$ is equal to the *second* element of the vector $\mathbf{x}$. However, note that this characterization is not well defined when the vector $\mathbf{x}$ contains duplicate values. Therefore, we give the following definition:

Definition 2 (Permutation of a real vector induced by a sorting function ξ)**.** *A permutation representation of a vector* $\mathbf{x} = [x_1, \ldots, x_N] \in \mathbb{R}^N$ *induced by a given sorting function* $\xi : \mathbb{R}^N \to \mathbb{R}^N$ *is the permutation* $\Pi^{\xi}_{\mathbf{x}} = [\pi_1, \ldots, \pi_N]$ *of the index identifiers* $\{1, \ldots, N\}$ *such that for any* $j = 1, \ldots, N$ *the element* π_j *is the smallest index for which* x_{π_j} *equals the* j*-th element of* $\xi(\mathbf{x})$*.*

The Deep Permutations can be formalized by using the above definition employing a descending order sort, although this characterization cannot be applied to describe the SPLX-perms or the traditional permutations. Nevertheless, these approaches share a common idea, which is using the distance to a set of pivots to first transform the metric object into a Cartesian coordinate space and then obtain the permutation by ordering the components of the resulting vectors. Therefore, for any function $f : (D, d) \to \mathbb{R}^N$ and a given *sorting* function we define a permutation representation of metric objects as follows:

Definition 3 (Permutation of a metric object induced by a space transformation f and a sorting function ξ)**.** *Let* $f : (D, d) \to \mathbb{R}^N$ *a space transformation, and* $\xi : \mathbb{R}^N \to \mathbb{R}^N$ *a function that sorts the components of a* N*-dimensional vector according to some predefined criteria. We define the permutation representation of a metric object* $o \in D$ *induced by the functions* f *and* ξ *as the permutation* $\Pi^{f,\xi}_{o} = [\pi_1, \ldots, \pi_N]$ *that lists the index identifiers* $\{1, \ldots, N\}$ *in an order such that for any* $j = 1, \ldots, N$ *the permutant* π_j *is the smallest index for which the* π_j*-th element of the vector* $f(o)$ *is equals to the* j*-th element of* $\xi(f(o))$*, i.e.,* $f(o)_{\pi_j} = \xi(f(o))_j$*.*

Figure 3 schematizes the idea behind this class of permutations and gives an example using the sorting in ascending order. Please note that the effect of using a different sorting function in most cases could be reproduced by changing the function f. For example, for a given f and object o the permutation obtained by sorting $f(o)$ in descending order is equal to the permutation obtained by applying the function $-f$ to the object o and then sorting the elements in ascending order. More generally, let $\xi_{\uparrow}$ and $\xi_{\downarrow}$ sorting functions in ascending and descending order, respectively, it can be easily proved that

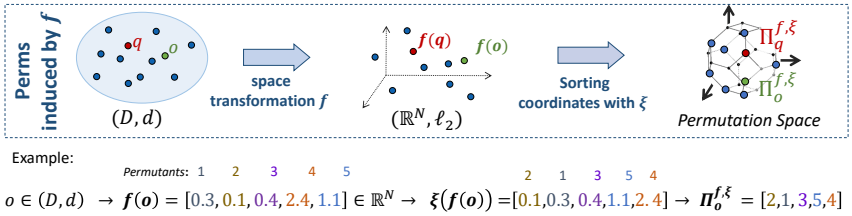


Figure 3: Illustration of the steps needed to compute a permutation induced by a space transformation f and a sorting function ξ. (Best view in color)

- if $g : \mathbb{R}^N \to \mathbb{R}^N$ is a monotonic increasing function then
$$\Pi_o^{(g\circ f),\xi_\uparrow} = \Pi_o^{f,\xi_\uparrow}$$
$$\Pi_o^{(g\circ f),\xi_\downarrow} = \Pi_o^{f,\xi_\downarrow}$$
for any $o \in D$;
- if $g : \mathbb{R}^N \to \mathbb{R}^N$ is a monotonic decreasing function then
$$\Pi_o^{(g\circ f),\xi_\uparrow} = \Pi_o^{f,\xi_\downarrow}$$
$$\Pi_o^{(g\circ f),\xi_\downarrow} = \Pi_o^{f,\xi_\uparrow}$$
for any $o \in D$.

For the sake of simplicity, in the following, we assume to use the sorting in *ascending* order and we omit the dependency of the sorting function ξ in the definition of the permutation. Therefore, we use the following characterization:

Definition 4 (Permutation of a metric object induced by a space transformation f)**.** *The permutation representation of a metric object $o \in (D, d)$ with respect to the transformation $f : (D, d) \to \mathbb{R}^N$ is the sequence $\Pi_o^f = [\pi_1, \ldots, \pi_N]$ that lists the permutants $\{1, \ldots, N\}$ in an order such that $\forall\, i \in \{1, \ldots, N-1\}$,*

$$f(o)_{\pi_i} < f(o)_{\pi_{i+1}} \tag{6}$$

or

$$\left[f(o)_{\pi_i} = f(o)_{\pi_{i+1}}\right] \wedge \left[\pi_i < \pi_{i+1}\right] \tag{7}$$

where $f(o)_j$ indicates the j-th coordinate value of the vector $f(o)$.

Note that, according to this definition, the traditional permutation is induced by the transformation $f(o) = [d(o, p_1), \ldots, d(o, p_N)]$, where $\{p_1, \ldots, p_N\}$ is a fixed set of pivots. The Deep Permutation is induced by $f = -Id$, where Id is the identity function. The SPLX-Perm, instead, is induced by the composition of the n-Simplex projection and a random rotation. Moreover, this generalization suggests that new permutation representations of generic metric objects can be defined but assuming that we use a transformation $f : (D, d) \to \mathbb{R}^N$ that relies only on distance computations and metric postulates to transform the objects. In some cases, the function f can also be generated using machine learning techniques, as in [27]. Nevertheless, for particular metric spaces, e.g., vector spaces, other operations or properties of the space can be employed when defining the transformation f. However, as discussed in the next section, not all the transformations may produce permutations that are suitable for metric search, as we would like similar objects to be projected into similar permutations.

4. Desired Properties of the Space Transformation f

In this section, we aim at investigating the properties that the function $f : (D, d) \to \mathbb{R}^N$ should possess in order to induce permutations suitable for approximate metric search. For simplicity, we denote the vector-valued function f by means of its coordinate functions $f_i : D \to \mathbb{R}$, $i \in \{1, \ldots, N\}$, such that

$$f(o) = [f_1(o), \ldots, f_N(o)]. \tag{8}$$

To make a parallel with the notation used in previous sections, note that each f_i is a real-valued function such that $f_i(o)$ is equal to the i-th component of the vector $f(o)$, i.e., $f_i(o) = f(o)_i$, for all $o \in D$ and $i \in \{1, \ldots, N\}$.

For the analysis, we assume that the metric used to assess the similarity of two permutations does not involve cross-correlation coefficients between different permutants. Specifically, we consider met-

rics of the form

$$\tilde{d}_p(\Pi_{o_1}, \Pi_{o_2}) = \left(\sum_{j=1}^{N} |\Pi_{o_1}^{-1}(j) - \Pi_{o_2}^{-1}(j)|^p \right)^{1/p} \quad (9)$$

where $p = 1$ and $p = 2$ correspond to the commonly used Spearman's Footrule and Spearman's rho distances, respectively. This choice guarantees that the distances are preserved if we "re-label" (i.e., permute) the coordinate functions of f, that is

$$\tilde{d}_p(\Pi_{o_1}^f, \Pi_{o_2}^f) = \tilde{d}_p(\Pi_{o_1}^{\hat{f}}, \Pi_{o_2}^{\hat{f}}) \quad (10)$$

for any vector-valued function $\hat{f}$ such that $\hat{f}(o) = [f_{i_1}(o), \ldots, f_{i_N}(o)]$ where $[i_1, \ldots, i_N]$ is a permutation of $[1, \ldots, N]$. With this assumption, we can infer that the permutations $\Pi_{o_1}^f$ and $\Pi_{o_2}^f$ will be more similar when the vectors $f(o_1), f(o_2) \in \mathbb{R}^N$ exhibit a similar ranking of their values across the different dimensions. In other words, if the largest and smallest values of the vectors are located in the same dimensional components, the permutations will be more similar. Clearly, as discussed in Sect. 4.2, the function f should also somehow provide an approximation of the original distance so that $f(o_1), f(o_2) \in \mathbb{R}^N$ are more similar when the metric objects o_1, $o_2 \in D$ are closer according to the distance function d.

4.1. On the distribution of the values of the function f

When considering permutations generated by sorting the components of the vectors $[f_1(o), \ldots, f_N(o)]$, $o \in D$, the various dimensional components of these vectors should ideally have the same distribution in order to obtain distinctive permutations that are representative of the data objects. In particular, we show that the ideal case involves having coordinate functions f_i whose values are independent and identically distributed (iid).

Suppose $\boldsymbol{\mathcal{X}} = [\mathcal{X}_1, \ldots, \mathcal{X}_N]$ be a random variable representing all possible outcomes $f(o) = [f_1(o), \ldots, f_N(o)]$, where $o \in D$. If the mean of distributions $\{f_i(o), o \in D\}$ varies by i and $\mathbb{E}[\mathcal{X}_i] << \mathbb{E}[\mathcal{X}_j]$, we would expect index "i" to precede index "j" in the permutations of almost all the data objects [2]. More generally, the differences in the distribution of the variables $\mathcal{X}_i$ affect the probability distribution of the permutation representations.

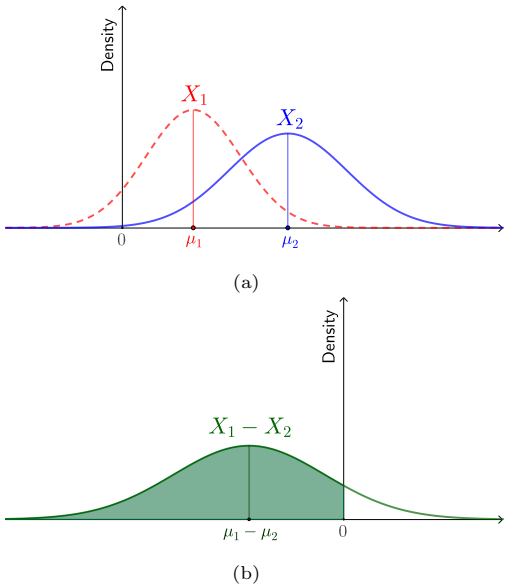


Figure 4: (a) Probability density function of two Gaussians $\mathcal{X}_1 \sim \mathcal{N}(\mu_1, \sigma_1)$ and $\mathcal{X}_2 \sim \mathcal{N}(\mu_i, \sigma_i)$ (b) Probability density function of the difference $\mathcal{X}_1 - \mathcal{X}_2$, which is the density of a Gaussian with mean $\mu_1 - \mu_2$. $P(\mathcal{X}_1 < \mathcal{X}_2)$ is the shaded area under the PDF curve of $\mathcal{X}_1 - \mathcal{X}_2$. (Best view in color)

This concept can be illustrated with a case in which all variables $\mathcal{X}_i$ follow a normal distribution with $\mathcal{X}_i \sim \mathcal{N}(\mu_i, \sigma_i)$ for $i = 1, \ldots, N$. In this case, the probability that index "1" precedes index "2" in the permutation is given by

$$P(\mathcal{X}_1 < \mathcal{X}_2) = P(\mathcal{X}_1 - \mathcal{X}_2 < 0),$$

where $\mathcal{X}_1 - \mathcal{X}_2$ follows a Gaussian distribution[3]. Figure 4 shows an example of this case, where the shaded area under the PDF curve of $\mathcal{X}_1 - \mathcal{X}_2$ corresponds to the probability $P(\mathcal{X}_1 < \mathcal{X}_2)$. When the mean values μ_1 and μ_2 are equal, the probability $P(\mathcal{X}_1 < \mathcal{X}_2)$ is 0.5, resulting in balanced rankings of indices "1" and "2" in the permutations of all data objects. If $\mu_1 << \mu_2$ or $\mu_2 << \mu_1$, the probability $P(\mathcal{X}_1 < \mathcal{X}_2)$ will be close to 1 or 0, respectively. This means that index "1" will almost always precede index "2" in the permutations, or vice versa.

Although not directly evident in the case of two Gaussian variables, the variance also plays a crucial

[2] We assume the *ascending* order is used to generate the permutation of o by reordering the coordinates of $f(o)$ In this case, the i-th and j-th components of f alone provide minimal information to distinguish different objects.

[3] $\mathcal{X}_1 - \mathcal{X}_2$ follows a Gaussian distribution because it is the difference of two Gaussian variables. Specifically, if $\mathcal{X}_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$ and $\mathcal{X}_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$, then $\mathcal{X}_1 - \mathcal{X}_2 \sim \mathcal{N}(\mu, \sigma^2)$, where $\mu = \mu_1 - \mu_2$ and $\sigma^2 = \sigma_1^2 + \sigma_2^2 - 2\rho_{12}\sigma_1\sigma_2$. Here, ρ_{12} represents the correlation between $\mathcal{X}_1$ and $\mathcal{X}_2$.

This is a pre-copyedit version of this article. The final authenticated version is available online at: https://doi.org/10.1016/j.is.2023.102286.

role in determining the distribution of the permutations. For example, when considering independent normal variables $\mathcal{X}_i$, if all the $\mathcal{X}_i$ have the same distribution, all permutations have the same probability of occurring. However, if the variables $\mathcal{X}_i$ have the same mean but different variances, a non-uniform distribution of induced permutations arises (proof in Appendix A).

These observations extend to non-Gaussian distributions, where different statistics for the $\mathcal{X}_i$ result in a non-uniform distribution of probabilities for induced permutations. The Gaussian case provides a tangible example of the effects of having different means and variances.

The statement that all permutations are equiprobable if the components of the vectors are iid random variables can be formalized as follows:

Proposition 1. *The induced permutations are uniformly distributed (i.e., every permutation occurs with probability $1/n!$) if and only if the continuous random variables $\mathcal{X}_1, \ldots, \mathcal{X}_n$ are independent and identically distributed.*

Proof. We first observe that having $\mathcal{X}_1, \ldots, \mathcal{X}_n$ iid is a necessary condition. In fact, if the variables were not iid, it is possible to construct counterexamples where the induced permutations are not uniformly distributed. The case of Gaussians with different variances studied in the Appendix A, in fact, provides a counterexample (Equation A.1 when $\sigma_1 \geq \cdots \geq \sigma_N$).

Let us now prove the sufficiency. Let $h(x)$ the probability density function of the considered iid variables and $[\![\cdot]\!]$ the Iverson bracket, which equals one if the argument is true, and zero otherwise. For any permutation $[\pi_1, \ldots \pi_n]$ of $[1, \ldots, N]$, we have $P(\mathcal{X}_{\pi_1} < \cdots < \mathcal{X}_{\pi_n}) = P(\mathcal{X}_1 < \cdots < \mathcal{X}_n) := p$, which can be seen from the following steps:

$$\int \cdots \int [\![x_{\pi_1} < \cdots < \pi_{x_n}]\!] \prod_{i=1}^{N} f(\pi_{x_i}) dx_{\pi_1} \ldots dx_{\pi_n}$$
$$\underset{y_i = x_{\pi_i}}{=} \int \cdots \int [\![y_1 < \cdots < y_n]\!] \prod_{i=1}^{N} f(y_i) dy_1 \ldots dy_n$$

Furthermore, since $\mathcal{X}_i$ are continuous random variables, we have $P(\mathcal{X}_i = \mathcal{X}_j) = 0$ for all $i \neq j$. Therefore, the event $A = \{\exists i, j \text{ with } i \neq j \text{ s.t. } \mathcal{X}_i = \mathcal{X}_j\}$ has probability zero since $P(A) \leq \sum_{i \neq j} P(\mathcal{X}_i = \mathcal{X}_j) = 0$. This implies that the complementary event $\{\forall i \neq j, \mathcal{X}_i \neq \mathcal{X}_j\}$ has probability one. As a result, we can write:

$$\begin{aligned} 1 =& P(\forall i \neq j, \mathcal{X}_i \neq \mathcal{X}_j) \\ =& \sum_{\pi \in S_N} P(\mathcal{X}_{\pi_1} < \cdots < \mathcal{X}_{\pi_n}) \\ =& \sum_{\pi \in S_N} p = n!p \end{aligned} \tag{11}$$

which proves that $p = 1/n!$. Thus, the induced permutations are uniformly distributed when $\mathcal{X}_1, \ldots, \mathcal{X}_n$ are iid. □

Our interest in the probability distribution of the induced permutations stems from the understanding that this probability significantly affects our ability to distinguish different data objects when transformed into the permutation space, and it may highly influence the performance of a given permutation-based index as well. For example, when using inverted files, if some permutation prefixes are much more frequent than others, some posting lists will be much longer than others. In the worst case, a posting list may contain the entire dataset and thus search performance may degenerate to a sequential scan.

While it may not always be feasible, the optimal scenario is to achieve a uniform distribution of induced permutations. However, it is worth noting that in practical cases the coordinate functions f_i may not be independent or identically distributed. For example, the distributions of the functions $f_i(o) = d(o, p_i)$ used by the traditional permutation approach cannot be independent due to constraints derived by the triangle inequality.

4.1.1. Geometrical interpretation

All these aspects can also be viewed from a geometric perspective using high-order Euclidean Voronoi diagrams. The main intuition here is to observe that there is an equivalence between the permutation induced by a function f, i.e., Π_o^f, and the "traditional" permutations $\Pi_{f(o)}$ (i.e., the one based on the object-pivot distances as in Def 1) computed after transforming the original metric objects through the function f and using a specific set of N pivots of a N-dimensional *Euclidean* space:

Proposition 2. *Let (D, d) a metric space, $f : (D, d) \to \mathbb{R}^N$ a space transformation, and Π_o^f the permutation of $o \in D$ induced by f using the sorting in ascending order.*
For any real number $c > 0$, the traditional permutation $\Pi_{f(o)}$ associated to the vector $f(o) \in \mathbb{R}^N$

with respect to the pivots $\hat{p}_1 = [-c, 0, \ldots, 0]$, $\hat{p}_2 = [0, -c, \ldots, 0]$, $\ldots$, $\hat{p}_N = [0, \ldots, 0, -c] \in \mathbb{R}^N$ *and the Euclidean distance is equal to* Π_o^f*:*

$$\Pi_o^f = \Pi_{f(o)} \tag{12}$$

Proof. For the given pivots $\{\hat{p}_1, \ldots, \hat{p}_N\}$ and the Euclidean distance, by definition, we have that $\Pi_{f(o)}$ is obtained by sorting the element of the vector $[\ell_2(f(o), \hat{p}_1), \ldots, \ell_2(f(o), \hat{p}_N)]$ in ascending order. For any $i = 1, \ldots, N$ we have that

$$\begin{aligned} \ell_2(f(o), \hat{p}_i) &= \sqrt{\|f(o)\|^2 + \|\hat{p}_i\|^2 - 2f(o)^T \hat{p}_i} \\ &= \sqrt{K + 2cf_i(o)} \end{aligned} \tag{13}$$

where $K = \|f(o)\|^2 + c^2$ is a constant that do not depend from the index i. It follows that

$$\begin{aligned} &[\ell_2(f(o), \hat{p}_1), \ldots, \ell_2(f(o), \hat{p}_N)] \\ &\quad = [g(f_1(o)), \ldots, g(f_N(o))] \end{aligned} \tag{14}$$

where $g(x) = \sqrt{K + 2cx}$. Since g is a monotonic increasing function we have that $g(f_i(o)) < g(f_j(o))$ if and only if $f_i(o) < f_j(o)$, and thus the permutation induced by ranking the elements in Eq. 14 in ascending order equals the permutation Π_o^f induced by ranking the element in $[f_1(o), \ldots, f_N(o)]$. In other words, $\Pi_{f(o)} = \Pi_o^{g \circ f} = \Pi_o^f$. □

If we divide the entire space $(\mathbb{R}^N, \ell_2)$ using the $(N-1)$-order Voronoi Partitioning [28, 29] associated to the specific pivots $\hat{p}_1, \ldots, \hat{p}_N$ we obtain exactly $N!$ cells corresponding to all the possible permutation of $\hat{p}_1, \ldots, \hat{p}_N$. This is due to the specific choice of these pivots.[4]. The boundaries of these cells are determined by the intersection of the bisectors between pivots pairs, which in our case are simply the hyperplanes of the form

$$H_{i,j} = \{[x_1, \ldots, x_N] \in \mathbb{R}^N : x_i = x_j\}, \quad \forall i \neq j$$

Therefore. any point with constant values $\gamma * [1, \ldots, 1]$ is in the intersection of all these hyperplanes, which means that even for very small values $\epsilon_1 < \epsilon_2 < \cdots < \epsilon_N$, any permutation $[\pi_1, \ldots, \pi_N]$ can be generated by considering the permutation $\Pi_{\mathbf{x}}$ associated to the vector $\mathbf{x} = [\gamma, \ldots, \gamma] + [\epsilon_{\pi_1}, \ldots, \epsilon_{\pi_N}]$, for any $\gamma \in \mathbb{R}$.

However, in our study, we are interested only in the permutation associated with the subset $f(D) \subset \mathbb{R}^N$. So if the codomain of f is uniformly distributed around a constant point $[\gamma, \ldots, \gamma]$ then its convex hull will intersect all the boundaries of the Voronoi cells, and thus all the possible distance permutations could, in theory, be generated. In a real scenario, it is difficult to have a perfect uniform distribution of $f(D)$ around a constant value. Nevertheless, it is important to keep in mind that the expressiveness of the generated permutations (in terms of equal probability to generate every possible permutation) is related to how much the expected value $\mathbb{E}[f(o)]$ deviates from a constant vector of the form $\gamma * [1, \ldots, 1]$ and how much all other statistics of the variables representing the values $f_i(o)$ differs across the coordinate index i.

4.1.2. Experimental evaluation

Let us now examine some experimental results which, in line with the observations made so far, will show the importance of having coordinates in f with an identical (or nearly identical) distribution to induce permutations that are effective for metric search.

To avoid introducing any approximation from the choice of a particular function f, we consider the case in which the starting space is already an Euclidean space. As a first base case, we analyze the choice of f as the identity function. The objective is to confirm that when the vector coordinates exhibit differing statistical properties, the permutations induced by just sorting the coordinates (i.e., using $f = Identity$) will perform worse than the permutations induced by a function that generates output components with similar distributions.

Setup and Evaluation Protocol. For this set of experiments, we used the *SISAP Colors* [31] benchmark, a real-world and publicly available dataset comprising approximately 113K feature vectors of dimensions 112. Each vector is a color histogram of a medical image. We compared these vectors using the Euclidean distance and we built a ground-truth for the exact k-nearest neighbors (k-NN) search related to 1,000 randomly selected queries[5]. The

[4] Note that as proved by Skala [30] for k pivots in a d dimensional Euclidean space the number of cells that occur in generalized Voronoi diagram on L_p spaces (i.e., the number of distinct distance permutations that can be generated) may be less than $k!$. However, if $k \leq d+1$, as in our case, k pivots can be chosen such that all $k!$ distinct distance permutations exist.

[5] The query objects were removed from the ground-truth.

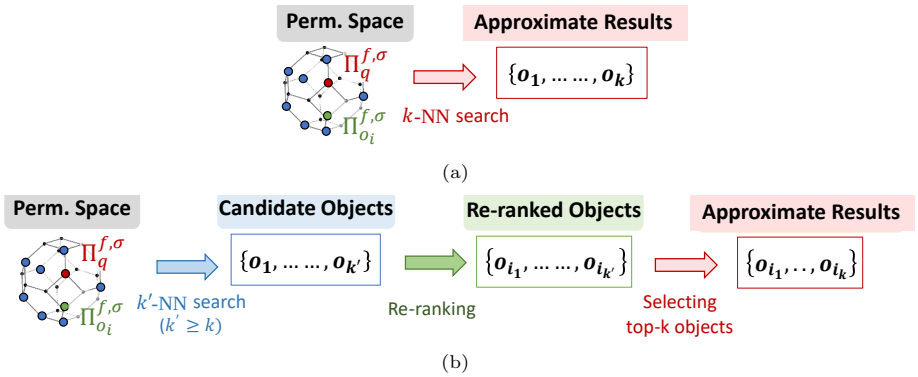


Figure 5: Illustration of how the permutations are typically used to compute an approximate result set to a k-NN query. In (a) the approximate result set is computed by performing a k-NN directly in the permutation space. In (b) the approximate result set is computed using a filter-and-refine approach. (Best view in color)

ground-truth was used to evaluate the quality of the approximate results obtained by performing a nearest neighbor search in the permutation space.

As quality measure, we used the *recall@k*, defined as $|\mathcal{R} \cap \mathcal{R}^A|/k$, where $\mathcal{R}$ is the result set of the exact k-NN search in the original metric space and $\mathcal{R}^A$ is the approximate result set obtained in the permutation space. We also evaluate a *filter-and-refine* approach in which, for each query object, a *candidate result set* is selected by performing a k'-NN search in the permutation space (filter step). Then the candidate results are *re-ranked* using the actual distance d and the top-k objects are selected to form the approximate answer to the query (refine step). Figure 5 illustrates these two searching approaches. We consider $k = 10$ and $k' = 100$, thus in the base case the approximate result set $\mathcal{R}^A$ is computed by performing a 10-NN search in the permutation space; in the filter-and-refine case, the candidate result set is computed by performing a 100-NN search in the permutation space and then it is refined using the actual distance d to form the final result set $\mathcal{R}^A$.

We evaluate the performance of both full-length permutations and permutation prefixes for various prefix lengths. We employed the *Spearman's rho* to compare the full-length permutations, while the permutation prefixes were compared using the *Spearman's rho with location parameter*.

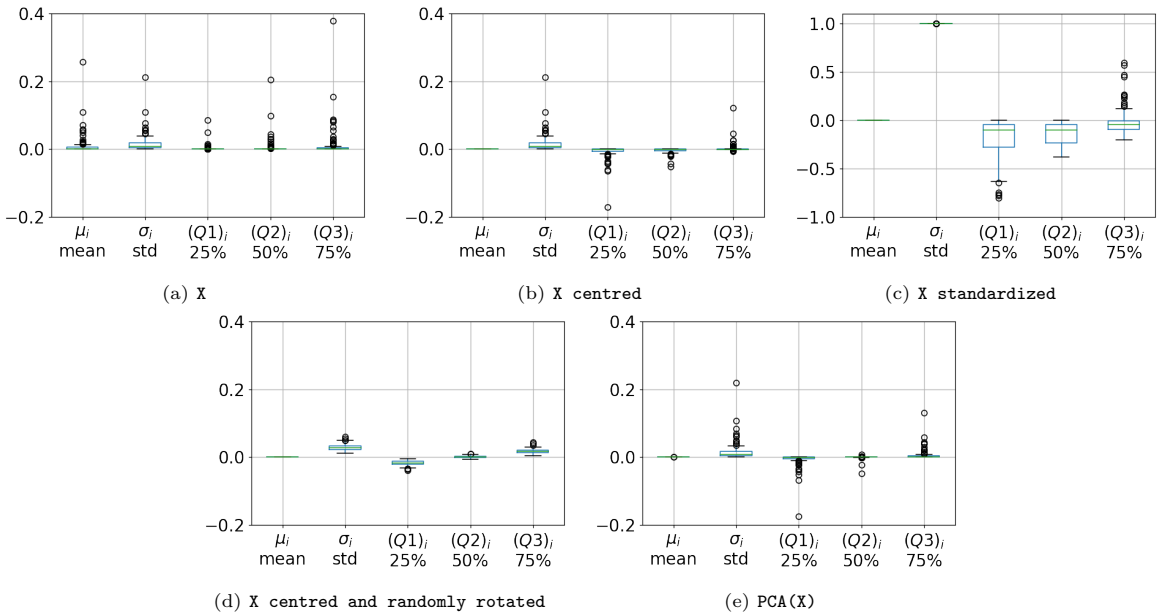
Results. We denote by $X = \{\mathbf{x}^{(j)}\}_{j=1}^{|X|}$ the original SISAP color data, where each element is a vector of the form $\mathbf{x}^{(j)} = [x_1^{(j)}, \ldots, x_N^{(j)}]$, with $N = 112$. Let $\boldsymbol{\mu} = [\mu_1, \ldots, \mu_N]$ and $\boldsymbol{\sigma} = [\sigma_1, \ldots, \sigma_N]$ the vectors with the mean and standard deviation values for each coordinate index i, that is

$$\mu_i = \text{mean}(\{x_i^{(j)} : j = 1 \ldots |X|\})$$
$$\sigma_i = \text{std}(\{x_i^{(j)} : j = 1 \ldots |X|\}).$$

In Figure 6a, we reported the box-plots depicting the distribution of the means μ_i, standard deviations σ_i and the quartiles of the original dataset X varying the coordinate index i, $i = 1, \ldots, N$. It can be observed that the box-plots contain many outliers and thus the statistics are not uniform along the various dimensions. This further assures us that the SISAP data provides an excellent example for our analysis as it is already a Euclidean dataset whose vectors have components not identically distributed.

To analyze the effect of the uniformity of vector component distributions on the quality of the permutations, we computed the permutations induced by sorting the elements of the vectors $f(\mathbf{x})$, $\mathbf{x} \in X$, for various choices of f:

- $f_{Id}(\mathbf{x}) = \mathbf{x}$ (identity function), so the transformed data is X itself;
- $f_c(\mathbf{x}) = \mathbf{x} - \boldsymbol{\mu}$, so that $\mathbb{E}[f(X)] = \mathbf{0}$. We refer this case to as `X_centred`;
- $f_s(\mathbf{x}) = \left[\frac{x_1 - \mu_i}{\sigma_1}, \ldots, \frac{x_N - \mu_N}{\sigma_N}\right]$ that standardizes each component by removing the mean and scaling to unit variance. We refer this case to as `X_standardized`;
- $f_R(\mathbf{x}) = R(\mathbf{x} - \boldsymbol{\mu})$ where R is a random rotation. We refer this case to as `X_centred_and_randomly_rotated`;



(a) `X` (b) `X centred` (c) `X standardized`

(d) `X centred and randomly rotated` (e) `PCA(X)`

Figure 6: *SISAP Colors*: box-plots showing the distribution of various statistics (mean, standard deviation, quartiles)

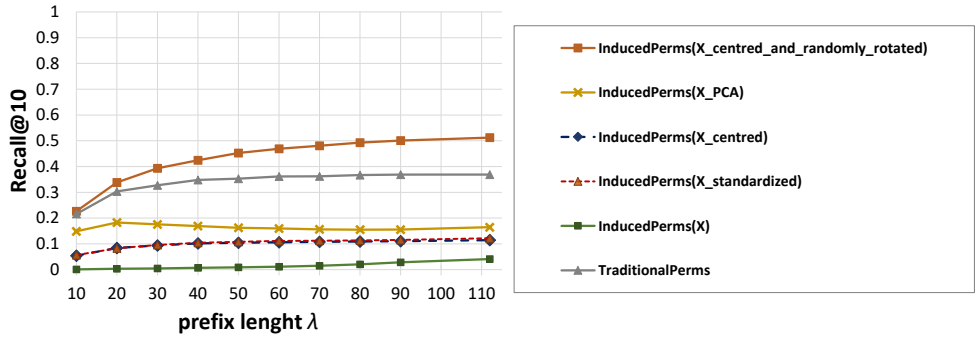
- $f_{PCA}(\mathbf{x}) = PCA(\mathbf{x})$ that centered the data and rotate it with the Principal component analysis (PCA) matrix (the full matrix is considered so that no dimensionality reduction is performed). We refer this case to as `X_PCA`.

Please note that f_c is a data translation that preserves the original Euclidean distances, i.e., $\ell_2(\mathbf{x}, \mathbf{y}) = \ell_2(f_c(\mathbf{x}), f_c(\mathbf{y}))$. The other functions f_s, f_R and f_{PCA} can be defined as the composition of f_c with other linear operations (i.e., a matrix multiplication):

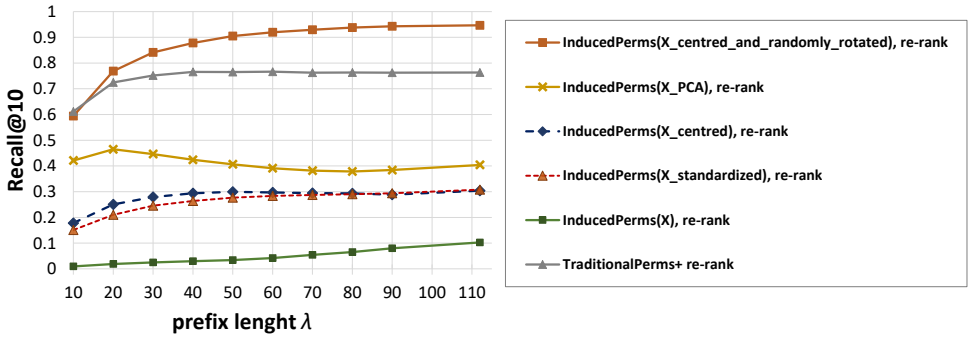
$$f_s(\mathbf{x}) = Df_c(\mathbf{x})$$
$$f_R(\mathbf{x}) = Rf_c(\mathbf{x})$$
$$f_{PCA}(\mathbf{x}) = Pf_c(\mathbf{x})$$

where $D = D = diag[1/\sigma_1, \ldots, 1/\sigma_N]$, R is a random orthogonal matrix, and P is is the PCA orthogonal matrix Since orthogonal matrices preserve the Euclidean distance, also f_R and f_{PCA} exactly preserve the distance between original data points. The function f_s preserve the distance if and only if all σ_i are equal to 1, that is the case in which the vector components of the original data already have unit variance. Moreover, f_R and f_{PCA} have in common the idea of rotating the centered data but with opposite effects. The PCA produces vectors whose first dimensional components have the highest variances. The random rotation, instead, helps in distributing the variance along all the dimensions. Figure 6 shows the box-plots of the distribution of mean, standard deviation, and quartiles of $f_c(X)_i$, $f_s(X)_i$, $f_R(X)_i$, and $f_{PCA}(X)_i$ varying the dimensional index i. Note that, according to the considered statistics, centering and randomly rotating the data is the transformation that produces the best results in terms of uniformity of the distributions along the various vector components (compact box-plots concentrated in their median values, and few outliers).

We computed the permutations induced by all the space transformations considered above. The total length of each permutation is $N = 112$, therefore we evaluated the performance of permutation prefixes with length $10 \leq \lambda \leq 112$. Figure 7 shows the *recall*@10 for the various approaches both in the case in which the result set is directly selected in the permutation space (Figure 7a) and when the actual distance is used to refine the results (Figure 7b). For reference, in the graphs, we also report the recalls obtained using the traditional permutations (Def. 1) for $N = 112$ randomly selected pivots. We observe that the permutations obtained by simply reordering the original vector components have extremely poor results and thus are not suitable for approximate metric search. Centring or standardizing each component of the data helps to slightly improve recall, which nevertheless remains low. Com-



(a) *Induced Permutations*



(b) *Induced Permutations with re-ranking using the actual distance*

Figure 7: *SISAP Colors*: Recall@10 for different induced permutation approaches varying the prefix lengths. Each induced permutation's length is equal to the original vector's dimensionality, i.e., N = 112. For reference the graphs report also the recall achieved using the traditional permutations computed using 112 pivots. (Best view in color)

pared with these transformations, PCA allows modest improvement in performance with small λ prefixes, but again the recall does not exceed 0.2 without re-ranking and 0.5 with the re-ranking. Centering and randomly rotating the data components, instead, gives much better results even better than the traditional permutations. For example, using a prefix length $\lambda = 70$ it reaches a recall of 0.48 (no re-ranking) and 0.93 (with re-ranking). For the same prefix length, the traditional permutations achieved a recall of 0.36 (no re-ranking) and 0.76 (with re-ranking).

4.2. Other desired properties of f

The experiments reported in the previous section were performed on a Euclidean space in order to show how different distributions of vector components influence the performance of the induced permutations. However, in the general case the starting space is not Euclidean but any metric space (D, d). The observations made so far suggest that the function f used to map the original data objects into a vector space ideally should produce coordinates with nearly identical distribution. However, this is not sufficient to have good permutations. Clearly the mapping f should preserve as much as possible object proximity relations. For example, suppose that f is a metric transformation of the form $f : (D, d) \to (\mathbb{R}^n, \ell_p)$ for some $1 \leq p \leq \infty$. We expect that f is suitable to compute induced permutations if it provides an approximation of the actual distance (e.g., $\ell_p(f(o_1), f(o_2)) \approx d(o_1, o_2)$) or if it is ranking preserving, i.e., $\ell_p(f(o_1), f(o_2)) < \ell_p(f(o_3), f(o_4))$ if $d(o_1, o_2) < d(o_3, o_4)$. For example, the traditional permutations are generated using a function $f : (D, d) \to (\mathbb{R}^n, \ell_\infty)$ for which the $\ell_\infty \circ f$ is a lower-bound of the actual distance. The SPLX-perms use a function $f : (D, d) \to (\mathbb{R}^n, \ell_2)$ that when composed with the Euclidean distance also provides a lower-bound of the actual distance, moreover, it uses a random rotation that preserves the Euclidean distance while helping in distributing the variance over all the dimensions of the vectors. The identity operator, which is of course ranking

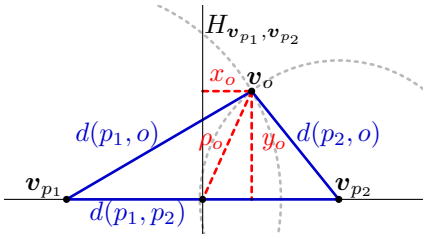


Figure 8: Planar Projection of two pivots p_1, p_2 and a data point o. (best view in color)

preserving, is used to generate the Deep Permutations.

Note that, on the one hand, it may be possible to simply compose the function f with other functions so that the components have similar distribution (e.g., by re-centering the vectors and applying a random orthogonal rotation, as shown in the ℓ_2 case). On the other hand, the condition of having an approximation of the original distance is more difficult to satisfy and cannot be theorized a priori. Therefore, the definition of a new function f that is good for inducing permutations is not a trivial task and may require specific theoretical investigations or machine learning procedures.

5. Pivot Pairs Permutations

In this section, we define a permutation-based representation that uses a transformation f, which not only depends on the distance of the objects to a predetermined set of pivots but also leverages information derived from the distances between pairs of pivots. The core idea is using the fact that any three points of a metric space can be isometrically embedded in a two-dimensional Euclidean space.

Specifically, let $p_1, p_2 \in D$ two fixed pivots and $o \in D$ an arbitrarily metric object. Without loss of generality, we could consider an isometric embedding that maps the points p_1, p_2, o to the vectors $\mathbf{v}_{p_1}, \mathbf{v}_{p_2}, \mathbf{v}_o \in (\mathbb{R}^2, \ell_2)$, such that

(i) $\mathbf{v}_{p_1}$ and $\mathbf{v}_{p_2}$ lie in the X-axis, e.g., $\mathbf{v}_{p_1} = [-\frac{\delta}{2}, 0]$ and $\mathbf{v}_{p_2} = [\frac{\delta}{2}, 0]$ where $\delta = d(p_1, p_2)$;

(ii) $\mathbf{v}_o$ is above the X-axis and its coordinates are given by the intersection of the ball centered on p_1 with radius $d(p_1, o)$ and the ball centered on p_2 with radius $d(p_2, o)$ (this intersection exists thanks to the triangle inequality).

Figure 8 depicts this situation in a 2D coordinate space where the two pivots are mapped in the X-axis symmetrically with respect to the origin and a single data object o is projected above the X-axis to the point $\mathbf{v}_o = (x_o, y_o)$ given by

$$x_o = \frac{d(o,p_1)^2 - d(o,p_2)^2}{2 \cdot d(p_1,p_2)} \tag{15}$$

$$y_o = \sqrt{d(o,p_1)^2 - \left(x_o + \frac{d(p_1,p_2)}{2}\right)^2} \tag{16}$$

Note that the only information used in the projection is the distances of the object o to the two pivots and the inter-pivot distance. Moreover, this projection preserves all the three distances between the points, i.e.,

$$\begin{aligned} \ell_2(\mathbf{v}_{p_1}, \mathbf{v}_{p_2}) &= d(p_1, p_2) \\ \ell_2(\mathbf{v}_{p_1}, \mathbf{v}_o) &= d(p_1, o) \\ \ell_2(\mathbf{v}_{p_2}, \mathbf{v}_o) &= d(p_2, o) \end{aligned}$$

This projection, called *planar projection* [32, 21], could be repeated for all the data points $o \in D$ while fixing the two pivots p_1, p_2. So we have a projection $\phi_{p_1,p_2} : (D, d) \to (\mathbb{R}^2, \ell_2)$ that preserves the distances of data objects to the two pivots. In [21] it was proved that if the space (D, d) has the four-point property then the planar projection[6] provides a lower-bound for the original distance, that is $\ell_2(\phi_{p_1,p_2}(o_1), \phi_{p_1,p_2}(o_2)) \leq d(o_1, o_2)$, for all $o_1, o_2 \in D$. Nevertheless, even if the space does not meet the four-point property the projection above can always be calculated. Moreover, since the distance to the pivots is preserved for each data point, it can be easily proved that all the objects in the hyperplane separating the two pivots in the original space are projected in the hyperplane $H_{\mathbf{v}_{p_1},\mathbf{v}_{p_2}} = \{\mathbf{v} \in \mathbb{R}^2 \,|\, \ell_2(\mathbf{v}, \mathbf{v}_{p_1}) = \ell_2(\mathbf{v}, \mathbf{v}_{p_2})\}$ separating the pivots in the 2D projection.

The Euclidean norm of a projected object ($\rho_o = \|\mathbf{v}_o\|$) could be interpreted as the distance of the point o to a *synthetic pivot* that is equidistant to the two original pivots, i.e., a sorting of midpoint which may not exist in the original metric space. Its calculation is immediate if we already know $d(p_1, p_2)$,

[6] Actually the planar projection used in [21] is a translation of the projection used here, as it projects the first pivot in the origin and the second pivots to the point $[d(p_1, p_2), 0]$. However, as the translation is an isometry the two projections are equivalent for our scopes.

$d(o, p_1)$, and $d(o, p_2)$ as it is equals to

$$\begin{aligned}\rho_o &= \sqrt{(x_o)^2 + (y_o)^2} \\ &= \frac{1}{2}\sqrt{2\,d(o,p_1)^2 + 2\,d(o,p_2)^2 - d(p_1,p_2)^2}\end{aligned} \tag{17}$$

We can repeat this procedure for several pairs of pivots to characterize a metric object based on the distribution of its distance from the synthetic midpoints between the original pivots. Formally, given a set $\{p_1, \ldots, p_n\} \subset D$ of n pivots, we select $m < \binom{n}{2}$ pivot pairs that we enumerate using an index i, so that (p_{i_1}, p_{i_2}) indicates the i-th pivot pair. For each object $o \in D$ and for each selected pair of pivots (p_{i_1}, p_{i_2}) we use Eq. 17 to compute the norm $\rho_o^{(i)}$ of the projected point $\phi_i(o) = (x_o^{(i)}, y_o^{(i)})$. Then we generate a permutation $\Pi_o^{f'}$ of length m by reordering the components of $f'(o) = \left(\rho_o^{(1)}, \ldots, \rho_o^{(m)}\right)$. Moreover, since we can interpret the values ρ_i as the distance to some synthetic pivots, we may combine this information with the distances to the actual pivots by computing the permutations induced by the function

$$\begin{aligned} f'' : D &\to \mathbb{R}^{n+m} \\ o &\to \left(d(o,p_1), \ldots, d(o,p_n), \rho_o^{(1)}, \ldots, \rho_o^{(m)}\right)\end{aligned} \tag{18}$$

In the following, we refer to the permutations $\Pi_o^{f'}$ and $\Pi_o^{f''}$ as *Pairs Permutation* (`P-Perms`), and *Pivot-Pairs Permutation* (`PP-Perms`), respectively.

5.1. Experiments

In our experiments, we compared the performance of three different permutation-based representations for approximate k-NN search: `P-Perms`, `PP-Perms`, and the traditional permutations (`Perms`). We conducted the experiments both on real-world and publicly available datasets (CoPhIR and ANN-SIFT) and on synthetic datasets. In the following, we first introduce the measures used for the evaluation and then we present the experimental results.

Evaluation Protocol. For each dataset, we build a ground-truth for the exact k-NN search related to 1,000 randomly selected queries. These ground-truths were used to evaluate the quality of the approximate results obtained either by performing a k-NN search in the permutation space or using the actual distance to re-rank a candidate result set of size $k' \geq k$ that was selected using a k'-NN search in the permutation space. The latter approach is a *filter-and-refine* approach that requires storing the original dataset and accessing to it at query time to refine the permutation-based candidate results. In the experiments, we used $k = 10$ and $k' = 100$.

The quality of the approximate results was evaluated using the *recall@k*.

As done by many PBI approaches [5, 7, 6], we index and search the data using fixed-length permutation prefixes instead of the full-length permutations. The metric $\tilde{d}_{\rho,\lambda}$ is used to compare permutation prefixes.

For indexing the permutation prefixes we used inverted files [7]. If N is the length of the full permutations (i.e., we have N different permutants that may appear in a permutation prefix), we have that

- The inverted index is composed of N posting lists, one for each permutant.
- Each object is stored in exactly λ posting lists, corresponding to the permutants appearing in its permutation prefix. Thus, the i-th posting list contains t_i entries related only to the data objects whose permutations prefixes contain the permutant i.
- Each entry of the i-th posting list is of the form $(ID_o, pos_o(i))$, where ID_o is the identifier of a data object, $pos_o(i)$ is the position of the permutant i in the permutation prefix associated to the object o.
- At query time, we access only the λ posting lists corresponding to the permutants in the query permutation prefix. For each object o in those selected posting lists, we use the stored $pos_o(i)$ to compute the $\tilde{d}_{\rho,\lambda}$ distance to the query permutation prefix.

In this setting, *the size in bits of the inverted index* is a function of the number of permutants N, the prefix length λ, and the number of data objects $|X|$:

$$\begin{aligned}\text{Size(Inverted Index)} = &\underbrace{N\,\lceil \log_2 N \rceil}_{\text{posting list identifiers}} \\ &+ \underbrace{\lambda\,|X|\,(\lceil \log_2 |X| \rceil + \lceil \log_2 \lambda \rceil)}_{\text{posting list entries}}\end{aligned} \tag{19}$$

The cost at query time includes 1) the cost of transforming the query into the permutation representation; 2) the search cost; 3) the cost of re-ranking the candidate set using the actual distance

(only for the filter-and-refine approach). *The cost for computing the permutations* (Table 2) varies with the employed permutation-based representation and the specific metric of the space. For a given set of n pivots the traditional permutation has $N = n$ permutants and requires the calculation of n object-pivot distances. For the same set of pivots and m selected pivot pairs, `P-Perms` and `PP-Perms` have $N = m$ and $N = n + m$ permutants, respectively. They require n object-pivot distance calculations plus m 2D Euclidean distances to calculate the ρ_o values (Eq. 17), which in most cases is a negligible cost with respect to object-pivot distance calculations. Additionally, `P-Perms` and `PP-Perms` require $\min[m, n(n-1)/2]$ pivot-pivot distances, which can be computed and stored once at indexing time and then reused for calculating all the objects/query permutations.

The *search cost* (SC), calculated as the *number of bits accessed per query*, is given by

$$SC = C(\text{pEntry}) \sum_{i=1}^{N} \delta_i t_i \qquad (20)$$

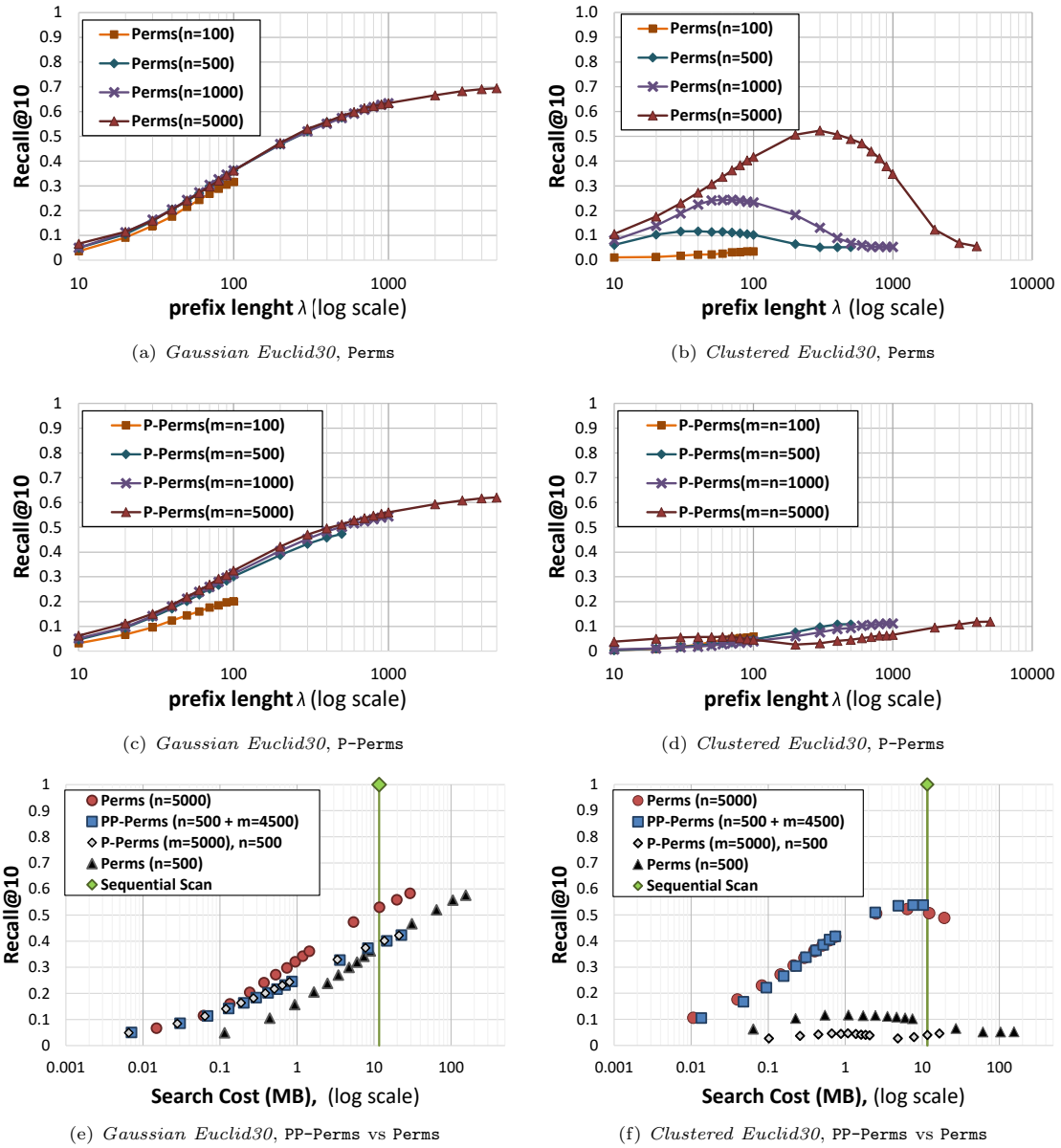
where t_i is the number of objects stored in the i-th posting list, δ_i is the fraction of samples in the database having the permutant i in their permutation prefixes (i.e., $\delta_i = t_i/|X|$). The term $C(\text{pEntry}) = \lceil \log_2 |X| \rceil + \lceil \log_2 \lambda \rceil$ denotes the size in bits of a single entry of a posting list. Given a query q, we access the i-th posting list only if the index i is in the permutation prefix associated with the query. This is true with probability δ_i since query and database objects share the same distribution. Consequently, the number of elements accessed per query is $\sum_{i=1}^{N} \delta_i t_i$ as the i-th posting list contains t_i entries, and we access it with δ_i probability. It is worth noting that for a fixed N, the larger the prefix λ, the greater t_i, resulting in a higher search cost. Moreover, for the filter-and-refine approach, the bytes accessed per query include those needed to select the k' candidate results (given by Eq 20) and those needed to re-rank the candidate results using the actual distance, i.e., $k' * C(Obj)$, where $C(Obj)$ is the size in bits of one original data object.

5.1.1. Experiments on Synthetic Data

The first question that may arise when considering the `P-Perms` representation as an alternative to the traditional permutations (`Perms`) is whether using the distances to the synthetic midpoint pivots instead of the actual pivots still helps in distinguishing similar data points from dissimilar ones in an approximate search scenario. Moreover, since the `P-Perms` and the `PP-Perms` allow producing permutations that are longer than the number of the employed pivots, it would be also interesting to analyze the performance of these permutations when the number m of pairs is increased while fixing the number n of pivots (i.e., fixing the number of actual distance computation needed to generate the permutations). To this end, we performed experiments on two representatives types of synthetic data: clustered and non-clustered Euclidean vectors. This choice was motivated by the observation that it was already proven in the literature [7] that the traditional permutations exhibit different behaviours on these two data types when varying the number n of pivots and the prefix length λ.

Specifically, we considered two datasets, each containing 100K vectors in a 30-dimensional Euclidean space. The first dataset, named *Gaussian Euclid30*, contains vectors whose coordinates are generated using a Gaussian distribution centered at the origin and with a standard deviation $\sigma = 0.1$. The second dataset, named *Clustered Euclid30*, contains vectors arranged in 20 clusters. The cluster centers were randomly selected in the hypercube $[0, 1]^{30}$. For each cluster, we generated $5K$ vectors using a Gaussian distribution with a small standard deviation ($\sigma = 0.01$).

Figures 9a and 9b show, for the two datasets, the *recall@10* achieved by the traditional permutation when varying the number n of pivots and the prefix length λ. It is important to note the different behaviors of the permutations on these two types of data. On Gaussian Euclid30, the recall increases when increasing both n and λ, but for a fixed λ the recall is almost unchanged when when only n is increased (Fig. 9a). For the clustered data, instead, the performance of the *full-length* permutations (i.e., the cases $\lambda = n$) is not improved when increasing the number of pivots beyond $n = 500$. However, for a fixed n, there exists an optimal prefix length $\lambda < n$ for which the recall achieves a maximum. Amato el al. [7] noted that this maximum is systematically achieved around the prefix length $\lambda = n/cl$, where cl is the number of clusters. Note that n/cl represents the average number of pivots taken from each cluster since we use n random pivots. This suggests that an object of a cluster is well represented by the pivots that belong to its same cluster, but when



(a) *Gaussian Euclid30*, `Perms`

(b) *Clustered Euclid30*, `Perms`

(c) *Gaussian Euclid30*, `P-Perms`

(d) *Clustered Euclid30*, `P-Perms`

(e) *Gaussian Euclid30*, `PP-Perms` vs `Perms`

(f) *Clustered Euclid30*, `PP-Perms` vs `Perms`

Figure 9: *Synthetic Datasets*: Recall@10 for the traditional permutations (graphs (a) and (b)) and the `P-Perms` (graphs (c) and (d)) varying the number of pivots and the prefix lengths. Graphs (e) and (f) show the recall for increasing prefix lengths as function of the Search Cost for `Perms` and `PP-Perms`. For each method, the points in the graphs correspond to the prefix lengths $\lambda = 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 200, 300, 400, 500$. (Best view in color)

This is a pre-copyedit version of this article. The final authenticated version is available online at: https://doi.org/10.1016/j.is.2023.102286.

Table 2: Distance computations needed for generating various permutation-based representations given the same set of n pivots. m is the number of pairs used in the Pairs and Pivot-Pair Permutations. N is the number of permutants.

Approach	**N**	**Number of Distance Calculations**	
		computed for each object/query	*computed once at indexing time*
`Perms`	n	n actual distances	
`P-Perms`	m	n actual distances + m 2D Euclidean distances	$\min(m, n(n-1)/2)$ actual distances
`PP-Perms`	$n+m$		

we increase the length of the permutation prefixes we also include pivots taken from other clusters which seems to introduce noisy information. In facts, when we fix n the recall begins to decrease sharply for $\lambda > n/cl$ (Fig. 9b).

Regarding our initial question, that is, whether `P-Perms` represent a valid alternative to classical permutations in distinguishing objects, for a preliminary analysis, we selected a number m of random pivot pairs equal to n, so that the Pair permutations have the same length of the traditional permutations (i.e., $N = n = m$). For this settings, we discovered that on *Gaussian Euclid30* the `P-Perms` has similar behavior and slightly lower effectiveness than the classical `Perms` when varying n and λ (Fig. 9c vs Fig. 9a), thus confirming us that the synthetic pivots computed from the pivot pairs could be used as alternative pivots for generating permutations. However, on the clustered data, the `P-Perms` seems to be completely ineffective (Fig. 9d), except for the case $\lambda = n$, for which the full-length `P-Perms` slightly outperforms the traditional full-length `Perms` (nevertheless, both the approaches achieved very low recall when using their full-length representations). One possible reason for the poor performance of the `P-Perms` on clustered data is that we are using as reference objects the synthetic midpoints of just $m = n$ random pivot pairs out of $n(n-1)/2$ possible pairs. In fact, since the data is uniformly distributed over the cl clusters, the synthetic midpoint of a pair (p_{i_1}, p_{i_2}) is representative of a cluster $\mathcal{C}$ if both p_{i_1} and p_{i_2} belong to the same cluster $\mathcal{C}$, which happen with probability $(n-cl)/cl(n-1)$. Conversely, with probability $n(cl-1)/cl(n-1)$ the two pivots belong to different clusters. For example, if $cl = 20$ and $n = 5K$ the probability of picking a pair of pivots of different clusters is about 95%, so when we use only $m = n = 5K$ random pairs we have on average $4,750$ pairs of pivots form different clusters and just about 12-13 pairs representative of each cluster. To mitigate this inconvenience we may try to use $m \gg n$ or, as proposed in the following, use the `PP-Perms` representation that employs both traditional and synthetic pivots. The latter approach guarantees to have a percentage of pivots that are still representative of the original data distribution which seems to be fundamental for clustered data. Note that for n pivots and m pairs the `PP-Perms` produces permutations of length $N = m + n$. For some prefix lengths, the effectiveness of the `PP-Perms` may decrease when considering $m \gg n$ as the percentage of synthetic pivots will be much larger than the percentage of actual pivots, which may be an issue for clustered data. Anyway, the loss in effectiveness is compensated by the more efficiency at searching time since the Search Cost (number of bits accessed per query) typically increases proportionally to λ^2/N. Therefore, since `PP-Perms` and `Perms` have different lengths N, in the rest of this paper, we report the *recall* values as a function of the Search Cost.

In Figures 9e and 9f, we compared the performance of the traditional `Perms` using $n = 500$ pivots ($N = 500$), the `PP-Perms` using $n = 500$ pivots and $m = 4,500$ pairs ($N = 5,000$), the `P-Perms` using $m = 5,000$ pairs selected from $n = 500$ pivots ($N = 5,000$), and the traditional `Perms` using $n = 5,000$ pivots ($N = 5,000$). The latter approach is plotted for reference as it has the same length as the tested `PP-Perms` and `P-Perms`, but note that it requires $5,000$ object-pivot distance computations, while the other approaches use an order of magnitude fewer object-pivot distances computations. For all the approaches we plot the recall versus the search cost when increasing the prefix length λ from 10 to 500. It is worth noting we also evaluated the percentage of actual pivots appearing in the permutation prefix of the `PP-Perms` when the prefix

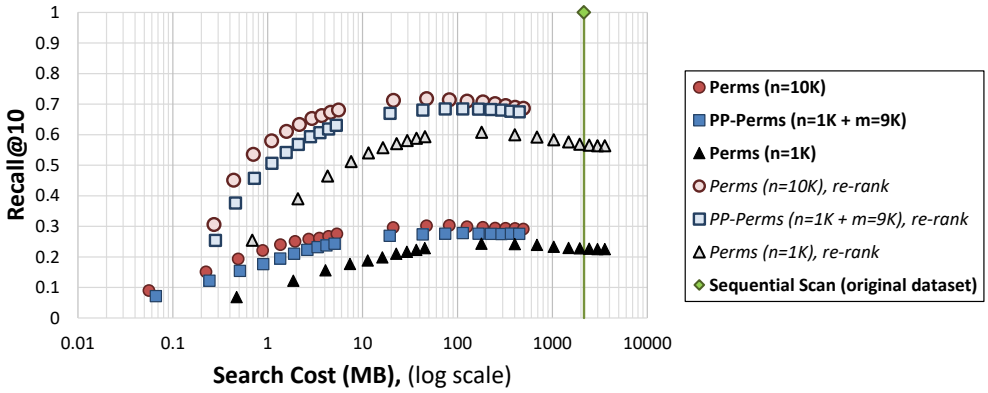
λ varies from 10 to 500. We found that for *Gaussian Euclid30*, on average, it is about 1.55% (with a very small standard deviation), and for *Clustered Euclid30*, it is about 8.14% (with the maximum percentage achieved for $\lambda = 200$, where approximately 9.54% are actual pivots). Specifically, for $\lambda = 200$, the permutation prefix on average contains distances to 3 real pivots and 197 synthetic pivots in the case of Gaussian data, and 19 real pivots and 181 synthetic pivots on average in the case of clustered data. As expected, the `P-Perms`, which rely only on synthetic pivots, has poor performance on the clustered data. However, on the clustered dataset, the `PP-Perms` approach, which uses both actual and synthetic pivots, not only outperforms the `Perms` techniques that use the same set of actual pivots ($n = 500$) but also reaches the performance of the traditional permutations built upon the larger set of pivots ($n = 5,000$). Thus we observed a great advantage in combining synthetic and real pivots to represent clustered data. In fact, the `PP-Perms` shows the best trade-off between recall, search cost, and the cost for computing the permutations (i.e., the actual object-pivot distance computations). On Gaussian data, both the `P-Perms` and the `PP-Perms` still outperform the traditional permutation built on the same pivot set (we are not interested in the recalls when the search costs are greater than the sequential scan). Moreover, for small search cost values, it achieves recalls in line with the more expensive traditional permutation built upon the larger pivot set. Given these outcomes, in the following, we focus our attention only on the `PP-Perms` and `Perms` approaches, omitting the `PP-Perms` technique.

5.1.2. Experiments on Real-World Data

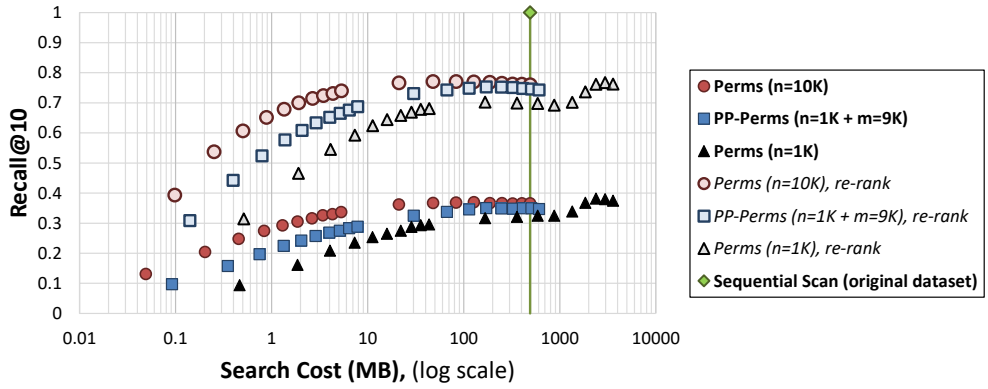
For the experiments on real-world data, we used two sets of 1M objects from the *CoPhIR* [33] and *ANN-SIFT* [34] datasets, for which we used different kinds of image features compared with distinct metrics. On the *CoPhIR* data we used as metric the linear combination of the five distance functions (Manhattan, Euclidean, and other special metrics) for the five MPEG-7 descriptors that have been extracted from each image. We adopted the weights proposed in [35, Table 1]. The *ANN-SIFT* contains SIFT local features (128-dimensional vectors) compared with the Euclidean distance. Note that the SIFT data contains some clusters as the distance distribution is a mixture of Gaussians (see [36, Fig. 1]). On both the datasets, we tested the traditional `Perms` using $n = 1,000$ pivots ($N = 1,000$), the `PP-Perms` using $n = 1,000$ pivots and $m = 9,000$ pairs ($N = 10,000$), and the traditional `Perms` using $n = 10,000$ pivots ($N = 10,000$). For each approach, we varied the prefix length λ from 10 to 1,000. We observed that, for the tested λ, the percentage of actual pivot appearing in the permutation prefixes of the `PP-Perms` is, on average, 3.4% for *CoPhIR* data and 5.5% for *SIFT* data. The results in terms of recall and search cost are depicted in Figures 10a and 10b for *CoPhIR* and *SIFT* data, respectively. For reference, we also reported the cost of the sequential scan for searching the original data descriptors using the actual distance. Moreover, we include the results when the actual distance is used to refine (*re-rank*) the candidate results selected in the permutation space. We observed that on both the datasets the `PP-Perms` outperforms the traditional permutations build upon the same set of pivots. Moreover, for $\lambda > 100$ it achieves recall values in line with that of the more expensive permutation built upon the 10 times larger set of pivots. Therefore, the `PP-Perms` can be profitably used as an alternative to the traditional permutation to generate long permutations while limiting the number of actual distance computations. For example, to search 1M SIFT data with a query cost of about 8 MB, the `PP-Perms` achieves a *recall@10* of 0.29 (0.69 when using the re-ranking) while the `Perms` that uses the same set of pivots has a recall of 0.24 (0.56 with the re-ranking). On the CoPhIR data, the improvement is even more evident: for a search cost of about 4 MB the `PP-Perms` reaches a recall of 0.24 (0.61 when using the re-ranking) while the traditional permutations have a recall of 0.16 (0.47 when using the re-ranking).

6. Conclusions

This paper introduced the concept of permutation representations induced by a metric transformation f, which generalizes the traditional definition of permutations associated with metric objects. As a practical example, we defined permutations induced by a combination of pivots and several planar projections related to some pivot pairs. In our experiments, this novel representation, called `PP-Perms`, achieved the best trade-off between effectiveness (recall) and efficiency (search cost and data distance computations) with respect to the traditional permutations.



(a) *CoPhIR*



(b) *SIFT*

Figure 10: Recall@10 as function of the Search Cost (with and without re-rank based on the actual distance), for increasing permutation prefix lengths. For each method, the points plotted in the graphs correspond to the prefix lengths $\lambda = 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000$. (Best view in color)

We also investigated both theoretically and experimentally some properties that the function f may have to produce permutations that are effective for approximate metric search. Nevertheless, the definition of novel functions f inducing good permutation representations of metric objects remains a challenging and open research problem. We believe that our generalization could pave the way for innovative research directions that go beyond the traditional use of distances between objects and pivots alone to calculate permutations. As future work, we aim to investigate the use of artificial intelligence techniques to automatically learn optimal functions f for the metric dataset of interest, which also goes in the new challenging research direction of defining learned metric indexes.

Acknowledgements

The work was partially supported by H2020 project AI4EU under GA 825619, by H2020 project AI4Media under GA 951911, and PNRR-National Centre for HPC, Big Data and Quantum Computing project CUP B93C22000620006.

References

[1] P. Zezula, G. Amato, V. Dohnal, M. Batko, Similarity search: the metric space approach, Vol. 32, Springer Science & Business Media, 2006.

[2] V. Pestov, Indexability, concentration, and vc theory, J. Discrete Algoritms 13 (2012) 2–18.

[3] B. Naidan, L. Boytsov, E. Nyberg, Permutation search methods are efficient, yet faster search is possible, Proceedings International Conference on Very Large Data Bases 8 (12) (2015) 1618–1629.

[4] R. Weber, H.-J. Schek, S. Blott, A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces, in: Proceedings International

Conference on Very Large Data Bases, Vol. 98, 1998, pp. 194–205.
[5] A. Esuli, Use of permutation prefixes for efficient and scalable approximate similarity search, Inf. Process. Manage. 48 (5) (2012) 889–902.
[6] D. Novak, P. Zezula, PPP-Codes for Large-Scale Similarity Searching, Vol. 24, Springer Berlin Heidelberg, 2016, pp. 61–87.
[7] G. Amato, C. Gennaro, P. Savino, MI-File: Using inverted files for scalable approximate similarity search, Multimed. Tools. Appl. (3) (2014) 1333–1362.
[8] E. S. Tellez, E. Chávez, G. Navarro, Succinct nearest neighbor search, Inf. Syst. 38 (7) (2013) 1019–1030.
[9] E. Chávez, K. Figueroa, G. Navarro, Proximity searching in high dimensional spaces with a proximity preserving order, in: MICAI 2005: Advances in Artificial Intelligence, Springer, 2005, pp. 405–414.
[10] G. Amato, P. Savino, Approximate similarity search in metric spaces using inverted files, in: Proceedings International ICST Conference on Scalable Information Systems, ICST / ACM, 2008, pp. 28:1–28:10.
[11] E. Chávez, K. Figueroa, G. Navarro, Effective proximity retrieval by ordering permutations, IEEE Trans. Pattern Anal. Mach. Intell. 30 (9) (2008) 1647–1658.
[12] G. Amato, F. Falchi, F. Rabitti, L. Vadicamo, Some theoretical and experimental observations on permutation spaces and similarity search, in: Similarity Search and Applications, Springer International Publishing, 2014, pp. 37–49.
[13] K. Figueroa, E. Chavez, G. Navarro, R. Paredes, Speeding up spatial approximation search in metric spaces, ACM J. Exp. Algorithmics 14.
[14] M. L. Hetland, T. Skopal, J. Lokoč, C. Beecks, Ptolemaic access methods: Challenging the reign of the metric space model, Inf. Syst. 38 (7) (2013) 989–1006.
[15] M. Kruliš, H. Osipyan, S. Marchand-Maillet, Employing gpu architectures for permutation-based indexing, Multimed. Tools. Appl. 76 (9) (2017) 11859–11887.
[16] H. Mohamed, S. Marchand-Maillet, Quantized ranking for permutation-based indexing, Inf. Syst. 52 (2015) 163–175, special Issue on Selected Papers from SISAP 2013.
[17] T. Murakami, R. Fujita, T. Ohki, Y. Kaga, M. Fujio, K. Takahashi, Cancelable permutation-based indexing for secure and efficient biometric identification, IEEE Access 7 (2019) 45563–45582.
[18] G. Amato, F. Falchi, C. Gennaro, L. Vadicamo, Deep Permutations: Deep convolutional neural networks and permutation-based indexing, in: Similarity Search and Applications, Springer International Publishing, 2016, pp. 93–106.
[19] L. Vadicamo, R. Connor, F. Falchi, C. Gennaro, F. Rabitti, SPLX-Perm: A novel permutation-based representation for approximate metric search, in: Similarity Search and Applications, Springer International Publishing, 2019, pp. 40–48.
[20] L. Vadicamo, C. Gennaro, G. Amato, On Generalizing Permutation-Based Representations for Approximate Search, in: Similarity Search and Applications, Springer International Publishing, Cham, 2021, pp. 66–80.
[21] R. Connor, L. Vadicamo, F. A. Cardillo, F. Rabitti, Supermetric search, Inf. Syst. 80 (2019) 108–123.
[22] M. Deza, T. Huang, Metrics on permutations, a survey, in: Journal of Combinatorics, Information and System Sciences, Citeseer, 1998.
[23] P. Diaconis, Group representations in probability and statistics, Lecture notes-monograph series 11 (1988) i–192.
[24] R. Fagin, R. Kumar, D. Sivakumar, Comparing top k lists, SIAM J. Discrete Math. 17 (1) (2003) 134–160.
[25] G. Amato, F. Carrara, F. Falchi, C. Gennaro, L. Vadicamo, Large-scale instance-level image retrieval, Inf. Process. Manag. 57 (6) (2020) 102100.
[26] R. Connor, L. Vadicamo, F. Rabitti, High-dimensional simplexes for supermetric search, in: Similarity Search and Applications, Springer International Publishing, 2017, pp. 96–109.
[27] F. Carrara, C. Gennaro, F. Falchi, G. Amato, Learning distance estimators from pivoted embeddings of metric objects, in: International Conference on Similarity Search and Applications, Springer, 2020, pp. 361–368.
[28] S. Marchand-Maillet, E. Roman-Rangel, H. Mohamed, F. Nielsen, Quantifying the invariance and robustness of permutation-based indexing schemes, in: International Conference on Similarity Search and Applications, Springer, 2016, pp. 79–92.
[29] F. Aurenhammer, Voronoi diagrams—a survey of a fundamental geometric data structure, ACM Computing Surveys (CSUR) 23 (3) (1991) 345–405.
[30] M. Skala, Counting distance permutations, Journal of Discrete Algorithms 7 (1) (2009) 49–61.
[31] K. Figueroa, G. Navarro, E. Chávez, Metric spaces library, www.sisap.org/library/manual.pdf (2007).
[32] R. Connor, L. Vadicamo, F. A. Cardillo, F. Rabitti, Supermetric search with the four-point property, in: Similarity Search and Applications, Springer International Publishing, 2016, pp. 51–64.
[33] P. Bolettieri, A. Esuli, F. Falchi, C. Lucchese, R. Perego, T. Piccioli, F. Rabitti, Cophir: a test collection for content-based image retrieval (2009). `arXiv:0905.4627`.
[34] H. Jegou, M. Douze, C. Schmid, Product quantization for nearest neighbor search, IEEE Trans. Pattern Anal. Mach. Intell. 33 (1) (2010) 117–128.
[35] M. Batko, F. Falchi, C. Lucchese, D. Novak, R. Perego, F. Rabitti, J. Sedmidubsky, P. Zezula, Building a web-scale image similarity search system, Multimed. Tools. Appl. 47 (3) (2010) 599–629.
[36] L. Vadicamo, V. Mic, F. Falchi, P. Zezula, Metric embedding into the hamming space with the n-simplex projection, in: Similarity Search and Applications, Springer International Publishing, 2019, pp. 265–272.

Appendix A.

In this appendix, we provide the detailed mathematical derivations that support the concepts discussed in Section 4.1. In particular, we demonstrate that having the random variables $\mathcal{X}_1, \ldots, \mathcal{X}_N$ representing all possible $f_1(o), \ldots, f_N(o)$ as independent and identically distributed (iid) is a necessary condition for the induced permutations by the transformation f to be uniformly distributed in the permutation space. To facilitate the derivation, we present a counterexample using a multivariate Gaussian distribution.

This is a pre-copyedit version of this article. The final authenticated version is available online at: https://doi.org/10.1016/j.is.2023.102286.

Suppose all the variable $\mathcal{X}_i$ follow a normal distribution with $\mathcal{X}_i \sim \mathcal{N}(\mu_i, \sigma_i)$ for $i = 1, \dots, N$. The probability that a generic permutation $\pi = [\pi_1, \dots, \pi_N]$ occurs is given by $p(\pi) = P(\mathcal{X}_{\pi_1} < \mathcal{X}_{\pi_2} < \dots < \mathcal{X}_{\pi_N})$. This probability, in the case of independent Gaussian variables, is given by

$$\int_{-\infty}^{\infty} \int_{-\infty}^{x_{\pi_N}} \cdots \int_{-\infty}^{x_{\pi_2}} \prod_{i=1}^{N} g_i(x_i) dx_{\pi_N} dx_{\pi_{N-1}} \cdots dx_{\pi_1}$$

where

$$g_i(x_i) = \frac{1}{\sigma_{\pi_i}} \phi\left(\frac{x_{\pi_i} - \mu_{\pi_i}}{\sigma_{\pi_i}}\right)$$

and

$$\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}.$$

Using the changing of variables

$$z_{\pi_i} = \frac{x_{\pi_i} - \mu_{\pi_i}}{\sigma_{\pi_i}},$$

for all $i = 1, \dots, N-1$, we have get

$$\int_{-\infty}^{x_{\pi_{i+1}}} g_i(x_i) dx_{\pi_i} = \int_{-\infty}^{z_{\pi_{i+1}} \frac{\sigma_{\pi_{i+1}}}{\sigma_{\pi_i}} + \frac{\mu_{\pi_{i+1}} - \mu_{\pi_i}}{\sigma_{\pi_i}}} \phi\left(z_{\pi_i}\right) dz_{\pi_i}.$$

Therefore, by indicating with $\delta_{i+1,i}^{(\pi)}$ the constants $\frac{\mu_{\pi_{i+1}} - \mu_{\pi_i}}{\sigma_{\pi_i}}$ we have

$$p(\pi) = \int_{-\infty}^{\infty} \int_{-\infty}^{z_{\pi_N} \frac{\sigma_{\pi_N}}{\sigma_{\pi_{N-1}}} + \delta_{N,N-1}^{(\pi)}} \cdots \int_{-\infty}^{z_{\pi_2} \frac{\sigma_{\pi_2}}{\sigma_{\pi_1}} + \delta_{2,1}^{(\pi)}} \prod_{i=1}^{N} \phi\left(z_{\pi_i}\right) dz_{\pi_N} dz_{\pi_{N-1}} \cdots dz_{\pi_1} \tag{A.1}$$

From the above derivation, for the example of independent normal variables $\mathcal{X}_i$, we can observe that

- if all the $\mathcal{X}_i$ have the same distribution, i.e., $\mathcal{X}_i \sim \mathcal{N}(\mu, \sigma)$, then all the permutations have the same probability to occur. In this case, the probability of any permutation $\pi = [\pi_1, \dots, \pi_N]$ is equal to:

$$\int_{-\infty}^{\infty} \int_{-\infty}^{z_N} \cdots \int_{-\infty}^{z_2} \prod_{i=1}^{N} \phi\left(z_i\right) dz_N dz_{N-1} \cdots dz_1;$$

- if the variables $\mathcal{X}_i$ have the same mean but different variances, i.e, $\mathcal{X}_i \sim \mathcal{N}(\mu, \sigma_i)$, then we have a non-uniform distribution of the induced permutations. In particular, the probability of the permutation $\pi = [\pi_1, \dots, \pi_N]$ is given by

$$\int_{-\infty}^{\infty} \int_{-\infty}^{z_{\pi_N} \frac{\sigma_{\pi_N}}{\sigma_{\pi_{N-1}}}} \cdots \int_{-\infty}^{z_{\pi_2} \frac{\sigma_{\pi_2}}{\sigma_{\pi_1}}} \prod_{i=1}^{N} \phi\left(z_{\pi_i}\right) dz_{\pi_N} dz_{\pi_{N-1}} \cdots dz_{\pi_1}$$

which depends on the ratios $\sigma_{\pi_i} / \sigma_{\pi_{i+1}}$. The higher these ratios are, the higher the probability $p(\pi)$. Therefore if $i_1, \dots, i_N$ are the indexes such that $\sigma_{i_1} \geq \dots \geq \sigma_{i_N}$ then the probability of $\pi_A = [i_1, \dots, i_N]$ will be greater than the probability of $\pi_B = [i_N, \dots, i_1]$.