Basic Triangle Inequality Approach Versus Metric VP-Tree and Projection in Determining Euclidean and Cosine Neighbors

Marzena Kryszkiewicz and Bartłomiej Jańczak

Abstract The Euclidean distance and the cosine similarity are often applied for clustering or classifying objects or simply for determining most similar objects or nearest neighbors. In fact, the determination of nearest neighbors is typically a subtask of both clustering and classification. In this chapter, we discuss three principal approaches to efficient determination of nearest neighbors: namely, using the triangle inequality when vectors are ordered with respect to their distances to one reference vector, using a metric VP-tree and using a projection onto a dimension. Also, we discuss a combined application of a number of reference vectors and/or projections onto dimensions and compare two variants of VP-tree. The techniques are well suited to any distance metrics such as the Euclidean distance, but they cannot be directly used for searching nearest neighbors with respect to the cosine similarity. However, we have shown recently that the problem of determining a cosine similarity neighborhood can be transformed to the problem of determining a Euclidean neighborhood among normalized forms of original vectors. In this chapter, we provide an experimental comparison of the discussed techniques for determining nearest neighbors with regard to the Euclidean distance and the cosine similarity.

Keywords The triangle inequality · Projection onto dimension · VP-tree · The Euclidean distance · The cosine similarity · Nearest neighbors

M. Kryszkiewicz (✉) · B. Jańczak
Institute of Computer Science, Warsaw University of Technology,
Nowowiejska 15/19, 00-665 Warsaw, Poland
e-mail: mkr@ii.pw.edu.pl

B. Jańczak
e-mail: b.janczak@ii.pw.edu.pl

R. Bembenik et al. (eds.), *Intelligent Tools for Building a Scientific Information Platform: From Research to Implementation*, Studies in Computational Intelligence 541,
DOI: 10.1007/978-3-319-04714-0_3,

1 Introduction

The Euclidean distance and the cosine similarity are often applied for clustering or classifying objects or simply for determining most similar objects or nearest neighbors. In particular, the cosine similarity is popular when dealing with texts [9]. It should be noted that the task of searching nearest neighbors is a typical subtask of the tasks of classification and clustering [1, 5–8]. It is challenging if datasets are large and high dimensional. In this chapter, we discuss three principal approaches to efficient determination of nearest neighbors: namely, a basic triangle inequality approach that applies ordering of vectors with respect to their distances to one reference vector [5, 6], using a VP-tree [13], which is a particular metric tree [10, 12, 14], and using a projection onto a dimension [3]. Also, we discuss a combined application of a number of reference vectors and/or projections onto dimensions and compare two variants of VP-tree. All the techniques have been proposed to prune large numbers of objects that certainly are not nearest neighbors of a given vector. Conceptually, the simplest technique is the one based on the projection and most complex is the one using a VP-tree.

These techniques are well suited to any distance metrics such as the Euclidean distance, but cannot be directly used for searching nearest neighbors with respect to the cosine similarity. However, we have shown recently in [4] that the problem of determining a cosine similarity neighborhood can be transformed to the problem of determining a Euclidean neighborhood among normalized forms of original vectors. Thus the techniques can be indirectly applied in the case of the cosine similarity as well. Nevertheless, the question raises if the transformation influences the efficiency of the techniques.

In this chapter, we will provide an experimental evaluation of the discussed techniques for determining nearest neighbors with regard to the Euclidean distance. We will also examine experimentally if the transformation of the problem of the search of cosine similar nearest neighbors to the search of Euclidean nearest neighbors is beneficial from practical point of view and to which degree in the case of each candidate reduction technique. We will use five benchmark datasets in our experiments with a few thousand vectors up to a few hundred thousand vectors and with a few dimensions up to more than a hundred thousand dimensions.

Our chapter has the following layout. In Sect. 2, we recall basic notions and relations between the Euclidean distance and the cosine similarity. In Sect. 3, we recall definitions of neighborhoods and recall how the problem of determining a cosine similarity neighborhood can be transformed to the problem of determining a neighborhood with regard to the Euclidean distance. The usage of the triangle inequality by means of reference vectors for efficient pruning of non-nearest neighbors is recalled in Sect. 4. Section 5 is devoted to presentation of a VP-tree as a means of reducing candidates for nearest neighbors. We also notice there a difference in properties of two variants of a VP-tree: the one based on medians and the other one based on bounds. In Sect. 6, we present pruning of candidates by means of the projection of vectors onto a dimension. Here, we also consider a

combined application of reference vectors and/or projections onto dimensions. In Sect. 7, we provide an experimental evaluation of the presented techniques on benchmark datasets. Section 8 summarizes our work.

2 The Euclidean Distance, the Cosine Similarity and Their Relation

In the chapter, we consider vectors of the same dimensionality, say n. A vector u will be also denoted as $[u_1, \ldots, u_n]$, where u_i is the value of the i-th dimension of u, $i = 1 \ldots n$. A vector will be called a *zero vector* if all its dimensions are equal zero. Otherwise, the vector will be called *non-zero*.

Vectors' similarity and dissimilarity can be defined in many ways. An important class of dissimilarity measures are distance metrics, which preserve the triangle inequality.

We say that a measure *dis preserves the triangle inequality* if for any vectors u, v, and r, $dis(u,r) \leq dis(u,v) + dis(v,r)$ or, alternatively $dis(u,v) \geq dis(u,r) - dis(v,r)$.

The most popular distance metric is the *Euclidean distance*. The *Euclidean distance* between vectors u and v is denoted by $Euclidean(u, v)$ and is defined as follows:

$$Euclidean(u,v) = \sqrt{\sum_{i=1\ldots n} (u_i - v_i)^2}$$

Among most popular similarity measures is the *cosine similarity*. The *cosine similarity* between vectors u and v is denoted by $cosSim(u, v)$ and is defined as the cosine of the angle between them; that is,

$$cosSim(u,v) = \frac{u \cdot v}{|u||v|}, \text{where:}$$

- $u \cdot v$ is the *standard vector dot product of vectors u and v* and equals $\sum_{i=1\ldots n} u_i v_i$;
- $|u|$ is *the length of vector u* and equals $\sqrt{u \cdot u}$.

In fact, the cosine similarity and the Euclidean distance are related by an equation:

Lemma 1 [4]. *Let u, v be non-zero vectors. Then*:

$$cosSim(u,v) = \frac{|u|^2 + |v|^2 - Euclidean^2(u,v)}{2|u||v|}.$$

Clearly, the cosine similarity between any vectors u and v depends solely on the angle between the vectors and does not depend on their lengths, hence the

calculation of the $cosSim(u, v)$ may be carried out on their *normalized forms* defined as follows:

A *normalized form of a vector u* is denoted by *NF(u)* and is defined as the ratio of u to its length $|u|$. A vector u is defined as a *normalized vector* if $u = NF(u)$. Obviously, the length of a normalized vector equals 1.

Theorem 1 [4]. *Let u, v be non-zero vectors. Then*:

$$cosSim(u, v) = cosSim(NF(u), NF(v)) = \frac{2 - Euclidean^2(NF(u), NF(v))}{2}.$$

Theorem 1 allows deducing that checking whether the cosine similarity between any two vectors equals or exceeds a threshold ε, where $\varepsilon \in [-1, 1]$, can be carried out as checking if the Euclidean distance between the normalized forms of the vectors does not exceed the modified threshold $\varepsilon' = \sqrt{2 - 2\varepsilon}$:

Corollary 1 [4]. *Let u, v be non-zero vectors,* $\varepsilon \in [-1, 1]$ *and* $\varepsilon' = \sqrt{2 - 2\varepsilon}$. *Then*:

$$cosSim(u, v) \geq \varepsilon \text{ iff } Euclidean(NF(u), NF(v)) \leq \varepsilon'.$$

In the case of very high dimensional vectors with, say, tens or hundreds of thousands of dimensions, there may be a problem with a correct representation of the values of dimensions of normalized vectors, which may result in erroneous calculation of the Euclidean distance between them. One may avoid this problem by applying an α *normalized form of a vector u* that is defined as α *NF(u)*, where $\alpha \neq 0$. Large value of α mitigates the problem of calculating distances between normalized forms of high dimensional vectors.

Let $\alpha \neq 0$. One may easily note that: $Euclidean(NF(u), NF(v)) \leq \sqrt{2 - 2\varepsilon} \Leftrightarrow |\alpha|\ Euclidean(NF(u), NF(v)) \leq |\alpha|\ \sqrt{2 - 2\varepsilon} \Leftrightarrow Euclidean(\alpha\ NF(u), \alpha\ NF(v)) \leq |\alpha|\ \sqrt{2 - 2\varepsilon}$. As a result, we may conclude:

Corollary 2 [4]. *Let* $\alpha \neq 0$, *u, v be non-zero vectors,* $\varepsilon \in [-1, 1]$ *and* $\varepsilon' = |\alpha|\sqrt{2 - 2\varepsilon}$. *Then*:

$$cosSim(u, v) \geq \varepsilon \text{ iff } Euclidean(\alpha NF(u), \alpha NF(v)) \leq \varepsilon'.$$

3 ε-Neighborhoods and *k*-Nearest Neighbors

In this section, we first recall the definitions of an ε-Euclidean neighborhood, ε-cosine similarity neighborhood and the method of transforming the latter problem to the former one [4]. Next, we recall how to calculate k-Euclidean nearest neighbors based on an ε-Euclidean neighborhood. Finally, we formally state how to transform the problem of calculating k-cosine similarity nearest neighbors to the

problem of calculating k-Euclidean nearest neighbors. In the next sections of the paper, we will assume that the determination of cosine similarity ε-neighborhoods and k-nearest neighbors is carried out as described in the current section; that is, by calculating the Euclidean distance between α normalized vectors instead of calculating the cosine similarity between original not necessarily α normalized vectors.

In the remainder of the paper, we assume that $\varepsilon \geq 0$ for the Euclidean distance and $\varepsilon \in (0, 1]$ for the cosine similarity if not otherwise stated.

3.1 Euclidean and Cosine ε-Neighborhoods

ε-Euclidean neighborhood of a vector p in D is denoted by $\varepsilon NB_{Euclidean}{}^{D}(p)$ and is defined as the set of all vectors in dataset $D\backslash\{p\}$ that are distant in the Euclidean sense from p by no more than ε; that is,

$$\varepsilon NB_{Euclidean}{}^{D}(p) = \{q \in D\backslash\{p\} | Euclidean(p, q) \leq \varepsilon\}.$$

ε-cosine similarity neighborhood of a vector p in D is denoted by $\varepsilon SNB_{cosSim}{}^{D}(p)$ and is defined as the set of all vectors in dataset $D\backslash\{p\}$ that are cosine similar to p by no less than ε; that is,

$$\varepsilon SNB_{cosSim}{}^{D}(p) = \{q \in D\backslash\{p\} | cosSim(p, q) \geq \varepsilon\}$$

Corollary 2 allows transforming the problem of determining a cosine similarity neighborhood of a given vector u within a set of vectors D (Problem P1) to the problem of determining a Euclidean neighborhood of α $NF(u)$ within the vector set D' consisting of α normalized forms of the vectors from D (Problem P2).

Theorem 2 [4]. *Let D be a set of m non-zero vectors $\{p_{(1)}, \ldots, p_{(m)}\}$, $\alpha \neq 0$, D' be the set of m vectors $\{u_{(1)}, \ldots, u_{(m)}\}$ such that $u_{(i)} = \alpha\, NF(p_{(i)})$, $i = 1 \ldots m$, $\varepsilon \in [-1, 1]$ and $\varepsilon' = |\alpha|\sqrt{2 - 2\varepsilon}$. Then*:

$$\varepsilon SNB_{cosSim}{}^{D}\left(p_{(i)}\right) = \left\{p_{(j)} \in D | u_{(j)} \in \varepsilon'\text{-}NB_{Euclidean}{}^{D'}\left(u_{(i)}\right)\right\}.$$

Please note that all vectors considered in the resultant problem P2 have α normalized forms and so have length equal to α.

Example 1. Let us consider determination of an ε-cosine similarity neighborhood of a vector $p_{(i)}$ in dataset $D = \{p_{(1)}, \ldots, p_{(8)}\}$ from Fig. 1 for the cosine similarity threshold $\varepsilon = 0.9856$ (which roughly corresponds to the angle of 9.74°). Let $\alpha = 1$. Then, the problem can be transformed to the problem of determining ε'-Euclidean neighborhood of $u_{(i)} = NF(p_{(i)})$ in the set $D' = \{u_{(1)}, \ldots, u_{(8)}\}$ containing normalized forms of the vectors from D, provided $\varepsilon' = \sqrt{2 - 2\varepsilon} \approx 0.17$. The resultant set D' is presented in Fig. 2.

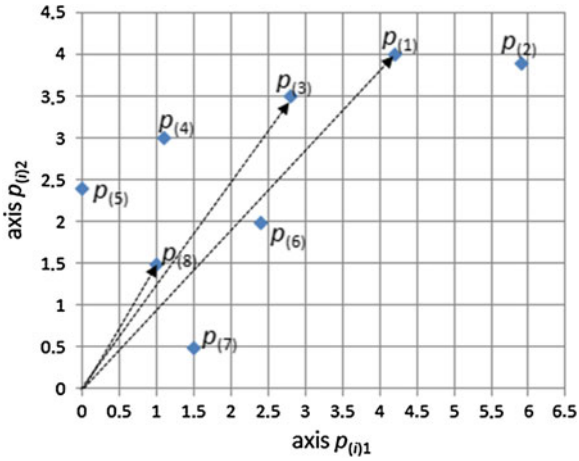

Fig. 1 Sample set D of vectors (after [4])

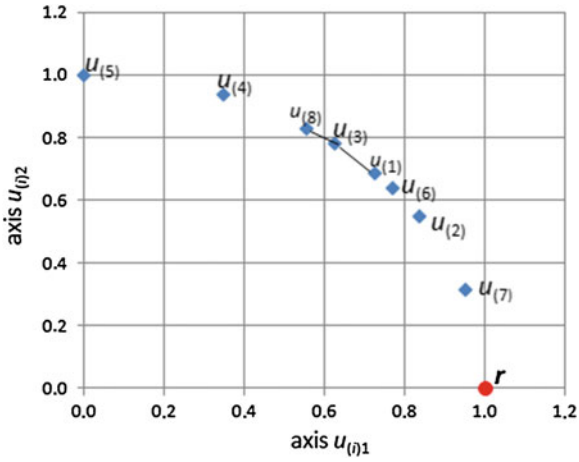

Fig. 2 Set D' containing normalized forms of vectors from D (after [4])

3.2 Euclidean and Cosine k-Nearest Neighbors

Instead of looking for an ε-Euclidean neighborhood (or an ε-cosine similarity neighborhood), one may be interested in determining *k-Euclidean nearest neighbors* (*k-cosine similarity nearest neighbors*, respectively). The task of searching k-Euclidean nearest neighbors can be still considered as searching an ε-Euclidean neighborhood for some ε value (possibly different for different vectors and adjusted dynamically) as follows:

Let K be a set containing any k vectors from $D\backslash\{p\}$ and $\varepsilon = \max\{Euclidean(p, q) | \ q \in K\}$. Then, k-Euclidean nearest neighbors are guaranteed to be found within ε Euclidean distance from vector p; that is, they are contained in

ε-$NB_{Euclidean}{}^{D}(p)$. In practice, one may apply some heuristics to determine possibly best value (that is, as little as possible) of ε within which k-nearest neighbors of p are guaranteed to be found and the value of ε can be re-estimated (and thus possibly narrowed) when calculating the distance between p and next vectors from $D\backslash(K \cup \{p\})$ [6].

The above approach to searching k-Euclidean nearest neighbors can be easily adapted to searching k-cosine similarity nearest neighbors based on the following observation.

Proposition 1. *Let D be a set of m non-zero vectors $\{p_{(1)}, \ldots, p_{(m)}\}$, $\alpha \neq 0$, D' be the set of m vectors $\{u_{(1)}, \ldots, u_{(m)}\}$ such that $u_{(i)} = \alpha\, NF(p_{(i)})$, $i = 1 \ldots m$. Then the following statements are equivalent*:

- $\{p_{(j1)}, \ldots, p_{(jk)}\}$ are k-cosine similarity nearest neighbors of $p_{(i)}$ in D;
- $\{u_{(j1)}, \ldots, u_{(jk)}\}$ are k-cosine similarity nearest neighbors of $u_{(i)}$ in D';
- $\{u_{(j1)}, \ldots, u_{(jk)}\}$ are k-Euclidean nearest neighbors of $u_{(i)}$ in D'.

Thus, the problem of determining k-cosine similarity nearest neighbors in a given set of vectors D is transformable to the problem of determining k-Euclidean nearest neighbors in set D' being the set of (α) normalized forms of vectors from D.

4 Basic Approach to Triangle Inequality-Based Search of Euclidean ε-Neighborhoods and Nearest Neighbors

4.1 Determining ε-Euclidean Neighborhoods with the Triangle Inequality

In this section, we recall the method of determining ε-Euclidean neighborhoods as proposed in [5]. We start with Lemma 2, which follows from the triangle inequality.

Lemma 2 [5]. *Let D be a set of vectors. Then, for any vectors $u, v \in D$ and any vector r, the following holds*:

$$Euclidean(u,r) - Euclidean(v,r) > \varepsilon \Rightarrow Euclidean(u,v) > \varepsilon \Rightarrow$$
$$v \notin \varepsilon NB_{Euclidean}{}^{D}(u) \wedge u \notin \varepsilon NB_{Euclidean}{}^{D}(v).$$

Now, let us consider vector q such that $Euclidean(q,r) > Euclidean(u,r)$. *If* $Euclidean(u,r) - Euclidean(v,r) > \varepsilon$, then also $Euclidean(q,r) - Euclidean(v,r) > \varepsilon$, and thus, one may conclude that $v \notin \varepsilon$-$NB_{Euclidean}{}^{D}(q)$ *and* $q \notin \varepsilon$-$NB_{Euclidean}{}^{D}(v)$ without calculating the real distance between q and v. This observation provides the intuition behind Theorem 3.

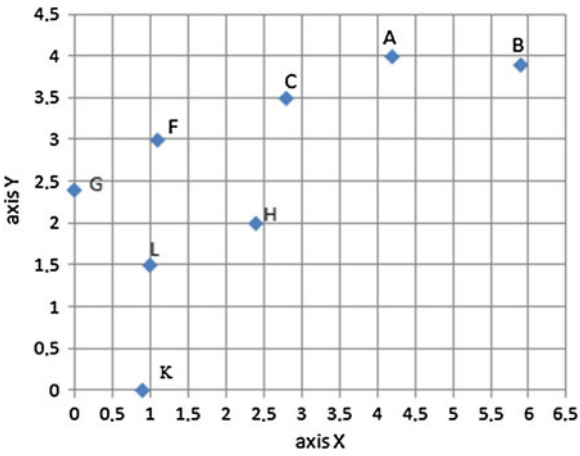

Fig. 3 Set of vectors D [5]

Theorem 3 [5]. *Let r be any vector and D be a set of vectors ordered in a non-decreasing way with regard to their distances to r. Let $u \in D$, f be a vector following vector u in D such that Euclidean(f,r) – Euclidean(u,r) $> \varepsilon$, and p be a vector preceding vector u in D such that Euclidean(u,r) – Euclidean(p,r) $> \varepsilon$. Then*:

(a) f and all vectors following f in D do not belong to $\varepsilon\text{-}NB_{Euclidean}{}^{D}(u)$;
(b) p and all vectors preceding p in D do not belong to $\varepsilon\text{-}NB_{Euclidean}{}^{D}(u)$.

As follows from Theorem 3, it makes sense to order all vectors in a given dataset D with regard to a reference vector r as this enables simple elimination of a potentially large subset of vectors that certainly do not belong to an ε-Euclidean neighborhood of an analyzed vector.

Example 2. (determining Euclidean ε-neighborhood with the triangle inequality). Let r be a vector [0, 0]. Figure 3 shows an example set D of two dimensional vectors. Table 1 illustrates the same set D ordered in a non-decreasing way with respect to the distances of its vectors to vector r. Let us consider the determination of the Euclidean ε-neighborhood of vector F, where $\varepsilon = 0.5$. We note that $Euclidean(F,r) = 3.2$, the first vector following F in D such that Euclidean(f,r) – $Euclidean(F,r) > \varepsilon$ is vector C ($Euclidean(C,r) - Euclidean(F,r) = 4.5 - 3.2 = 1.3 > \varepsilon$), and the first vector preceding F in D such that $Euclidean(F,r) - Euclidean(p,r) > \varepsilon$ is G ($Euclidean(F,r) - Euclidean(G,r) = 3.2 - 2.4 = 0.8 > \varepsilon$). By Theorem 3, neither C nor any vectors following C belong to $\varepsilon\text{-}NB_{Euclidean}{}^{D}(F)$ as well as neither G nor any vectors preceding G belong to $\varepsilon\text{-}NB_{Euclidean}{}^{D}(F)$. As a result, H is the only vector for which it is necessary to calculate its actual distance to F in order to determine $\varepsilon\text{-}NB_{Euclidean}{}^{D}(F)$ properly.

In the sequel, a vector to which the distances of all vectors in D have been determined will be called a *reference vector*.

Table 1 Ordered set of vectors D from Fig. 3 with their distances to reference vector $r(0, 0)$

$q \in D$	X	Y	$Euclidean(q,r)$
K	0.9	0.0	0.9
L	1.0	1.5	1.8
G	0.0	2.4	2.4
H	2.4	2.0	3.1
F	**1.1**	**3.0**	**3.2**
C	2.8	3.5	4.5
A	4.2	4.0	5.8
B	5.9	3.9	7.1

In fact, one may use more than one reference vector for estimating the distance among pairs of vectors [5]. Additional reference vectors should be used only when the basic reference vector according to which the vectors in D are ordered is not sufficient to state that a given vector u does not belong to ε-neighborhood of another vector v. The estimation of the distance between u and v by means of an additional reference vector is based on Lemma 2. The actual distance between the two vectors u and v is calculated only when none of reference vectors is sufficient to state that $v \notin \mathfrak{E}NB_{Euclidean}{}^{D}(u)$.

Let us note that the presented method assumes that an object u for which we wish to find its ε-neighborhood among vectors in a set D ordered with respect to vectors' Euclidean distances to a reference vector r also belongs to D. The method, however, can be easily adapted in the case when u does not belong to D. In such a case, it would be useful to calculate the Euclidean distance from u to reference vector r for determining the position in D where u could be inserted without violating the maintained order of vectors. Knowing this position of u, one may determine a respective vector p preceding u in D and a respective vector f following u in D as specified in Theorem 3. This would allow for the candidate reduction which follows from Theorem 3. Clearly, the determination of a respective position of u in D has logarithmic time complexity with regard to the number of vectors in D. In fact, the determination of this position does not require the real insertion of u into D. Let us also note that eventual using of additional reference vectors according to Lemma 2 may require the calculation of the Euclidean distance from u to some or all of these reference vectors.

4.2 Determining k-Euclidean Nearest Neighbors with the Triangle Inequality

In this section, we recall the method of determining k-Euclidean nearest neighbors, as introduced in [4, 6]. The method assumes that all vectors in a given dataset D are ordered with respect to some reference vector r. Then, for each vector u in D, its k-nearest neighbors can be determined in the following steps:

(1) The radius, say ε, within which k-nearest neighbors of u are guaranteed to be found is estimated based on the real distances of k vectors located directly before and/or after u in the ordered set D.[1]
(2) Next, ε-$NB_{Euclidean}{}^{D}(u)$ is determined in a way similar to the one described in Sect. 4.1. Clearly, the real distances to u from vectors considered in phase 1, do not need to be calculated again.
(3) k-nearest neighbors of u are determined as a subset of ε-$NB_{Euclidean}{}^{D}(u)$ found in step 2.

The above description is a bit simplified. In [6], steps 2 and 3 were not split, and the value of ε was adapted (narrowed) with each new candidate vector having a chance to belong to k nearest neighbors of u. Also, more than one reference vector can be used to potentially reduce the number of calculations of real distances between vectors [6].

5 Using VP-tree for Determining Euclidean ε-Neighborhoods and Nearest Neighbors

A VP-tree (Vantage Points tree) index was originally offered as a tool using the triangle inequality for determining a nearest neighbor of a vector u in a set of vectors D provided the distance between the nearest neighbor and u does not exceed a user-specified threshold value ε [13]. Thus, the problem consisted in searching one nearest neighbor of u in ε-$NB_{Euclidean}{}^{D}(u)$. Clearly, it can be easily adapted to searching all vectors in ε-$NB_{Euclidean}{}^{D}(u)$ as well as to searching any k nearest neighbors. In the latter case, the estimation of the ε radius within which k nearest neighbors are guaranteed to be found can be carried out as discussed in Sect. 3.2. In the remainder of this section, we focus on describing VP-tree in the context of determining ε-Euclidean neighborhoods.

VP-tree is a binary tree and its structure is as follows: The number of nodes in a VP-tree is equal to the cardinality of D. Each vector v (or its identifier) from D is stored in one node in the VP-tree altogether with the following information:

- *the median* $\mu(v)$ of the distances from v to all vectors stored in the subtree whose root is the node containing v; later on, we will denote this subtree by $S(v)$;
- *a reference to its left child node* being a root of the $LS(v)$ subtree that stores those vectors from $S(v)\backslash\{v\}$ whose distances to v are less than μ;
- *a reference to its right child node* being a root of the $RS(v)$ subtree that stores those vectors from $S(v)\backslash\{v\}$ whose distances to v are at least μ.

[1] Please note that one may estimate the radius ε within which k nearest neighbors of u are guaranteed to be found based on the real distances of any k vectors in set D that are different from u; restricting calculations to vectors located directly before and/or after u in the ordered set D is a heuristic, which is anticipated to lead to smaller values of ε.

Let us now illustrate the way in which ε-$NB_{Euclidean}{}^{D}(u)$ of a given vector u can be determined by means of a VP-tree. First, the distance is calculated between u and the vector stored in the root, say r, of the VP-tree. Let $\mu(r)$ be the value of the median stored in the root. The median is used to determine a'priori if any of the subtrees ($LS(r)$ or $RS(r)$) does not contain the sought neighbors of u. The decision about eventual ignoring any of the subtrees is made based on the following conditions:

(C1) $Euclidean(u,r) - \mu(r) \geq \varepsilon$. Then, for each vector v stored in $LS(r)$, the following holds: $Euclidean(u,r) - Euclidean(v,r) > \varepsilon$ and thus, by Lemma 2, $Euclidean(u,v) > \varepsilon$, so $LS(r)$ does not contain any nearest neighbor of u within the ε radius and should not be hence visited.

(C2) $\mu(r) - Euclidean(u,r) > \varepsilon$. Then, for each vector v stored in $RS(r)$, the following holds: $Euclidean(v,r) - Euclidean(u,r) > \varepsilon$ and thus, by Lemma 2, $Euclidean(u,v) > \varepsilon$, so $RS(r)$ does not contain any nearest neighbor of u within the ε radius and should not be hence visited.

Observation 1. Conditions C1 and C2 are mutually exclusive for $\varepsilon \geq 0$, so at most one subtree of $S(v)$ may be ignored while searching an ε-Euclidean neighborhood based on these conditions.

The presented procedure is repeated recursively for roots of subtrees that were not eliminated by the conditions C1, C2.

The structure and performance of a VP-tree strongly depends on the way vectors are assigned to nodes. As proposed in [13], each vector v in a node of VP-tree should imply the maximal variance among distances from v to all vectors in the subtree $S(v)$. As fulfillment of this condition would result in an unacceptably large time of building the tree, the authors of [13] decided to treat only a random sample of vectors as candidates to be assigned to a node and to determine corresponding medians and variances based on another random sample of vectors. For details of this procedure, please see [13].

As noted in [13], one may use a pair of values called $left\_bound(r)$ and $right\_bound(r)$, respectively, instead of the median, to potentially increase the usefulness of the conditions C1, C2. If $S(r)$ is a considered subtree, than $left\_bound(r)$ is defined as the maximum of the distances from r to all vectors in $LS(r)$, while $right\_bound(r)$ is defined as the minimum of the distances from r to all vectors in $RS(r)$. Clearly, $left\_bound(r) < \mu(r) \leq right\_bound(r)$ for any vector r. So, $Euclidean(u,r) - \mu(r) \geq \varepsilon$ implies $Euclidean(u,r) - left\_bound(r) > \varepsilon$ and $\mu(r) - Euclidean(u,r) > \varepsilon$ implies $right\_bound(r) - Euclidean(u,r) > \varepsilon$. In fact, the conditions C1 and C2 can be replaced by the following less restrictive conditions C1′ and C2′, respectively:

(C1′) $Euclidean(u,r) - left\_bound(r) > \varepsilon$ (then $LS(r)$ does not contain any nearest neighbor of u within the ε radius and should not be hence visited).

(C2′) $right\_bound(r) - Euclidean(u,r) > \varepsilon$ (then $RS(r)$ does not contain any nearest neighbor of u within the ε radius and should not be hence visited).

Observation 2. Conditions C1′ and C2′ are not guaranteed to be mutually exclusive for $\varepsilon \geq 0$, so it is possible that both subtrees of $S(r)$ may be ignored while searching an ε-Euclidean neighborhood based on these conditions.

Example 3. Let $\varepsilon = 1$ and r be a vector represented by a node in a VP-tree such that $left\_bound(r) = 8.5$ and $right\_bound(r) = 12$. Let u be a vector such that $Euclidean(u,r) = 10$. Then, $Euclidean(u,r) - left\_bound(r) > \varepsilon$ and $right\_bound(r) - Euclidean(u, r) > \varepsilon$, which means that neither $LS(r)$ nor $RS(r)$ contains any nearest neighbor of r within the ε radius.

Corollary 3. *Let u be any vector. Then, the determination ε-Euclidean neighborhood of u, where $\varepsilon \geq 0$, by means of a VP-tree*:

(a) Requires visiting at least one entire path in the VP-tree leading from its root to a leaf and calculating the Euclidean distances from u to all vectors stored in the nodes of this path when using conditions C1 and C2.
(b) May not require visiting any entire path in the VP-tree that ends in a leaf when using conditions C1′ and C2′.

So, if u is a vector for which we wish to determine its ε-Euclidean neighborhood, where $\varepsilon \geq 0$, then at least one leaf and all its ancestors in the VP-tree will be visited when applying conditions C1 and C2 (which involve medians), while it may happen that no leaf will be visited when applying conditions C1′ and C2′ (which involve left_bounds and right_bounds).

6 Using Vector Projection onto a Dimension for Determining Euclidean ε-Neighborhoods and Nearest Neighbors

In this section, we recall after [3] how to use vector projection onto a dimension to reduce the number of vectors that should be verified as candidates for ε-Euclidean neighborhoods or k-Euclidean nearest neighbors.

One may easily note that for any dimension l, $l \in [1, \ldots, n]$, and any two vectors u and v, $|u_l - v_l| = \sqrt{(u_l - v_l)^2} \leq \sqrt{\sum_{i=1\ldots n}(u_i - v_i)^2} = Euclidean(u, v)$. This implies Property 1.

Property 1. *Let l be an index of a dimension, $l \in [1, \ldots, n]$, and u, v be any vectors. Then*:

$$|u_l - v_l| > \varepsilon \Rightarrow Euclidean(u, v) > \varepsilon \Rightarrow$$
$$v \notin \varepsilon NB_{Euclidean}{}^{D}(u) \wedge u \notin \varepsilon NB_{Euclidean}{}^{D}(v).$$

Property 1 provides the intuition behind Theorem 4.

Theorem 4 [3]. *Let l be an index of a dimension, $l \in [1, \ldots, n]$, and D be a set of vectors ordered in a non-decreasing way with regard to the values of their l-th dimension. Let $u \in D$, f be a vector following vector u in D such that $f_l - u_l > \varepsilon$, and p be a vector preceding vector u in D such that $u_l - p_l > \varepsilon$. Then:*

(a) f and all vectors following f in D do not belong to $\varepsilon\text{-}NB_{Euclidean}{}^{D}(u)$;
(b) p and all vectors preceding p in D do not belong to $\varepsilon\text{-}NB_{Euclidean}{}^{D}(u)$.

Please note that Theorem 4 is analogical to Theorem 3, which is based on using the triangle inequality. Both theorems assume that vectors are ordered and use this ordering to ignore some false candidates for ε-Euclidean neighborhoods members. The only difference between the two theorems is that the ordering value for a vector v in Theorem 3 is calculated as the Euclidean distance between v and some fixed reference vector r, while in Theorem 4, the ordering value for v equals the value of some fixed l-th dimension of v.

Clearly, one may also use projections onto additional dimensions to reduce the number of candidates for ε-Euclidean neighborhoods. Projections onto additional dimensions should be used only when the basic dimension according to which the vectors in D are ordered is not sufficient to state that a given vector u does not belong to an ε-neighborhood of another vector v. The estimation of the distance between u and v by means of projection onto any additional dimension is based on Property 1.

In the sequel, we assume that the calculation of k-Euclidean nearest neighbors by means of projection onto a dimension is carried out in an analogical way as their determination by means of the triangle inequality, as described in Sect. 4.2.

Let us also note that one may use both reference vectors and projections onto dimensions for determining Euclidean ε-neighborhoods and k-nearest neighbors, irrespective of whether the set of vectors is ordered with regard to a basic reference vector or with regard to the value of a basic dimension. This approach to reducing candidate vectors was also evaluated by us and its results are provided in Sect. 7.

7 Experiments

In this section, we report the results of the experiments we have carried out on five benchmark datasets (see Sect. 7.1) in order to evaluate the presented techniques of reducing the number of candidates for k nearest neighbors for the Euclidean distance (see Sect. 7.2) as well as cosine similarity measure (see Sect. 7.3).

7.1 Description of Datasets Used in Experiments

In Table 2, we provide a short description of used benchmark datasets.

Table 2 Characteristics of used benchmark datasets

Name	No. of vectors	No. of dimensions	Source
Sequoia	62,556	2	[11]
Birch	100,000	2	[15]
Cup98	96,367	56	http://kdd.ics.uci.edu/databases/kddcup98/kddcup98.html
Covtype	581,012	55	http://ics.uci.edu/pub/machine-learning-databases/covtype/covtype.info
Sports	8,580	126,373	http://glaros.dtc.umn.edu/gkhome/cluto/cluto/download

We have tested how the representation of data influences the efficiency of searching similar vectors. It turned out that sparse datasets (*sports*) should be internally also represented as sparse, dense datasets (*sequoia*, *birch*, *cup98*) should be internally represented as dense or otherwise the calculation time is considerably longer [2]. The *covtype* dataset is neither typically sparse, not typically dense. For this dataset, it turned that the efficiency was more or less the same irrespectively of its internal representation [2].

7.2 Determining k-Euclidean Nearest Neighborhoods

In this section, we compare the performance of calculating k-Euclidean nearest neighbors for 10 % of vectors present in the examined datasets by means of the following algorithms[2]:

- TI$[r_1][r_2]\ldots[r_m]$—this variant requires vectors to be ordered with respect to their Euclidean distances to reference vector r_1. In order to reduce the number of candidates, the triangle inequality is used with respect to at most m reference vectors: the first reference vector r_1 is used according to Theorem 3; remaining specified reference vectors $r_2,\ldots, r_m$ (if any) are used when needed according to Lemma 2;
- Proj$(dim_1)(dim_2)\ldots(dim_m)$—this variant requires vectors to be ordered with respect to their values of dimension dim_1. In order to reduce the number of candidates, projections onto at most m dimensions are used. Projection onto dimension dim_1 is used in accordance with Theorem 4; remaining specified dimensions $dim_2,\ldots, dim_m$ (if any) are used when needed according to Property 1;

[2] In the case when more than one vector is in a same distance from a given vector u, there may be a number of alternative sets containing exactly k nearest neighbors of u. The algorithms we tested return all vectors that are no more distant than a most distant k-th Euclidean nearest neighbor of a given vector u. So, the number of returned neighbors of u may happen to be larger than k.

- TI[r_1][r_2]…[r_m]_Proj(dim_1)(dim_2)…(dim_s)—this variant requires vectors to be ordered with respect to their Euclidean distances to reference vector r_1. In order to reduce the number of candidates, reference vector r_1 is used according to Theorem 3, remaining specified reference vectors r_2,…, r_m (if any) are used when needed according to Lemma 2 and specified dimensions dim_1,…, dim_s are used when needed according to Property 1;
- Proj(dim_1)(dim_2)…(dim_m)_TI[r_1][r_2]…[r_s]—this variant requires vectors to be ordered with respect to their values of dimension dim_1. In order to reduce the number of candidates, projection onto dimension dim_1 is used in accordance with Theorem 4, remaining specified dimensions dim_2,…, dim_m (if any) are used when needed according to Property 1 and specified reference vectors r_1,…, r_s are used when needed according to Lemma 2;
- VP—this variant requires vectors to be stored in a VP-tree and uses medians to determine subtrees that should (not) be visited;
- [VP]—this variant requires vectors to be stored in a VP-tree and uses left_ bounds and right_bounds to determine subtrees that should (not) be visited.

In order to underline that more than one reference vector is used by a TI algorithm, we will denote this algorithm interchangeably as TI+. Similarly, if more than one dimension for projecting is used, we will write Proj+ interchangeably with Proj.

All the algorithms which use reference vectors assume that the information about Euclidean distances from each vector in a source dataset to all reference vectors is pre-calculated. We need to mention, nevertheless, that in our experiments each vector u for which its neighbors were looked for played a double role: (i) a role of a vector belonging to the dataset D in which neighbors were searched (in this case, the information pre-calculated for u was treated by an algorithm as available) and (ii) a role of an external vector for which its neighbors were searched in D (in this case, the algorithm treated the information pre-calculated for u as unavailable and calculated it again if needed).

In the sequel, we will apply the following notation related to applied reference vectors and dimensions:

- [max]—a vector that has maximal domain values for all dimensions;
- [min]—a vector that has minimal domain values for all dimensions;
- [max_min]—a vector that has maximal domain values for all odd dimensions and minimal domain values for all even dimensions;
- [rand]—a vector that has randomly selected domain values for all dimensions;
- [rand] × m—a sequence of m vectors [r_1][r_2]…[r_m], where each vector r_i, where $i = 1…m$, has randomly selected domain values for all dimensions;
- (dmax)—a dimension with most wide domain;
- (drand)—a random dimension;
- (l)—dimension l, where $l \in [1, \ldots, n]$.

Figures 4, 5, 6, 7 and 8 show in a logarithmic scale the average number of the Euclidean distance calculations per an evaluated vector for $k = 1, 2, 5, 10, 20$ and

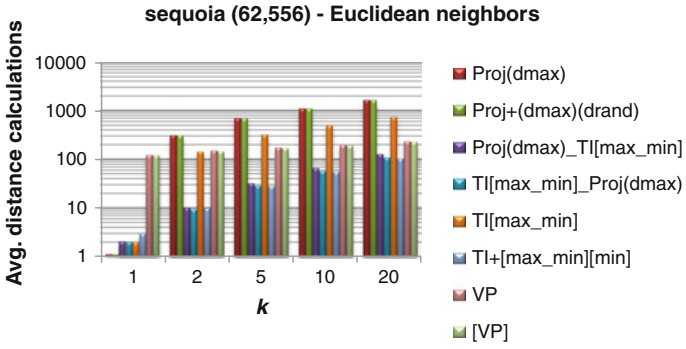

Fig. 4 The avg. no. of the Euclidean distance calculations when searching *k*-Euclidean nearest neighbors of a vector in 2-dimensional *sequoia* dataset of 62,556 vectors (log. scale)

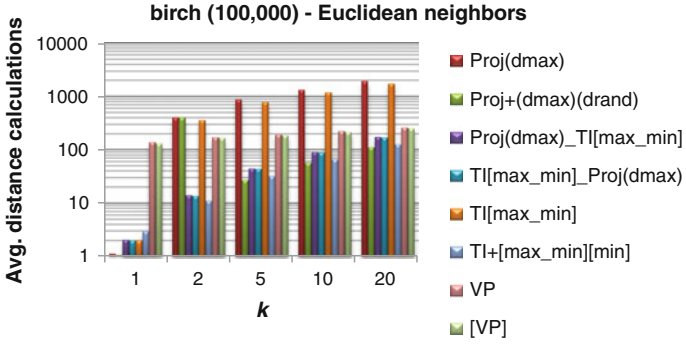

Fig. 5 The avg. no. of the Euclidean distance calculations when searching *k*-Euclidean nearest neighbors of a vector in 2-dimensional *birch* dataset of 100,000 vectors (log. scale)

respective datasets. In the case of the algorithms that do not use reference vectors, this number is equal to the average number of the candidate vectors in D to which the Euclidean distance from an evaluated vector was calculated. In the current implementation of the algorithms that use reference vectors, the Euclidean distances are calculated from an evaluated vector also to the all reference vectors. These additional Euclidean distance calculations are included in the presented statistics.

For all non-sparse datasets (Figs. 4, 5, 6 and 7), the application of each of the presented optimization techniques reduced the number of the Euclidean distance calculations by orders of magnitude in comparison with the number of vectors in a searched dataset, though to a different degree. For these datasets, it was always possible to determine a respective number of reference vectors that made TI algorithms more efficient than the variants using VP-tree for all tested values of k. In the case of sparse *sports* dataset, the usefulness of the optimization techniques turned out much lower (Figs. 8 and 9 present the experimental results for *sports* in a logarithmic scale and linear scale, respectively).

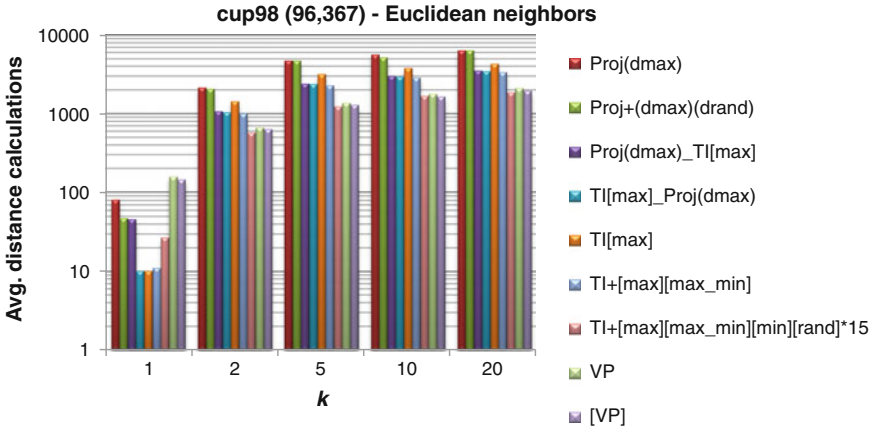

Fig. 6 The avg. no. of the Euclidean distance calculations when searching k-Euclidean nearest neighbors of a vector in 56-dimensional *cup98* dataset of 96,367 vectors (log. scale)

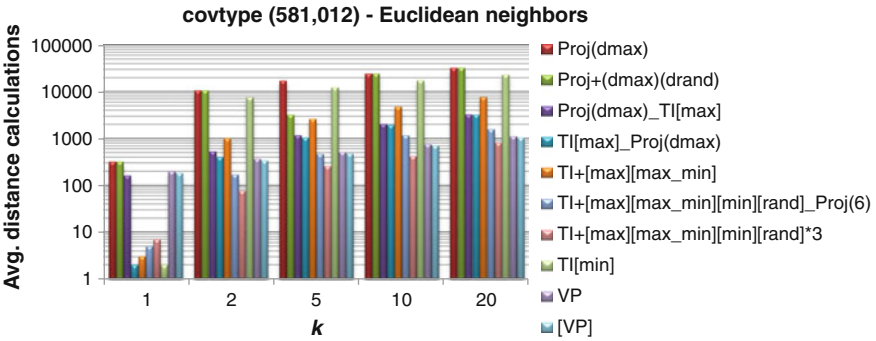

Fig. 7 The avg. no. of the Euclidean distance calculations when searching k-Euclidean nearest neighbors of a vector in 55-dimensional *covtype* dataset of 581,012 vectors (log. scale)

7.3 Determining k-Cosine Similarity Nearest Neighborhoods

In this section, we compare the performance of calculating k-cosine similarity nearest neighbors by equivalent calculation of k-Euclidean nearest neighbors among α normalized forms of original vectors, where $\alpha = 1000$.[3] We tested the same algorithms determining k-Euclidean nearest neighbors as in Sect. 7.2, but the calculations were carried out this time on α normalized forms of original vectors

[3] When applying $\alpha = 1$, the nearest neighbors happen to be incorrectly determined because of errors introduced during normalization of vectors. Hence, we decided to apply larger value of α.

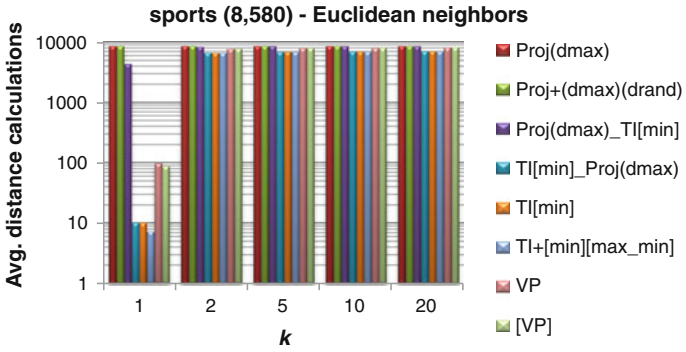

Fig. 8 The avg. no. of the Euclidean distance calculations when searching *k*-Euclidean nearest neighbors of a vector in 126,373-dimensional *sports* dataset of 8,580 vectors (log. scale)

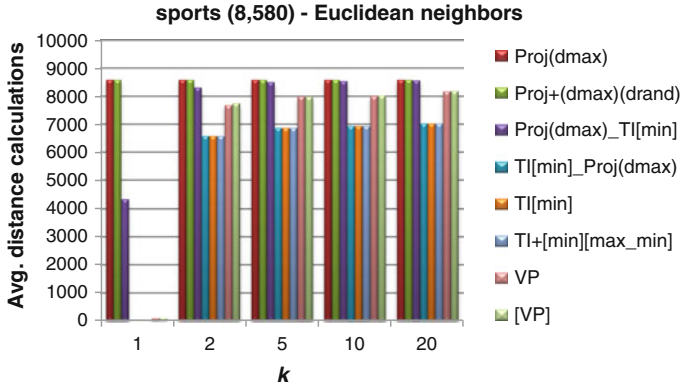

Fig. 9 The avg. no. of the Euclidean distance calculations when searching *k*-Euclidean nearest neighbors of a vector in 126,373-dimensional *sports* dataset of 8,580 vectors (linear scale)

instead of original vectors themselves and reference vectors were also chosen with respect to α normalized values rather than original ones.

Figures 10, 11, 12, 13 and 14 show in a logarithmic scale the average number of the Euclidean distance calculations per an evaluated vector for $k = 1, 2, 5, 10, 20$ and respective datasets. For all non-sparse datasets (Figs. 10, 11, 12 and 13), the application of each of the presented optimization techniques reduced the number of the Euclidean distance calculations by orders of magnitude in comparison with the number of vectors in a searched dataset, though to a different degree. For three datasets (*sequoia*, *birch*, *covtype*) out of four non-sparse datasets, it was always possible to determine a respective number of reference vectors that made TI algorithms more efficient than the variants using VP-tree. In the case of sparse

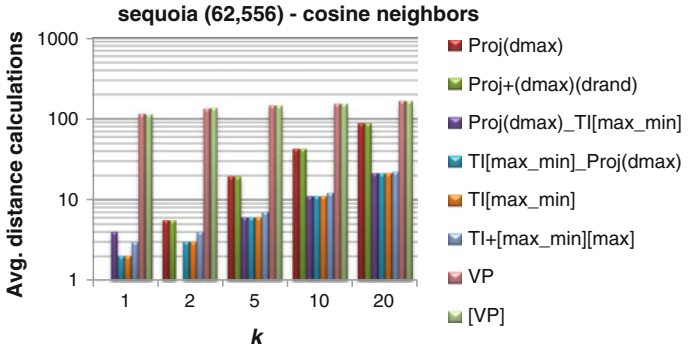

Fig. 10 The avg. no. of the Euclidean distance calculations when searching *k*-cosine nearest neighbors of a vector in 2-dimensional *sequoia* dataset of 62,556 vectors (log. scale)

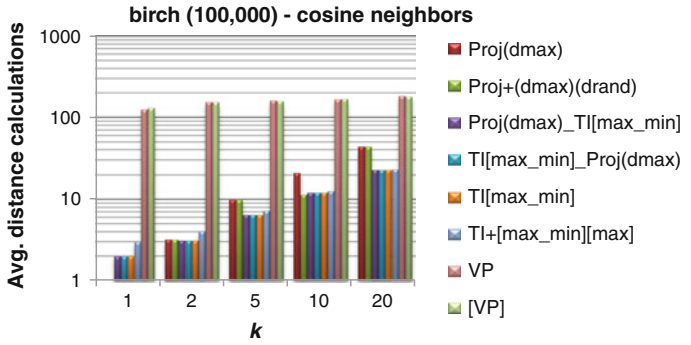

Fig. 11 The avg. no. of the Euclidean distance calculations when searching *k*-cosine nearest neighbors of a vector in 2-dimensional *birch* dataset of 100,000 vectors (log. scale)

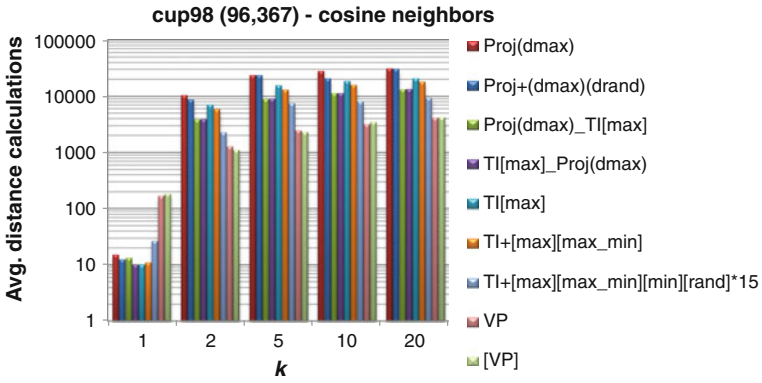

Fig. 12 The avg. no. of the Euclidean distance calculations when searching *k*-cosine nearest neighbors of a vector in 56-dimensional *cup98* dataset of 96,367 vectors (log. scale)

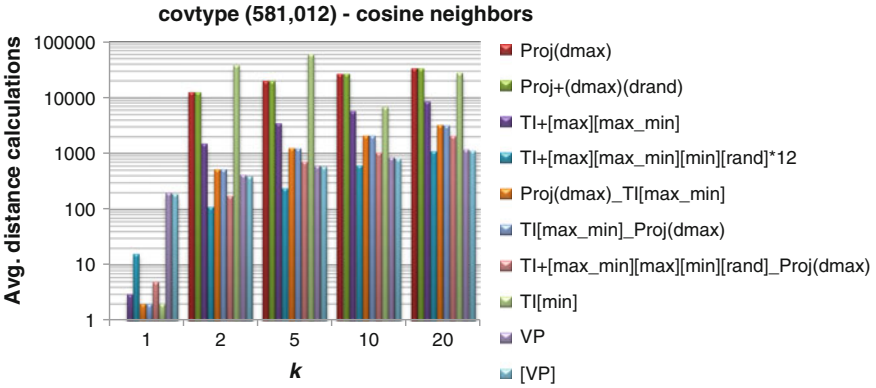

Fig. 13 The avg. no. of the Euclidean distance calculations when searching k-cosine nearest neighbors of a vector in 55-dimensional *covtype* dataset of 581,012 vectors (log. scale)

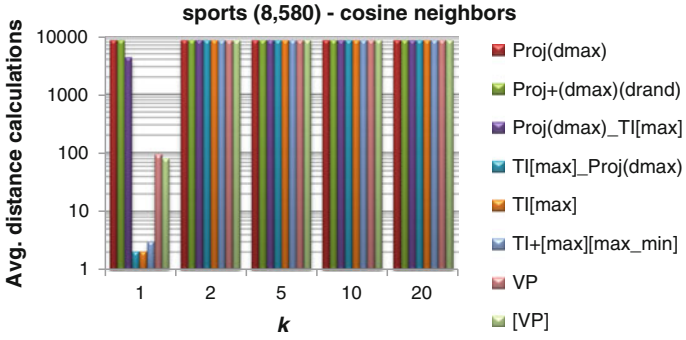

Fig. 14 The avg. no. of the Euclidean distance calculations when searching k-cosine nearest neighbors of a vector in 126,373-dimensional *sports* dataset of 8,580 vectors (log. scale)

sports dataset, the optimization techniques did not turned out useful for $k > 1$ (Figs. 14 and 15 present the experimental results for *sports* in a logarithmic scale and linear scale, respectively).

8 Summary

In this paper, we have recalled three principal approaches to efficient determination of nearest neighbors: namely, using the triangle inequality when vectors are ordered with respect to their distances to one reference vector, using a metric VP-tree and using a projection onto a dimension. Also, we have discussed a combined application of a number of reference vectors and/or projections onto dimensions

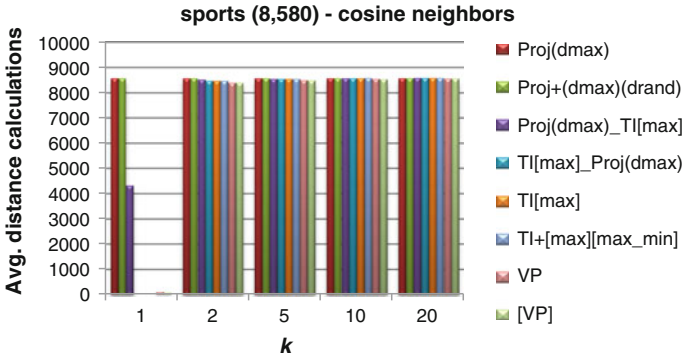

Fig. 15 The avg. no. of the Euclidean distance calculations when searching *k*-cosine nearest neighbors of a vector in 126,373-dimensional *sports* dataset of 8,580 vectors (linear scale)

and have compared two variants of VP-tree. The techniques are well suited to any distance metrics such as the Euclidean distance. They can also be applied directly in the case of the cosine similarity after transforming the problem of finding most cosine similar neighbors into the problem of finding nearest Euclidean neighbors.

In this paper, we provided an experimental comparison of the discussed techniques for determining nearest neighbors with regard to the Euclidean distance and the cosine similarity. All the techniques turned out powerful both for the Euclidean distance and the cosine similarity in the case of used non-sparse benchmark datasets (*sequoia*, *birch*, *cup98*, *covtype*)—the number of calculations of the Euclidean distances turned out by orders of magnitude lower than the number of such calculations when applying brute force approach in which an evaluated vector had to be compared with all vectors in a searched dataset. In the case of the Euclidean distance, it was always possible to obtain a more efficient variant of the algorithm using a few reference vectors (typically one or two, but in the case of the *cup98* dataset-18) than either variant using VP-tree. In the case of the cosine similarity, it was possible to obtain a more efficient variant of the algorithm using a few reference vectors than either variant using VP-tree for all non-sparse datasets except for the *cup98* dataset. Using reference vectors turned out useful for determining Euclidean nearest neighbors also in the case of the sparse *sports* dataset, though to much lower degree than in the case of the non-sparse datasets. Using VP-tree for determining Euclidean nearest neighbors also reduced the number of Euclidean distance calculations for this dataset, but was definitely less efficient than using the algorithm applying even one reference vector. No candidate reduction technique turned out practically useful when searching most cosine similar neighbors in the *sports* dataset.

We note that unlike vantage points that are vectors from a given set of vectors D in which neighbors are searched, reference vectors may be arbitrary. In our experiments, we typically constructed a basic reference vector as a vector with maximal and/or minimal domain values and then ordered vectors in D with respect to their distances to this basic reference vector.

In the paper, we noticed that the determination of an ε-Euclidean distance neighborhood by means of VP-tree that uses left_bounds and right_bounds may not require visiting any entire path in the VP-tree that ends in a leaf. To the contrary, the determination of an ε-Euclidean distance neighborhood by means of VP-tree that uses medians requires calculating the Euclidean distances to all vectors stored in nodes of at least one entire path in the VP-tree that starts in its root and ends in a leaf. This observation suggests that VP-tree using bounds should be not less efficient than VP-tree using medians. Our experiments show that the variant of VP-tree that uses left_bounds and right_bounds performs usually slightly better than the basic variant of VP-tree that uses medians. Sometimes, nevertheless, we observed an opposite effect. This unexpected effect might have been caused among other by randomness aspects in building a VP-tree.

In addition, we would like to note that ordering of vectors with respect to their distances to a basic reference vector can be achieved not necessarily by means of files sorted with respect to distances of source vectors to a first reference vector, but, for instance, by means of B+-trees, which are very easy to create and maintain.

Finally, we would like to note that all our theoretical conclusions related to determining ε-Euclidean neighborhoods and ε-Euclidean nearest neighbors remain valid if we replace the Euclidean distance with any distance metric.

Acknowledgments This work was supported by the National Centre for Research and Development (NCBiR) under Grant No. SP/I/1/77065/10 devoted to the Strategic scientific research and experimental development program: "Interdisciplinary System for Interactive Scientific and Scientific-Technical Information".

References

1. Elkan, C.: Using the triangle inequality to accelerate k-means. In: ICML'03, pp. 147–153. Washington (2003)
2. Jańczak, B.: Density-based clustering and nearest neighborood search by means of the triangle inequality. M.Sc. Thesis, Warsaw University of Technology (2013)
3. Kryszkiewicz, M.: The triangle inequality versus projection onto a dimension in determining cosine similarity neighborhoods of non-negative vectors. In: RSCTC 2012, LNCS (LNAI) 7413, pp. 229–236. Springer, Berlin (2012)
4. Kryszkiewicz, M.: Determining cosine similarity neighborhoods by means of the euclidean distance. In: Rough Sets and Intelligent Systems, Intelligent Systems Reference Library 43, pp. 323–345. Springer, Berlin (2013)
5. Kryszkiewicz M., Lasek P.: TI-DBSCAN: clustering with DBSCAN by means of the triangle inequality. In: RSCTC 2010, LNCS (LNAI) 6086, pp. 60–69. Springer (2010)
6. Kryszkiewicz M., Lasek P.: A neighborhood-based clustering by means of the triangle inequality. In: IDEAL 2010, LNCS 6283, pp. 284–291. Springer (2010)
7. Moore, A.W.: The anchors hierarchy: using the triangle inequality to survive high dimensional data. In: Proceeding of UAI, pp. 397–405. Stanford (2000)
8. Patra, B.K., Hubballi, N., Biswas, S., Nandi, S.: Distance based fast hierarchical clustering method for large datasets. In: RSCTC 2010, pp. 50–59. Springer, Heidelberg (2010)

9. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Commun. ACM **18**(11), 613–620 (1975)
10. Samet, H.: Foundations of Multidimensional and Metric Data Structures. Morgan Kaufmann, San Francisco (2006)
11. Stonebraker, M., Frew, J., Gardels, K., Meredith, J.: The SEQUOIA 2000 storage benchmark. In: Proceeding of ACM SIGMOD, pp. 2–11. Washington (1993)
12. Uhlmann, J.: Satisfying general proximity/similarity queries with metric trees. Inf. Process. Lett. **40**(4), 175–179 (1991)
13. Yanilos, P.N.: Data structures and algorithms of nearest neighbor search in general metric spaces. In: Proceedings of 4th ACM-SIAM Symposium on Descrete Algorithms, pp. 311–321. Philadelphia (1993)
14. Zezula, P., Amato, G., Dohnal, V., Bratko, M.: Similarity Search: The Metric Space Approach. Springer, Heidelberg (2006)
15. Zhang, T., Ramakrishnan, R., Livny, M.: BIRCH: a new data clustering algorithm and its applications. Data Min. Knowl. Disc. **1**(2), 141–182 (1997)