The VLDB Journal manuscript No.
(will be inserted by the editor)

Pivot Selection Algorithms in Metric Spaces: A Survey and Experimental Study

Yifan Zhu · Lu Chen · Yunjun Gao · Christian S. Jensen

Received: date / Accepted: date

Abstract Similarity search in metric spaces is used widely in areas such as multimedia retrieval, data mining, data integration, to name but a few. To accelerate metric similarity search, pivot-based indexing is often employed. Pivot-based indexing first computes the distances between data objects and pivots and then exploits filtering techniques that use the triangle inequality on pre-computed distances to prune search space during search. The performance of pivot-based indexing depends on the quality of the pivots used, and many algorithms have been proposed for selecting high-quality pivots. We present a comprehensive empirical study of pivot selection algorithms. Specifically, we classify all existing algorithms into three categories according to the types of distances they use for selecting pivots. We also propose a new pivot selection algorithm that exploits the power law probabilistic distribution. Next, we report on a comprehensive empirical study of the search performance enabled by different pivot selection approaches, using different datasets and indexes, thus contributing new insight into the strengths and weaknesses of existing selection techniques. Finally, we offer advice on how to select appropriate pivot selection algorithms for different settings.

Yifan Zhu · Lu Chen (Corresponding Author) · Yunjun Gao
College of Computer Science
Zhejiang University, Hangzhou, China
E-mail: {xtf_z, luchen, gaoyj}@zju.edu.cn

Christian S. Jensen
Department of Computer Science
Aalborg University, Aalborg, Denmark
E-mail: csj@cs.aau.dk

Keywords Similarity Search, Metric Space, Metric Index, Pivot

1 Introduction

Similarity search aims to find objects similar to a query object under a given metric. When using different metrics (such as Euclidean distance or edit distance), similarity search is capable of accommodating a wide variety of data types (e.g., locations, strings and images), thus having applications in a wide range of areas [12,13,23,31,32]. For instance, similarity search can be employed to multimedia searching to find images of similar shapes, colors, and layouts. As the notion of metric space is general and can accommodate a wide variety of data types and distance metrics, we focus on solutions to similarity search in metric spaces that make no limiting assumptions and thus are able to support a wide range of applications.

In order to accelerate similarity search in metric spaces, a number of indexing techniques have been developed. Existing metric indexes can be classified into two categories, i.e., compact partitioning techniques and pivot-based techniques [8]. The former divide space into compact regions, and enable pruning of such regions during search. The latter represent objects as vectors of distances to a set of pivots. These distances can be employed to avoid unnecessary distance computations by pivot filtering and validation. Given a query q, a pivot p, and an object o, we have $|d(q,p) - d(p,o)| \leq d(q,o) \leq d(q,p) + d(p,o)$ according to the triangle inequality. By storing the distances $d(p,o)$ from p to all data objects o, we can derive upper and lower bounds on $d(q,o)$ for pruning and validation. Therefore, pivot-based techniques are able to reduce

the number of distance computations, and to improve performance accordingly [21].

The performance of pivot-based methods depends on the quality of the pivots used. Thus, many proposals [1,4,5,9,16,17,20,22,24,27,29,34,36] exist on how to design algorithms for selecting high-quality pivots for use in metric indexes. Pivot-based indexing has attracted recent attention [6,10,11,28,30,33], and in two previous studies [10,11], we survey all metric indexes, including pivot-based metric indexes. While these studies focus on the indexing structures, little attention has been given to pivot selection. In this paper, we investigate methods for high-quality pivot selection that can be used to improve the performance of pivot-based indexing. To enhance the quality of selected pivots, additional features, such as the distribution of queries [21,34] and the similarity metrics employed [2,14,18,19,35], can be utilized. For instance, when Euclidean distance is used as the metric in L-dimensional vector space, machine learning can be exploited to improve the quality of the selected pivots [19]. Nevertheless, such specific features are not available in a general setting where the specifics of the datasets and queries are not known in advance. We consider general pivot selection algorithms that can find a specified number of pivots without relying on any specialized features of the data and setting.

Although some studies [1,7,15,25,33] compare a few pivot selection techniques, no comprehensive comparisons have been reported. In particular, the HKvp index [33] and permutation-based indexes [1] are used to compare different pivot selection algorithms, as the latter indexes belong to the class of pivot-based indexes that support approximate similarity search. In contrast, we employ three typical pivot-based indexes (e.g., LAESA [26] MVPT [3], and SPB-tree [9]) for comparisons. Another study compares four pivot selection methods [7], and a study also has been reported that proposes an incremental sampling pivot selection framework to combine existing pivot selection algorithms [25]. A further study [15] conducts experiments in vector space and thus does not take into account the variety of data types (e.g., strings) in metric spaces. Motivated by this state of affairs, we provide a comprehensive evaluation of all the pivot selection algorithms in the same experimental settings (i.e., the same numbers of pivots, the same size of candidate pivots, and the same sample size w.r.t. the dataset), considering three types of pivot-based indexes, to enable balanced comparisons.

We survey all existing general pivot selection algorithms known to us, and classify them into three categories: (i) P-P distribution based algorithms [1,16, 17,27,29] that select pivots according to the distance distribution among pivots; (ii) P-O distribution based algorithms [22,24,34] that select pivots according to the distance distribution between pivots and data objects; and (iii) O-O distribution based algorithms [4, 5,9] that select pivots according to the distance distribution between data objects, and aim to maximize the similarity between original metric distances and lower bound distances derived by using pivots. We provide detailed descriptions of the algorithms, and analyze their time complexities. To better illustrate the variations among the algorithms, a case study is included in the experiments.

To sum up, we make the following contributions.

- We present a compact survey of pivot selection algorithms for metric indexing, classifying the algorithms into three categories according to the distance distributions (i.e., P-P, P-O, and O-O distributions) they use. We also provide corresponding time complexity analyses.
- We present a new pivot selection algorithm that uses a power law probabilistic distribution to further improve the quality of selected pivots.
- We report on a comprehensive experimental comparison of all the pivot selection algorithms while considering the effects of different kinds of pivot-based metric index structures and considering both real and synthetic datasets with different data types and distance metrics.
- We conduct a case study to offer new insights into the strengths and the weaknesses of existing techniques, and we provide a combined effectiveness-efficiency analysis to offer guidance on how to choose an appropriate pivot selection algorithm for a specific setting.

The rest of the paper is organized as follows. Section 2 provides definitions and terminology related to pivot selection. Sections 3, 4, and 5 cover the three categories of pivot selection algorithms. Section 6 reports on the comprehensive experimental study. Finally, Section 7 concludes the paper and offers directions for future research.

2 Preliminaries

We first provide definitions of metric similarity search. Then, we introduce the pivot filtering and validation techniques used to accelerate similarity search. Finally, we present an overview of all the pivot selection algorithms. Table 1 summarizes frequently used notations.

Table 1 Frequently Used Notation

Symbol	Description
O	the object set
o, q	an object or a query object
S	the sample object set
A	the set of object pairs
P	the set of pivots
P_{cand}	the set of candidate pivots
n	the number of objects
$\widehat{n}$	the number of sample objects
m	the number of pivots
$\widehat{m}$	the number of candidate pivots
$dcmp$	the metric distance computation cost
$compdists$	the number of distance computations
R	the number of pivot replacements
α	the distance threshold
$sp(\cdot)$	the spacing between objects
$\sigma^2_{sp}(\cdot)$	the variance of spacing
$sc(\cdot,\cdot)$	the linear correlation coefficient
ϵ_{sp}	the spacing variance threshold
ϵ_{sc}	the linear correlation coefficient threshold
$d(\cdot)$	the distance function
$d_P(\cdot)$	the lower bound distance function
$dis(o,\ P)$	the distance from o to a pivot set P

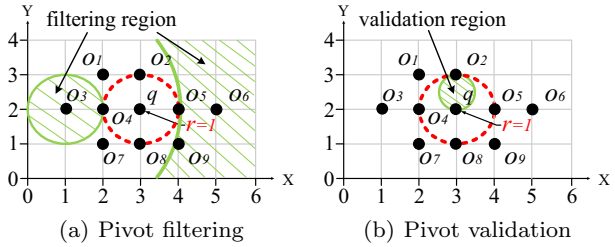
2.1 Metric similarity search

A metric space is a pair (M, d), in which M is an object domain and d is a distance function for measuring the "similarity" between objects in M. In particular, the distance function d has four properties: (1) *symmetry*: $d(q,o) = d(o,q)$; (2) *non-negativity*: $d(q,o) \geq 0$; (3) *identity*: $d(q,o) = 0$ iff $q = o$; and (4) *triangle inequality*: $d(q,o) \leq d(q,p) + d(p,o)$. Based on these properties, we define metric similarity search, including the metric range query and the metric k nearest neighbor query.

Definition 1 (Metric Range Query) Given an object set O, a query object q, and a search radius r in a metric space, a metric range query (MRQ) finds the objects in O that are within distance r of q, i.e., $MRQ(q,r) = \{o|\ o \in O \wedge d(q,o) \leq r\}$.

Definition 2 (Metric k Nearest Neighbor Query) Given an object set O, a query object q, and an integer k in a metric space, a metric k nearest neighbor query (MkNNQ) finds k objects in O that are most similar to q, i.e., $MkNNQ(q,k) = \{S|\ S \subseteq O \wedge |S| = k \wedge \forall s \in S\ (\forall o \in (O - S)(d(q,s) \leq d(q,o)))\}$.

Consider an English word set O = {"rank", "france", "far", "friend", "brand"}, where edit distance is employed to measure similarity between the words. Taking "frank" as a query word q, an example of metric range query with search radius 2 finds the words that can be edited to become the query word in 2 steps, i.e., MRQ("frank", 2) = {"rank", "france", "brand"}. An example of a metric k nearest neighbor query finds (k =)1 word from O with the smallest edit distance to the query word "frank", i.e., $MkNNQ(q,k)$("frank", 1) = {"rank"}. If we know the distance from q to its k-th nearest neighbour in advance, an MkNNQ can be answered by means of an MRQ.



(a) Pivot filtering (b) Pivot validation

Fig. 1 Pivot Filtering and Validation

2.2 Pivot filtering and validation

When having to answer the above queries, a brute-force solution is computing the distances from all the objects to the query object and then determining the final results according to the definitions. However, the distance computations (e.g., edit distance) are usually costly. To improve efficiency, we can use well-chosen pivots and the triangle inequality to prune the search space. Next, we introduce filtering and validation based on pivots.

Lemma 1 (Pivot Filtering) *Given a pivot set P, a query object q, and a search radius r in a metric space, an object o can be pruned if* $\exists p_i \in P\ (|d(o,p_i) - d(q,p_i)| > r$.

Proof Given any pivot p_i in P, a query object q, and an object o, we have $|d(o,p_i) - d(q,p_i)| \leq d(o,q)$ due to the triangle inequality. If $|d(o,p_i) - d(q,p_i)| > r$, then $d(o,q) > r$. Thus, o can be pruned safely.

Lemma 2 (Pivot Validation) *Given a pivot set P, a query object q, and a search radius r in a metric space, an object o is validated to be in the answer to $MRQ(q,r)$ if* $\exists p_i \in P\ (d(o,p_i) + d(q,p_i) \leq r)$.

Proof Given any pivot p_i in P, a query object q, and an object o, we have $d(o,q) \leq d(o,p_i) + d(q,p_i)$ due to the triangle inequality. Hence, $d(o,q) \leq d(o,p_i) + d(q,p_i) \leq r$, meaning that o is contained in the result.

Table 2 Pivot Selection Algorithms

Category	Method	Para.	Time Complexity
P-P distribution	SSS [29]	α	$O(nm \cdot dcmp)$
	FFT [16]	none	$O(nm^2 \cdot dcmp)$
	BPP [1]	none	$O(\widehat{n}\widehat{m}^2 + \widehat{n}\widehat{m} \cdot dcmp)$
	BPS [27]	none	$O(nm^2 \cdot dcmp)$
	HF [17]	none	$O(nm^2 \cdot dcmp)$
P-O distribution	MV [34]	α	$O(n \log n + n\widehat{n} \cdot dcmp)$
	SC [22]	ϵ_{sp}, ϵ_{sc}	$O(nmR \cdot dcmp)$
	PCA [24]	none	$O(n\widehat{m}^2 \cdot dcmp)$
O-O distribution	IS [4]	none	$O(m\widehat{m}\|A\| \cdot dcmp)$
	DSSS [5]	α	$O(n\widehat{m} + m\|A\|R \cdot dcmp)$
	HFI [9]	none	$O(m\widehat{m}\|A\| + n\widehat{m} \cdot dcmp)$
	WRR	λ	$O(\|A\|(mR + \widehat{m} \cdot dcmp))$

Filtering regions can be calculated according to the condition in Lemma 1, while validation regions can be computed using the condition in Lemma 2. As mentioned in Section 2.1, a MkNNQ can be regarded as an MRQ, where the radius equals the k-th nearest neighbor distance. Thus, Lemmas 1 and 2 apply to both MkNNQ and MRQ. Examples of the pivot filtering and validation when answering $MRQ(q,\ 1)$ are shown in Figs. 1(a) and 1(b). Fig. 1(a) uses o_3 as a pivot, and the objects (i.e., o_3, o_6, and o_9) that are located in the filtering regions (shaded green) can be pruned according to Lemma 1. The two separate regions that are shaded green represent the cases $d(o, o_3) - d(q, o_3) > r$ and $d(q, o_3) - d(o, o_3) > r$. More specifically, o_3 satisfies $d(q, o_3) - d(o_3, o_3) > r$ while o_6 satisfies $d(o_6, o_3) - d(q, o_3) > r$; thus, they can be discarded. Fig. 1(b) utilizes o_2 as a pivot, and the object o_2 located in the validation region (i.e., the region shaded green) is validated to be in the result via Lemma 2 due to $d(o_2, o_2) + d(q, o_2) \leq r$.

To enable pivot filtering and validation techniques, many pivot-based metric indexes store pre-computed distances $d(o, p_i)$ from every object o to each pivot p_i. These indexes use pre-computed distances to derive the lower bound $|d(o, p_i) - d(q, p_i)|$ and upper bound $d(o, p_i) + d(q, p_i)$ of the original distances (stated in Lemmas 1 and 2) in order to prune and validate objects during search. As summarized elsewhere [11], pivot-based indexes can be classified into three categories, namely pivot-based tables, pivot-based trees, and pivot-based external indexes, according to the structures they use for storing the pre-computed distances. The performance of an index is dependent on its pruning and validation ability. Therefore, how to select high-quality pivots is important for pivot-based metric indexes and many pivot selection algorithms have been proposed in the literature.

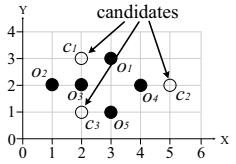
2.3 Pivot selection

Table 2 summarizes all pivot selection algorithms. We consider only pivot selection algorithms designed for metric spaces, and that do not rely on specific assumptions of the setting (query type, search radius, etc.). These general pivot selection algorithms can be classified into three categories, namely P-P distribution based techniques, P-O distribution based techniques, and O-O distribution based techniques, according to the distance distributions they use.

The first category selects pivots by controlling the distance distribution of pivots, i.e., pivots cannot be too close to each other. Spatial Selection of Sparse Pivots (SSS) [29] and Farthest First Traversal (FFT) [16] use cluster-based functions to choose well-distributed pivots. Balancing Pivot-Position Occurrences (BPP) [1] tends to choose centers as pivots. In contrast, algorithms such as Base-Prototypes Selection (BPS) [27] and Hull of Foci (HF) [17] choose outliers as pivots.

The second category selects pivots by utilizing the distribution of distances between pivots and objects. Maximum Variance (MV) [34] selects pivots to maximize the distance variance between pivots and objects, while Spacing-Correlation Based Selection (SC) [22] considers the correlation of pivots and the distance distribution between pivots and objects. PCA for Pivot Selection (PCA) [24] selects pivots using dimensionality reduction techniques.

Methods in the last category select pivots by using the distribution of distances between objects, i.e., aim at maximizing the similarity between the original metric distances and the lower bound distances derived in Lemma 1. Incremental Selection (IS) [4] chooses pivots that maximize the lower bound distances for sampled object pairs. Dynamic Pivot Selection (DSSS) [5] is a dynamic version of SSS. Instead of selecting pivots directly from the dataset, DSSS first uses SSS to obtain candidates and then selects pivots among the candidates using the same strategy as IS. Similarly, HF-Based Incremental Selection (HFI) [9] uses HF to obtain candidates, and then, it selects pivots that can maximize the ratio of the lower bound distance to the original distance. In addition, we propose a new algorithm, Weighted Distribution Ratio Selection (WDR), that selects pivots by using the power law probabilistic distribution to control the contribution of each object pair.



(a) Pivot candidates

	o_1	o_2	o_3	o_4	o_5	σ^2
c_1	1	1.4	1	2.2	2.2	0.29
c_2	2.2	4	3	1	2.2	0.99
c_3	2.2	1.4	1	2.2	1	0.29

(b) Distance table

	o_1	o_2	o_3	o_4	o_5
c_1	1	1	1	2	2
c_2	2	3	3	1	2
c_3	2	1	1	2	1

removing c_3

rank	1	2
$p_1 = c_1$	4	1
$p_2 = c_2$	2	3

removing c_2

rank	1	2
$p_1 = c_1$	4	1
$p_2 = c_3$	4	1

(c) Rank table and permutation

	$\Delta 1$	$\Delta 2$	$\Delta 3$	$\Delta 4$	σ_{sp}^2
c_1	0	0.4	0.8	0	0.33
c_2	1.2	0	0.8	1	0.46
c_3	0	0.4	0.8	0	0.33

(d) Difference distances

Fig. 2 Illustration of BPP, MV, and SC

Algorithm 1: SSS

Input: an object set O, a threshold α, and the maximum distance MD
Output: a set of pivots P

1 $P = \{p\}$; //p is randomly selected
2 **foreach** $o \in O$ **do**
3 **if** $dis(o, P) \geq \alpha \cdot MD$ **then**
4 $P = P \cup \{o\}$;
5 **end**
6 **end**
7 **return** P

Algorithm 2: FFT

Input: an object set O and the number m of pivots
Output: a set of pivots

1 $P = \{p\}$; //p is randomly selected
2 **while** $|P| < m$ **do**
3 $p = \arg\max_{o \in O} dis(o, P)$;
4 $P = P \cup \{p\}$;
5 **end**
6 **return** P

3 P-P distribution based methods

We introduce P-P distribution based pivot selection methods with corresponding complexity analyses.

3.1 Spatial Selection of Sparse Pivots

Spatial Selection of Sparse Pivots (SSS) [29] selects sparse objects as pivots. To control the distribution of pivots, the distance between pivots should exceed $\alpha \cdot MD$, where α is a threshold and MD denotes the maximum distance between any pair of data objects in a dataset. It can be computed easily based on the space in which the data is located and the distance metric used. For example, the string dataset uses edit distance as the metric, making its MD equal to the length of the longest string.

SSS defines the distance between an object o and the pivot set P as follows.

$$dis(o, P) = \min_{p \in P} d(o, p) \tag{1}$$

Stated briefly, a new pivot p selected by SSS should satisfy $dis(p, P) \geq \alpha \cdot MD$.

Algorithm 1 depicts the pseudo-code of SSS. It takes as input an object set O, a threshold α, and the maximum distance MD. First, the pivot set P is initialized using a randomly selected object (line 1). Then, for each object $o \in O$ satisfying $dis(o, P) \geq \alpha \cdot MD$, o is chosen as the next pivot (lines 2–6). Finally, SSS returns the selected pivot set P. Since each object is checked by SSS only once, the result is produced in $O(nm \cdot dcmp)$ time, where n is the cardinality of the object set, m is the number of pivots, and $dcmp$ is the cost of computing the metric distance between two objects. Although SSS is efficient (i.e., has linear cost w.r.t. the object set cardinality), it cannot flexibly choose a fixed number of pivots. The number of selected pivots depends on threshold α, the data distribution, the first randomly chosen pivot, and the following chosen order of items in object set O. The larger the α is, the fewer pivots are selected.

3.2 Farthest First Traversal

Like SSS, Farthest First Traversal (FFT) [16] controls the distance distribution between pivots. However, unlike SSS that cannot control the number of selected pivots, FFT can find a fixed number of pivots. Algorithm 2 depicts the pseudo-code of FFT, which takes as input an object set O and the number m of pivots. First, a pivot is selected at random. Then, a while-loop finds the remaining pivots (lines 2–5). In each iteration, FFT chooses the object o with the maximal $dis(o, P)$ value as a pivot.

FFT needs to find the farthest object w.r.t. the previously selected pivots in each iteration; thus, the time complexity of an iteration is $O(nm \cdot dcmp)$. As we have m iterations, the complexity of FFT is $O(nm^2 \cdot dcmp)$, which exceeds that of SSS. Since $dis(o, P)$ ($o \in O, o \notin P$) is no larger than the distance between the

Algorithm 3: BPP

Input: an object set O, the number $\widehat{m}$ of candidate pivots, and the number m of pivots
Output: a set of pivots

```
1 generate P of m̂ randomly selected objects;
2 while |P| > m do
3 |   p = arg min_{p∈P} std(P − {p});
4 |   P = P − {p};
5 end
6 return P
```

previous two selected pivots in P (otherwise, this object would have been chosen as a pivot), FFT is also widely used to choose samples.

3.3 Balancing Pivot-Position Occurrences

Balancing Pivot-Position Occurrences (BPP) [1] is a permutation-based pivot selection algorithm. Specifically, in the permutation-based index, each object has a permutation of pivots (ordered by the distance between the pivot and the object), based on which the similarity between objects is measured by the difference between their permutations.

To better illustrate, an example is shown in Fig. 2, where the object set is $\{o_1, o_2, o_3, o_4, o_5\}$ and the candidate pivots are c_1, c_2, and c_3. The distance table depicted in Fig. 2(b) records the distances between pivots and objects. Next, the rank table (the table to the left in Fig. 2(c)) can be computed by using the rank of each pivot p_i w.r.t. an object o, as is defined below.

$$rank(p_i, o) = 1 + \sum_{p_j \in P, d(p_j,o) < d(p_i,o)} 1 \tag{2}$$

For instance, as $d(c_1, o_2) = d(c_3, o_2) < d(c_2, o_2)$, c_2 ranks third among all candidate pivots w.r.t. o_2, meaning that $rank(c_2, o_2) = 3$. In addition, the permutation of o_1 w.r.t. the ranks of the pivots is (1, 2, 2) (i.e., row 2 in the rank table), while the permutation of o_2 is (1, 3, 1). If pivots are distributed unevenly (i.e., appear in the same position in all the permutations w.r.t. objects in the dataset), the similarity computation quality degrades. In the example, pivots c_1, c_2, and c_3 appear in the same positions for objects o_2 and o_3, so the similarity between o_2 and o_3 is 0 because their permutations are the same (i.e., (1, 3, 1)), which is inaccurate. Hence, the distribution of pivots, in different positions of the permutations, should be made as uniform as possible. We define the rank permutation of the pivot p_i, which utilizes *rank* values over all the objects, as follows:

$$pr(p_i, k) = \sum_{o \in O, rank(p_i,o) = k} 1 \tag{3}$$

The two small tables to the right in Fig. 2(c) represent the $pr(p_i, k)$ ($p_i \in P, 1 \le k \le |P|$) values when removing c_2 or c_3. More specifically, after removing c_3, we have $pr(c_1, 1) = 4$ and $pr(c_1, 2) = 1$ for candidate pivot c_1, and we have $pr(c_2, 1) = 2$ and $pr(c_2, 2) = 3$ for candidate pivot c_2; and after removing c_2, we have $pr(c_1, 1) = 4$ and $pr(c_1, 2) = 1$ for candidate pivot c_1, and we have $pr(c_2, 1) = 4$ and $pr(c_2, 2) = 1$ for candidate pivot c_2. Based on the rank permutations, the distribution of the pivot permutation positions can be evaluated. Since the pivot rank permutations after removing c_3 are distributed more evenly, removing c_3 is the better choice.

In order to measure the evenness of pivot rank permutations w.r.t. the pivot set P, the standard deviation is adopted.

$$std(P) = \sqrt{\frac{1}{|P|^2} \sum_{p_i \in P, 1 \le k \le |P|} (pr(p_i, k) - \mu)^2}, \tag{4}$$

where μ denotes the mean for values of $pr(p_i, k)$, $p_i \in P$, and $1 \le k \le |P|$. The larger the $std(P)$ is, the more differently the rank positions permutate (i.e., the more uniform the distribution of the pivot set P is).

Algorithm 3 presents the pseudo-code of BPP. It takes as input an object set O, the number $\widehat{m}$ of candidate pivots, and the number m of pivots. It first randomly generates a candidate pivot set P from O with cardinality $\widehat{m}$. Then, a while loop is performed to reduce the set to m pivots (lines 2–5). In each iteration, BPP removes a candidate from P that has the minimum standard deviation for pr.

The cost of computing the distances between candidate pivots and objects is $O(n\widehat{m} \cdot dcmp)$ in order to compute the rank defined in Equation 2. In each iteration, the complexity of re-computing pr and std are $O(n\widehat{m})$ and $O(\widehat{m}^2)$, respectively. Thus, the total complexity of BPP is $O(n\widehat{m} \cdot dcmp + n\widehat{m}^2)$. Due to the high time complexity, a sample of O with cardinality $\widehat{n}$ is used in BPP. Hence, the time complexity is $O(\widehat{n}\widehat{m} \cdot dcmp + \widehat{n}\widehat{m}^2)$.

3.4 Base-Prototypes Selection

Similar to FFT, Base-Prototypes Selection (BPS) [27] selects pivots far from each other. However, instead of using the maximum distance between the pivot and all

Algorithm 4: HF

Input: an object set O and the number m of pivots
Output: a set of pivots

1 p_{tmp} = a randomly selected object from O;
2 $p_1 = \arg\max_{o \in O}$ *diff_dis*(o, p_{tmp});
3 $p_2 = \arg\max_{o \in O}$ *diff_dis*(o, p_1);
4 $P = \{p_1, p_2\}$;
5 **while** $|P| < m$ **do**
6 $\quad p = \arg\min_{o \in O-P}$ *diff_dis*(o, P);
7 $\quad P = P \cup \{p\}$;
8 **end**
9 **return** P

the previous pivots, BPS uses the accumulated distance defined as follows.

$$accum\_dis(o, P) = \sum_{p \in P} d(o, p) \tag{5}$$

In each iteration, BPS selects an object with the maximum accumulated distance to the previous pivots as a new pivot. Since the pseudo-code of BPS is very similar to that of FFT, it is omitted. The time complexity of BPS is also the same as that of FFT, i.e., $O(nm^2 \cdot dcmp)$.

There also exist other similar algorithms [7] that choose a new pivot with the maximum (accumulated) distance to the previous pivots. However, these algorithms need prior knowledge (i.e., the search radius) to obtain pivots, which renders them inapplicable for use in a general setting with no specialized assumptions and no prior knowledge.

3.5 Hull of Foci

Hull of Foci (HF) [17] aims to choose the objects that are near the hull of the dataset. In order to choose a new pivot, HF minimizes the following distance:

$$diff\_dis(o, P) = \sum_{p \in P} |d(p_1, p_2) - d(o, p)|, \tag{6}$$

where p_1 and p_2 are the two most distant objects in the pivot set (i.e., the first two pivots).

Algorithm 4 depicts the pseudo-code of HF. It takes as input an object set O and the number m of pivots. First, HF randomly selects an object p_{tmp}, and then, it finds the farthest object p_1 w.r.t. p_{tmp} and the farthest object p_2 w.r.t. p_1 (lines 1–3). These are the first two pivots (line 4). Next, a while loop finds the remaining pivots (lines 5–8). In each iteration, the object that has the minimum *diff_dis*(o, P) is chosen as the new pivot.

Algorithm 5: MV

Input: an object set O, the number m of pivots, a threshold α, the maximum distance *MD*
Output: a set of pivots

1 $P = \varnothing$, $Cand = O$;
2 Compute μ_o and σ_o for each $o \in Cand$;
3 sort $Cand$ in descending order of σ_o;
4 $\omega = MD \cdot \alpha$;
5 **while** $|P| < m$ *and* $|Cand| > 0$ **do**
6 $\quad p = Cand_1$;
7 $\quad P = P \cup \{p\}$;
8 $\quad$ **foreach** $o \in Cand$ **do**
9 $\quad\quad$ **if** $|d(o, p) - \mu_p| > \omega$ **then**
10 $\quad\quad\quad Cand = Cand - \{o\}$;
11 $\quad\quad$ **end**
12 $\quad$ **end**
13 **end**
14 **return** P

To better understand algorithm HF, if we use the first two pivots of HF (i.e., the two most distant objects) as two foci to draw an ellipse, the ellipse is like the hull of the dataset. As HF selects objects that have similar distances to the previously chosen pivots as subsequent pivots (i.e., HF selects the objects close to the ellipse as subsequent pivots), the newly chosen pivots are also near the hull of the dataset. Hence, HF has a strong capability of choosing outliers. The cost of finding the first two pivots is $O(n \cdot dcmp)$ while finding the remaining pivots costs $O(nm^2 \cdot dcmp)$. Thus, the time complexity of HF is $O(nm^2 \cdot dcmp)$.

4 P-O distribution based methods

We proceed to cover the algorithms that consider the distances between pivots and data objects.

4.1 Maximum Variance

Maximum Variance (MV) [34] aims to maximize the variance of the distances between pivots and objects. Inspired by two properties of the triangle inequality, that (i) pivots close to the query objects can prune far-away objects and validate nearby objects and (ii) pivots far from query objects can prune objects near the queries, MV finds that (i) pivots that are either too close to, or too far from, each other have similar pruning capabilities and (ii) pivots with high variance offer better pruning capabilities. Based on these two findings, MV uses a distance threshold α, the max distance *MD*, and the mean of distances μ to discard the

candidates that are close to, or far away from, existing pivots. Specifically, an object o satisfying $d(o,p) < \mu - \alpha \cdot MD$ is close to existing pivots while an object o satisfying $d(o,p) > \mu + \alpha \cdot MD$ is far away from existing pivots (the two conditions can be simplified as $|d(o,p) - \mu| > \alpha \cdot MD$). Further, MV selects objects with high variance (i.e., strong filtering capability) as new pivots. In Fig. 2(b), the candidate c_2 has the largest variance compared with those of the other candidates, indicating that the distances between c_2 and data objects vary substantially, enabling c_2 to better distinguish objects. Thus, MV selects c_2 as the first pivot, and sets μ to 2.56, which is the average distance of the five objects to c_2. With $\alpha \cdot MD$ being 1.34, o_2 (that is far away from c_2) and o_4 (that is close to c_2) are disregarded as pivot candidates because $d(o_2, c_2) = 4 > \mu + \alpha \cdot MD$ and $d(o_4, c_2) = 1 < \mu - \alpha \cdot MD$.

Algorithm 5 shows the pseudo-code of MV. It takes as input the set O of objects, the maximum distance MD, a threshold α, and the number m of pivots. First, MV uses the entire dataset as a candidate pivot set. For each candidate o, it obtains the distance distribution between o and all other data objects, and computes the corresponding μ_o and σ_o (line 2). Then, it sorts all the candidates in descending order of σ_o and computes the threshold ω (lines 3–4). Next, a while loop is performed to select m pivots (lines 5–13). In each iteration, MV uses the first candidate in $Cand$ as a pivot (line 6–7) and then filters the candidates that are too close or too far from the newly chosen pivots using threshold ω (lines 8–12).

To speed up the computation, a sample set instead of the entire object set can be used to compute μ and σ. Thus, the cost of computing each object's variance and sorting them is reduced to $O(n\widehat{n} \cdot dcmp + nlogn)$. Choosing each pivot and filtering unqualified candidates have time complexity $O(n \cdot dcmp)$. Since $\widehat{n} \gg m$, the time complexity of MV is $O(n\widehat{n} \cdot dcmp + n \log n)$.

4.2 Spacing-Correlation based Selection

Spacing-Correlation based Selection (SC) [22] utilizes the variance of spacing to evaluate the quality of a single pivot, and uses a linear correlation coefficient to measure the correlation between pivots. It first sort objects in ascending order of their distances to the pivot p and lets o_i denote the object having the i^{th} smallest distance. Then, SC defines the i^{th} spacing between objects w.r.t. the pivot p as:

$$sp(p,i) = dis(o_{i+1}, p) - dis(o_i, p), \tag{7}$$

Algorithm 6: SC

Input: an object set O, thresholds ϵ_{sp} and ϵ_{sc}, and the number m of pivots
Output: a set of pivots

```
1  generate P of m randomly selected objects;
2  S = ∅;
3  foreach o ∈ O in a random order do
4      S = S ∪ {o};
5      foreach p ∈ P do
6          use o to update σ²_sp(p);
7          if σ²_sp(p) > ε_sp then
8              replace p with a random object
               p_tmp ∈ O;
9          end
10     end
11     foreach (p1, p2) ∈ P and sc(p1, p2) > ε_sc do
12         if σ²_sp(p1) > σ²_sp(p2) then
13             replace p1 with a random object
               p_tmp ∈ O;
14         end
15         else
16             replace p2 with a random object
               p_tmp ∈ O;
17         end
18     end
19 end
20 return P
```

Next, it defines the variance of spacing w.r.t. the pivot p as:

$$\sigma_{sp}^2(p) = \frac{1}{n-1} \sum_{i=1}^{n-1} (sp(p,i) - \mu)^2, \tag{8}$$

where μ is the average value of spacing $sp(p,i)$, $1 \le i < n$. Using d_{1i} and d_{2i} to denote $d(p_1, o_i)$ and $d(p_2, o_i)$ respectively, the linear correlation coefficient between p_1 and p_2 is defined as follows.

$$sc(p_1,p_2) = \frac{n \sum_{i=1}^{n} d_{1i} d_{2i} - \sum_{i=1}^{n} d_{1i} \sum_i d_{2i}}{\sqrt{n \sum_{i=1}^{n} {d_{1i}}^2 - (\sum_{i=1}^{n} d_{1i})^2} \sqrt{n \sum_{i=1}^{n} {d_{2i}}^2 - (\sum_{i=1}^{n} d_{2i})^2}} \tag{9}$$

Figs. 2(b) and 2(d) illustrate the linear correlation coefficient and the variance of distance differences, respectively. The former shows that pivots c_1 and c_3 are highly correlated since they have the same distances to o_2, o_3, and o_4, meaning that the effect of using either of the two pivots to separate objects is the same. The latter shows the distance differences (i.e., $d(o_j, c_i) - d(o_{j-1}, c_i)$) between the sorted objects and

the corresponding σ_{sp}^2 of each candidate c_i. Taking c_1 as an example, the sorted distances between candidate c_1 and all objects are (1, 1, 1.4, 2.2, 2.2). Thus, the differences between the sorted distances for c_1 are (0, 0.4, 0.8, 0), and the corresponding σ_{sp}^2 is 0.29. However, to reduce the computation cost of σ_{sp}^2, a random order can be used instead of sorting all the objects. Pivots having smaller σ_{sp}^2 are better, as the distance difference is distributed uniformly when σ_{sp}^2 is small [22]. In this case, c_1 and c_3 have smaller σ_{sp}^2, and hence they are better than c_2.

SC selects pivots having high quality and low correlation with each other. Algorithm 6 depicts the pseudo-code. It takes as input an object set O, two thresholds ϵ_{sp} and ϵ_{sc}, and the number m of pivots. SC first generates P using m random objects and then initializes an empty set S (lines 1–2). Next, a while loop is performed until O is empty. In each iteration, the algorithm randomly picks an object o from O and then updates S (lines 3–4). For each pivot $p \in P$, it recomputes σ_{sp}^2 if S is updated, and it replaces p with a random object if $\sigma_{sp}^2(p) \geq \epsilon_{sp}$ (lines 5–10). Finally, for each pivot pair with sc no less than ϵ_{sc}, SC replaces the pivot having larger σ_{sp}^2 with a new random object (lines 11–18).

SC iteratively picks a random object from the object set and performs two steps: (1) it updates the σ_{sp}^2 of each pivot and replaces the pivots having large σ_{sp}^2 with random objects; and (2) it finds pivot pairs that have large coefficients and replaces one pivot (having the larger σ_{sp}^2) of each pair with a random object. Let R_{sp} and R_{sc} denote the total replacements in step 1 and step 2, respectively. Then, step 1 takes $O(mnR_{sp} \cdot dcmp)$ time, while step 2 takes $O(nm^2 + mnR_{sc} \cdot dcmp)$ time. Let R be $R_{sc} + R_{sc}$, i.e., the number of replacements caused by violations of the thresholds ϵ_{sp} and ϵ_{sc} (lines 7 and 11 in Algorithm 6). The time complexity of SC is then $O(nmR \cdot dcmp)$. The slower the algorithm runs (i.e., the stricter ϵ_{sp} and ϵ_{sc} are), the smaller spacing variance and lower linear correlation coefficient the chosen pivots will have. In the worst case, when ϵ_{sp} and ϵ_{sc} are too strict, each object in the dataset will cause the replacement of pivots; thus, the time complexity of SC becomes $O(n^2 m \cdot dcmp)$.

4.3 PCA for Pivot Selection

PCA for Pivot Selection (PCA) [24] performs dimensionality reduction to select high-quality pivots. First, FFT is used to select $\widehat{m}$ high-quality candidate pivots. Then, a matrix containing the distances between candidate pivots and all objects is obtained. After that, PCA performs dimension reduction (i.e., eigenvalue decomposition on the corresponding covariance matrix) on the matrix, and finds the top-m eigenvectors with the highest eigenvalues, thus keeping the reduced space most similar to the initial space. Finally, m pivots among the candidates that have the maximum projection values w.r.t. the top-m eigenvectors are selected.

Algorithm 7: PCA

Input: an object set O, a candidate size $\widehat{m}$, and the number m of pivots
Output: a set of pivots

```
1  P_cand = {m̂ candidate pivots chosen by FFT};
2  P = ∅;
3  V = P_cand × O; // compute the distance matrix
4  compute eigenvalues E_value and eigenvectors
   E_vector of V using principal component analysis;
5  sort E_value in descending order and E_vector in
   the same order as E_value;
6  while |P| < m and |E_vector| > 0 do
7  |  pop the top vector v from E_vector;
8  |  p = arg max_{p∈P_cand} projection(p, v);
9  |  P = P ∪ {p};
10 end
11 return P
```

Algorithm 7 takes as input the set O of objects, the number $\widehat{m}$ of candidate pivots, and the number m of pivots. First, it generates P_{cand} using $\widehat{m}$ candidate pivots chosen by FFT and initializes the pivot set P to be empty. Then, it computes the distance matrix V of distances between candidate pivots and objects. Next, it reduces the dimensionality of V to get corresponding eigenvalues E_{value} and eigenvectors E_{vector} and sorts them in descending order (lines 4–5). Then, a while loop is performed to select m pivots (lines 6–10). In each iteration, the algorithm pops the top vector v from E_{vector}, finds the candidate pivot $p \in P_{cand}$ that has the maximum projection value w.r.t. v, and updates P to be $P \cup \{p\}$. Finally, the pivot set is returned.

The algorithm needs $O(n\widehat{m} \cdot dcmp)$ time to perform FFT, and $O(n\widehat{m}^2 \cdot dcmp)$ to compute the corresponding matrix. Since $n \gg \widehat{m}$, the cost of dimension reduction, sorting on E_{value}, and selecting pivots with maximum projection can be disregarded. Consequently, the total time complexity of PCA is $O(n\widehat{m}^2 \cdot dcmp)$.

5 O-O distribution based methods

We proceed to describe pivot selection algorithms that operate on the distribution of distances between objects. As stated in Section 2.2, pivot filtering (i.e., Lemma 1) is employed to avoid unnecessary distance computations during search. According to the lemma,

Algorithm 8: IS

Input: an object set O, a set A of object pairs, and the number m of pivots
Output: a set of pivots

1 generate P_{cand} of $\widehat{m}$ randomly selected objects;
2 $P = \varnothing$;
3 **while** $|P| < m$ **do**
4 $\quad p = \arg \max_{p \in P_{cand}} \sum_{(o_i, o_j) \in A} d_{P \cup \{p\}}(o_i, o_j)$;
5 $\quad P = P \cup \{p\}$;
6 **end**
7 **return** P

given a pivot p, $|d(p,o) - d(p,q)|$ is the lower bound of $d(q,o)$. Given a pivot set P, the following equation also defines the lower bound.

$$d_P(o,q) = \max_{p \in P} |d(p,o) - d(p,q)| \tag{10}$$

To achieve the best pruning, $d_P(q,o)$ should be as close to $d(q,o)$ as possible. In other words, $d_P(q,o)$ should be as large as possible. Since the query objects are not known in advance and the objects in the datasets are assumed to have the same distribution of query objects, we aim to maximize $d_P(o_i,o_j)$ ($o_i, o_j \in O$) instead of maximizing $d_P(q,o)$. However, the cost of maximizing $d_P(o_i,o_j)$ is very high, especially for large datasets. Thus, a sample set A of object pairs is used.

5.1 Incremental Selection

Incremental Selection (IS) [4] selects the pivots to maximize the sum of the lower bound distances d_P of sampled object pairs defined in Equation 10. The pseudo-code of IS is shown in Algorithm 8. It takes as input an object set O, an object pair set A, and the number m of pivots. First, the pivot set P is initialized to be empty. Then, a while-loop is conducted to find m pivots (lines 3–6). In each iteration, IS chooses the object $p \in P_{cand}$ having the maximum sum of $d_{P \cup \{p\}}$ among all the object pairs of A as a new pivot, and updates the pivot set. Finally, the pivot set is returned (line 7).

IS has the capability to select the pivots having low correlation coefficient (i.e., no pivots share similar filtering and validation capabilities) and high singular performance (i.e., each pivot has strong capabilities at filtering and validating objects). This is because the objects that are highly correlated with previously selected pivots are discarded because of making no effort at maximizing the sum of $d_{P \cup \{o\}}$ among object pairs, while the objects with stronger filtering capability have larger lower bound distances, and are chosen as new pivots. The time complexity of IS is $O(m\widehat{m}|A| \cdot dcmp)$, as it needs to recompute $d_{P \cup \{o\}}$ among all object pairs in each iteration.

Algorithm 9: DSSS

Input: an object set O, the maximum distance MD, a threshold α, a set A of object pairs, and the number m of pivots
Output: a set of pivots

1 $P = \varnothing$;
2 **foreach** $o \in O$, $dis(o,P) \geq \alpha \cdot MD$ **do**
3 $\quad$ **if** $|P| < m$ **then**
4 $\quad\quad P = P \cup \{o\}$;
5 $\quad$ **end**
6 $\quad$ **else**
7 $\quad\quad p = \arg\min_{p \in P} ctr(p,P,A)$;
8 $\quad\quad$ **if** $ctr(p,P,A) < ctr(o, P - \{p\} \cup \{o\}, A)$ **then**
9 $\quad\quad\quad P = P - \{p\} \cup \{o\}$;
10 $\quad\quad$ **end**
11 $\quad$ **end**
12 **end**
13 **return** P

5.2 Dynamic Pivot Selection

Unlike IS that chooses pivots one by one, Dynamic Pivot Selection (DSSS) [5] first selects m pivots using SSS, and then repeatedly replaces the pivot that contributes the least to the pivot set with a better one. Given an object pair (o_i,o_j) and a set of pivots P, the contribution of a pivot $p \in P$ to the pivot set P w.r.t. the object pair (o_i,o_j) is defined as follows.

$$ctr(p,P,(o_i,o_j)) = d_P(o_i,o_j) - d_{P-\{p\}}(o_i,o_j), \tag{11}$$

where the last term $d_{P-\{p\}}(o_i,o_j)$ refers to the lower bound distance of object pair (o_i,o_j) using the pivot set $P - \{p\}$ (i.e., the pivot set after removing p). The total contribution of a pivot p w.r.t. a pivot set P is the sum of its contributions for all pairs in the set A of object pairs:

$$ctr(p,P,A) = \sum_{(o_i,o_j) \in A} ctr(p,P,(o_i,o_j)) \tag{12}$$

According to Equation 12, DSSS selects pivots with high contributions in order to maximize $d_P(o,q)$. Algorithm 9 depicts the pseudo-code. It takes as input an object set O, the maximum distance MD, a threshold α, and the number m of pivots. For each object $o \in O$ satisfying $dis(o,P) \geq \alpha \cdot MD$, if too few pivots are chosen, DSSS chooses o as the next pivot (lines 3–5).

Algorithm 10: HFI

Input: an object set O, the number $\widehat{m}$ of candidate pivots, a set A of object pairs, and the number m of pivots
Output: a set of pivots

1 generate P_{cand} of $\widehat{m}$ objects selected by HF;
2 compute the distances between P_{cand} and A;
3 $P = \varnothing$;
4 **while** $|P| < m$ **do**
5 $\quad p = \arg\max_{o \in P_{cand}} prec(P \cup \{o\})$;
6 $\quad P_{cand} = P_{cand} - \{p\}$;
7 $\quad P = P \cup \{p\}$;
8 **end**
9 **return** P

Otherwise (i.e., $|P| \geq m$), DSSS first removes the pivot $p \in P$ with the least contribution from the pivot set (line 7). Next, it determines whether p or o is to be the next pivot according to their contribution (lines 8–10). Finally, the pivot set is returned.

DSSS includes two steps: (i) finding the objects satisfying the condition $dis(o, P) \geq \alpha \cdot MD$ as candidate pivots, which has complexity $O(n\widehat{m} \cdot dcmp)$; and (ii) for each candidate pivot, deciding whether or not to replace the pivot with the least contribution, which has complexity $O(m|A| \cdot dcmp)$. In the second step, the dominant cost is to compute the contribution of a candidate pivot w.r.t. a pivot set, which takes $O(m|A| \cdot dcmp)$ time. The cost of replacements in *if*-conditions (line 9 of Algorithm 9) needs only O(1) time and thus, does not affect on the total time complexity. As $n \gg m|A|$, the time complexity of DSSS is $O(n\widehat{m} \cdot dcmp)$. Note that since DSSS applies SSS to select candidate pivots, the size of the set of candidate pivots is not necessarily fixed at $\widehat{m}$.

5.3 HF-Based Incremental Selection

Unlike IS and DSSS that maximize the sum of lower bound distances of object pairs, HF-Based Incremental Selection (HFI) [9] operates on the deviation of the lower bound distance from the original distance:

$$prec(p) = \frac{1}{|A|} \cdot \sum_{(o_i, o_j) \in A} \frac{d_P(o_i, o_j)}{d(o_i, o_j)} \tag{13}$$

Algorithm 10 shows the pseudo-code of HFI. It takes as input an object set O, the number $\widehat{m}$ of candidate pivots, a set A of object pairs, and the number m of pivots. First, it finds $\widehat{m}$ candidate pivots using HF and computes the distances between candidate pivots and objects in A, in order to reduce subsequent duplicate distance computations. Then, a while loop is performed to select m pivots (lines 4–8). In each iteration, HFI chooses an object o from the pivot candidates that maximizes $prec(P \cup \{o\})$, and updates the pivot set and the pivot candidate set. Finally, the pivot set is returned (line 9).

The cost of choosing pivot candidates and computing the distances is $O(n\widehat{m} \cdot dcmp + \widehat{m}|A| \cdot dcmp)$, which can be simplified to $O(n\widehat{m} \cdot dcmp)$ due to $|A| \ll n$. In addition, the cost of choosing each pivot is similar to that of IS. Hence, the total time complexity of HFI is $O(m\widehat{m}|A| + n\widehat{m} \cdot dcmp)$.

Algorithm 11: WDR

Input: an object set O, a set A of object pairs, a threshold λ, and the number m of pivots
Output: a set of pivots

1 generate P_{cand} of $\widehat{m}$ randomly selected objects;
2 compute the distances between P_{cand} and A;
3 $P = \varnothing$;
4 **repeat**
5 $\quad$ **if** $|P| < m$ **then**
6 $\quad\quad p = \arg\min_{p \in p_{cand}} wr(P \cup \{p\}, \lambda)$;
7 $\quad\quad P_{cand} = P_{cand} - \{p\}$;
8 $\quad\quad P = P \cup \{p\}$;
9 $\quad$ **end**
10 $\quad temp = wr(P, \lambda)$;
11 $\quad$ **foreach** $p_1 \in P$, $p_2 \in P_{cand}$ **do**
12 $\quad\quad$ **if** $temp > wr(P \cup \{p_2\} - \{p_1\}, \lambda)$ **then**
13 $\quad\quad\quad P_{cand} = P_{cand} - \{p_1\}$;
14 $\quad\quad\quad P = P \cup \{p_2\}$;
15 $\quad\quad\quad temp = wr(P, \lambda)$;
16 $\quad\quad$ **end**
17 $\quad$ **end**
18 **until** *no pivot is replaced*
19 **return** P

5.4 Weighted Distribution Ratio Selection

To further improve HFI, we develop a Weighted Distribution Ratio Selection (WDR) algorithm to select high quality pivots by utilizing the power law to control the contribution of each object pair. According to HFI, the higher the distance ratio $\frac{d_P(o_i, o_j)}{d(o_i, o_j)}$ is, the higher the quality of the pivot set P is. Usually, the larger the difference between the lower bound distance $d_P(o_i, o_j)$ and the real distance $d(o_i, o_j)$ (i.e., the lower bound distance estimation is not accurate), the more difficult it is to avoid the unnecessary $d(o_i, o_j)$ computation via pivot filtering and validation, which is consistent with the power law, an unscientific rule of thumb, i.e., 80% of the effect comes from 20% of the causes. Based on this, we use the power law probabilistic distribution to model

the importance of distance ratios, as defined below.

$$f(x, \lambda) = (1 - x)^{\lambda}, \lambda \in \mathbb{R} \tag{14}$$

According to this definition, given a positive value of λ, the larger x is, the smaller $f(x, \lambda)$ is. If we take the distance ratio $\frac{d_P(o_i,o_j)}{d(o_i,o_j)} \in [0, 1]$ as x, $f(x, \lambda)$ can be used to measure the contribution of each object pair (o_i, o_j). The larger the distance ratio is, the smaller the f value is (i.e., the better the performance of P when pruning (o_i, o_j)). Hence, we define the following function based on the power law function $f(x, \lambda)$ to evaluate the quality of the pivot set P by estimating the number of potential unnecessary distance computations on sample object pair set A.

$$wr(P, \lambda) = \sum_{(o_i,o_j)\in A} f(\frac{d_P(o_i, o_j)}{d(o_i, o_j)}, \lambda) \cdot d(o_i, o_j) \tag{15}$$

WDR aims to minimize $wr(P, \lambda)$, $\lambda \geq 0$. Note that function $wr(P, \lambda)$ contains two factors: (i) the power law impact $f(\cdot, \cdot)$ on the ratios between lower bound distances and real distances and (ii) the weighted contribution defined by the real distances $d(o_i, o_j)$. The distance weight is applied because for the same ratio $\frac{d_P(o_i,o_j)}{d(o_i,o_j)}$, the larger $d(o_i, o_j)$ is, the larger the difference $d_P(o_i, o_j) - d(o_i, o_j)$ is. Here, parameter λ can be adjusted dynamically based on the data distribution and the distance metric. λ is set to 2 in our experiments.

Algorithm 11 presents the pseudo-code of WDR. It takes as input an object set O, a set A of object pairs, a threshold λ, and the number m of pivots. First, WDR generates a candidate pivot set P_{cand} using m randomly selected objects, and then it computes the distances between candidate pivots and objects belonging to $|A|$ (lines 1–2). In each iteration, WDR selects a new pivot if too few pivots are selected (lines 5–9) and replaces an old pivot with a better one to minimize $wr(P, \lambda)$ (lines 10–17). WDR repeats until no more pivots are changed (line 18). Finally, the pivot set P is returned (line 19).

Note that, by setting λ to 1, the function $wr(P, \lambda)$ can be computed as $\sum_{(o_i,o_j)\in A}(d(o_i, o_j) - d_P(o_i, o_j))$. As $\sum_{(o_i,o_j)\in A} d(o_i, o_j)$ is a constant value, minimizing $wr(p, 1)$ is equivalent to maximizing Equation 10, i.e., WDR degenerates to IS. The cost of selecting candidates can be omitted, while that of pre-computing distances is $O(\widehat{m}|A| \cdot dcmp)$. In each iteration of selecting pivots, $O(m|A|)$ time is needed for finding a new pivot or replacing an old pivot with a better choice. Thus, the time complexity is $O(|A|(mR + \widehat{m}dcmp))$, where R is the number of pivot replacements.

Table 3 Statistics of the Datasets Used in Our Experiments

Dataset	Card.	Dim.	Dis. Metric	Max Dis.
LA	107,327	2	L_2-norm	14,142
Words	611,756	1∼34	*Edit Dis.*	34
Color	$1M$	282	L_1-norm	143,820
INT	$10M$	20	L_∞-norm	10,000

Table 4 Query Parameters

Parameters	Value	Default
k	5, 10, 20, 50, 100	20
r	2%, 4%, 8%, 16%, 32%	8%
Cardinality	20%, 40%, 60%, 80%, 100%	100%

6 Experimental evaluation

We first present a case study, and then, we experimentally evaluate the efficiency of the existing pivot selection algorithms and the quality of the selected pivots according to their similarity search performance.

6.1 Experimental setup

All the pivot selection algorithms and associated similarity search algorithms were implemented in C++. The only exception is that the dimensionality reduction in PCA, which accounts for only a small fraction of PCA's overall cost, is implemented in Python. All experiments were conducted on an Intel Core i7-7700 3.6GHz PC with 32GB memory. All source code of the implemented algorithms is publicy available[1].

Datasets. We use three real-life datasets: (i) *LA*[2] that contains geographical locations in Los Angeles, where the L_2-norm is used as the distance metric; (ii) *Words*[3] that consists of proper nouns, acronyms, and compound words taken from the Moby project, using edit distance as the metric; and (iii) *Color*[4] that contains standard MPEG-7 image features, extracted from *Frickr*, using the L_1-norm distance as the metric. In addition, we create a synthetic dataset *INT*, where the values of the first five dimensions are randomly generated, while the values of the remaining dimensions are linear combinations of the previous ones. We employ the L_∞-norm distance as the metric for *INT*. To control the maximum distance of each dataset, each dimension value in *LA* and *Color* is mapped to [-255, 255] while values in *INT* are mapped to [0, 10,000]. Table 3 offers summary statistics of the datasets, where cardinality, dimensionality, and distance are abbreviated as Card., Dim., and Dis..

[1] https://github.com/ZJU-DBL/PSAMS
[2] http://www.dbs.informatik.uni-muenchen.de/∼seidl
[3] http://icon.shef.ac.uk/Moby/
[4] http://cophir.isti.cnr.it/

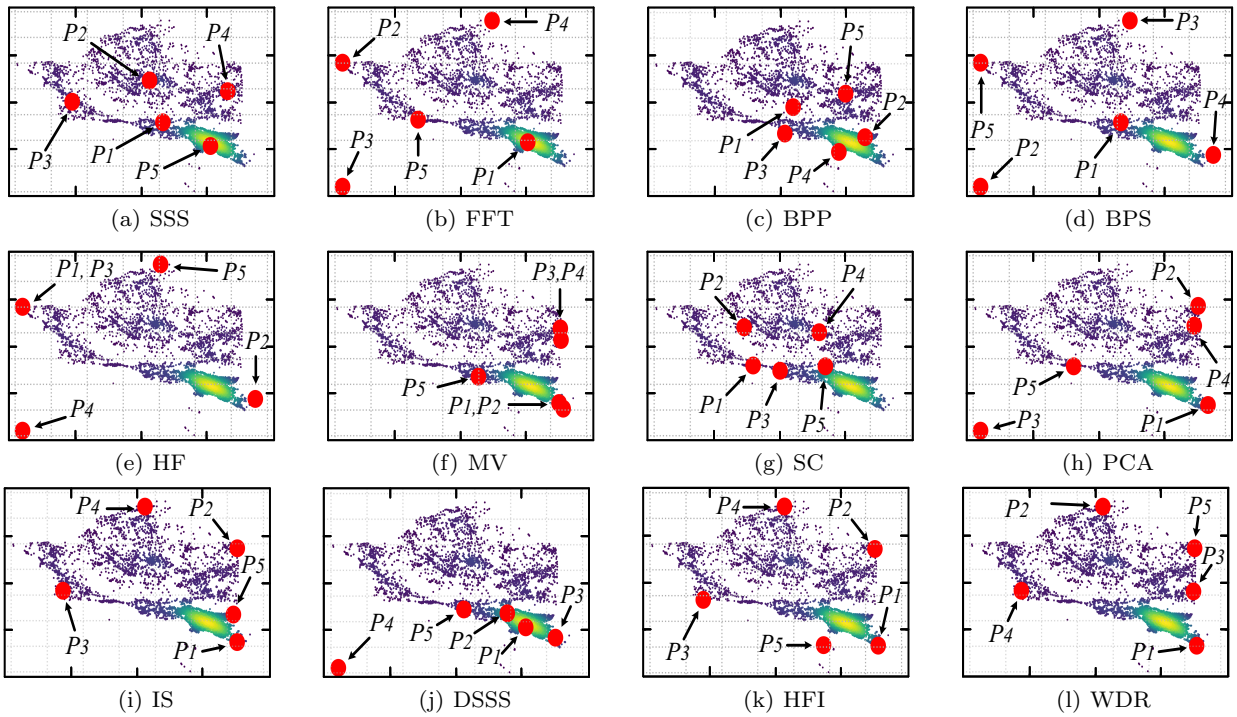


Fig. 3 Pivots Chosen by Different Algorithms on *LA*

Performance metrics. We quantify the quality of pivots according to the performance of (i.e., how well they support) metric similarity queries (including MRQ and MkNNQ) using three typical pivot-based indexes belonging to different categories [9]: the pivot-based table LAESA [26], the pivot-based tree MVPT [3], and the pivot-based external index SPB-tree [9]. Moreover, LAESA and MVPT only use the pivot filtering for similarity search, while the SPB-tree utilizes both pivot filtering and validation. The query performance is measured in terms of the number of distance computations (*compdists*) and the CPU time. The *compdists*, captures directly the performance of pivot filtering and validation. The fewer *compdists*, the stronger the filtering and validation ability of the pivots.

Parameters. We conduct experiments to evaluate metric similarity search when varying parameters k, r, and the cardinality percentage w.r.t. the entire dataset. Parameter k is used in the MkNNQ, and parameter r is used in the MRQ. Detailed information is presented in Table 4. Specifically, r denotes the percentage of the search radius w.r.t. the maximum distance. Each reported measurement is an average over 500 random queries. In the experiments, we explore the performance of 11 existing pivot selection algorithms and the proposed algorithm WDR, using a fixed number of pivots. In order to identify an appropriate pivot set cardinality, we also conduct experiments that assess the effect of the number of pivots. No new observations compared to a previous study [11] are found, which suggests to use 5 pivots. Hence, we adjust the parameters used in the different algorithms to ensure that 5 pivots are chosen (i.e., adjusting α used in SSS), that the candidate pivot set has cardinality around 300 (i.e., adjusting α used in MV and DSSS, and $\widehat{m}$), and that the sample size is around 1% the dataset size (i.e., adjusting $\widehat{n}$ and $|A|$). In SC, we set the strictest settings of ϵ_{sp} and ϵ_{sc} to make the pivots unchanged w.r.t. 1% the entire dataset. In addition, we set parameter λ to 2 for WDR based on a study on the four datasets where we vary λ.

6.2 Case study

We present a case study on *LA*, which is a 2D dataset that makes it easy to visualize the selected pivots. This can help to understand the characteristics and objective functions of the pivot selection methods. However, additional findings can be obtained when considering higher dimensions due to different data distributions and features. Note that visualization for different types of datasets is left as a promising future research direction.

Fig. 3 shows the 5 pivots selected by the 12 pivot selection methods. The figure displays the objects in the datasets, and the red points are the pivots. The color implies the data distribution, where the yellow part is

the most dense region in *LA*. In addition, an analysis of the quality of the pivots is provided in Section 6.5. The key findings are the following:

• SSS selects pivots that are uniformly distributed, which includes a pivot located in the most dense region.

• The first pivot of FFT is randomly selected, and thus is located in the densest region. Then, the farthest object P_2 w.r.t. the first pivot is chosen as a pivot. The other pivots are selected similarly.

• The pivots selected by BPP surrounds the dense region, and hence, they have low standard deviation of the rank permutation.

• BPS is similar to FFT. Both select four distant pivots w.r.t. the first randomly chosen pivot.

• Although the first and third pivots of HF are very close, the others are outliers, indicating that HF is good at finding outliers.

• Some of the pivots selected by MV are close to each other. MV has the potential drawback of choosing pivots near each other, which have similar variance.

• SC improve on MV by considering the correlation coefficient of selected pivots; thus, no pivots are close to each other.

• The pivots of PCA are obtained by dimensionality reduction. The second and forth pivots are close to each other, while the others distribute well.

• The pivots selected by IS, HFI, and WDR have similar distributions. However, DSSS, where four pivots surround the densest region, has a very different distribution. This is because DSSS focuses on the contribution of each pivot—it is easier for objects close to the dense region to make a large contribution.

6.3 Efficiency of pivot selection algorithms

Fig. 4 reports the CPU time of all the pivot selection algorithms on the four datasets when varying the cardinality from 20% to 100%. The first observation is that the run time of most pivot selection algorithms increases as the cardinality of the dataset grows, which is consistent with the complexity analyses summarized in Table 2. However, there are two exceptions: (i) The runtime of DSSS decreases on *Color* when the cardinality increases from 40% to 60%. This is because the time complexity of DSSS depends on the cardinality of the dataset and the number of candidate pivots. Although the cardinality of the dataset increases from 40% to 60%, the number of candidate pivots drops from 335 to 133, yielding a decreased runtime. (ii) WDR fluctuates due to its local optimization strategy. We also observed that SSS achieves the best performance in terms of CPU time, while BPS, HF, and FFT are also very efficient. Each of these three algorithms needs to compute all the distances between the objects and the pivots, meaning that they have the same time complexity. In contrast, BPP, MV, and PCA have the highest CPU cost. The reasons are that i) BPP re-computes the permutation rank in each iteration; and ii) MV and PCA have to compute all the distances between candidate pivots and objects.

The P-P distribution based methods select pivots by mainly considering the distances among pivots, the O-O distribution based methods sample object pairs in the dataset and select pivots based on the distances of sampled object pairs, while the P-O distribution based methods consider the distances between pivots and all the objects in the dataset. In general, the P-P distribution based methods are the most efficient, followed by the O-O distribution based methods, while the P-O distribution based methods have the highest CPU cost. This is because (i) few pivots are needed for the P-P distribution based methods, resulting in few distance computations; and (ii) the sampling technique used by the O-O distribution based methods avoid distance computations across the entire dataset, which significantly reduces the number of distance computations compared with the P-O distribution based methods. Although BPP is a P-P method, it has the highest time cost. This is because P-P methods focus mostly on the distances between pivots to achieve high efficiency, while BPP also considers the distribution between pivots w.r.t. all objects.

6.4 The quality of pivots

We compare the metric similarity search performance enabled by different pivot selection methods using three representative metric indexes: LAESA [26] (belonging to the pivot-based table category, which utilizes a table structure to store the pre-computed distances between objects and pivots), MVPT [3] (belonging to the pivot-based tree category, which uses a main-memory tree structure to index the pre-computed distances), and the SPB-tree [9] (belonging to the pivot-based external index category, which employs an external B^+-tree to maintain the pre-computed distances) when varying the search radius r for MRQ and k for MkNNQ.

6.4.1 Metric range query

We first compare the quality of pivots according to MRQ performance. Fig. 5 depicts the MRQ performance on *LA* and *Words*, while Fig. 6 presents the MRQ performance on *INT* and *Color*. Because no new findings are observed in the MRQ results on *INT*

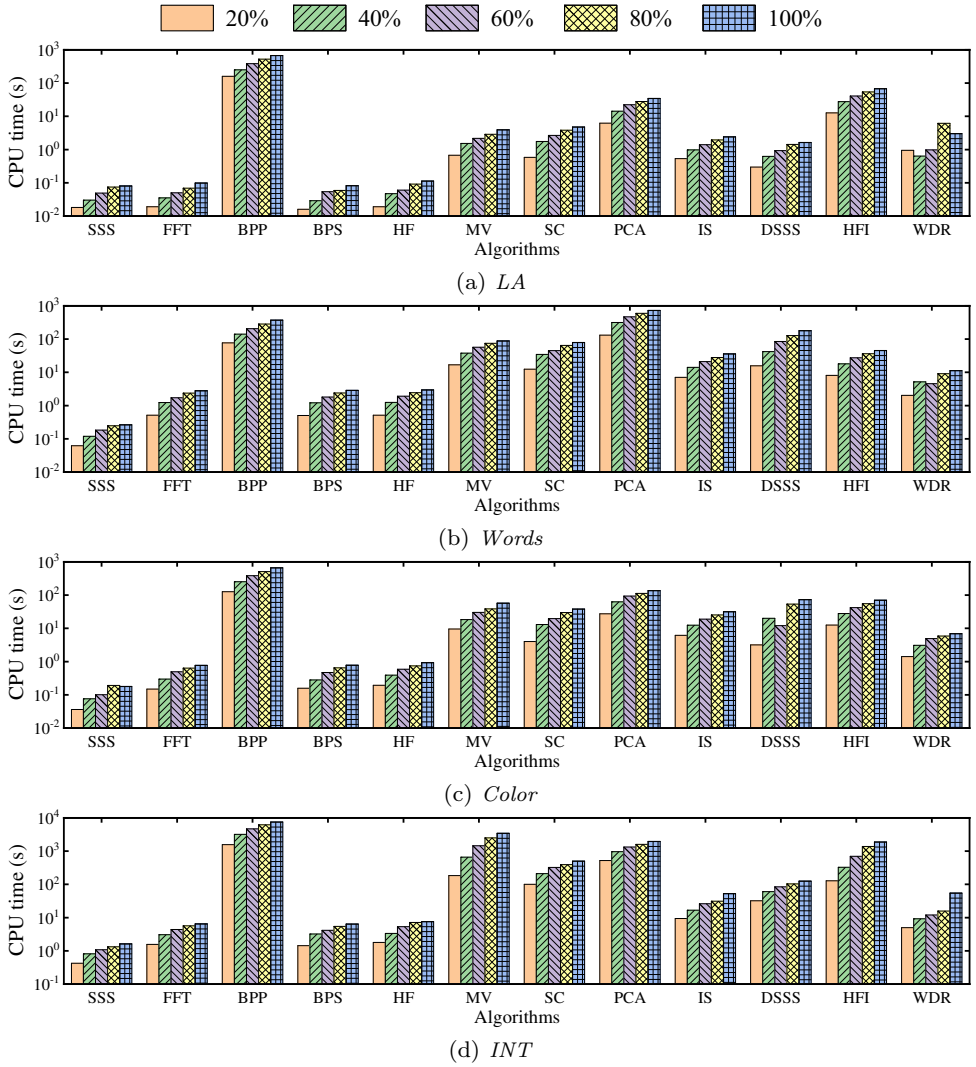


Fig. 4 CPU Time Performance of Pivot Selection Algorithms

and *Color* compared to those on *LA* and *Word*, we omit a detailed analysis. As can be seen, the CPU time and *compdists* increase with the growth of r in most cases. However, due to the strong validation ability of the SPB-tree, the costs of the SPB-tree on *LA* decreases as r increases, especially for large search radii.

MRQ performance using LAESA. As can be observed, the performance (i.e., *compdists*) achieved when using the different pivot selection methods is very similar to that on *LA* while it varies considerably on *Words*. This is because the (intrinsic) dimensionality of *LA* is very small, enabling the 5 pivots to offer good performance. The second observation is that FFT, BPS, HF, IS, and HFI have the lowest CPU cost on *LA*, although all the algorithms share similar *compdists* performance (see Figs. 5(a) and 5(d)). This is because LAESA uses pivots one by one to filter objects, indicating that the order of pivots is very important. If the pruning ability of the first few pivots is high, LAESA can achieve high efficiency, especially when the distance metric is simple. However, HF, MV, IS, HFI, and WDR have the best performance on *Words* (see Figs. 5(g) and 5(j)). The reason is that, the distances between objects in this dataset are very similar (i.e., discrete distances in the range of [0, 34]), making some P-P methods (including FFT, SSS, BPP, and HF) unable to find a small number of well-distributed pivots.

MRQ performance using MVPT. The performance of MVPT is similar to that of LAESA. The difference between the two is that LAESA uses a table to store all the objects, while MVPT sorts all the objects according to their distances w.r.t. pivots and stores them in a balanced tree. When performing MRQ, MVPT can prune all objects in a sub-tree as a whole, while LAESA can only prune objects one by one; thus, the pruning efficiency of MVPT is

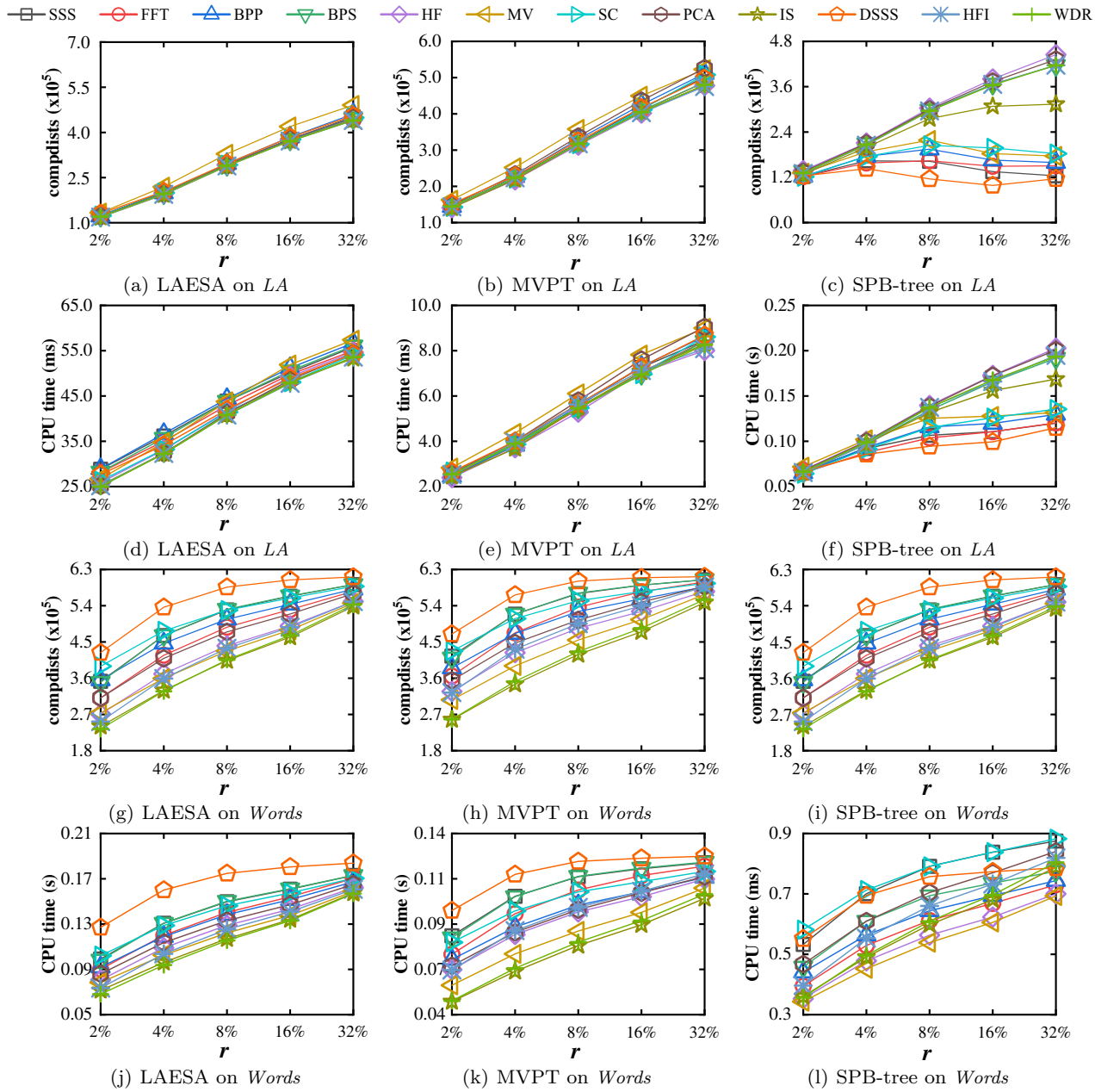


Fig. 5 MRQ Performance vs. Search Radius r on *LA* and *Words*

higher than that of LASEA. As can be observed (in Figs. 5(d), 5(e), 5(j), and 5(k)), the CPU time when using MVPT is much smaller than that achieved when using LASEA.

MRQ performance using the SPB-tree. By exploiting the pivot validation technique, SSS, FFT, and DSSS achieve the best performance on *LA* (see Figs. 5(c) and 5(f)), while HF, MV, IS, and WDR perform the best on *Words* (see Figs. 5(i) and 5(l)). Recalling the observations in Section 3, outliers have better filtering capabilities while centers have stronger validation abilities.

Based on the above observations, the O-O distribution based algorithms have better MRQ performance in most cases.

6.4.2 Metric k nearest neighbor query

Next, we evaluate the quality of pivots according to MkNNQ performance. Fig. 7 depicts the results using *Color* and *INT*.

MkNNQ performance using LAESA. We see that IS, HFI, and WDR have the best performance on both datasets (see Figs. 7(a), 7(d), 7(g), and 7(j)).

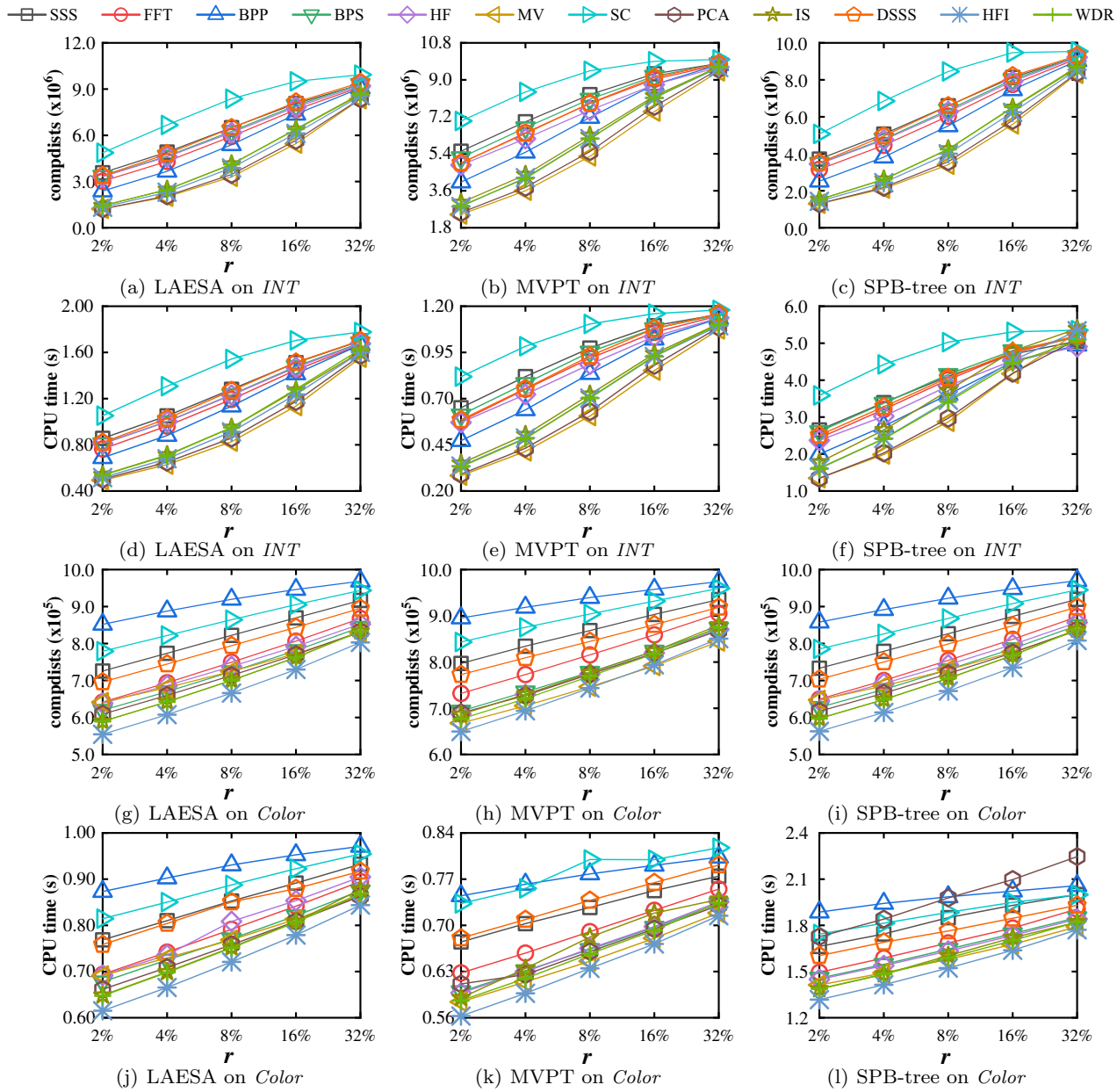


Fig. 6 MRQ Performance vs. Search Radius r on *INT* and *Color*

This is because the O-O distribution methods aim to maximize the pruning ability of the selected pivots; thus, they can achieve high performance. In addition, among the P-P methods, FFT, HF, and BPS have better performance than SSS and BPP on *Color* (see Figs. 7(a) and 7(d)). However, BPP performs better than FFT, HF, and BPS on *INT* (see Figs. 7(g) and 7(j)). The reason is that *INT* is generated randomly and has a relatively uniform distance distribution, which enables BPP to find uniformly distributed pivots. However, for skewed distributions, the outliers selected by FFT, HF, and BPS are better.

M*k*NNQ performance using MVPT. The observations for MVPT are similar to those for LAESA. However, on *INT*, LAESA incurs much fewer distance computations than does MVPT (see Figs. 7(g) and 7(h)). This is because, for M*k*NNQ, LAESA reduces the search space quickly and has fewer distance computations. It does this by using the first few visited objects to achieve better pruning. In contrast, MVPT visits the tree in a top-down manner and can only reduce the search space when it reaches the objects in the leaf nodes.

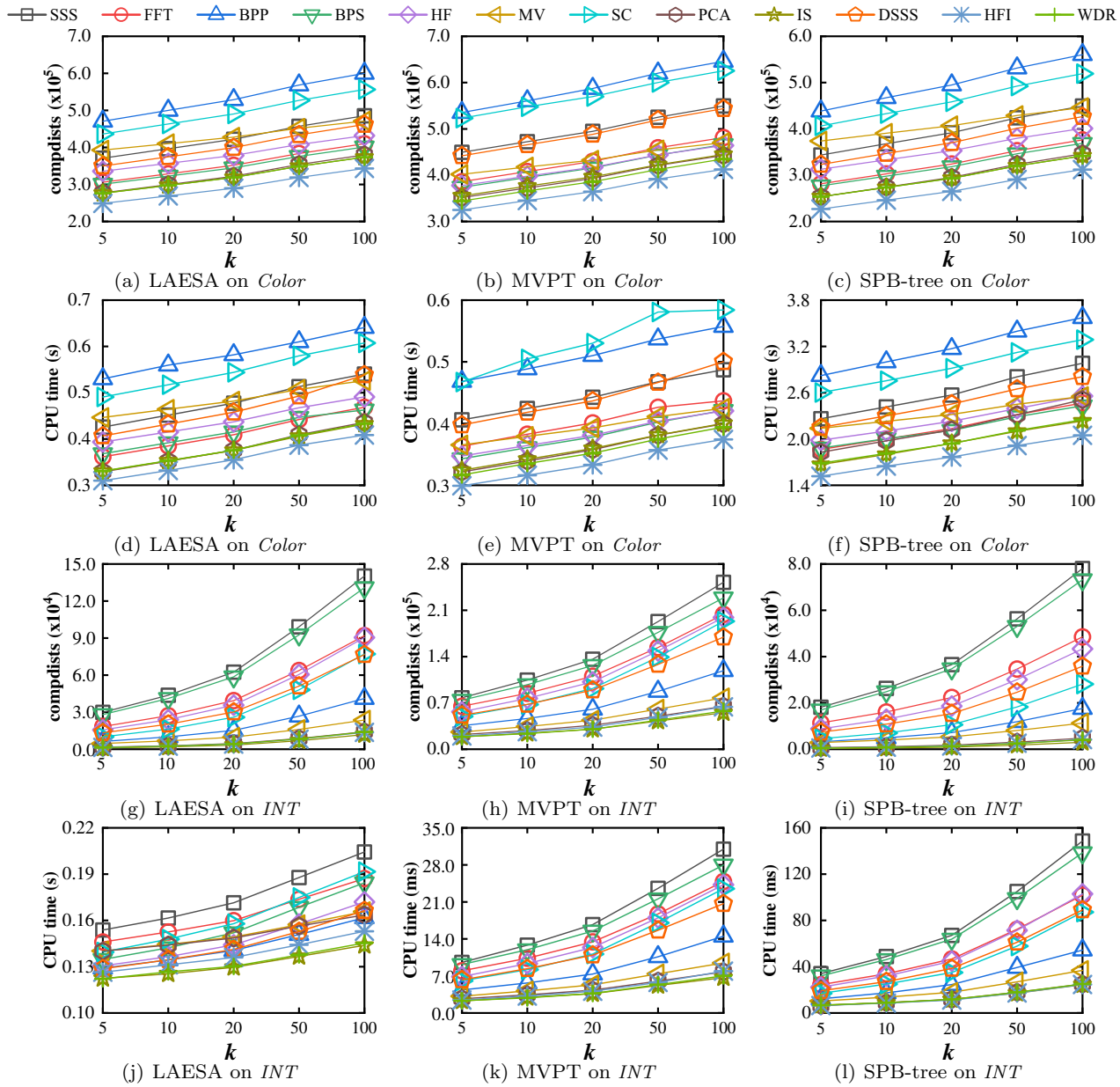


Fig. 7 MkNNQ Performance vs. k

MkNNQ performance using the SPB-tree. Overall, the use of the SPB-tree leads to similar observations as for LAESA on *Color* and *INT*. Note that, the CPU time depends on two factors, i.e., the pruning ability and the cost of pruning. Therefore, although FFT, HF, SC, and DSSS have different *compdists* (see Fig. 7(i)), their CPU times are similar (see Fig. 7(l)).

6.4.3 Statistical significance test

As different algorithms share the same query performance, we also conduct statistical significance tests to determine whether statistically significant differences exist among the pivot selection algorithms. We report statistical significance tests for MkNNQ and MRQ results on four datasets. As the distribution of performance results w.r.t. different algorithms is unknown, we apply the Friedman test to quantify the statistical differences among all pivot selection algorithms in terms of 60 metrics. These 60 metrics are all the *compdists* and CPU time results when performing MkNNQ and MRQ while varying k or r and using the LAESA, MVPT, and the SPB-tree on each dataset. The Friedman test is a non-parametric test that quantifies the differences between groups

Table 5 Significance Values among All Algorithms using Friedmans Test on *LA*

	SSS	FFT	BPP	BPS	HF	MV	SC	PCA	IS	DSSS	HFI	WDR
SSS	–	**0.003**	**0.909**	**0.859**	**0.899**	0	**0.016**	**0.83**	0	**0.586**	0	0
FFT	**0.003**	–	**0.002**	**0.002**	**0.004**	0	**0.569**	**0.006**	**0.1**	0	**0.403**	**0.519**
BPP	**0.909**	**0.002**	–	**0.95**	**0.81**	0	**0.011**	**0.742**	0	**0.667**	0	0
BPS	**0.859**	**0.002**	**0.95**	–	**0.761**	0	**0.009**	**0.695**	0	**0.714**	0	0
HF	**0.899**	**0.004**	**0.81**	**0.761**	–	0	**0.022**	**0.929**	0	**0.502**	0	0
MV	0	0	0	0	0	–	0	0	0	0	0	0
SC	**0.016**	**0.569**	**0.011**	**0.009**	**0.022**	0	–	**0.028**	**0.027**	**0.003**	**0.16**	**0.224**
PCA	**0.83**	**0.006**	**0.742**	**0.695**	**0.929**	0	**0.028**	–	0	**0.448**	0	0
IS	0	**0.1**	0	0	0	0	**0.027**	0	–	0	**0.418**	**0.317**
DSSS	**0.586**	0	**0.667**	**0.714**	**0.502**	0	**0.003**	**0.448**	0	–	0	0
HFI	0	**0.403**	0	0	0	0	**0.16**	0	**0.418**	0	–	**0.849**
WDR	0	**0.519**	0	0	0	0	**0.224**	0	**0.317**	0	**0.849**	–

Table 6 Significance Values among All Algorithms using Friedmans Test on *Words*

	SSS	FFT	BPP	BPS	HF	MV	SC	PCA	IS	DSSS	HFI	WDR
SSS	–	0	**0.001**	**0.667**	0	0	**0.667**	0	0	**0.008**	0	0
FFT	0	–	**0.436**	0	0	0	0	**0.156**	0	0	0	0
BPP	**0.001**	**0.436**	–	**0.004**	0	0	0	0	0	0	0	0
BPS	**0.667**	0	**0.004**	–	0	0	**0.389**	0	0	**0.002**	0	0
HF	0	0	0	0	–	0	0	**0.005**	0	0	**0.839**	0
MV	0	0	0	0	0	–	0	0	**0.028**	0	**0.085**	**0.061**
SC	**0.667**	0	0	**0.389**	0	0	–	0	0	0	0	0
PCA	0	**0.156**	0	0	**0.005**	0	0	–	0	0	**0.002**	0
IS	0	0	0	0	0	**0.028**	0	0	–	0	0	**0.742**
DSSS	**0.008**	0	0	**0.002**	0	0	0	0	0	–	0	0
HFI	0	0	0	0	**0.839**	**0.085**	0	**0.002**	0	0	–	0
WDR	0	0	0	0	0	**0.061**	0	0	**0.742**	0	0	–

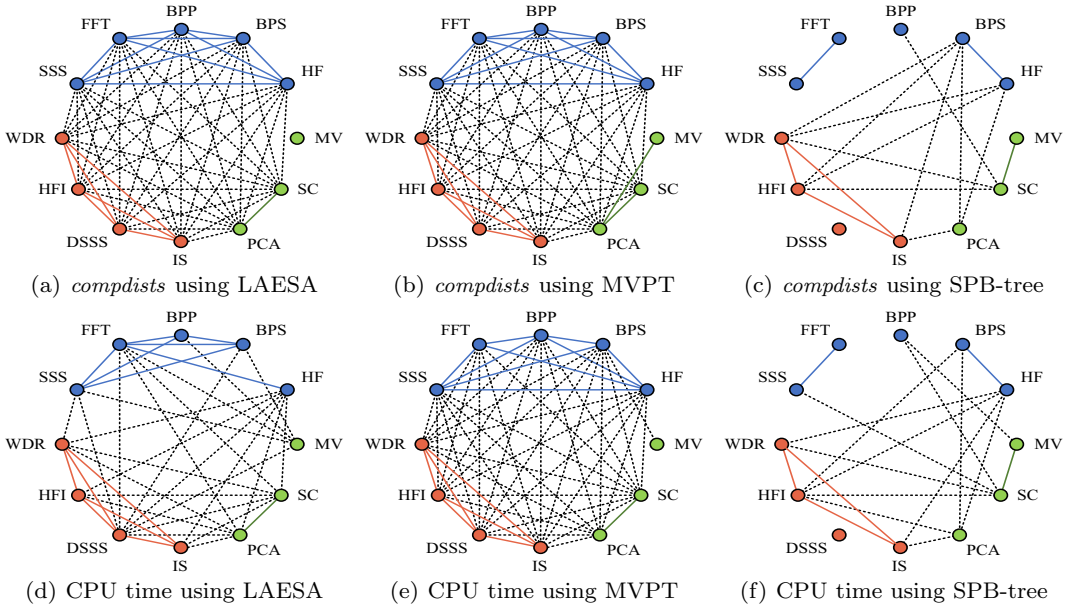
Table 7 Significance Values among All Algorithms using Friedmans Test on *Color*

	SSS	FFT	BPP	BPS	HF	MV	SC	PCA	IS	DSSS	HFI	WDR
SSS	–	0	**0.001**	0	0	0	**0.038**	0	0	**0.18**	0	0
FFT	0	–	0	**0.016**	**0.323**	**0.156**	0	0	0	**0.007**	0	0
BPP	**0.001**	0	–	0	0	0	**0.215**	0	0	0	0	0
BPS	0	**0.016**	0	–	**0.156**	**0.323**	0	**0.111**	**0.004**	0	0	0
HF	0	**0.323**	0	**0.156**	–	**0.667**	0	**0.003**	0	0	0	0
MV	0	**0.156**	0	**0.323**	**0.667**	–	0	**0.01**	0	0	0	0
SC	**0.038**	0	**0.215**	0	0	0	–	0	0	**0.001**	0	0
PCA	0	0	0	**0.111**	**0.003**	**0.01**	0	–	**0.197**	0	0	**0.017**
IS	0	0	0	**0.004**	0	0	0	**0.197**	–	0	0	**0.276**
DSSS	**0.18**	**0.007**	0	0	0	0	**0.001**	0	0	–	0	0
HFI	0	0	0	0	0	0	0	0	0	0	–	**0.007**
WDR	0	0	0	0	0	0	0	**0.017**	**0.276**	0	**0.007**	–

when the dependent variable being measured is ordinal. For this test, we have two hypotheses: (i) the null hypothesis that the metric similarity search results of different algorithms share the same distribution and (ii) the alternative hypothesis that their distributions are different. Tables 5 through 8 provide the significance values among all the pivot selection algorithms. The values are reported as "0" when "0.000" is obtained after rounding the values and using three significant decimals. The tables support two conclusions. First, the significances for the similarity search performance of all pivot selection algorithms on different datasets (i.e., 0.000) indicate significant differences between the different algorithms, i.e., the null hypothesis does not hold. This also shows that our analyses of all the pivot selection algorithms are meaningful. Second, as a larger significance value between two algorithms corresponds to a higher probability that they share similar query performance, Tables 5 to 8 demonstrate that the algorithms with similar performance have limited significance differences, meaning that the distributions of the pivots chosen by those algorithms have common characteristics. In particular, *LA* has the most non-zero significances among the four datasets. This is because, the number of pivots is fixed at 5 for each dataset in the experiments. Due to the low (intrinsic) dimensionality

Table 8 Significance Values among All Algorithms using Friedmans Test on *INT*

	SSS	**FFT**	**BPP**	**BPS**	**HF**	**MV**	**SC**	**PCA**	**IS**	**DSSS**	**HFI**	**WDR**
SSS	–	0	0	**0.08**	0	0	**0.05**	0	0	0	0	0
FFT	0	–	0	**0.66**	**0.206**	0	**0.097**	0	0	**0.714**	0	0
BPP	0	0	–	0	**0.001**	0	0	0	0	0	0	0
BPS	**0.076**	**0.66**	0	–	**0.002**	0	**0.859**	0	0	**0.28**	0	0
HF	0	**0.206**	0	**0.002**	–	0	**0.003**	0	0	**0.369**	0	0
MV	0	0	0	0	0	–	0	**0.704**	**0.761**	0	**0.622**	**0.714**
SC	**0.051**	**0.97**	0	**0.859**	**0.003**	0	–	0	0	0	**0.43**	0
PCA	0	0	0	0	0	**0.704**	0	–	**0.939**	0	**0.909**	**0.99**
IS	0	0	0	0	0	**0.761**	0	**0.939**	–	0	**0.849**	**0.95**
DSSS	0	**0.714**	0	**0.28**	**0.369**	0	**0.043**	0	0	–	0	0
HFI	0	0	0	0	0	**0.622**	0	**0.909**	**0.849**	0	–	**0.899**
WDR	0	0	0	0	0	**0.714**	0	**0.99**	**0.95**	0	**0.899**	–



(a) *compdists* using LAESA (b) *compdists* using MVPT (c) *compdists* using SPB-tree

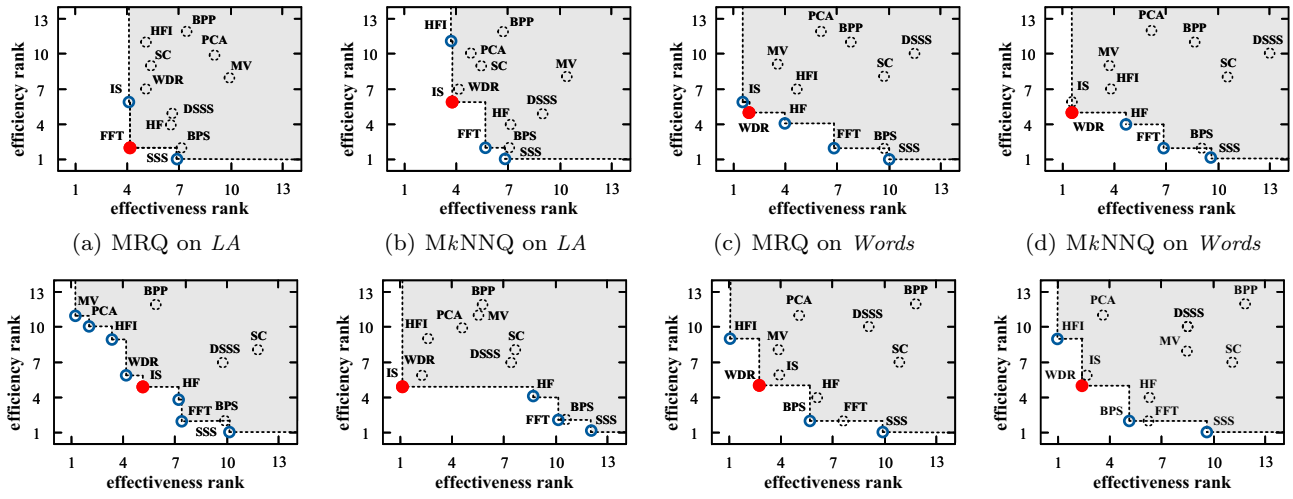
(d) CPU time using LAESA (e) CPU time using MVPT (f) CPU time using SPB-tree

Fig. 8 *Z*-test on *LA*

of *LA*, 5 pivots are sufficient to achieve very good performance, making that all the algorithms achieve similar performance on *LA*.

Friedman test can detect the differences among all the algorithms over multiple test attempts, i.e., by analyzing all the performance metrics (namely, *compdists* and the CPU time) of MkNNQ and MRQ using three typical indexes (viz., LAESA, MVPT, and SPB-tree) under all the parameter settings. In contrast, *Z*-test aims at a single attempt, which detects the differences of all the algorithms by analyzing a single performance metric when performing MRQ or MkNNQ with a specific parameter setting using a specific index.

We present a *Z*-test to detect the differences among all the algorithms on *LA* using MRQ. Note that, only *LA* is used because the performance of all the algorithms is similar on *LA* as shown in Figs. 5(a) to 5(f). More specifically, we run a *Z*-test with the p-value of 0.05 to test whether the performance (including *compdists* and the CPU time) of all the algorithms when performing MRQ (with $r = 8\%$) is actually similar to each other. The *Z*-test results are depicted in Fig. 8, where the algorithms belonging to different categories are plotted as circles with different colors, i.e., P-P algorithms are in blue, P-O algorithms are in green, and O-O algorithms are in orange. Two algorithms that have the same MRQ performance (i.e., the p-value smaller than 0.05) are connected with solid lines if they belong to the same category, and are connected with dotted lines if they belong to different categories. As observed, the dense connections among all the algorithms indicate their *compdists* and CPU time share similar distributions. It is consistent with our previous conclusion that the pivot selection algorithms achieve similar performance on *LA*. However, since the SPB-tree index employs both the pivot pruning and pivot validation techniques, while LAESA and MVPT only employ the pivot pruning techniques, the



(a) MRQ on *LA* (b) MkNNQ on *LA* (c) MRQ on *Words* (d) MkNNQ on *Words*

(e) MRQ on *INT* (f) MkNNQ on *INT* (g) MRQ on *Color* (h) MkNNQ on *Color*

Fig. 9 Effectiveness Rank vs. Efficiency Rank

connections among all the algorithms using SPB-tree index are sparse. It means that the performance of all the algorithms is different using the SPB-tree on *LA*, which is consistent with the MRQ experimental results illustrated in Figs. 5(a) to 5(f).

6.5 Performance analysis

Considering that it is hard to find the top pivot selection algorithm on all datasets in terms of the different performance metrics, we aggregate the effectiveness (in terms of the similarity search performance when using the selected pivots) and the efficiency (in terms of the runtime of pivot selection methods) of all the algorithms to enable a fair comparison. Inspired by the skyline concept, we plot the effectiveness rank (i.e., the mean rank of the distance computations and the CPU time when using different indexes and when varying the parameter k in MkNNQ or the radius in MRQ) and the efficiency rank in Fig. 9. In the figure, the blue and red circles indicate algorithms on the skyline, i.e., algorithms that cannot be dominated by any other algorithms. An algorithm is not dominated by another algorithm if the other has lower effectiveness or efficiency rank. The skyline algorithms thus constitute appropriate pivot selection methods for a given setting. Further, the algorithms represented by red circles are the best choices (according to the sum of the effectiveness and efficiency ranks) in the particular setting.

As can be observed, FFT, SSS, and IS are skyline algorithms on the *LA* dataset (see Figs 9(a) and 9(b)), among which FFT is the best choice as it has relatively high effectiveness and low time cost. Next, SSS, FFT, HF, IS, and WDR are skyline algorithms on the *Words* dataset (see Figs 9(c) and 9(d)), among which WDR is the best choice. In the case of the *INT* dataset, there are many skyline algorithms, among which IS is the best choice (see Figs 9(e) and 9(f)). This is because *INT* is a synthetic dataset while the others are real-life datasets. *INT* has two notable characteristics: a random data distribution and linearly correlated dimensions. Thus, P-O methods (MV and PCA) are able to leverage statistical techniques (distance variance and dimensionality reduction) to learn the data distribution and select representative objects as pivots, giving them the best MRQ performance. However, in terms of MkNNQ results, the O-O methods (such as IS, WDR, and HFI) are better. This is because MRQ returns a large fraction of the dataset, while MkNNQ only considers a small fraction of the data. On the*Color* dataset, SSS, BPS, HFI, and WDR are skyline algorithms (see Figs 9(g) and 9(h)), and WDR is the best choice.

Based on the above observations, the P-P methods FFT and SSS are always skyline algorithms because of their efficiency, while the O-O methods IS and WDR are skyline algorithms because they are able to select high-quality pivots (i.e., the best MRQ and MkNNQ performance). This indicates that pivot selection algorithms with higher CPU cost do not necessarily yield pivots of higher quality. As a summary, we recommend the O-O distribution based algorithms IS and WDR for most datasets if the pivot quality is more important (e.g., for off-line analysis), and we recommend the P-P distribution based methods FFT and SSS if pivot selection efficiency is more important

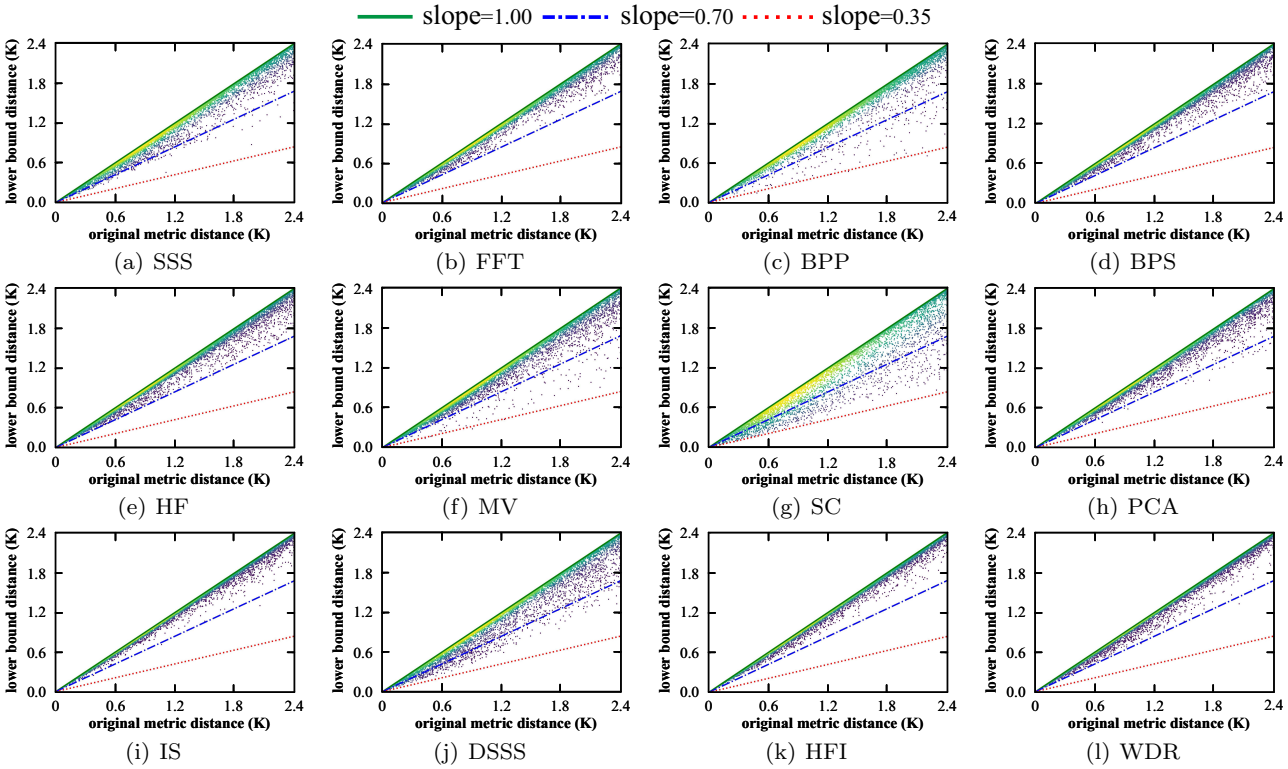


Fig. 10 Lower Bound Distance vs. Original Metric Distance

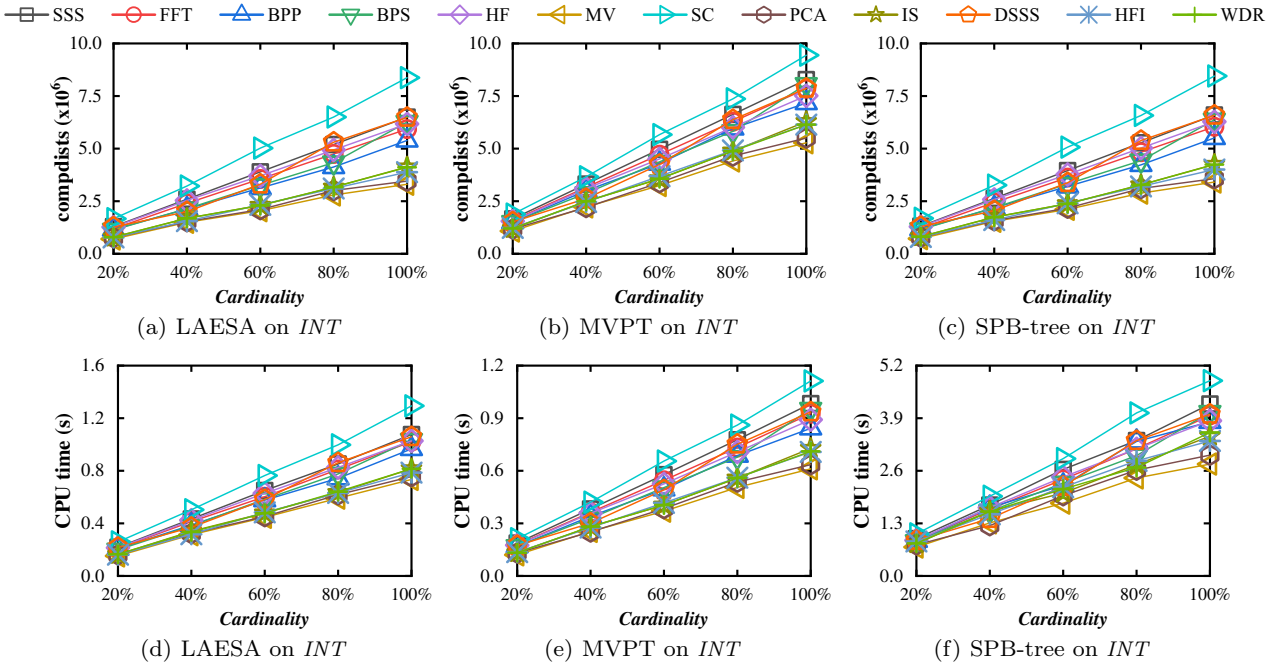
(e.g., in online analysis in dynamic environments). In addition, for high (intrinsic) dimensional datasets, the O-O distribution based algorithms IS and WDR are recommended; and for low (intrinsic) dimensional real-life datasets, the P-P distribution based algorithm FFT is recommended. The reason is that for low (intrinsic) dimensional datasets, a few pivots are sufficient to achieve high search performance; thus, the P-P distribution based methods are recommended due to their high efficiency and quality.

In order to study the quality of pivots, we use *LA* as an representative example to compare the object pair distances with corresponding lower bound distances, as shown in Fig. 10, where the X-axis denotes the original metric distance while the Y-axis represents the lower bound distance. To better present the detailed information, we only consider x values in the range [0, 2400], as 99.9% distances are located in this range. The red dotted line, the blue dashed-and-dotted line, and the green solid line have slopes of 0.35, 0.70, and 1, respectively. The more points that are located close to the green line, the better the pruning by the pivots is; and the more points that are located close to the red line, the worse the pruning by the pivots is. In addition, we utilize different colors to denote different distance distributions. The purple color indicates sparse distributions, while the yellow color indicates dense distributions.

As can be observed, FFT, BPS, HF, IS, HFI, and WDR have nearly no points between the blue dashed-and-dotted line and the red dotted line. These algorithms perform the best, as seen in Fig 9. This is because the pruning ability gets stronger when the lower bound distance approaches the original metric distance. In other words, the more points that are near the solid green line (the slope equals 1), the better the pivot selection algorithm is. Moreover, the points of MV are denser around the green solid line and farther from the red dotted line, compared with PCA. This is consistent with the observation that MV performs slightly better than PCA on *LA*, as shown in Fig. 5. As a summary, good pivots have higher ratios between the lower bound distance and the original metric distance. This motivates WDR that uses a power law probabilistic distribution to better control the distribution of distance ratios.

6.6 Scalability analysis

The running time performance (i.e., efficiency) of all pivot selection algorithms when varying the cardinality of a dataset is studied in Section 6.3. Here, we further



(a) LAESA on *INT* (b) MVPT on *INT* (c) SPB-tree on *INT*

(d) LAESA on *INT* (e) MVPT on *INT* (f) SPB-tree on *INT*

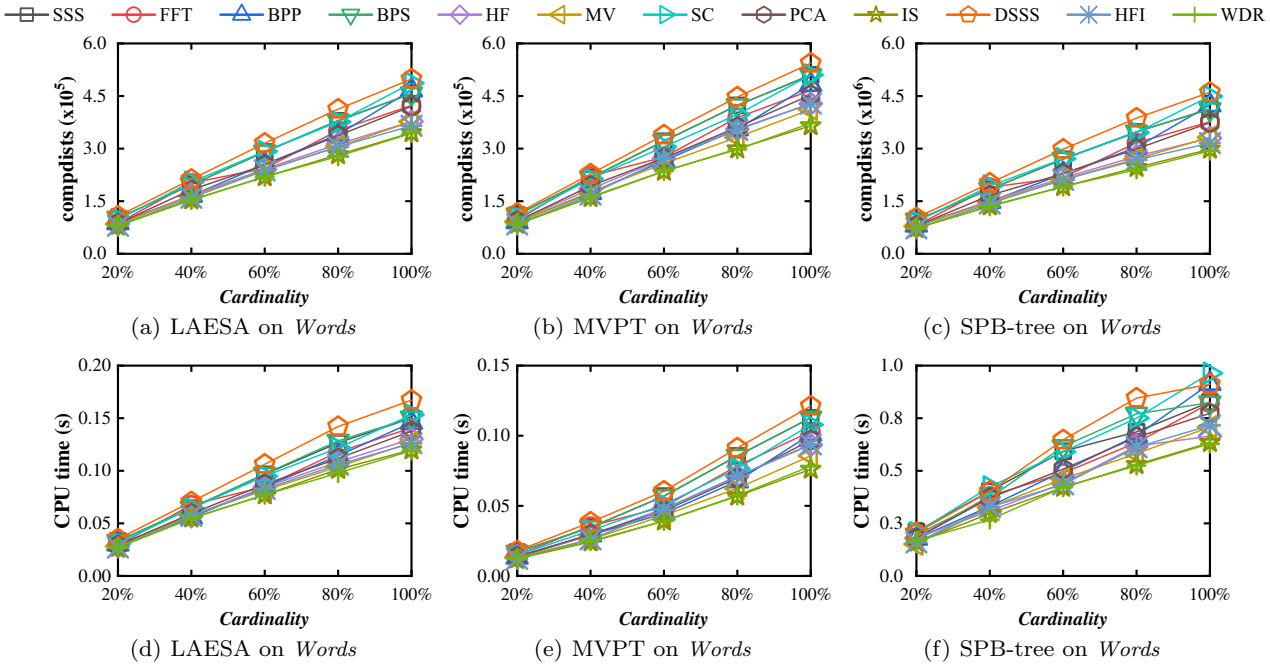
Fig. 11 MRQ Performance vs. Cardinality of Dataset

explore the quality of pivots for scalability analysis. Here, we use the pivots selected in Section 6.3 and perform MRQ and MkNNQ using indexes LAESA, MVPT, and the SPB-Tree. For brevity, we only report the MRQ and MkNNQ results on one dataset each. Fig. 11 shows the MRQ performance results (i.e., the *compdists* and the CPU time) on *INT* when the dataset size is varied from $1M$ to $10M$. Fig. 12 reports the MkNNQ performance results on *Words* when the dataset size is varied from 122,351 to 611,756. It is observed that, the number of distance computations and the CPU time increase linearly as the size of the dataset grows. The results imply that the pivots selected by all algorithms offer good scalability.

7 Conclusions

We classify all existing pivot selection algorithms known to us into three categories according to the different distance distributions they exploit, i.e., P-P, P-O, and O-O distribution based algorithms. Moreover, we present a new O-O distribution based algorithm, and we provide time complexity analyses of all the pivot selection algorithms. We conduct an comprehensive experimental evaluation of all the pivot selection algorithms using four datasets and three different pivot-based metric indexes. The findings and insights are summarized as follows.

- In most cases, the O-O distribution based algorithms yield the best search performance. This conclusion also shows that a higher time cost of a pivot selection algorithm may not necessarily yield higher-quality pivots, as the P-O distribution based algorithms are time consuming. Good pivot sets have good distributions, and the pivots have low correlations with each other. The P-P distribution based algorithms consider the pivot distribution, while the P-O distribution based algorithms mostly focus on the correlations between pivots. The O-O distribution based algorithms take both criteria into account in a different way—they maximize the ratio between lower bound distances and original metric distances.
- The structure of a pivot-based index affects the utility of the selected pivots. The order of pivots is more important to pivot-based trees compared with pivot-based tables and external indexes, as pivot-based trees use only one pivot for pruning at each tree level, while the others use the pivots as a whole for pruning. We note that PCA, IS, HFI, and WDR select pivots in the order of their quality according to different criteria. In addition, when we have many pivots (the number of pivots exceeds the (intrinsic) dimensionality of the dataset), algorithms SSS, FFT, and DSSS that choose uniformly distributed pivots are better due to their pivot validation. This holds especially true for large search spaces.



(a) LAESA on *Words* (b) MVPT on *Words* (c) SPB-tree on *Words*

(d) LAESA on *Words* (e) MVPT on *Words* (f) SPB-tree on *Words*

Fig. 12 MkNNQ Performance vs. Cardinality of Dataset

– Based on the above findings, we recommend the O-O distribution based algorithms IS and WDR for most datasets if the pivot quality is more important (e.g., for off-line analysis), and we recommend the P-P distribution based methods FFT and SSS if pivot selection efficiency is more important (e.g., for online analysis in dynamic environments). In addition, for high (intrinsic) dimensional datasets, the O-O distribution based algorithms IS and WDR are recommended; and for low (intrinsic) dimensional real-life datasets, the P-P distribution based algorithm FFT is recommended. The reason is that for low (intrinsic) dimensional datasets, a few pivots are sufficient to achieve high search performance; thus, the P-P distribution based methods are recommended due to their high efficiency and quality.

Although we conduct a comprehensive empirical study of pivot selection algorithms, a number of open issues remain that require further attention. Several promising research directions for pivot selections in metric spaces are summarized below:

– As stated in Section 1, specific contexts (e.g., data distribution, query types, similarity search types) can be leveraged to further improve the performance of pivot selection algorithms. For example, we use the power law probabilistic distribution in the proposed WDR algorithm to improve the pruning and validation capabilities. In addition, visualization of pivot distributions is a promising direction of future work for comprehending the features of specific metric datasets, potentially offering deeper insights into pivot selection algorithms.
– A recent study [6] utilizes machine learning to embed the distances between pivots and objects when selecting pivots. This approach is shown to be efficient and applicable to generic metric spaces. Inspired by this finding, it is of interest to develop pivot selection algorithms based on different machine learning techniques, in order to efficiently select high-quality pivots.
– All existing studies of pivot selection algorithms, including our experimental evaluation, are conducted on a single machine. With the growing volumes of data, it is highly relevant to study distributed data processing. Consequently, the last, but not the least important, direction is to develop pivot selection algorithms for use in distributed environments and on new hardware platforms.

Acknowledgments

This work was supported in part by the NSFC under Grants No. 62025206 and 61972338. Yunjun Gao is the corresponding author of the work.

References

1. Amato, G., Esuli, A., Falchi, F.: A comparison of pivot selection techniques for permutation-based indexing. Inf. Syst. **52**, 176–188 (2015)
2. Angiulli, F., Fassetti, F.: Principal directions-based pivot placement. In: SISAP, pp. 85–90 (2013)
3. Bozkaya, T., Özsoyoglu, Z.M.: Distance-based indexing for high-dimensional metric spaces. In: SIGMOD, pp. 357–368 (1997)
4. Bustos, B., Navarro, G., Chávez, E.: Pivot selection techniques for proximity searching in metric spaces. Pattern Recognit. Lett. **24**(14), 2357–2366 (2003)
5. Bustos, B., Pedreira, O., Brisaboa, N.R.: A dynamic pivot selection technique for similarity search. In: SISAP, pp. 105–112 (2008)
6. Carrara, F., Gennaro, C., Falchi, F., Amato, G.: Learning distance estimators from pivoted embeddings of metric objects. In: SISAP, pp. 361–368 (2020)
7. Chávez, E., Navarro, G.: A compact space decomposition for effective metric indexing. Pattern Recognit. Lett. **26**(9), 1363–1376 (2005)
8. Chávez, E., Navarro, G., Baeza-Yates, R., Maproquín, J.L.: Proximity searching in metric spaces. ACM Computing Surveys **33**(3), 273–321 (2001)
9. Chen, L., Gao, Y., Li, X., Jensen, C.S., Chen, G.: Efficient metric indexing for similarity search. In: ICDE, pp. 591–602 (2015)
10. Chen, L., Gao, Y., Song, X., Li, Z., Miao, X., Jensen, C.S.: Indexing metric spaces for exact similarity search. CoRR **abs/2005.03468** (2020)
11. Chen, L., Gao, Y., Zheng, B., Jensen, C.S., Yang, H., Yang, K.: Pivot-based metric indexing. PVLDB **10**(10), 1058–1069 (2017)
12. Dallachiesa, M., Palpanas, T., Ilyas, I.F.: Top-k nearest neighbor search in uncertain data series. PVLDB **8**(1), 13–24 (2014)
13. Echihabi, K., Zoumpatianos, K., Palpanas, T., Benbrahim, H.: Return of the lernaean hydra: Experimental evaluation of data series approximate similarity search. PVLDB **13**(3), 403–420 (2019)
14. Figueroa, K., Paredes, R.: An effective permutant selection heuristic for proximity searching in metric spaces. In: MCPR, pp. 102–111 (2014)
15. Gómez-Tostón, C., Barrena, M., Cortés, Á.: Characterizing the optimal pivots for efficient similarity searches in vector space databases with minkowski distances. Appl. Math. Comput. **328**, 203–223 (2018)
16. Hochbaum, D.S., Shmoys, D.B.: A best possible heuristic for the k-center problem. Math. Oper. Res. **10**(2), 180–184 (1985)
17. Jr., C.T., Filho, R.F.S., Traina, A.J.M., Vieira, M.R., Faloutsos, C.: The omni-family of all-purpose access methods: a simple and effective way to make similarity search more efficient. VLDB J. **16**(4), 483–505 (2007)
18. Kim, S., Lee, D., Cho, H.: An eigenvalue-based pivot selection strategy for improving search efficiency in metric spaces. In: BigComp, pp. 207–214 (2016)
19. Kimura, M., Saito, K., Ueda, N.: Pivot learning for efficient similarity search. In: KES, pp. 227–234 (2007)
20. Kurasawa, H., Fukagawa, D., Takasu, A., Adachi, J.: Margin-based pivot selection for similarity search indexes. IEICE Transactions **93-D**(6), 1422–1432 (2010)
21. Kurasawa, H., Fukagawa, D., Takasu, A., Adachi, J.: Optimal pivot selection method based on the partition and the pruning effect for metric space indexes. IEICE Transactions **94-D**(3), 504–514 (2011)
22. Leuken, R.H.V., Veltkamp, R.C., Typke, R.: Selecting vantage objects for similarity indexing. In: ICPR, pp. 453–456 (2006)
23. Li, W., Zhang, Y., Sun, Y., Wang, W., Li, M., Zhang, W., Lin, X.: Approximate nearest neighbor search on high dimensional data-experiments, analyses, and improvement. TKDE **32**(8), 1475–1488 (2020)
24. Mao, R., Miranker, W.L., Miranker, D.P.: Pivot selection: Dimension reduction for distance-based indexing. J. Discrete Algorithms **13**, 32–46 (2012)
25. Mao, R., Zhang, P., Li, X., Liu, X., Lu, M.: Pivot selection for metric-space indexing. Int. J. Mach. Learn. Cybern. **7**(2), 311–323 (2016)
26. Micó, L., Oncina, J., Carrasco, R.C.: A fast branch & bound nearest neighbour classifier in metric spaces. Pattern Recognit. Lett. **17**(7), 731–739 (1996)
27. Micó, L., Oncina, J., Vidal, E.: A new version of the nearest-neighbour approximating and eliminating search algorithm (AESA) with linear preprocessing time and memory requirements. Pattern Recognit. Lett. **15**(1), 9–17 (1994)
28. Nathan, V., Ding, J., Alizadeh, M., Kraska, T.: Learning multi-dimensional indexes. In: SIGMOD, pp. 985–1000 (2020)
29. Pedreira, O., Brisaboa, N.R.: Spatial selection of sparse pivots for similarity search in metric spaces. In: SOFSEM, pp. 434–445 (2007)
30. Sprenger, S., Schäfer, P., Leser, U.: Bb-tree: A main-memory index structure for multidimensional range queries. In: ICDE, pp. 1566–1569 (2019)
31. Sun, Y., Wang, W., Qin, J., Zhang, Y., Lin, X.: SRS: solving c-approximate nearest neighbor queries in high dimensional euclidean space with a tiny index. PVLDB **8**(1), 1–12 (2014)
32. Sundaram, N., Turmukhametova, A., Satish, N., Mostak, T., Indyk, P., Madden, S., Dubey, P.: Streaming similarity search over one billion tweets using parallel locality-sensitive hashing. PVLDB **6**(14), 1930–1941 (2013)
33. Tosun, U.: A novel indexing scheme for similarity search in metric spaces. Pattern Recognit. Lett. **54**, 69–74 (2015)
34. Venkateswaran, J., Kahveci, T., Jermaine, C.M., Lachwani, D.: Reference-based indexing for metric spaces with costly distance measures. VLDB J. **17**(5), 1231–1251 (2008)
35. Watve, A., Pramanik, S., Jung, S., Lim, C.Y.: Data-independent vantage point selection for range queries. The Journal of Supercomputing **75**(12), 7952–7978 (2019)
36. Yamagishi, Y., Aoyama, K., Saito, K., Ikeda, T.: Pivot generation algorithm with a complete binary tree for efficient exact similarity search. IEICE Transactions **101-D**(1), 142–151 (2018)