ProMIPS: Efficient High-Dimensional c-Approximate Maximum Inner Product Search with a Lightweight Index

Yang Song
School of Computer Science and Engineering
Northeastern University
Shenyang, China
ysqyw1994@163.com

Yu Gu*
School of Computer Science and Engineering
Northeastern University
Shenyang, China
guyu@mail.neu.edu.cn

Rui Zhang
www.ruizhang.info
rui.zhang@ieee.org

Ge Yu
School of Computer Science and Engineering
Northeastern University
Shenyang, China
yuge@mail.neu.edu.cn

***Abstract*—Due to the wide applications in recommendation systems, multi-class label prediction and deep learning, the Maximum Inner Product (MIP) search problem has received extensive attention in recent years. Faced with large-scale datasets containing high-dimensional feature vectors, the state-of-the-art LSH-based methods usually require a large number of hash tables or long hash codes to ensure the searching quality, which takes up lots of index space and causes excessive disk page accesses. In this paper, we relax the guarantee of accuracy for efficiency and propose an efficient method for c-Approximate Maximum Inner Product (c-AMIP) search with a lightweight iDistance index. We project high-dimensional points to low-dimensional ones via 2-stable random projections and derive probability-guaranteed searching conditions, by which the c-AMIP results can be guaranteed in accuracy with arbitrary probabilities. To further improve the efficiency, we propose Quick-Probe for quickly determining the searching bound satisfying the derived condition in advance, avoiding the inefficient incremental searching process. Extensive experimental evaluations on four real datasets demonstrate that our method requires less pre-processing cost including index size and pre-processing time. In addition, compared to the state-of-the-art benchmark methods, it provides superior results on searching quality in terms of overall ratio and recall, and efficiency in terms of page access and running time.**

***Index Terms*—Maximum Inner Product Search, Probability-Guaranteed, Lightweight Index**

I. Introduction

Given a dataset D of n data points and a query point q in d-dimensional space R^d, a Maximum Inner Product (MIP) search returns the point $o^* \in D$ maximizing the inner product with q. Mathematically, it is represented as $o^* = \arg\max_{o \in D} \langle o, q \rangle$. The so-called MIP search is a fundamental problem and it has been widely applied in various domain areas, such as matrix factorization based recommendation systems [2], [5], [22], [26], multi-class label prediction [10] and deep learning [37]. Typically, in matrix factorization based recommendation systems, the vectors q and o are viewed as latent features for a user and a product, respectively. The inner product between q and o reflects the user's interest in the product. Therefore, MIP search is an important concern in these recommendation systems to recommend popular products to users.

The phenomenon of the "Dimensionality Curse" makes exact MIP search in high-dimensional space very expensive. Therefore, many researchers set their sights on the approximate version of the MIP search problem [1], [2], [15], [17], [30], [34], [35], [41], [44], which is called c-Approximate MIP (c-AMIP) search problem. Mathematically, given an approximation ratio c ($0 < c < 1$) and a query point q, c-AMIP search returns a point $o \in D$ such that $\langle o, q \rangle \geq c\langle o^*, q \rangle$, where o^* is the exact MIP point of q. In this way, a good accuracy-efficiency trade-off can be provided where the efficiency can be improved significantly while only a small amount of errors occur.

At present, the state-of-the-art methods for c-AMIP search are transformation-based. In these methods, a MIP search is converted into a Nearest Neighbor (NN) search or a Maximum Cosine-similarity (MC) search by transforming the given data points and the query point asymmetrically or symmetrically, and employ Locality-Sensitive Hashing (LSH) to solve the NN or MC search problem. These LSH-based methods improve the searching efficiency, but to achieve satisfactory accuracy, they require more hash vectors to project high-dimensional points onto more hash values, indexed by heavyweight structures in terms of massive hash tables. These heavyweight structures require more maintenance overhead, which increases linearly as the number of hash tables increases. Especially in commonly used mobile devices or IoT devices, a huge amount of data

* Corresponding author (email: guyu@mail.neu.edu.cn)

will be frequently inserted or deleted in a short time, where the heavyweight index requiring more maintenance overhead may cause delays. Besides, hundreds or thousands of hash tables may also lead to more disk page accesses which degrades the efficiency when storing data points on disks.

Motivated by these existing restrictions, we attempt to design an efficient method for c-AMIP search with a lightweight index. A recent method, SRS [38], which can be considered as a special version of LSH technique, projects high-dimensional points onto low-dimensional ones via 2-stable random projections to reduce high-dimensional c-ANN search to low-dimensional NN search, and perform the low-dimensional search through a lightweight index in terms of R-tree. Compared to the standard LSH, SRS can directly project high-dimensional points onto lower-dimensional ones with fewer projections, avoiding the heavyweight index. Although it's designed for Euclidean distance, it presents a new angle to solve c-AMIP search problem since the Euclidean distance between two points can be computed by their inner product and 2-norms. Even though, it's still challenging to follow the direction of SRS to solve c-AMIP search problem. Since inner product isn't a metric measurement, some basic necessary properties such as non-negativity and triangle inequality are not satisfied. Without these properties, we can't derive the probability-guaranteed searching conditions for c-AMIP search directly like SRS.

Inspired by SRS, we also project high-dimensional points onto low-dimensional ones via 2-stable random projections. Based on the projection and the properties of inner product, we theoretically derive two conditions specifically for c-AMIP search. According to the conditions, we perform an incremental NN search in low-dimensional space to collect the candidate points until a point satisfying either of the conditions is searched. And the required c-AMIP point is guaranteed to appear among these candidate points with the given probability. However, during the incremental NN search, every time a point is returned, it is required to determine whether it satisfies the condition, which is a time-consuming procedure. To avoid the procedure, we come up with a quick method named Quick-Probe for directly locating the point satisfying the searching condition to determine the searching range, which enables us to replace the incremental NN search with a range search without testing each returned NN point. Meanwhile, based on Quick-Probe, we can also compute an optimized projected dimension to pursue a more efficient searching process. With respect to the index used for search, since the optimized dimensions are usually greater than 3, R-tree used in SRS isn't applicable. Hence, in order to search in higher-dimensional space, we adopt iDistance [18], which is an efficient index, and design a new partition pattern for it. Compared to LSH-based methods, iDistance is a typical lightweight index, which only requires a single B+-tree to orderly organize points on disks, rather than a large number of hash tables or long hash codes.

As can be seen from the above descriptions, we propose an efficient method for the probability-guaranteed high-dimensional c-AMIP search with a lightweight index. Our contributions are summarized as follows:

- We employ 2-stable random projections to project high-dimensional points onto low-dimensional points and theoretically derive two searching conditions for c-AMIP search. Relying on the conditions, the c-AMIP result can be guaranteed in accuracy with arbitrary probabilities.
- Quick-Probe is proposed for quickly locating the point to determine the searching range, which avoids testing each returned point repeatedly to accelerate the searching process. Besides, an optimized projected dimension can be computed based on Quick-Probe.
- Extensive experimental evaluations on four real datasets show that our method occupies a smaller index size and requires less pre-processing time compared to benchmark methods. Furthermore, our method is also superior in accuracy measured by overall ratio and recall, and efficiency measured by page access and running time.

The rest of the paper is structured as follows. Section II presents the preliminaries. We introduce the overall framework in Section III. The searching conditions are presented in Section IV. We propose Quick-Probe in Section V. In Section VI, we describe the indexing technique. The time and space complexities are theoretically analyzed in Section VII. Experimental evaluations are discussed in Section VIII. The related works are introduced in Section IX. Finally, we conclude our work in Section X.

II. Preliminaries

A. Problem Definition

Given a dataset D containing n data points in a d-dimensional space R^d, the inner product $\langle o, q\rangle$ between two points $o = (o_1, o_2, ..., o_d)$ and $q = (q_1, q_2, ..., q_d)$ can be computed as $\langle o, q\rangle = \Sigma_{i=1}^{d} o_i q_i$. Inner product is widely used in real applications where the MIP search problem plays an important role. For example, in recommender systems, o and q are used as a user vector and an item vector, respectively. A higher inner product between o and q indicates that the item better suits the user's preference [2].

In this paper, to handle the high-dimensional cases, we allow a trade-off between accuracy and efficiency, and focus on c-AMIP search problem formally defined as follows:

Definition 1 (c-AMIP search problem)**.** *Given a query point $q \in R^d$ and an approximation ratio c ($0 < c < 1$), c-AMIP search is to find a point $o \in D$ such that $\langle o, q\rangle \geq c\langle o^*, q\rangle$, where o^* is the q's exact MIP point in D.*

Similarly, c-k-AMIP search is to find k points $o_i \in D$ ($1 \leq i \leq k$) such that $\langle o_i, q\rangle \geq c\langle o_i^*, q\rangle$, where o_i^* is the i^{th} exact MIP point of q in D.

B. 2-Stable Random Projection

Definition 2 (2-Stable Random Projections)**.** *Given a d-dimensional point o, which can be considered as a vector $\overrightarrow{o}$, and a d-dimensional vector $\overrightarrow{v}$, whose entries are i.i.d. random*

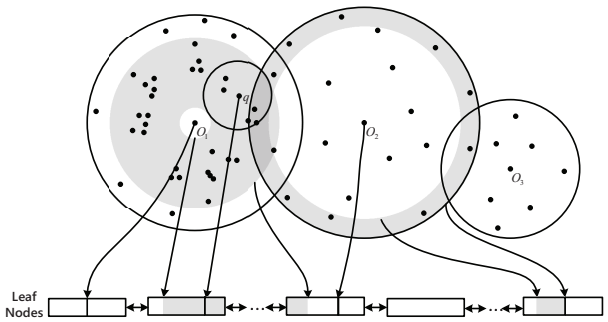


Fig. 1. IDistance

variables following the standard normal distribution $N(0,1)$, 2-Stable Random Projections is to compute $f(o) = \overrightarrow{v} \cdot \overrightarrow{o}$.

Based on 2-stable random projections, we can obtain the following Lemma [31].

Lemma 1. *For any $o_1, o_2 \in R^d$, $f(o_1) - f(o_2)$ follows the normal distribution $N(0, dis^2(o_1, o_2))$.*

In our method, the projected dimension of each point is m. Therefore, we perform m 2-stable random projections to obtain m-dimensional projected points.

C. IDistance

IDistance is an efficient index based on B+-tree for the exact similarity search [18], which is illustrated in Fig. 1. In iDistance, the whole indexing space is divided into several partitions centered at their reference points. In these partitions, points are transformed into a single dimensional value based on their distances to their corresponding reference points and these values are indexed by a B+-tree. Based on the B+-tree, similarity search can be performed. For example, in Fig. 1, given a query point and a searching radius, the grey area in the B+-tree will be searched, so that the points in the gray areas of the space are fetched for determining the final searching results. In this paper, we utilize iDistance as the index to accelerate the searching process.

We summarize the frequently-used symbols in Table I.

III. Overall Framework

In this paper, to solve the probability-guaranteed c-AMIP search problem in high-dimensional space, we project high-dimensional points onto low-dimensional ones via 2-stable random projections. Since the ratio of points' Euclidean distance in high-dimensional space and low-dimensional space follows the chi-square distribution, and points' Euclidean distance is related to their inner product, we can derive two searching conditions. Based on these conditions, we perform incremental NN search in low-dimensional space for the probability-guaranteed c-AMIP point. In detail, every time a point is returned, we test if the point satisfies either of the conditions to determine whether to terminate the incremental NN search. If satisfied, the c-AMIP point exists in the searched points with the given probability at least. The searching conditions are elaborated in Section IV.

TABLE I
Frequently used symbols

Symbol	Explanation
D	dataset
n	number of data points
o, q	data point, query point
$P(o)$, $P(q)$	projected data point, projected query point
d	original dimensionality of each point
m	projected dimensionality of each point
o^*, o_i^*	the MIP point, the i-th MIP point of q
$N(a,b)$	the normal distribution with mean a and variance b
$dis(o_1,o_2)$	the Euclidean distance between o_1 and o_2
$\|o\|$	the norm of point o
o_M	the point with the maximum norm in the original space
$\langle o_1,o_2\rangle$	the inner product between o_1 and o_2
$\chi^2(m)$	the chi-square distribution with m degrees of freedom
$\Psi_m(x)$	cumulative distribution function of $\chi^2(m)$
$\Psi_m^{-1}(p)$	inverse function of $\Psi_m(x)$
k	number of the returned points
c	approximation ratio
p	guaranteed probability

However, it is time-consuming to perform the incremental NN search and test each returned point one by one. To avoid it, we propose a method to quickly locate the point satisfying the searching condition, called Quick-Probe. The method quickly locates the point through binary transformation and data norm's properties. In this way, the searching range is directly determined by the located point and we can perform range search instead of the incremental NN search elaborated in Section IV, to collect the candidate points. Benefiting from Quick-Probe, we no longer do any incremental NN search and the time-consuming process of testing the returned point one by one can be avoided. Quick-Probe is elaborated in Section V. In addition, we adopt iDistance as the index and design a new partition pattern for it for performing searching tasks more efficiently in low-dimensional space, which is elaborated in Section VI.

Based on the searching conditions and Quick-Probe, our method is described in two parts including the pre-process and the searching process. In the pre-process, the original high-dimensional points are projected onto projected low-dimensional ones. In the low-dimensional space, the index structure is constructed for performing the searching tasks and the low-dimensional points and their corresponding high-dimensional ones are organized on disks. In addition, the projected points are also converted into binary codes and each point's norms are computed, for determining the searching range according to Quick-Probe. In the searching process, Quick-Probe is applied to find the point satisfying the condition and determine the searching range, by which the range search is performed in the projected space to find the candidate points. These candidate points are verified using their inner products in the original space for returning the c-AMIP search results. To clearly summarize our method's overall framework, we give Fig. 2 to describe it.

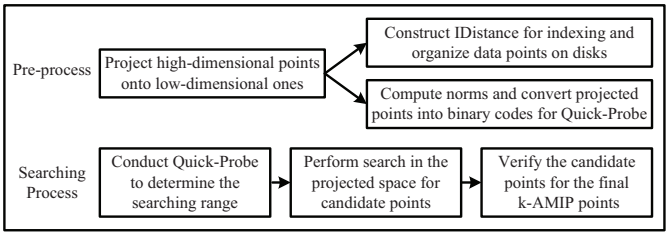


Fig. 2. Overall Framework

IV. Searching Conditions

Our method aims to guarantee the c-AMIP search in accuracy with arbitrary probabilities by proposing two searching conditions. In this section, we will introduce the conditions and prove their validity.

A. Condition A

As stated above, we perform incremental NN search in the projected space. During the searching process, we fetch every returned point as the candidate points. If the current returned $P(q)$'s i-th NN point $P(o_i)$ satisfies:

$$\|o_M\|^2 + \|q\|^2 - \frac{2\langle o_i, q\rangle}{c} \leq 0, \tag{1}$$

a c-AMIP point must exist among these candidate points, and the searching process can be terminated, where o and q are the corresponding original points of $P(o)$ and $P(q)$, o_M is the point with the maximum norm in the original space. Formula 1 is considered as Condition A and the following Theorem 1 proves its validity.

Theorem 1. *If the current returned NN point satisfies Formula 1, a c-AMIP point must exist among the points having been returned.*

Proof. We assume that o^* is the exact MIP point of the query point q. If $\|o_M\|^2 + \|q\|^2 - \frac{2\langle o_i,q\rangle}{c} \leq 0$, since o_M is the point with the maximum norm, we have $\|o^*\|^2 + \|q\|^2 - \frac{2\langle o_i,q\rangle}{c} \leq 0$.

Since $\|o^*\|^2 + \|q\|^2 - 2\langle o^*, q\rangle \geq 0$, we have $\langle o_i, q\rangle \geq c\langle o^*, q\rangle$. Therefore, when $\|o_M\|^2 + \|q\|^2 - \frac{2\langle o_i,q\rangle}{c} \leq 0$, a c-AMIP point must have been accessed when o_i is searched. □

B. Condition B

During the incremental NN search in the projected space, if the current returned NN point doesn't satisfy Condition A, $\|o_M\|^2 + \|q\|^2 - \frac{2\langle o_i,q\rangle}{c} > 0$ is true. Based on it, we turn to test the returned NN point by the following Formula 2. If $P(q)$'s i-th NN point $P(o_i)$ satisfies:

$$\Psi_m\left(\frac{dis^2(P(o_i), P(q))}{\|o_M\|^2 + \|q\|^2 - \frac{2\langle o_{max},q\rangle}{c}}\right) \geq p, \tag{2}$$

a c-AMIP point must exist among these candidate points with the given probability p at least, and the searching process can be terminated, where o_{max} is the point with the maximum inner product to q among all the candidate points having been returned so far. Formula 2 is considered as Condition B and the following Theorem 2 proves its validity.

Before proving Theorem 2, we firstly give the following Lemma 2 as the preparations.

Lemma 2. $\frac{dis^2(P(o),P(q))}{\|o\|^2+\|q\|^2-2\langle o,q\rangle}$ *follows the* $\chi^2(m)$ *distribution.*

Proof. According to Definition 2, we select m d-dimensional vectors whose entries are i.i.d random variables following $N(0,1)$ for performing m 2-stable random projections to get m-dimensional projected points. The m-dimensional projected points are denoted as $P(o) = (f_1(o), f_2(o), ..., f_m(o))$ and $P(q) = (f_1(q), f_2(q), ..., f_m(q))$.

According to Lemma 1, we have $\frac{f_i(o)-f_i(q)}{dis(o,q)} \sim N(0,1)$ $(1 \leq i \leq m)$.

Therefore, we can obtain $\sum_{i=1}^{m} \left(\frac{f_i(o)-f_i(q)}{dis(o,q)}\right)^2 \sim \chi^2(m)$.

Since $dis^2(P(o), P(q)) = \sum_{i=1}^{m}(f_i(o) - f_i(q))^2$ and $dis^2(o,q) = \|o\|^2 + \|q\|^2 - 2\langle o, q\rangle$, the lemma can be proved. □

Theorem 2. *If the current returned NN point satisfies Formula 2, a c-AMIP point must exist among the points having been returned with probability p at least.*

Proof. Assume that o^* is the exact MIP point, we consider the relationship between $dis(P(o^*), P(q))$ and $dis(P(o_i), P(q))$. We discuss their relationship in two cases:

- **C1:** $dis(P(o^*), P(q)) \leq dis(P(o_i), P(q))$.
  In this case, since we perform the incremental NN search, o^* must have been accessed when o_i is searched.
- **C2:** $dis(P(o^*), P(q)) > dis(P(o_i), P(q))$.
  In this case, our method may produce incorrect results if none of the c-AMIP points has appeared so far, which can also be represented as $\langle o_{max}, q\rangle < c \cdot \langle o^*, q\rangle$. However, we can prove that the probability of such incorrect case is less than $1 - p$.
  According to Lemma 2, for any $x > 0$ and o, we have

$$Pr[dis(P(o), P(q)) \leq x] = \Psi_m\left(\frac{x^2}{\|o\|^2 + \|q\|^2 - 2\langle o, q\rangle}\right).$$

Based on it, we have:

$$Pr[dis(P(o^*), P(q)) > dis(P(o_i), P(q))]$$
$$= 1 - \Psi_m\left(\frac{dis^2(P(o_i), P(q))}{\|o^*\|^2 + \|q\|^2 - 2\langle o^*, q\rangle}\right).$$

Since $\langle o_{max}, q\rangle < c \cdot \langle o^*, q\rangle$, we can derive

$$\Psi_m\left(\frac{dis^2(P(o_i), P(q))}{\|o^*\|^2 + \|q\|^2 - 2\langle o^*, q\rangle}\right)$$
$$> \Psi_m\left(\frac{dis^2(P(o_i), P(q))}{\|o^*\|^2 + \|q\|^2 - \frac{2\langle o_{max},q\rangle}{c}}\right).$$

Since o_M is the point with the maximum norm, we have

$$\Psi_m\left(\frac{dis^2(P(o_i), P(q))}{\|o^*\|^2 + \|q\|^2 - \frac{2\langle o_{max},q\rangle}{c}}\right)$$
$$> \Psi_m\left(\frac{dis^2(P(o_i), P(q))}{\|o_M\|^2 + \|q\|^2 - \frac{2\langle o_{max},q\rangle}{c}}\right).$$

Therefore, if o_i satisfies $\Psi_m(\frac{dis^2(P(o_i),P(q))}{\|o_M\|^2+\|q\|^2-\frac{2\langle o_{max},q\rangle}{c}}) \geq p$, we have $Pr[dis(P(o^*),P(q)) > dis(P(o_i),P(q))] \leq 1-p$.

□

Algorithm 1 gives the pseudo-code of the searching process. The algorithm can also be extended to solve the c-k-MIP search problem by some simple changes. In Condition A, it's required to test the current k-th MIP point o^k_{max}. Similarly, we should use o^k_{max} in Condition B instead of o_{max}.

Algorithm 1: MIP-Search-I (D, n, c, p, q)

```
o_max ← Null;
i ← 1;
// Perform incremental NN search
while i ≤ n do
    P(o_i) ← P(q)'s i-NN point;
    if ⟨o_max, q⟩ ≤ ⟨o_i, q⟩ then
        o_max ← o_i; // Update MIP point
    if Condition A then
        return o_max;
    else if Condition B then
        return o_max;
    i ← i + 1;
return o_max;
```

V. Quick-Probe

As can be seen from Algorithm 1, we have to perform the incremental NN search to find the point satisfying the searching condition. Whenever a point is returned, it's required to test it using Condition A or Condition B. Especially in Condition B, Euclidean distance in the projected space is computed, which is time-consuming when the projected dimension is high. Besides, it also incurs extra page accesses when fetching points from disks. Therefore, we attempt to avoid testing the points one by one.

A. Method

For the purpose, we introduce a method named Quick-Probe, by which we can quickly locate a point satisfying Condition B as much as possible. The distance between the point and the query in the projected space is used as an estimation of the searching range. It enables us to perform range search instead of incremental NN search to find the candidates points within the searching range.

Nevertheless, it's hard to determine the bound of $\frac{dis^2(P(o),P(q))}{\|o_M\|^2+\|q\|^2-\frac{2\langle o_{max},q\rangle}{c}}$ in Condition B and locate a point satisfying the condition. But we observe that the point satisfying Formula 3 is more likely to satisfy Condition B and the determined searching range can infinitely approach the range determined by Condition B. So we turn our attention to determine the bound of $\frac{dis^2(P(o),P(q))}{c\times dis^2(o,q)}$, and locate a point satisfying Formula 3.

$$\Psi_m(\frac{dis^2(P(o),P(q))}{c\times dis^2(o,q)}) \geq p \tag{3}$$

The bound is determined through binary transformation and data norm's properties. In detail, we transform each projected point into a binary code $c(o) = (c_1(o), c_2(o), ..., c_m(o))$, where $c_i(o) = 1$ if $P_i(o)$ is non-negative and $c_i(o) = 0$ otherwise ($i = 1, 2, ..., m$). According to Theorem 3, we can derive the lower bound of $dis(P(o), P(q))$. The upper bound of $dis(o, q)$ can be derived through Theorem 4 using the property of data norm. By Theorem 3 and Theorem 4, a lower bound of $\frac{dis^2(P(o),P(q))}{c\times dis^2(o,q)}$ can be computed. If the lower bound referring to a point o is greater than $\Psi^{-1}_m(p)$, Formula 3 must be satisfied. Therefore, we can use $dis(P(o), P(q))$ as the searching range in the projected space. The process of finding o is described as below.

Theorem 3. *The lower bound of the Euclidean distance between $P(o)$ and $P(q)$ is $\frac{1}{\sqrt{m}}\sum_{i=1}^{m}(c_i(o)\oplus c_i(q))\times|P_i(q)|$.*

Proof. For any m-dimensional vector x, it holds that $\sqrt{m}\|x\|_2 \geq \|x\|_1$ [23], [47]. Therefore, we have $\|P(o) - P(q)\|_2 \geq \frac{1}{\sqrt{m}}\|P(o)-P(q)\|_1$. When $c_i(o) = c_i(q)$, $P_i(o)$ and $P_i(q)$ have the same sign and $c_i(o) \oplus c_i(q) = 0$ holds. Since $|P_i(o)-P_i(q)| \geq 0$, we have $|P_i(o)-P_i(q)| \geq (c_i(o)\oplus c_i(q))\times|P_i(q)|$. When $c_i(o) \neq c_i(q)$, $P_i(o)$ and $P_i(q)$ have different signs and $c_i(o) \oplus c_i(q) = 1$ holds. Therefore, we also have $|P_i(o)-P_i(q)| = |P_i(o)|+|P_i(q)| \geq (c_i(o)\oplus c_i(q))\times|P_i(q)|$. Therefore, it holds that $|P_i(o) - P_i(q)| \geq (c_i(o) \oplus c_i(q)) \times |P_i(q)|$ and we can obtain that

$$\|P(o) - P(q)\|_2 \geq \frac{1}{\sqrt{m}}\sum_{i=1}^{m}(c_i(o) \oplus c_i(q)) \times |P_i(q)| \tag{4}$$

□

Theorem 4. *The upper bound of the Euclidean distance between o and q is $\sum_{i=1}^{m}|o_i| + \sum_{i=1}^{m}|q_i|$.*

Proof. According to the property of vector norm and absolute value equality [47], we can simply derive:

$$\|o-q\|_2 \leq \|o-q\|_1 \leq \sum_{i=1}^{m}|o_i|+\sum_{i=1}^{m}|q_i| = \|o\|_1+\|q\|_1. \tag{5}$$

□

In the pre-process, the projected points with the same binary code will be divided into the same group, and the 1-norms of their original points are computed and sorted. In the searching process, the lower bounds of Euclidean distance between each group and the query point are computed through Formula 4. We search the groups in ascending order of their lower bounds. In each group, its lower bound is denoted as LB and we fetch the point o whose $\|o\|_1$ is the smallest among the points in the group to find the largest value of $\frac{LB^2}{c\times(\|o\|_1+\|q\|_1)^2}$. Then we test whether it satisfies $\Psi_m(\frac{LB^2}{c\times(\|o\|_1+\|q\|_1)^2}) \geq p$, which is

denoted as Test A. If Test A is satisfied, we fetch the point to determine the searching range. If not satisfied, we record the point's value of $\frac{LB^2}{c\times(\|o\|_1+\|q\|_1)^2}$ and continue to search in the next group until the point is found. If there is no point satisfying it in all groups, we fetch the point with the largest recorded value of $\frac{LB^2}{c\times(\|o\|_1+\|q\|_1)^2}$ as the result. The following Algorithm 2 describes the whole process. In Algorithm 2, $G = \{G_1, G_2, ..., G_K\}$ are the input set of groups with the same binary codes. In each group, the points o are sorted in the ascending order of $\|o\|_1$.

Algorithm 2: Quick-Probe (G, c, p, q)

```
Compute each group G_i's lower bound LB_i;
{GS_1, GS_2, ..., GS_K} ← the sorted groups in the ascending order of the lower bounds;
point ← Null;
value ← 0;
for i = 1 to K do
    // Test A
    if Ψ_m(LB_i^2 / (c×(‖o_i1‖_1+‖q‖_1)^2)) ≥ p then
        return o_i1;
    // Update the point with the largest value
    if LB_i^2 / (c×(‖o_i1‖_1+‖q‖_1)^2) ≥ value then
        value ← LB_i^2 / (c×(‖o_i1‖_1+‖q‖_1)^2);
        point ← o_i1;
return point;
```

Combining Quick-Probe and the aforementioned Condition A and Condition B, the searching process is described in Algorithm 3. Quick-Probe is applied to find the point o as the input to determine the searching range in the projected space. During the range search in the projected space, when a point is returned, its inner product to the query point is recorded for the final verification. Besides, Condition A is also tested to determine whether to terminate the searching process (Unlike Condition B, Condition A doesn't require too much computation).

Because the searching range obtained by Quick-Probe is an estimated value, the obtained point may not satisfy Condition B completely, which indicates that the searching range may not completely guarantee c-AMIP point with the given probability p. Faced with this problem, we compensate it by expanding the searching range to ensure the probability-guaranteed c-AMIP result. If the entire range search has been performed, the recorded maximum inner product is brought into Condition B to test whether the result satisfies the condition. If satisfied, terminate the searching process and return the result. If not satisfied, according to Formula 2, the searching range will be extended to $r^{'} = \sqrt{\Psi_m^{-1}(p) \times (\|o_M\|^2 + \|q\|^2 - \frac{2\langle o_{max}, q\rangle}{c})}$ as compensation to find the final results. Since the obtained maximum inner product later is greater than or equal to current $\langle o_{max}, q\rangle$, the extended $r^{'}$ is larger than or equal to the actual searching range satisfying the probability-guaranteed requirements.

Algorithm 3: MIP-Search-II (D, c, p, q, o)

```
r ← dis(P(o), P(q)); // Determined searching range
o_max ← Null;
i ← 0;
// Perform range search
while dis(P(o_i), P(q)) < r do
    i ← i + 1;
    o_i ← the original form of P(o_i);
    if ⟨o_i, q⟩ > ⟨o_max, q⟩ then
        o_max ← o_i; // Update MIP point
        if Condition A then
            return o_max;
if Condition B then
    return o_max;
else
    Update the searching range to r';
    while dis(P(o_i), P(q)) < r' do
        i ← i + 1;
        o_i ← the original form of P(o_i);
        if ⟨o_i, q⟩ > ⟨o_max, q⟩ then
            o_max ← o_i;
            if Condition A then
                return o_max;
return o_max;
```

B. Optimized Projected Dimension

In Quick-Probe, the projected points are transformed into binary codes. It indicates that m projected dimensions will bring 2^m binary codes. If assuming that each binary code represents the same number of points, 2^m groups will bring $n/2^m$ points in each group. It can be observed that more projected dimensions may bring more groups, while bring fewer points in each group. If the point can be located by directly searching one group, fewer points in one group will lead to less time consumption. However, more groups also require more time to compute their lower bounds. Therefore, there exists a trade-off and we can derive an optimized projected dimension to improve the efficiency of Quick-Probe.

Binary codes with m bits can divide the whole dataset into up to 2^m groups. The time consumption to compute the groups' lower bounds and find the group with the smallest lower bound is $2^m(m+1)$. We assume that the whole dataset can be equally divided and the point satisfying Formula 3 can be located by searching only one group. Therefore, each group contains $n/2^m$ points and the time consumption of searching the point is $n/2^m$. The total time consumption is $2^m(m+1) + n/2^m$. We set the function $f(m) = 2^m(m+1)+n/2^m$. Since the second derivative of $f(m)$ is greater than

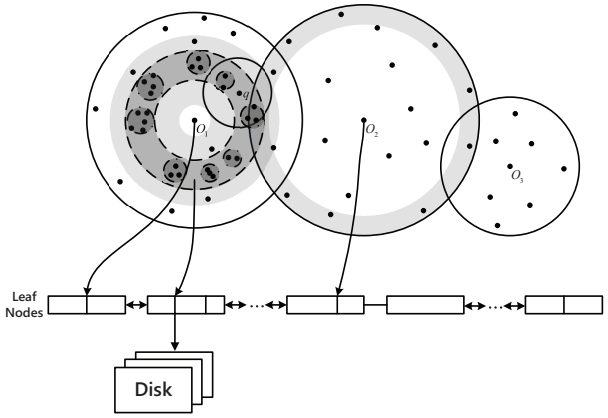


Fig. 3. IDistance with New Partition Pattern

0, $f(m)$ has the minimum value. Our objective is to compute $m = \arg\min f(m)$, which is considered as the optimized projected dimension.

VI. Index Structure

In the standard iDistance shown in Fig. 1, when performing range search, the searching area is much larger than the given searching sphere, which indicates that a large portion of searching area is unnecessary.

Different from the standard iDistance, to avoid much unnecessary searching area, we adopt a different partition pattern as shown in Fig. 3. We use the following Formula 6 to compute each point's index key,

$$I(p) = \lfloor i * C + dis(p, O_i)/\varepsilon \rfloor \tag{6}$$

where ε is a constant determined by the data distribution. In detail, taking a two-dimensional space as an example, we obtain the clusters' radii after the first stage of clustering and compute their average. Then, we make a circle with the average as the radius denoted as r_{avg}, and the value of ε is equal to r_{avg}/N_{key} to divide the circle into N_{key} rings with equal ring widths, which also means that points can be mapped to N_{key} keys. We continue to employ k-means to divide the sets of points in the rings into several sub-partitions, while the clusters' centers and radii are the sub-partitions' pivots and radii, respectively. In the searching process, points can be filtered in sub-partitions by whether they intersect with the given searching sphere. In addition, the points in the same sub-partition can be collectively organized on disks in order, which means that the adjacent points belonging to the same sub-partition are likely to be organized on the same disk, while the adjacent sub-partitions are also likely to be organized on the adjacent disks. It's beneficial to reduce page accesses since points can be read from disks in sub-partitions to avoid random readings. As shown in Fig. 3, an index key indexes a deep grey ring in the partition. The points in this ring are divided into eight sub-partitions. Given a searching sphere centered at the query point, two of the eight sub-partitions intersect with the given sphere and the points in these sub-partitions are selected as the candidate points. In our partition pattern, it's required to select appropriate values of the number of partitions k_p and the number of sub-partitions k_{sp} to ensure that each sub-partition contains a certain number of points to make the filter effective. To the end, we introduce a parameter called selectivity μ. That is, we try to make nearly μn points in each sub-partition by setting appropriate k_p and k_{sp}. We assume that, after the first clustering stage via k_p-means, the number of points in each cluster is the same, which is $\frac{n}{k_p}$. We determine the value of ε in Formula 6 according to the data distribution to control the number of keys in each cluster, and the number of points corresponding to each key is also assumed to be the same. We denote the number of keys in a cluster as N_{key}, thereby the number of points represented by a key is $\frac{n}{k_p * N_{key}}$. Based on the aforementioned assumptions, the number of points in each sub-partition is $\frac{n}{k_p * N_{key} * k_{sp}}$ after clustering by k_{sp}-means. Therefore, the selectivity $\mu = \frac{1}{k_p * N_{key} * k_{sp}}$. In the experimental evaluations, we will give the parameter settings on the testing datasets.

Algorithm 4 introduces the index construction containing the dividing process, computing the index keys and constructing the B+-tree to index these points.

Algorithm 4: Index-Construct(D)

```
Project original dataset D onto projected dataset D_p;
Divide D_p into k_p partitions {P_1, P_2, ..., P_{k_p}};
for i = 1 to k_p do
    for every point p in P_i do
        I(p) = ⌊i * C + dis(p, O_i)/ε⌋; // Formula 6
    Divide points with the same index keys in P_i into
    k_sp sub-partitions;
Construct B+-tree index and organize points on disks;
```

VII. Time and Space Complexities

The time cost of our method consists of five parts. Firstly, according to Section V, we have the computed optimized projected dimension $m = O(\log n)$ and the time cost of locating the point through Quick-Probe is $2^m m + 2^m + \frac{n}{2^m} = O(n \log n)$. Secondly, the time complexity of computing q's projection is $O(d)$. Thirdly, the time cost of locating the partition containing the projected query point and computing the projected query point's key is $k_p m + 1 = O(1)$. Then, since there are $k_p N_{key}$ keys in B+-tree, locating the key in the B+-tree costs $\log(k_p N_{key})$. In the B+-tree, assuming that $\alpha k_p N_{key}$ $(0 < \alpha < 1)$ keys are searched, it costs $\alpha k_p N_{key} \log(k_p N_{key})$. The process of determining whether the searching range intersects with $\alpha k_p N_{key} k_{sp}$ sub-partitions costs $\alpha k_p N_{key} k_{sp} m$. Summing them up, the whole searching process costs $\log k_p N_{key} + \alpha k_p N_{key} \log k_p N_{key} + \alpha k_p N_{key} k_{sp} m = O(\log n)$. Finally, we denote that the filtering rate is β $(0 < \beta < 1)$, which indicates βn are selected as candidate points. Hence computing the inner products for candidate points costs $\beta n d = O(d)$. Therefore, the time complexity of our method is $O(n \log n + d + 1 + \log n + d) = O(d + n \log n)$.

We also analyze the space cost of our proposed method. The space complexity of our method consists of the space complexities of storing n original high-dimensional points and n projected low-dimensional points, which are $O(nd)$ and $nm = O(n \log n)$, respectively. In addition, In Quick-Probe, the space complexity of storing the binary codes and each point o's $\|o\|_1$ are $nm = O(n \log n)$ and $O(n)$, respectively. Thus, the total space cost is $O(nd + n \log n + n \log n + n) = O(nd + n \log n)$.

We also list the time and space complexities of two benchmark methods, H2-ALSH [17] and Norm Ranging-LSH [44] in Table II. From Table II, the time complexity of our method outperforms two benchmark methods. Although the space complexities of three methods are the same, in fact, the projected space in our method is much smaller than the number of hash tables in H2-ALSH or the hash codes' length in Norm Ranging-LSH.

TABLE II
TIME AND SPACE COMPLEXITIES

	Time Complexity	Space Complexity
ProMIPS	$O(d + n \log n)$	$O(nd + n \log n)$
L2-ALSH	$O(d \log n + n \log n)$	$O(nd + n \log n)$
Norm Ranging-LSH	$O(d \log n + n \log n)$	$O(nd + n \log n)$

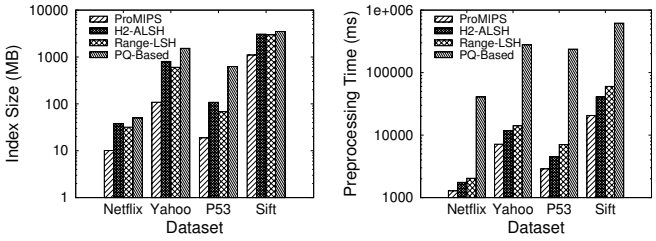
VIII. EXPERIMENTAL EVALUATIONS

A. Experiment Setup

1) Benchmark Methods: We select two state-of-the-art methods with probability guarantee in accuracy, H2-ALSH [17] and Norm Ranging-LSH [44], as two benchmark methods. In addition, to compare our method with the method without probability guarantee in accuracy, we adopt the asymmetric transformation in H2-ALSH to convert MIP search problem into NN search problem, and select the latest product quantization-based NN search technique [19] which performs well in accuracy and efficiency to solve the problem as a benchmark method. In the experiments, our method is denoted as "ProMIPS". Three benchmark methods are denoted as "H2-ALSH", "Range-LSH" and "PQ-Based", respectively. To evaluate the page access, we employ the disk-resident QALSH in the implementation of H2-ALSH. In Range-LSH, we organize the data in each subset sequentially on disks according to the descending order of each subset's maximum norm. In PQ-based method, we organize the data according to each cell's inverted list. All methods are implemented in Java and all experiments are conducted on an ECS with Intel Core Processor (Haswell, no TSX) 2.29GHZ, 48GB main memory, and 512GB hard disk, running under Windows 10. We use the buffering management in the operating system.

2) Datasets and queries: Four real datasets Netflix [3], Yahoo [12], P53[1] and Sift[2] are summarized in Table III. On Netflix and Yahoo, the user vectors and item vectors are generated by PureSVD [6], [17]. For all datasets, 100 points are randomly selected as the query points.

[1]http://archive.ics.uci.edu/ml/datasets/p53+Mutants
[2]http://archive.ics.uci.edu/ml/datasets/SIFT10M



(a) Index Size (b) Pre-processing Time

Fig. 4. Index Size and Pre-processing Time

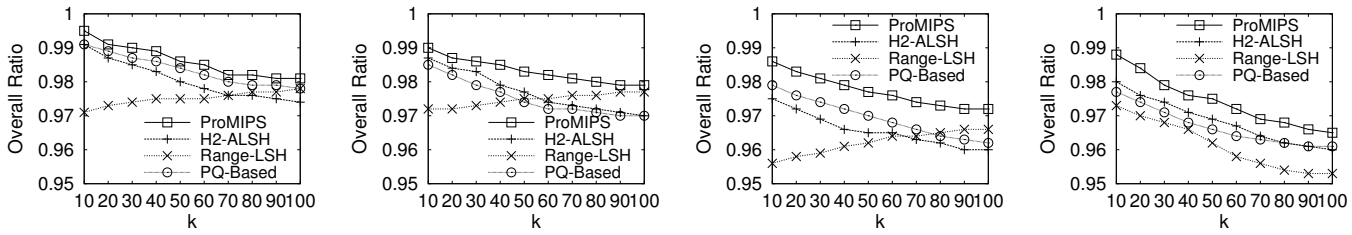
TABLE III
DATASETS

Parameter	n	d	Data Size
Netflix	17770	300	84.2MB
Yahoo	624961	300	2.3GB
P53	31420	5408	1.07GB
Sift	11164866	128	7.3GB

3) Evaluation metrics:

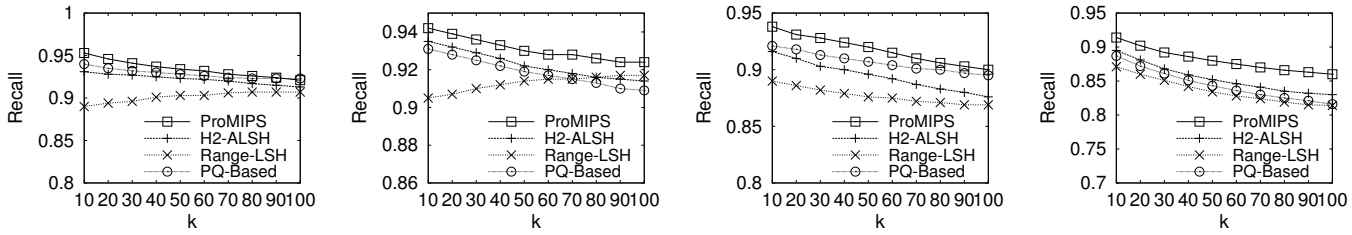
- Index Size. It is defined as the size of each evaluated method's index.
- Pre-processing Time. It is defined as the pre-computation and the index construction time of each evaluated method.
- Overall Ratio. It is defined as: $\frac{1}{k}\sum_{i=1}^{k}\frac{\langle o_i, q\rangle}{\langle o_i^*, q\rangle}$ in c-k-AMIP search problem, where o_i is the i-th returned AMIP point and o_i^* is the exact i-th MIP point of the query point. Intuitively, the overall ratio is between 0 and 1 and a larger overall ratio indicates a higher accuracy.
- Recall. It is defined as: t/k in c-k-AMIP search problem. t is the number of the returned AMIP points which are actually in the set of exact k-MIP points. A larger recall means more exact k-MIP points are returned, indicating a higher accuracy.
- Page Access. It is defined as the number of disk pages to be accessed during the searching process.
- CPU Time. It is defined as the CPU time for performing a c-k-AMIP search.
- Total Time. It is defined as the running time for reading data from disks and performing a c-k-AMIP search.

4) Parameter Settings: The performance of ProMIPS is evaluated under different parameter settings. According to Section V-B, the projected dimensions m are set to 6 on Netflix and P53. On Yahoo and Sift, the projected dimensions m are set to 8 and 10, respectively. Through experiments, we find that it doesn't have much effect on efficiency when the values of k_p, N_{key} and k_{sp} are set in the ranges of 5-15, 20-50 and 5-25, respectively. Therefore, we set $k_p = 5$, $N_{key} = 40$ and $k_{sp} = 10$ as the default values for all testing datasets. The values of ε on Netflix, Yahoo, P53 and Sift are 0.02, 40, 0.1 and 250, respectively. The default approximation ratio c is set to 0.9 and we vary c to 0.7, 0.8 and 0.9 to evaluate its impact on ProMIPS's searching accuracy and efficiency. The default guaranteed probability p is set to 0.5 and we vary p to 0.3, 0.5, 0.7 and 0.9 to evaluate its impact on ProMIPS's searching accuracy and efficiency. In H2-ALSH, the value of



(a) Netflix (b) Yahoo (c) P53 (d) Sift

Fig. 5. Overall Ratio



(a) Netflix (b) Yahoo (c) P53 (d) Sift

Fig. 6. Recall

c_0 is fixed to 2.0 [17]. In Range-LSH, we divide the datasets into 32 partitions under a code length of 16 [44]. In PQ-based method, the whole space is divided into 16 subspaces. The number of centroids in each subspace is 256 and the number of searched nearest cells is 16 in the searching process [19]. The required k is set from 10 to 100 in all testing cases. When evaluating the page access, the disk page's size is set to 4KB on Netflix, Yahoo and Sift. On P53, the disk page's size is set to 64KB due to its high dimension.

B. Pre-processing Time and Index Size

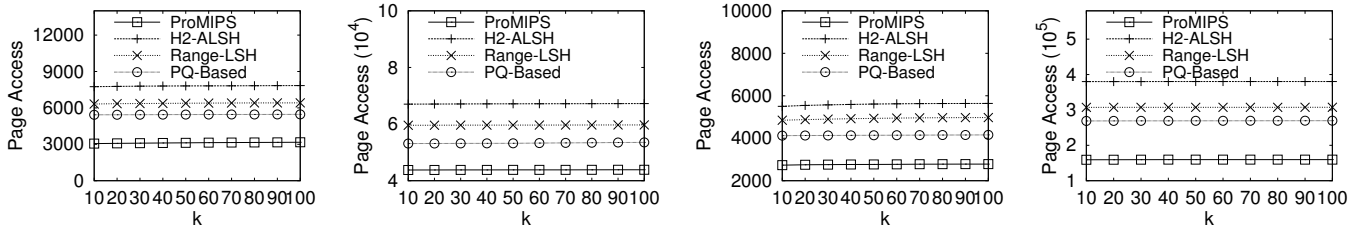
The pre-process of our method contains generating each point's projection, computing each point's norms and converting the projected points into binary codes for Quick-Probe, and constructing the index. The pre-processes of H2-ALSH and Range-LSH contain constructing multiple hash tables and transforming data points. The pre-process of PQ-based method contains constructing quantizers with multiple cells, computing the residuals, training for the rotation matrices and maintaining each cell's corresponding inverted list. The index size and the pre-processing time of four evaluated methods are illustrated in Figs. 4(a) and (b), respectively. On all datasets, the index size and the pre-processing time of ProMIPS beat other methods. This is because H2-ALSH and Range-LSH construct multiple hash tables and PQ-based method stores many local rotation matrices and cells incurring large space overheads, while ProMIPS constructs iDistance with a single B+-tree. In ProMIPS, although the two-stage dividing process in the index construction is time-consuming, only one B+-tree is required, which reduces the time overhead. Compared to H2-ALSH, Range-LSH uses more hash vectors to generate each point's bit vector and their proposed single-table multi-probe strategy requires more time to rank the hash tables. Therefore, it takes more pre-processing time in Range-LSH. Nevertheless, since the points' bit vectors take up less space, the index size of Range-LSH is smaller than that of H2-ALSH. Since the training process to obtain the optimized rotation matrices is costly and it's space-consuming to store rotation matrices and cells, the performances of PQ-based method on the index size and the pre-processing time are the worst.

C. Overall Ratio and Recall

Fig. 5 reports the results on overall ratio when varying the value of k from 10 to 100. Four methods perform well on all datasets while the values of overall ratio are over 0.95. From the experimental results, the overall ratio of ProMIPS is higher than those of the other three methods by up to 3%. Meanwhile, the overall ratio of ProMIPS is larger than the default approximation ratio when varying k. The phenomenon demonstrates that ProMIPS can guarantee c-k-AMIP search in accuracy. In addition, we test the recall of four methods on four datasets and the results are shown in Fig. 6. In Fig. 6, the similar trends are observed. Both of the experimental results on the overall ratio and recall illustrate that ProMIPS can provide c-k-AMIP point with a high accuracy.

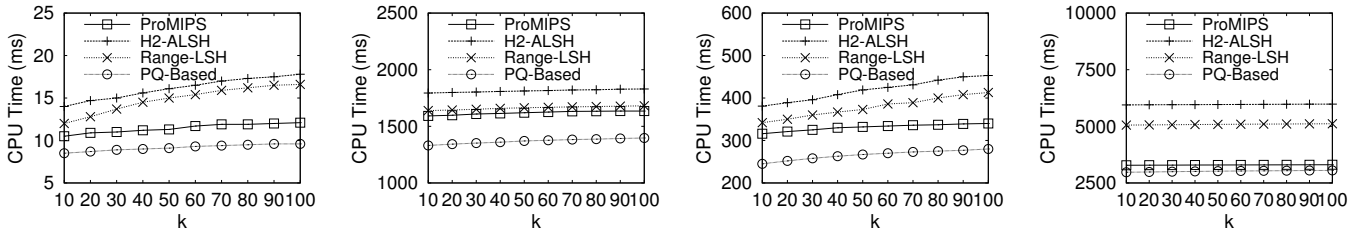
D. Page Access

We evaluate the page access of four methods by varying k from 10 to 100 as well and show the experimental results in Fig. 7. In Fig. 7, ProMIPS outperforms the other three methods in all testing cases as k increases. It is because iDistance used in ProMIPS only requires one B+-tree as index, while both H2-ALSH and Range-LSH require more hash tables to ensure the accuracy, leading to more candidate points. In addition, the searching conditions in our method enable us to verify fewer candidate points to obtain satisfactory results. Meanwhile, benefiting from Quick-Probe, we can avoid reading the projected points from disks and testing them one by



(a) Netflix (b) Yahoo (c) P53 (d) Sift

Fig. 7. Page Access



(a) Netflix (b) Yahoo (c) P53 (d) Sift

Fig. 8. CPU's Time

one. Besides, using the iDistance with our proposed partition pattern, the points can be collectively organized on disks in sub-partitions. The points can be read from disks sequentially to reduce page accesses. The experimental results on four datasets also illustrate that ProMIPS provides good efficiency in all data dimensions and at all data scales, which reflects our method's high scalability. In PQ-based method, we have to check many PQ-encoded residuals, which incurs more page accesses. Compared to H2-ALSH, Range-LSH performs better in terms of the page access because fewer hash buckets are probed during the searching process in Range-LSH benefitting from their proposed single-table multi-probe strategy, which brings fewer selected candidate points.

E. CPU Time and Total Time

In Fig. 8, we evaluate the CPU time to test the efficiency of four methods. From the experimental results, the performance of ProMIPS on CPU time is comparable. PQ-based method performs the best on CPU time because the distances between PQ-encoded residuals are pre-computed in the pre-process. Compared to H2-ALSH and Range-LSH, ProMIPS requires fewer candidate points to guarantee the accuracy benefiting from the derived effective searching conditions. In addition, the process of Quick-Probe determines a certain searching range in the projected space, which avoids testing each returned point to reduce the CPU time. With respect to H2-ALSH, it's more complex to count points' frequencies to fetch the candidate points in more hash tables compared to directly scanning points in hash tables for candidate points in Range-LSH. Therefore, it takes more CPU's running time in H2-ALSH.

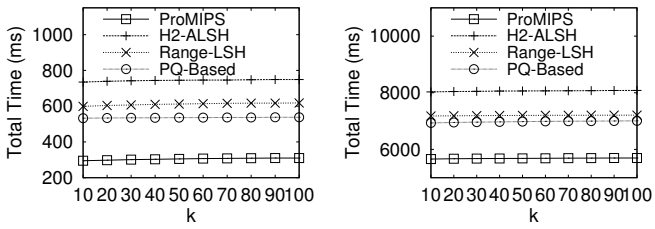
Furthermore, we also evaluate the total time to verify the efficiency. Due to the space limits, we only show the experimental results on Netflix and Yahoo in Fig. 9. In the whole searching process, a large portion of the time consumption comes from reading data from disks. Since ProMIPS performs the best on page access, it obtains the superior performance on total time.

F. Impact of c and p

Since ProMIPS guarantees c-k-MIP search in accuracy, we vary the approximation ratio c and the guaranteed probability p to evaluate how the performances of ProMIPS vary with c and p. We test overall ratio, recall, page access, CPU time and total time on four datasets. Due to the space limits, we only show the results on the overall ratio and page access to demonstrate our method's accuracy and efficiency. The recall and running time show similar trends with the overall ratio and page access, respectively. The experimental results are reported in Fig. 10 and Fig. 11.

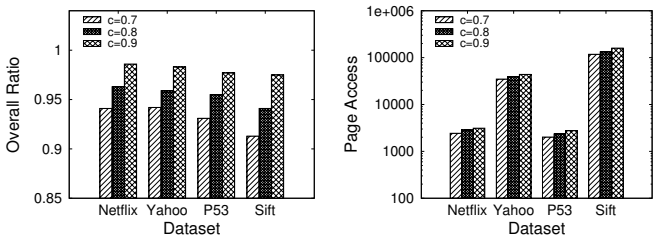
In Fig. 10, the overall ratio decreases as c decreases. This is because a smaller c leads to a smaller range according to the searching conditions and fewer candidate points are selected, which leads to a lower accuracy. Although the overall ratio decreases, it's still larger than the given approximation ratio c. It demonstrates that ProMIPS can guarantee c-k-MIP search in accuracy. In Fig. 10, a larger overall ratio leads to more page accesses, which shows that ProMIPS enjoys a better trade-off between the accuracy and efficiency.

In Fig. 11, it shows that a higher probability leads to a higher overall ratio. This is because a higher p leads to a larger searching range containing more candidate points. But more candidate points also incur more page accesses. Although we can obtain a higher overall ratio when $p = 0.9$, it incurs much more page accesses at the same time. It demonstrates that the increasing rate of accuracy is lower than the decreasing rate of efficiency as p increases.



(a) Netflix (b) Yahoo

Fig. 9. Total Time



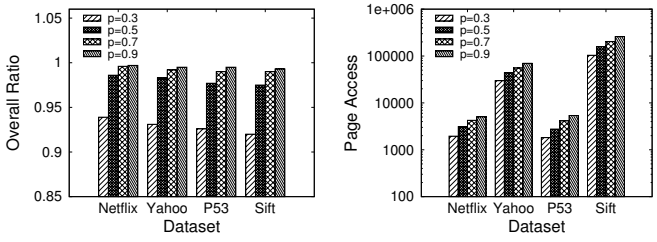
(a) Overall Ratio (b) Page Access

Fig. 10. Impact of c

IX. Related Work

In recent years, MIP search problem has received widespread attention and various types of methods have been proposed to solve both exact and approximate MIP search problems. In the beginning, some tree-based searching methods [7], [8], [21], [32] are presented for the exact MIP search problem. In addition, several methods based on linear search [22], [39], [40] are also proposed. However, these methods suffer from the curse of dimensionality and their performances will degrade sharply when the feature dimension is high (more than 20) [17], [44].

To address the MIP search problem in high-dimensional space, there exists a line of research on approximate solutions by trading off the accuracy and efficiency. Since inner product doesn't satisfy some important metric properties such as non-negativity and triangle inequality, it's not a metric measurement. Existing methods for metric measurements [24], [25], [48], such as Locality Sensitive Hashing (LSH) [16] and some quantization-based methods [19], [42], [45], can't be applied to MIP search problem. In addition, some methods proposed for a class of measurements such as Bregman distance [36] can't be employed. For this reason, most existing methods employ asymmetric (data points and query point are transformed in different manners) or symmetric (data points and query point are transformed in the same manner) transformations to convert a MIP search problem into a Nearest Neighbor (NN) search problem (called MIPS-NNS reduction) or a Maximum Cosine-similarity (MC) search problem (called MIPS-MCS reduction) [2], [17], [30], [34], [35], [44]. Benefiting from these transformations, the order of MIP points can be preserved by the order of NN/MC points as much as possible, and the traditional metric search methods represented by LSH can be applied. These methods are considered as transformation-based methods and they are introduced as follows.



(a) Overall Ratio (b) Page Access

Fig. 11. Impact of p

In L2-ALSH [34] and Sign-ALSH [35], the MIP search problem is respectively converted into an NN search problem or an MC search problem by various asymmetric transformations, and the NN and MC search problems are solved by E2LSH [9] and SimHash [4], respectively. Nonetheless, they both introduce transformation errors affecting the accuracy. Besides, L2-ALSH leads to distortion errors after the transformation, which indicates that the Euclidean distance between most data points and the query point will be close to each other [17], and the efficiency will decrease. To avoid the transformation errors, an exact asymmetric transformation based solution named X-BOX is proposed. It takes advantage of the MIPS-NNS reduction and solves the NN search problem by PCA-tree, but its transformation also causes distortion errors. In addition to the aforementioned asymmetric solutions, Simple-LSH [30] employs a symmetric transformation for a MIPS-MCS reduction. Nevertheless, it suffers from long tails in the 2-norm distribution of real datasets [44].

Recently, two LSH-based methods, named H2-ALSH [17] and Norm Ranging-LSH [44] are devised. H2-ALSH proposes an asymmetric transformation without transformation errors named QNF transformation to convert MIP search problem into NN search problem. Furthermore, to reduce the distortion errors for the higher efficiency, a novel homocentric hypersphere partition strategy is designed. Norm-ranging LSH partitions the whole dataset into several subsets, where the searching process is performed by several independent indexes, to solve the excessive normalization problem caused by the long tails. Nevertheless, these methods require a large number of hash tables or long hash codes to ensure the accuracy, which takes up lots of pre-processing overheads. In this paper, we choose these two advanced methods as the benchmark methods.

There is also a plethora of data-dependent methods [11], [13], [14], [20], [27]–[29], [33], [43], [46], which are dedicated to the MIP search problem recently. These methods require learning-based techniques in the preprocess, which is difficult to maintain when large volumes of data are being updated. More importantly, they are not tailored to our concerned c-AMIP search problem with the probability guarantee in accuracy.

X. Conclusion

In this paper, we address the important issue of c-AMIP search on high-dimensional and large-scale datasets by introducing an efficient method with a lightweight index. In our method, we employ 2-stable random projections to reduce the high-dimensional c-AMIP search problem to a low-dimensional search problem. With two derived searching conditions and the proposed Quick-Probe, our method can efficiently guarantee c-AMIP search in accuracy with arbitrary probabilities. In addition, to accelerate the searching process, we utilize the lightweight iDistance as the index to perform the range search in the low-dimensional space. Experimental results on four real datasets demonstrate that our method requires less pre-processing cost and provides c-AMIP results with a probability guarantee in accuracy efficiently.

Acknowledgment

The work is supported by the National Key Research & Development Program of China (No. 2018YFB1003400), the National Natural Science Foundation of China (Nos. 62072083 and U1811261) and Liaoning Revitalization Talents Program (XLYC1807158). Yang Song is supported by the Chinese Scholarship Council.

References

[1] A. Auvolat and P. Vincent. Clustering is efficient for approximate maximum inner product search. *CoRR*, abs/1507.05910, 2015.

[2] Y. Bachrach, Y. Finkelstein, R. Gilad-Bachrach, L. Katzir, N. Koenigstein, N. Nice, and U. Paquet. Speeding up the xbox recommender system using a euclidean transformation for inner-product spaces. In *RecSys 2014*.

[3] J. Bennett, S. Lanning, et al. The netflix prize. In *Proceedings of KDD cup and workshop*, page 35, 2007.

[4] M. Charikar. Similarity estimation techniques from rounding algorithms. In *STOC 2002*.

[5] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *RecSys 2010*.

[6] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *RecSys 2010*.

[7] R. R. Curtin, A. G. Gray, and P. Ram. Fast exact max-kernel search. In *SIAM 2013*.

[8] R. R. Curtin and P. Ram. Dual-tree fast exact max-kernel search. *Statistical Analysis and Data Mining*, 7(4):229–253, 2014.

[9] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *SoCG 2004*.

[10] T. L. Dean, M. A. Ruzon, M. Segal, J. Shlens, S. Vijayanarasimhan, and J. Yagnik. Fast, accurate detection of 100, 000 object classes on a single machine. In *CVPR 2013*.

[11] Q. Ding, H. Yu, and C. Hsieh. A fast sampling algorithm for maximum inner product search. In *AISTATS 2019*.

[12] G. Dror, N. Koenigstein, Y. Koren, and M. Weimer. The yahoo! music dataset and kdd-cup'11. In *Proceedings of the 2011 International Conference on KDD Cup*, pages 3–18, 2011.

[13] M. Fraccaro, U. Paquet, and O. Winther. Indexable probabilistic matrix factorization for maximum inner product search. In *AAAI 2016*.

[14] R. Guo, Q. Geng, D. Simcha, F. Chern, S. Kumar, and X. Wu. New loss functions for fast maximum inner product search. *CoRR*, abs/1908.10396, 2019.

[15] R. Guo, S. Kumar, K. Choromanski, and D. Simcha. Quantization based fast inner product search. In *AISTATS 2016*.

[16] Q. Huang, J. Feng, Y. Zhang, Q. Fang, and W. Ng. Query-aware locality-sensitive hashing for approximate nearest neighbor search. *PVLDB*, 2015.

[17] Q. Huang, G. Ma, J. Feng, Q. Fang, and A. K. H. Tung. Accurate and fast asymmetric locality-sensitive hashing scheme for maximum inner product search. In *KDD 2018*.

[18] H. V. Jagadish, B. C. Ooi, K. Tan, C. Yu, and R. Zhang. idistance: An adaptive b^+-tree based indexing method for nearest neighbor search. *TODS*, 30(2):364–397, 2005.

[19] Y. Kalantidis and Y. Avrithis. Locally optimized product quantization for approximate nearest neighbor search. In *CVPR 2014*.

[20] O. Keivani, K. Sinha, and P. Ram. Improved maximum inner product search with better theoretical guarantees. In *IJCNN 2017*.

[21] N. Koenigstein, P. Ram, and Y. Shavitt. Efficient retrieval of recommendations in a matrix factorization framework. In *CIKM 2012*.

[22] H. Li, T. N. Chan, M. L. Yiu, and N. Mamoulis. FEXIPRO: fast and exact inner product retrieval in recommender systems. In *SIGMOD 2017*.

[23] J. Li, X. Yan, J. Zhang, A. Xu, J. Cheng, J. Liu, K. K. W. Ng, and T. Cheng. A general and efficient querying method for learning to hash. In *SIGMOD 2018*.

[24] K. Li and G. Li. Approximate query processing: What is new and where to go? - A survey on approximate query processing. *DSE*, 3(4):379–397, 2018.

[25] W. Li, Y. Zhang, Y. Sun, W. Wang, M. Li, W. Zhang, and X. Lin. Approximate nearest neighbor search on high dimensional data - experiments, analyses, and improvement. *TKDE*, 32(8):1475–1488, 2020.

[26] R. Liu, W. Cheng, H. Tong, W. Wang, and X. Zhang. Robust multi-network clustering via joint cross-domain cluster alignment. In *ICDM 2015*.

[27] R. Liu, T. Wu, and B. Mozafari. A bandit approach to maximum inner product search. In *AAAI 2019*.

[28] S. S. Lorenzen and N. Pham. Revisiting wedge sampling for budgeted maximum inner product search. *CoRR*, abs/1908.08656, 2019.

[29] S. Morozov and A. Babenko. Non-metric similarity graphs for maximum inner product search. In *NeurIPS 2018*.

[30] B. Neyshabur and N. Srebro. On symmetric and asymmetric lshs for inner product search. In *ICML 2015*.

[31] R. Panigrahy. Entropy based nearest neighbor search in high dimensions. In *SODA 2006*.

[32] P. Ram and A. G. Gray. Maximum inner-product search using cone trees. In *KDD 2012*.

[33] F. Shen, W. Liu, S. Zhang, Y. Yang, and H. T. Shen. Learning binary codes for maximum inner product search. In *ICCV 2015*.

[34] A. Shrivastava and P. Li. Asymmetric LSH (ALSH) for sublinear time maximum inner product search (MIPS). In *NIPS 2014*.

[35] A. Shrivastava and P. Li. Improved asymmetric locality sensitive hashing (ALSH) for maximum inner product search (MIPS). In *UAI 2015*.

[36] Y. Song, Y. Gu, R. Zhang, and G. Yu. Brepartition: Optimized high-dimensional knn search with bregman distances. *CoRR*, abs/2006.00227, 2020.

[37] R. Spring and A. Shrivastava. Scalable and sustainable deep learning via randomized hashing. In *KDD 2017*.

[38] Y. Sun, W. Wang, J. Qin, Y. Zhang, and X. Lin. SRS: solving c-approximate nearest neighbor queries in high dimensional euclidean space with a tiny index. *PVLDB*, 2014.

[39] C. Teflioudi and R. Gemulla. Exact and approximate maximum inner product search with LEMP. *TODS*, 42(1):5:1–5:49, 2017.

[40] C. Teflioudi, R. Gemulla, and O. Mykytiuk. LEMP: fast retrieval of large entries in a matrix product. In *SIGMOD 2015*.

[41] S. Vijayanarasimhan, J. Shlens, R. Monga, and J. Yagnik. Deep networks with large output spaces. In *ICLR 2015*.

[42] C. Wei, B. Wu, S. Wang, R. Lou, C. Zhan, F. Li, and Y. Cai. Analyticdb-v: A hybrid analytical engine towards query fusion for structured and unstructured data. *PVLDB*, 2020.

[43] X. Wu, R. Guo, S. Kumar, and D. Simcha. Local orthogonal decomposition for maximum inner product search. *CoRR*, abs/1903.10391, 2019.

[44] X. Yan, J. Li, X. Dai, H. Chen, and J. Cheng. Norm-ranging LSH for maximum inner product search. In *NeurIPS 2018*.

[45] W. Yang, T. Li, G. Fang, and H. Wei. PASE: postgresql ultra-high-dimensional approximate nearest neighbor search extension. In *SIGMOD 2020*.

[46] H. Yu, C. Hsieh, Q. Lei, and I. S. Dhillon. A greedy approach for budgeted maximum inner product search. In *NIPS 2017*.

[47] F. Zhang. *Matrix theory: basic results and techniques*. Springer Science & Business Media, 2011.

[48] R. Zhang, B. C. Ooi, and K. Tan. Making the pyramid technique robust to query types and workloads. In *ICDE 2004*.