Liu et al. EURASIP Journal on Wireless Communications and Networking (2018) 2018:208
https://doi.org/10.1186/s13638-018-1221-3

EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

# OPRCP: approximate nearest neighbor binary search algorithm for hybrid data over WMSN blockchain

Huakun Liu[1], Xin Wei[1], Ruliang Xiao[1,2,3*], Lifei Chen[1,2], Xin Du[1] and Shi Zhang[1]

## Abstract

In order to prevent sensitive data tampering in the application of security monitoring, intelligent traffic, and other sensitive Internet of Things, the research on WMSN (wireless multimedia sensor networks) application system based on blockchain and IPFS (InterPlanetary File System) is of great significance. However, WMSN data are characterized by high dimensionality, large scale, and multiple types, so it is challenging to search WMSN data efficiently over blockchain system. This paper proposed a novel One Permutation with Rotation and cross-polytope locality-sensitive hashing (OPRCP) method of approximate nearest neighbor binary query for querying binary hybrid data in the form of WMSN multimedia data (containing two hybrid types of data, such as image-text and image-audio). Firstly, a binary hybrid data index was built with the method of locality-sensitive hashing (LSH) to retain content similarity among original data objects for performing accurate queries. Secondly, the approximate nearest neighbor search strategy was used in place of the nearest neighbor strategy, to reduce querying time. Finally, a binary hybrid data model was employed to cope with multiple types of data in WMSN and carry out collaborative search of binary hybrid data. The experimental results show that compared with other mainstream methods, the proposed OPRCP method is widely adaptive to massive high-dimensional data in multiple types and can improve the accuracy of query results. The OPRCP method exhibits good performance, effectively saves resources, and reduces query time for a variety of datasets. It is an effective solution to the binary hybrid search of approximate neighbors, and it is applicable to the WMSN data search based on smart contracts in WMSN blockchain systems.

**Keywords:** WMSN hybrid data, Approximate nearest neighbor binary search, LSH, Blockchain, IPFS

## 1 Introduction

The wireless multimedia sensor network (WMSN) is a new wireless sensor network developed based on wireless sensor networks (WSN) with multimedia data such as videos, audios, and images. To date, WMSN has been widely used in security monitoring, intelligent transportation, environmental monitoring, etc. We should prevent some sensitive application data from tampering, such as supervision data of farm products and violation evidence data from intelligent transportation systems. At present, a cutting-edge idea is to protect highly sensitive data by using currently hot technologies of blockchain and IPFS to build blockchain systems based on WMSN [1–5]. Therefore, the solution of multimedia data search will be basic design in the development of WMSN blockchain systems. Generally, after preprocessing of WMSN data [6], we filter massive data from the same network through search operations based on multimedia data, to obtain data results similar to query objects. It can be seen from literature [3–5] that quick querying of high-dimensional massive datasets has shown great potential. Hence, for diversified WMSN data, how to use different types of data to carry out collaborative search and improve the precision of query results is of great significance to the application of WMSN blockchain.

People usually construct a WMSN blockchain application solution based on Ethereum and IPFS [7]. WMSN data are processed on the blockchain after being stored in the IPFS distributed structure, where such data can be

* Correspondence: xiaoruliang@fjnu.edu.cn
[1]College of Mathematics and Informatics, Fujian Normal University, Fuzhou 350117, China
[2]Digit Fujian Internet-of-Things Laboratory of Environmental Monitoring, Fuzhou 350117, China
Full list of author information is available at the end of the article

searched or operated in other ways through smart contracts. Previous work [8] indicates that query of massive multimedia data is composed of three procedures: (1) extraction of multimedia data features, (2) creation of query indexes on feature data in datasets, and (3) mapping query objects into the query index structure. To be specific, the first step is to extract features of multimedia data, which are usually converted into feature vectors for data preparation for the following procedures. The second procedure of creating query indexes is very important and aims to reduce comparisons with data objects during search, thus improving search efficiency. With regard to multimedia data search on WMSN blockchain, existing methods are challenged in three aspects as follows:

Problem 1 (curse of space): massive data storage on WMSN blockchain and IPFS need a large space for data themselves, but existing methods tend to consume space several times larger than that for datasets when creating index structures on the premise of ensuring accuracy, which is undoubtedly a "curse of space" to massive datasets.

Problem 2 (curse of dimensionality): The processing of data of a single type such as images or text can create feature vectors which are high-dimensional data, let alone the processing of hybrid data requires the consideration of multiple data features at the same time, so the processing of massive IPFS data on WMSN blockchain is a "curse of dimensionality."

Problem 3 (curse of growth): Against the background of WMSN blockchain IPFS distributed storage, the rapid growth of WMSN data requires relatively excellent scalability of established index structures. The processing of massive WMSN blockchain IPFS data is a "curse of growth."

In view of the above problems in WMSN blockchain IPFS systems, we proposed a novel hybrid data query method named OPRCP (One Permutation with Rotation and Cross-Polytope locality-sensitive hashing) in this paper, in which we used a kind of approximate nearest neighbor binary search of WMSN binary hybrid data, such as image-text and image-audio. The OPRCP method implements the hybrid locality-sensitive hashing method, effectively solving the problem of binary hybrid approximate nearest neighbor search of WMSN hybrid data.

The OPRCP method is completely different from all the existing methods, the highlights of which mainly include the following: First, we adopted the collaborative filtering strategy rather than the existing method of separately querying and filtering data in multiple types, to implement hybrid hash mapping of multiple types and thus realize much more accurate query than previous methods. Second, in hash mapping of a single type, the methods of One Permutation with Rotation [9] and cross-polytope LSH [10] were used respectively to map similar data to the same index value, which compared with the original LSH method [11] significantly improves in terms of time and space. Third, the feature hashing method [12] was applied to reduce the dimension of high-dimensional sparse data from $d$ to $d^{'}(d^{'} \ll d)$, thus maintaining precision to a large extent and reducing time complexity. Lastly, the multi-probe locality-sensitive hashing method [13] was used to cut down the space occupied for storing indexes, overcoming the shortcoming of consuming large amounts of storage space in the LSH method.

Through a lot of comparative experiments, we find that the OPRCP algorithm presented in this paper needs less query time and storage space to achieve ideal query results, showing good scalability in terms of data scale and data types.

The remainder of this paper is structured as follows. Section 2 introduces related work. Section 3 describes relevant basic knowledge involved in this paper. Section 4 presents the OPRCP method. Section 5 is an analysis of experiments and experiment results. Section 6 draws conclusions on the paper.

## 2 Related work

The privacy protection of the WMSN is a challenging research hotspot due to the lack of related Internet of Things (IoT) standards [4, 5]. Blockchain is usually used as a basic decentralized technology for encrypting digital currency, such as Bitcoin and Ethereum. In general, IoT is a centralized distributed structure. The centralized IoT data management and access control model has many problems, especially the scalability issues of IoT systems, forcing users to trust third-party intermediaries to manage their data [14]. But the literature [7] presented a novel decentralized privacy-preserving access control model based on blockchain technology in IoT. So, for protecting privacy data and sensitive data on the opened IoT of WMSN, it provided us a decentralized and secure technical guarantee example [7]. But there is still a problem because that many blockchain technologies such as Bitcoin and Ethereum do not provide decentralized data storage capabilities. So, we must borrow the InterPlanetary File System (IPFS) because it provides us a high-throughput content-addressable block storage model with content-addressable hyperlinks [15], which forms a generic Melker-DAG. IPFS is well integrated with blockchain technology [3, 5, 7, 14, 16]. Therefore, the decentralized security and privacy protection model based on IPFS data storage technology has become one of the research hotspots of the WMSN Internet of Things.

In the field of data search over IoT of WMSN, search strategy and index structure have always been the research hotspots. These studies are generally devoted to improving search performance in terms of structure and algorithm. In terms of search strategy, for the similarity search problem of low-dimensional data, many excellent solutions have emerged for nearest neighbor queries. However, in the high-dimensional case, these methods

tend to have only a slight improvement in the linear query time. A lot of researches [17–19] show that using the approximate nearest neighbor search strategy, people can break the bottleneck of linear search time, instead of nearest neighbor search. In fact, in most application scenarios, the approximate nearest neighbor can achieve similar results as the nearest neighbor. At present, there are many feasible and effective methods for the problem of approximate nearest neighbor search [17, 20].

In terms of index structure, for a single type of data, people performed well with approximate similarity search by constructing feature vector indexes. For example, for image data, there are two index structures commonly. The first one is the tree index, which was proposed in [21] by Sunil Arya of the Hong Kong University of Science and Technology and David M. Mount of the Maryland University. Another is the hash index, which was introduced in [21] by TTIC's Greg Shakhnarovich, Trevor Darrell of UC Berkeley, and Piotr Indyk of MIT. In particular, for the hash index structure, Andrei Broder proposed a min-hashing method to effectively solve the approximate nearest neighbor search problem in Hamming space [22]. For Euclidean space, Mayau Datar (Stanford University) and Nicole Immorlica (MIT) introduced a LSH method based on p-stable distribution in [11].

However, we need to consider multiple features at the same time for hybrid data. At present, many researchers usually try to first index a single type respectively, then to integrate them. For example, such as image and text data, Chen Liu, who comes from the National University of Singapore, proposes a method of linking tagged resources to concepts extracted from Wikipedia and implements cross-model search in [23]; Ju Fan, from Tsinghua University, introduced a similarity search method SEAL based on region-oriented space-oriented text data in [24].

A mixed LSH method based on p-stable LSH [11] and min-hashing [22] method is proposed by Yu Ge, who comes from Northeastern University, and proves that using the LSH method can better solve the approximate nearest neighbor search problem of multiple types of data [25]. However, the main disadvantages of the method in [25] are the large storage consumption and high computational complexity, which is not conducive to distributing the calculations to the child computing nodes.

Up to date, the existing hybrid-type approximate search methods have a large overhead in space, and there is still much room for improvement in query efficiency.

Therefore, in view of these deficiencies, we propose a novel hybrid data query method. We ingeniously combine LSH method based on OPR and CP, which makes further improvements. The proposed OPRCP method reduces the query time, consumes less storage space, and gets the same accuracy as other methods. In addition, ORPCP still guarantees better query performance with the growth of data dimension. In the WMSN-blockchain IoT, combining with the access control mechanism, by constructing a query index based on the OPRCP in the form of a smart contract, we can construct a search transaction and obtain a good application effect.

## 3 Preliminaries

We use the $l_p^d$ to denote the space $\mathbb{R}^d$ under $l_p$ norm. For any point $v \in \mathbb{R}^d$, we denote by $\|\overrightarrow{v}\|_p$ the $l_p$ norm of vector $v$. We use $S^{d-1}$ to denote the unit Euclidean sphere in $\mathbb{R}^d$ with the center being the origin.

The data model in this paper is a data object oriented to the binary hybrid data type. To simplify the problem, we mainly focus on the binary hybrid data in Hamming space and Euclidean space. To be convenient for theoretical analysis, we assume that the two data types are independent of each other. For Hamming space, we often use Jaccard distance to represent the similarity between two objects, see Formula (1), and for the Euclidean space, we use the normalized Euclidean distance, see Formula (3).

$$\left\|\overrightarrow{v}\right\|_p = \left(\sum_{i=1}^{d} v_i^p\right)^{\frac{1}{p}} \tag{1}$$

$$\mathrm{J}(x,y) = 1 - \frac{|x \cap y|}{|x \cup y|} \tag{2}$$

$$\mathrm{D}(x,y) = \frac{\mathrm{Euclidean}(x,y)}{D_{\max}} \tag{3}$$

The linear weighted sum is widely used in the problem of calculating the similarity of multivariate hybrid data types [26, 27]. We define the binary hybrid data similarity as follows: Suppose $X$ be the whole dataset, there are two data types for every $x \in X$; we denote the Hamming space data by $x^1$, while the Euclidean space data by $x^2$, and $x = (x^1, x^2)$. Then, for any $x_1$, $x_2 \in X$, $\alpha \in (0, 1)$ indicates the proportion of data types, we have

$$\mathrm{dist}(x_1, x_2) = \alpha \times J\left(x_1^1, x_2^1\right) + (1-\alpha) \times D\left(x_1^2, x_2^2\right) \tag{4}$$

### 3.1 Approximate nearest neighbor

In this paper, we focus on the approximate nearest neighbor problem in Hamming space and Euclidean space.

**Definition 1** [28] *The (c, r)-approximate near neighbor problem (ANN) with failure probability f is to construct a data structure over a set of points P in metric space (X, D) supporting the following query: given any fixed query point $q \in X$, if there exits $p \in P$ with $D(p, q) \le r$, then report some $p' \in P$ such that $D(p', q) \le cr$, with probability at least 1 − f.*

We extend it to binary hybrid data and get Definition 2.

**Definition 2** *The (c, r, d)-approximate near neighbor problem with failure probability f is to construct a data structure over a set of points P in metric space* $(X, D)$, $X = (X^1, X^2)$, *supporting the following query: given any fixed query point* $q \in X$, *if there exits* $p \in P$ *with* $D(p^1, q^1) \le r$, $D(p^2, q^2) \le d$, *then report some* $p' \in P$ *such that* $D(p'^1, q^1) \le cr$, $D(p'^2, q^2) \le cd$ *with probability at least* $1 - f$.

### 3.2 Locality-sensitive hashing

Searching on high-dimensional datasets, most solutions are not entirely satisfactory and can only provide little improvement over a linear algorithm. In 1998, Indyk and Motwani of MIT introduced an approximate similarity search method with sublinear dependence on the data size, called locality-sensitive hashing (LSH) [17]. The key idea is to hash the points using hash functions so as to ensure that, for each function, the probability of collision for the similar objects is much higher than those dissimilar objects. This method is an important technique for solving the $(c, r)$ – NNproblem. A LSH family is defined as:

**Definition 3** [28] *A family* $\mathcal{H} = \{h : X \to U\}$ *is* $(r_1, r_2, p_1, p_2)$*-sensitive for* $(X, D)$ *if for any* $q, p \in X$ *we have*

(1) If $D(p, q) \le r_1$, then $\Pr_{\mathcal{H}}[h(q) = h(p)] \ge p_1$

(2) If $D(p, q) \ge r_2$, then $\Pr_{\mathcal{H}}[h(q) = h(p)] \le p_2$

In order for a locality-sensitive hash family to be useful, it has to satisfy inequalities $p_1 > p_2$ and $r_1 < r_2$.

To perform a hybrid search on two types of data, we use two different LSH families for different types of data. We describe them below.

### 3.3 One permutation with rotation and cross-polytope LSH

The min-hashing [22] method is popular for build data structure for Hamming space. In [9], Anshumali from Cornell University and Ping Li from Rutgers University introduced an improved method based on one permutation hashing, called One Permutation with Rotation Hashing (OPR), which costs less computation and resource-consumption, and gets the same level of query performance as min-hashing. Next, we describe it.

We use $D$ to denote the dimension of data in Hamming space, then we can consider binary vectors in $\mathbb{R}^D$ the same as sets in $\Omega = \{0, 1, 2, \ldots, D-1\}$.

Let $S \subseteq \Omega = \{0, 1, 2, \ldots, D-1\}$ and consider a random permutation $\pi \subseteq \Omega \to \Omega$, and we divide the space into $k$ bins. For the $j$th bin, where $0 \le j \le k-1$, we define the set

$$M_j(\pi(S)) = \left\{ \pi\{S\} \cap \left[ \frac{Dj}{k}, \frac{D(j+1)}{k} \right) \right\} \tag{5}$$

We need to clarify two concepts:

1) If $M_j(\pi(S))$ is empty, then $M_j(\pi(S)) = \varnothing$.
2) The minimum of $M_j(\pi(S))$ is the smallest nonzero index in the bin.

If $M_j(\pi(S)) = \varnothing$, then we denote the hash value of this bin by $OPR_j(\pi(S)) = E$. If the set is not empty, $OPR_j(\pi(S))$ is the minimum of $M_j(\pi(S))$. In this paper, to simplify the problem, we always assume $D$ is divisible by $k$. That is, when $M_j(\pi(S)) \ne \varnothing$, we have

$$OPR_j(\pi(S)) = \min\left(M_j(\pi(S))\right) \mod \frac{D}{k} \tag{6}$$

Formally, if $M_j(\pi(S)) = \varnothing$, the hash value is $C$ plus the hash value of the first non-empty bin on the right (circular). We define

$$H_j(\pi(S)) = \begin{cases} \mathrm{OPR}_j(\pi(S)) & \text{if } \mathrm{OPR}_j(\pi(S)) \ne E \\ \mathrm{OPR}_{(j+t) \bmod k}(\pi(S)) + \mathrm{tC} & \text{otherwise} \end{cases} \tag{7}$$

$$t = \min(\mathrm{z}), \mathrm{s.t.} \mathrm{OPR}_{(j+z) \bmod k}(\pi(S)) \ne E \tag{8}$$

Here $C$ is a constant to avoid wrong collisions, and $C \ge \frac{D}{k} - 1$.

In [29], Kengo Terasawa and Yuzuru Tanaka proposed a novel and efficient LSH method to solve ANN problem in Euclidean space, called cross-polytope LSH. We recall the definition of it.

**Definition 4** [29] *Consider the following hash family H for points on a unit sphere* $S^{d-1} \subset \mathbb{R}^d$. *Let* $A \in \mathbb{R}^{d \times d}$ *be a random matrix with i.i.d. Gaussian entries. To hash a point* $x \in S^{d-1}$, *we compute* $y = \frac{Ax}{\|Ax\|} \in S^{d-1}$ *and then find the point closest to y from* $\{\pm e_i\}_{1 \le i < d}$, *where* $e_i$ *is the ith standard basis vector of* $\mathbb{R}^d$. *We use the closest neighbor as a hash of x.*

## 4 The OPRCP method

### 4.1 The overall framework of ANN binary query

In the WMSN blockchain IoT application, we referred to [1, 3, 4] and proposed an inquiry transaction mechanism based on the OPRCP method. Moreover, the intelligence contract can be built and provide search applications for WMSN blockchain users.

The same as the process of hybrid query that was mentioned in the first part (1), the hybrid search of multimedia consists of three steps (Fig. 1):

(1) Extracting feature of multimedia data.

This is the phase of data preprocessing. After input of the original WMSN multi-type data, the multimedia data can be converted into a feature vector by characterizing, see Phase 1, steps 1–2 in Fig. 1. In this step, TF-IDF method is usually used to convert the text data into a

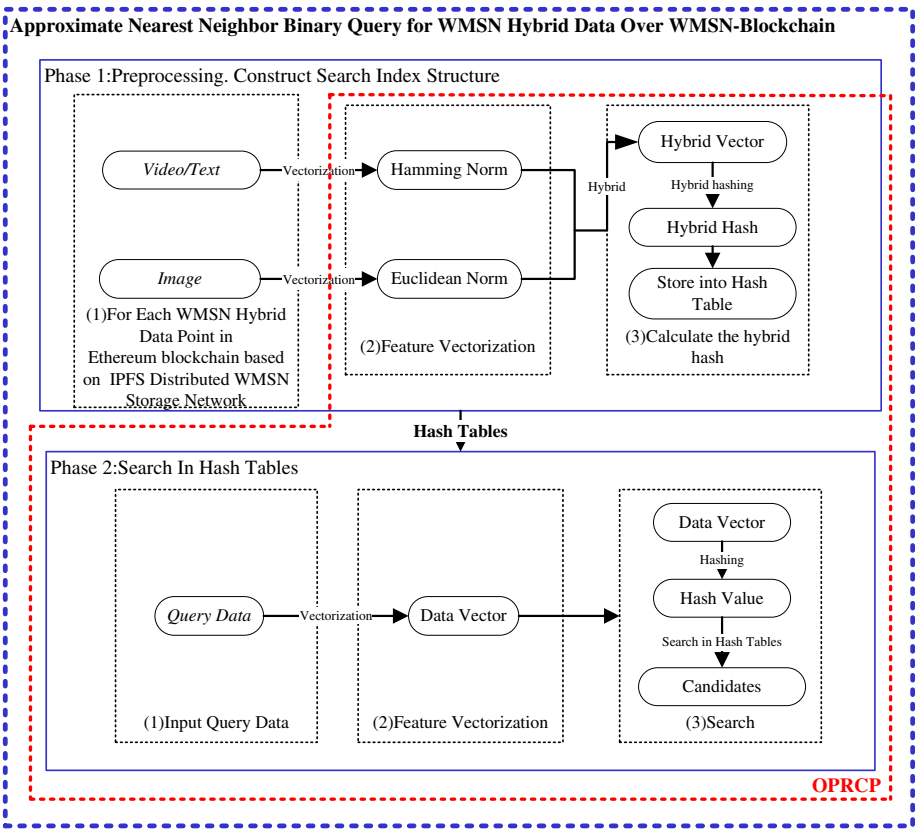


**Fig. 1** The flowchart of approximate similarity binary search for WMSN hybrid data over Ethereum blockchain based on IPFS distributed WMSN storage network

feature vector, and extracting SIFT feature values is often used to process image data.

(2) Constructing a data structure with feature vectors.

In Fig. 1 Phase 1 steps 2–3, after feature extraction phase is finished.

(3) Mapping query object to data structure.

In the query phase (Fig. 1 Phase 2), given a query object, the same hash calculation process is performed after characterizing, then the query results are generated from the data structure.

This paper mainly addresses the problems of the last two steps of the WMSN data query process (Fig. 1 Phase 1 steps 2–3, Phase 2 steps 2–3).

### 4.2 Hybrid query data structure

For a large-scale WMSN hybrid dataset (such as picture-text and picture-video), a simple processing method is to construct a data structure for each data type, then query each data type separately during the query phase, and finally, filter the results. We show a binary hybrid LSH framework based on the method. This two-level structure consists of two LSH methods that process a single data type. After input of the hybrid data, we search the approximate nearest neighbors of the corresponding type on each level, then return the intersection of two levels of query results, as shown in Fig. 2.

Intuitively, the performance of this filter structure depends on the order in which types of queries are performed. If the performance of the second level is higher than the first level, it is clear that it will be better to process the second level first. However, these characteristics are often different for different datasets. In order to avoid this problem, we adopted a collaborative filtering binary hybrid LSH framework.

The basic idea of the large-scale WMSN data binary hybrid LSH is the same as the basic LSH method, that is, to hash the hybrid data using hash functions to ensure that the probability of collision is much higher for similar objects than dissimilar objects. The process consists of three steps:

(1) Convert $x^1$ to $k_1$ hash values by $(r_1, r_2, p_1, p_2)$-sensitive hash,

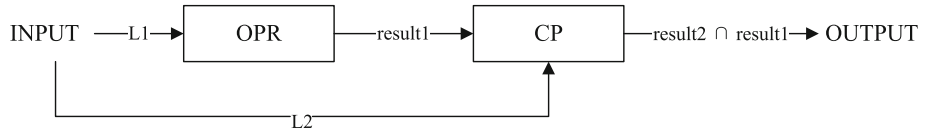


**Fig. 2** Two-level LSH filter structure

(2) Convert $x^2$ to $k_2$ hash values by $(d_1, d_2, p_1, p_2)$-sensitive hash,
(3) Combine $k_1$, $k_2$ hash values to get the hybrid hash value,

as shown in Fig. 3.

In order to perform the binary hybrid approximate nearest neighbor search, we need to preprocess data and construct a hash index for each of the data types in the dataset. Suppose parameters as are follows: (1) approximation factor $c$, (2) Hamming sensitive distance $r$, and (3) Euclidean sensitive distance $d$. For $X^1$, given a family $\mathcal{H}^1$ of hash functions with parameters $(r, cr, p_1^1, p_2^1)$ as in Definition 3, and for $X^2$, the hash family is $\mathcal{H}^2$. We choose $k_1$ functions from $\mathcal{H}^1$, $k_2$ functions from $\mathcal{H}^2$, and combine them to get a new hybrid hash function. Define a hybrid hash function $\mathcal{G} = (g : S \rightarrow U^{k_1+k_2})$ such that $g_i(v) = (h_1^1, h_2^1, \ldots, h_{k_1}^1, h_1^2, h_2^2, \ldots, h_{k_2}^2)$, where $h_i^1 \in \mathcal{H}^1$, $h_i^2 \in \mathcal{H}^2$. For an integer $L$, we choose $L$ functions $g_1$, $g_2$, …, $g_L$ from $\mathcal{G}$, independently and uniformly at random. For each hybrid data, we will get $L$ hash values, then store data into corresponding buckets of $L$ different hash tables. The construct process is as follows.

**Algorithm 1PROCESS**

**Input**: $P$
**Output**:Hash Tables $L$

1. Initialization $H^1, H^2$
2. **for**$i \leftarrow 1 \ldots L$**do**
3. $g_i = (h_1^1, h_2^1, \ldots h_{k_1}^1, h_1^2, h_2^2, \ldots h_{k_2}^2)$, // $h^1 \sim H^1, h^2 \sim H^2$
4. **for** p$\in P$**do**
5. **for**$i \leftarrow 1 \ldots L$ **do**
6. idx$\leftarrow g_i(p)$
7. store $p$ into buckets$_i(idx)$

**Return**:Hash Tables $L$

For an $n$-point dataset in a $d$-dimensional space, Algorithm 1 achieves process time $O(ndkL)$ and space $O(dn + nL)$. The space consumption mainly depends on the number of hash tables $L$, which is linear with the size of dataset. The processing time depends on $k_1 + k_2$ and the attributes of datasets such as $d$ and $n$.

### 4.3 Hybrid query OPRCP method

There are many efficient LSH methods for Hamming space and Euclidean space. In order to adapt to the characteristics of the binary hybrid data, we combined a variety of efficient methods to improve them. The proposed OPRCP method is a fusion of the OPR method and the CP method:

(1) For the Hamming space, we use the One Permutation with Rotation (OPR) method. Compared with the well-known min-hashing, OPR has a large improvement in time and space. Specifically, OPR requires only one permutation, which makes the running time greatly reduced.
(2) For the Euclidean space, we use the cross-polytope LSH method (CP). This method has a query time of $O(n^\rho)$ for $\rho = \frac{1}{2c^2-1}$. The LSH method based on p-stable distributions in [11] can only obtain $\rho = \frac{1}{c^2}$. In addition, CP method is easy to reduce storage space by extending the query method.

Next, we introduce the basic OPRCP.

During construction of the data structure (see Algorithm 1), for every $p \in P$, in order to get hybrid hash value of $p$, we need to perform OPR processing on the Hamming space part of $p$, to get a $k_1$-dimensional vector $H_1 = [h_1, h_2, \ldots, h_{k_1}]$. Then for the Euclidean space, we apply cross-polytope LSH to get an index of the nearest standard basis vector from point $p$, denoted as $H_2 = i$, $i \in \{1, 2, \ldots, 2d\}$. Finally, we combine $H_1$, $H_2$ and get the hybrid hash $H = [H_1, H_2]$, then we use md5 to hash $H$ again to get last hash value of $p$ and store $p$ in corresponding hash bucket. To increase the probability of collision for similar object, we need to repeat above steps $L$ times. After all points have been processed, we will get $L$ hash tables. Each hash table has multiple hash buckets. The data points stored in the same bucket are called collision points.

The pseudocode appears as Algorithm 2.

**Algorithm 2OPRCP : PROCESS**

**Input**: $p$
**Output**:The hybrid hash of $p$

1. $p_1$is the Hamming norm point
2. $p_2$is the Euclidean norm point
3. Initialization $H_1, H_2$
4. Permutation $p_1$
5. Split $p_1$ into $k$ bins
6. for i$\leftarrow 1 \ldots k$ do
7. if the $i$th bin is empty
8. $H_{1i}$is the value of first non-empty bin on the right + C
9. else
10. $H_{1i}$ is the $i$th bin first nonzero index
11. $A \in \mathbb{R}^{d \times d}$is a random matrix with i.i.d. Gaussian entries
12. Compute$y = \frac{Ap_2}{\|Ap_2\|_2}$
13. $\{\pm e_i\}_{1 \le i \le d}$. is the $i$-th standard basis vector of $\mathbb{R}^d$
14. $H_2$is the point closest to $y$from $\{\pm e_i\}_{1 \le i \le d}$
15. The final hybrid hash is MD5$(H_1, H_2)$

**Return**: The hybrid hash of $p$

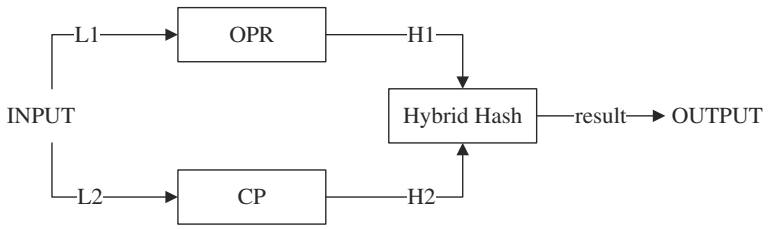


**Fig. 3** Binary hybrid LSH filter structure

For once hash process, we use time $O(kd_1 + d_2^2)$, where $d_1$, $d_2$ are the dimensions of Hamming space and Euclidean space of point $p$, as described in Algorithm 2. The bottleneck is applying a random rotation, to multiply a random matrix with a vector. It takes $O(d_2^2)$ time. To reduce this time, we instead use fast Hadamard transform, like the pseudo-rotation process in [30]. This method make the random rotation can calculate in time $O(d_2 \log d_2)$. Then, for $L$ times and $n$ points, we get the total time $O(nL(kd_1 + d_2 \log d_2))$.

### 4.4 Multi-probe OPRCP for large-scale WMSN data environment

To cope with the problem of storing large-scale LSH data structure of WMSN datasets, we proposed a multi-probe query method for OPRCP. This method can effectively reduce the number of independent hash tables used in the OPRCP data structure to achieve the effect of reducing space.

The standard LSH method is a single-probe query method. The main idea is as follows: Given a query point $q$, we calculate its corresponding $L$ hash values, and all points stored in the same hash buckets are taken as a candidate set. One of the major disadvantages of the standard LSH method is the huge space consumption, which uses $O(n\mathrm{L})$ space to store hash tables. To reduce the space of storing independent hash tables, a multi-probe query method has been proposed in [13]. This method considers candidates from multiple hash buckets in each table. Points that are close to $q$ but fail to collide with $q$ under hash function $h_i$ are still likely to hash to a value that is close to $h_i(q)$. We can probe multiple hash buckets close to $h_i(q)$, so that we appropriately increase the "collision" probability with approximate nearest neighbors (in this case, the collision means be queried by multi-probe) and so we can reduce the number of hash tables to reduce storage space. Multi-probe LSH has been shown to perform well in practice [13, 31].

The remaining problem is how to define a specific multi-probe query method for OPRCP. In the next, we describe the multi-probe version of the two data processing methods in OPRCP:

(1) First, we introduced the multi-probe version of cross-polytope LSH for Euclidean space. For the standard cross-polytope LSH, we define $h_i(q)$ as the point closest to $q$ from $\{\pm e_i\}_{1 \le i \le d}$ after $q$ rotation. Based on the method, we consider that some points with a high probability of collision with $q$ are still likely to hash different $m$ values close to $h_i(q)$. Let $m$ be the range of multi-probe. Given a query point $q$, we define $H_i(q) = \mathrm{argsort}_j \mathrm{Dist}(q, e)_{1 \le j \le m}$.
(2) For the OPR method for Hamming space, it is difficult to directly define its multi-probe version. As a result, we use the feature hashing approach convert it to a feature vector through an intermediate mapping. The vector can be processed by cross-polytope LSH. In particular, feature vectors correspond to a high-dimensional and extremely sparse data in binary Hamming space generated by characterizing data such as text. This feature allows feature hashing to effectively reduce dimensionality on the premise of ensuring no deviation [12]. Therefore, we not only obtain feature vectors that can be used for cross-polytope LSH processing, but also use the method of multi-probe version to achieve the effect of reducing space, and the search time can be further reduced after dimensionality reduction. Specifically, we reduce the dimension from $d$ to $d' \ll d$ by applying a linear map $x \to Sx$, where $S$ is a random sparse $d' \times d$ matrix, whose columns have *one* non-zero ±1 entry sampled uniformly. The feature hashing is also valid for Euclidean space vectors.

The pseudocode for the multi-probe OPRCP appears as Algorithm 3.

**Algorithm 3OPRCP : Multi-Probe Query**

**Input**: $q$
**Output**: *the approximate near neighbor of query q*

1. $q_1$ is the Hamming norm point
2. $q_2$ is the Euclidean norm point
3. $S \in \mathbb{R}^{d' \times d}$, each column have one non-zero $\pm 1$entry sampled uniformly
4. $A \in \mathbb{R}^{d \times d}$is a random matrix with i.i.d. Gaussian entries
5. $\{\pm e_{1i}\}_{1 \le i \le d'}$. is the $i$-th standard basis vector of $\mathbb{R}^{d'}$
6. $\{\pm e_{2i}\}_{1 \le i \le d}$. is the $i$-th standard basis vector of $\mathbb{R}^{d}$
7. Compute $q_1 = Sq_1$
8. Compute $y_1 = \frac{Aq_1}{\|Aq_1\|_2}$
9. **for** $i \leftarrow 1 \ldots m$ **do**
10. Compute $H_1$ is the $i$th closest point to y from $\{\pm e_{1i}\}_{1 \le i \le d'}\}$
11. Compute $y_2 = \frac{Aq_2}{\|Aq_2\|_2}$
12. **for** $i \leftarrow 1 \ldots m$ **do**
13. Compute $H_2$ is the $i$th closest point to y from $\{\pm e_{2i}\}_{1 \le i \le d'}\}$
14. Combination $H_1, H_2$and get the points in correspond bucket

**Return**: the approximate near neighbor of query $q$

In the process of approximate nearest neighbor search, this algorithm has a hashing time of $O(kdL)$. Then, one can determine nearest neighbor by retrieving points taken from the corresponding buckets. In general, the number of candidates is much smaller than the size of the dataset. So, the linear search process on the whole dataset can be avoided by using OPRCP to filter dissimilar data.

In this section, we introduce the OPRCP method, which aims to solve the problem of constructing data structure and query phase in hybrid query process in the WMSN blockchain IoT application. There are respective merits for both hash indexing frameworks. The two-level filter method is simple and easy to implement, and it can directly be extended by standard algorithm. However, due to the impact of the actual application scenario, the universality is poor; OPRCP adopted a hybrid collaborative query framework. By combining the OPR method and the CP method, the hybrid hash data structure was constructed. The query time is greatly improved compared to previous methods. In particular, by designing multi-probe OPRCP method, storage space of data structure is further reduced.

## 5 Experiments

### 5.1 Datasets

This experiment uses real datasets and synthetically generated datasets to evaluate the accuracy and efficiency of the algorithm, and we implement the simple OPRCP method and the complex OPRCP method of a two-layer filter structure (TLOPRCP).

The data types in this paper are binary data for Hamming space (text-characterized data) and numerical data for Euclidean space (image-characterized data). In order to verify the performance of the algorithm, we use real datasets and synthetically generated datasets for testing. The real dataset is MNIST [32]. MNIST is a standard handwritten digit dataset, which is oriented to the Euclidean space. We convert it to binary data by binarization and combine it with the original Euclidean spatial data to construct a binary hybrid dataset. The dimensions of the six synthetic datasets are 32, 128, 256, 512, 1024, 2048, and a data amount of 10,000. We considered randomly generating all points in the Euclidean space, and then, the Euclidean data is binarized. Binary data is obtained, which is finally mixed into binary hybrid data. Using synthetic datasets can adjust a condition while fixing another condition to evaluate the unilateral performance of the algorithm. In this experiment, we adjusted the dimension while fixing dataset size, in order to verify the performance and the optimal parameters of the algorithm in different dimensions; the performance of the algorithm changes with the dimensional change. The specific datasets are described in Table 1.

We implemented the related method BHL [25], TLBHL [25], and TLE2LSH [11] to compare with our method. BHL [25] is an LSH method for binary hybrid data that combines min-hashing [22] and E2LSH [11] and builds a hybrid data structure on the data to implement the query. TLBHL is an extended version of the BHL, which is a two-level filter structure, and TLE2LSH is an extended version of E2LSH [11].

The experiment was run on a PC with an operating system Arch Linux, Intel Corei7-4510U 3.1 GHz CPU and 8 GB of memory. All the code used in the experiment was implemented based on Python 3.6.

### 5.2 Metrics

In order to discuss the performance of the algorithm in the experiment, each query can be divided into four cases: true positive, false positive, true negative, and false negative, according to the combination of its true condition and query result, as shown in Table 2, using *TP*, *FP*, *TN*, and *FN*, respectively, indicating their corresponding set size.

Seven performance metrics were used in all experiments: hashing time, query time, recall, precision, F1-score, number of candidates, and memory occupied:

(1) Hashing time: calculating hash value is the main step of the query. The hashing time means the time consumed by applying hash functions.

**Table 1** Description of the datasets

| Datasets | Dimension (Hamming space, Euclidean space) | Dataset size |
|---|---|---|
| MNIST[32] | (784,784) | 70,000 |
| SynSet-1e4d16 | (16,16) | 10,000 |
| SynSet-1e4d64 | (64,64) | 10,000 |
| SynSet-1e4d128 | (128,128) | 10,000 |
| SynSet-1e4d256 | (256,256) | 10,000 |
| SynSet-1e4d512 | (512,512) | 10,000 |
| SynSet-1e4d1024 | (1024,1024) | 10,000 |

**Table 2** Confusion matrix of search result

| True condition | Query result | |
|---|---|---|
| | Positive | Negative |
| True | TP (true positive) | FN (false negative) |
| False | FP (false positive) | TN (true negative) |

(2) Query time: the time from the beginning of a query process to return to the exact nearest neighbor, which is also the total query time. This experiment focuses on this metric.
(3) Recall: it is the fraction of the relevant instances that has been retrieved over the total amount of relevant instances. In search application, it can be used to measure the coverage of query results.

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{9}$$

(4) Precision: it is the fraction of relevant instances among the retrieved instances, which can be used to measure the quality of query results.

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{10}$$

(5) F1-score: it is a measure of a test's accuracy. It considers both the precision and the recall. In search application, a good method should ensure that both the recall and the precision are optimal. However, the two metrics may conflict in practical process. To balance the recall and the precision, F-score is often used to conduct a comprehensive assessment of the two. The most common of these is the F1-score.

$$\text{F1} = \frac{2 \times \text{recall} \times \text{precision}}{\text{recall} + \text{precision}} \tag{11}$$

(6) Number of candidates: it refers to the total number of points stored in the same buckets, indicating the number of filtered results. In query process, number of candidates is the number of points that need to be compared for similarity, which is a critical factor for query time.
(7) Memory occupied: one of the disadvantages of LSH method is that it needs to construct a relatively large number of data structure to ensure a high probability of querying the nearest neighbor for sufficiently high dimensions. It leads to an increase in memory occupied, so we compare memory occupied as a metric.

The standard for setting each parameter in this experiment is to ensure the probability of finding the exact nearest neighbor $\text{Pr}_{\text{fn}}$ is at least 0.9.

OPRCP method has three parameters: (1) the number of hash tables $L$. To ensure $\text{Pr}_{\text{fn}}$ is acceptable, we need to choose an appropriate number of hash tables, so that near neighbors collide with query point under a hash family at least once; (2) the number of OPR bins $k$, which need to be set to decrease the collision probability of dissimilar points. Theoretically, with the increasing of $k$, the collision probability of dissimilar points decreased. In the meantime, the collision probability of approximate nearest points also decreased; we should improve it to achieve an acceptable value by increased $L$; and (3) probe factor $m$,$L$ is increased in order to ensure $\text{Pr}_{fn}$ is acceptable, which leads to a corresponding increase in memory occupied. The setting of $m$ compromised less precision, in exchange for a smaller $L$ for the same collision probability.

Our experiment (on the same synthetic dataset: SynSet-1e4d128, $d = 128$, $n = 1e4$) analyzes the impact of different parameters by fixing other parameters and adjusting the parameter to be analyzed.

### 5.3 The influence of probe factor *m*

In this section, we discuss the impact of probe factor $m$ on various metrics. Figure 4 presents the plots of the performance of our method corresponding to $L = 10$, $k = 2$ with different choices of $m$. In order to reduce memory occupied of hash tables, we increase $m$ to make up for the reduction in $\text{Pr}_{fn}$ caused by decreased $L$.

As $m$ increases, $\text{Pr}_{fn}$ and the recall gradually increase and eventually approach 1. But at the same time, the precision shows a decreasing trend, and F1-score also begins to decrease when it increases to about 0.7 (Fig. 4a).

In Fig. 4b, the number of candidates also increases with the increase of $m$, which increases the number of near neighbors in candidates. This reflected in the increase in recall and decrease in precision. Observing the F1-score, we can find that $m = 3$ is a critical value, which makes the F1-score reach the maximum. That is, when $m < 3$, the rate of increase of recall is faster than the rate of decrease of precision, but after that, the quality of candidate tends to decrease.

The hashing time depends on the number of hash tables $L$. As described above, we already fixed $L = 10$, so the hashing time grows slowly (Fig. 4c). But with the increasing of number of candidates, in order to find the nearest neighbor, we have to calculate the similarity between query object and all the points in candidates. The total query time is mainly composed of two parts, one is

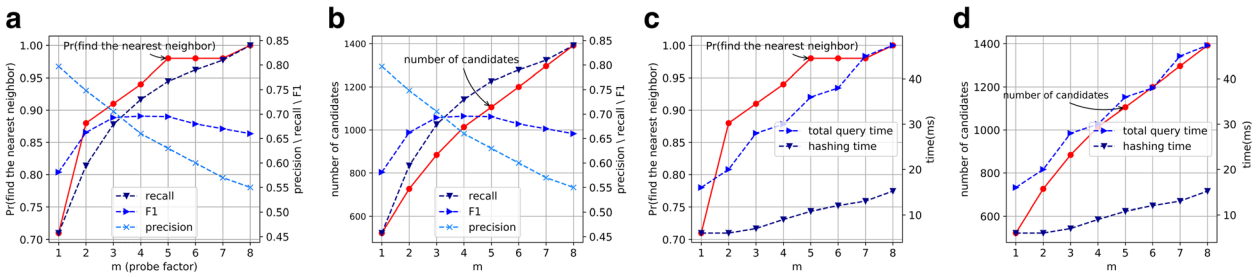


**Fig. 4** The effect of the probe factor $m$ on the algorithm. **a** $m$ versus the probability of find the nearest neighbor and precision/recall/F1. **b** $m$ versus the number of candidates and precision/recall/F1. **c** $m$ versus the probability of find the nearest neighbor and query/hashing time. **d** $m$ versus the number of candidates and query/hashing time

the hashing time and the other is the time of calculation of similarity. So, it is not hard to see the increase of number of candidates will inevitably lead to the increase of query time (Fig. 4d).

Based on the above experiments, we can conclude that for this experiment, $m = 3$ is a balanced value. Approximate nearest neighbor search algorithm needs to constantly balance the query time and the performance of search, like precision, recall. For example, in order to reduce the query time, we need to decrease the number of candidates, and this will inevitably lead to a decline in the recall.

### 5.4 The effect of the number of hash tables *L*

We need take an appropriate $L$ to ensure $\mathrm{Pr}_{fn}$. Figure 5 presents $L$ effects on the performance of the algorithm when $m = 1$, $k = 2$.

With the increase in $L$, there is no doubt that $\mathrm{Pr}_{fn}$will increase rapidly. But, unlike the impact of $m$ on the performance, the rate of precision declines very slowly (Fig. 5a). This shows that the increase of $L$ has less impact on the quality of the candidates. On the other word, OPRCP can guarantee that most (usually greater than 70%) points in candidates are the near neighbor of the query object. This feature can also makes the F1-score up to 0.8 (Fig. 5b).

In this experiment, the hashing time is rising quickly due to the increase in $L$. Although the hashing time accounts for a larger percentage of total query time, the query time still depends on the number of candidates (Fig. 5d).

### 5.5 The effect of the number of OPR bins *k*

In order to reduce the probability of collision for dissimilar points, we can increase the number of OPR bins $k$. Figure 6 shows that the performance of OPRCP corresponding to $L = 10$, $k = 2$, 4, comparing the performance of $m = \{1, 2, \ldots, 8\}$.

Increasing the value of $k$ , we decrease the probability of collision for any two points. Thus, the $\mathrm{Pr}_{fn}$ of $k = 2$ is always higher than $k = 4$ (Fig. 6a). Given $\mathrm{Pr}_{fn} = 0.9$, $k = 2$, it can achieve in $m = 3$, while $k = 4$ needs $m = 5$.

In Fig. 6b, as $m$ increases, the change trend of precision or recall is the same under both conditions. Specifically, increasing $m$ decreases the precision and increases the recall. We have analyzed the reason in 5.4. More interesting is that the trends are completely different in a comprehensive view. For $k = 2$, precision starts at 0.8, recall starts at 0.7, then, increasing the $m$ precision quickly dropped to 0.2 and the recall rose to 0.9. The gap between precision and recall is increasing. On the contrary, precision starts at 0.95 and recall starts at 0.55 in $k = 4$.

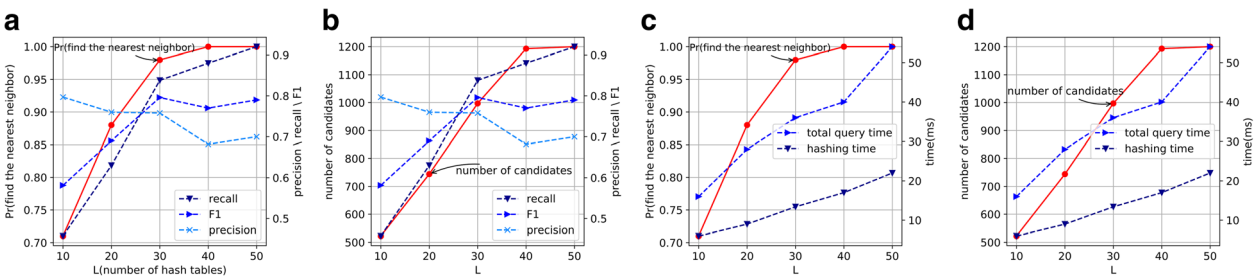


**Fig. 5** The effect of the number of hash table $L$ on the algorithm. **a** $L$ versus the probability of find the nearest neighbor and precision/recall/F1. **b** $L$ versus the number of candidates and precision/recall/F1. **c** $L$ versus the probability of find the nearest neighbor and query/hashing time. **d** $L$ versus the number of candidates and query/hashing time

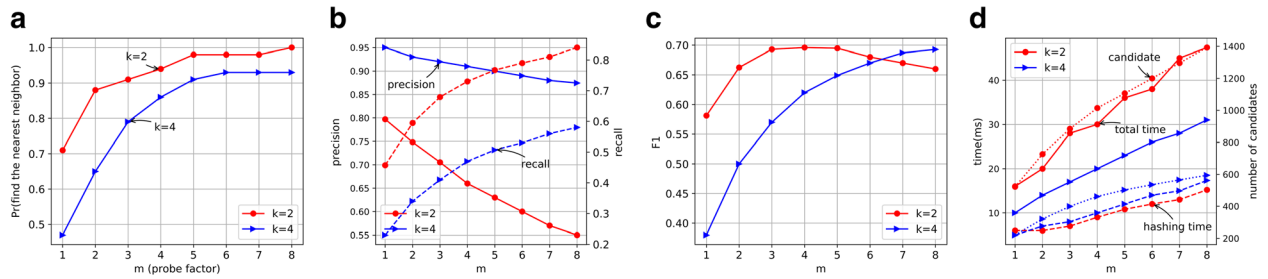


**Fig. 6** The effect of the bin number $k$ on the algorithm. **a** $m$ versus the probability of find the nearest neighbor and precision/recall/F1. **b** $m$ versus the number of candidates and precision/recall/F1. **c** $m$ versus the probability of find the nearest neighbor and query/hashing time. **d** $m$ versus the number of candidates and query/hashing time

Then, the precision slowly decreased to 0.75 and the recall increased to 0.6. The gap between them is decreasing. As $k$ increases, the probability of collision for dissimilar points will decrease, so the precision at $k = 4$ is always higher than $k = 2$. At the same time, the probability of collision for similar points will decrease too. This causes the recall at $k = 4$ is always lower than $k = 2$. So, we can adjust the value of $k$ according to different application requirements for each metric. For example, if we need high precision, we can set a bigger $k$, while a smaller $k$ should be taken if high recall is required.

We can further compare the precision and the recall under two conditions by F1-score. Figure 6c shows that when we achieve the same $\Pr_{fn}$ ($\Pr_{fn} = 0.9$, $m = 3$ at $k = 2$, $m = 5$ at $k = 4$), the F1-score of $k = 2$ is better than $k = 4$. With the increase in $m$, the gap between $k = 2$ and $k = 4$ gradually decreases due to the decline of F1-score of $k = 2$. This also reflects trends in both. More specifically, $k = 2$, the gap between the precision and recall is increasing, which leads to F1-score becoming smaller and smaller. $k = 4$, the gap between the precision and recall is getting smaller and more stable, which leads to F1-score increase and eventually converge.

Due to the higher precision, the number of candidates of $k = 4$ is always significantly less than $k = 2$. This makes the query time of $k = 4$ slightly lower than $k = 2$ when $\Pr_{fn}$ is the same (Fig. 6d).

### 5.6 Experimental results and analysis

Through experimental analysis of algorithm parameters, we will compare the algorithm on the real dataset and the synthetic dataset. The standard of setting parameter is the minimized query time, and $\Pr_{fn}$ must achieve at least 0.9.

Figures 7 and 8 show the comparisons of the various metrics on the synthetic dataset. Figure 7a shows OPRCP and TLOPRCP growth rates are lower than other algorithms in hashing time as dimension increases exponentially, which means that the two algorithms can solve the problem of "curse of dimensionality" better than others. Although TLBHL can achieve low time in low dimensions, its growth rate is gradually accelerating as dimension increases.

The number of candidates determines the amount of data to be calculated for similarity when finding the nearest neighbor. The number of candidates should be as small as possible, so as to get a smaller query time. In Fig. 7c, the candidates of BHL is the least and much lower than other algorithms. In order to get the least number of candidates, BHL used more numbers of hash

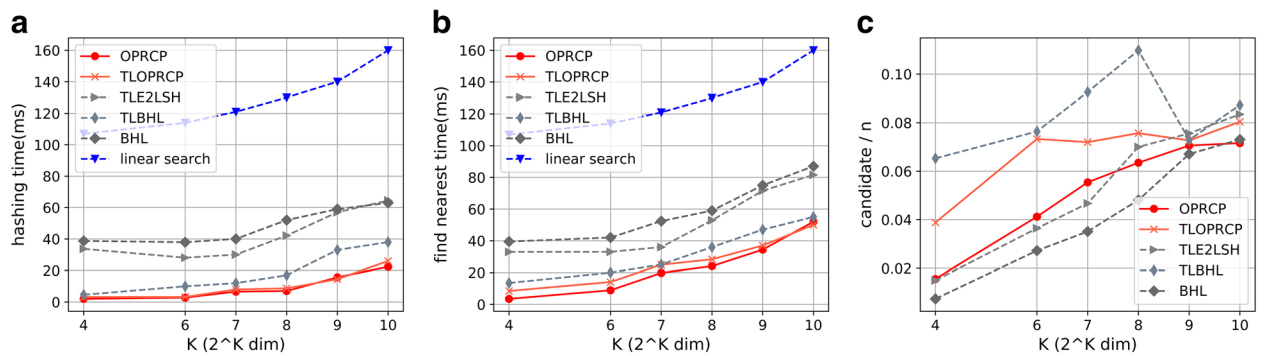


**Fig. 7** Comparison of algorithms in different dimensional data-1. **a** $k$ versus the hashing time between different methods. **b** $k$ versus the find nearest time between different methods. **c** $k$ versus the rate of candidates between different methods

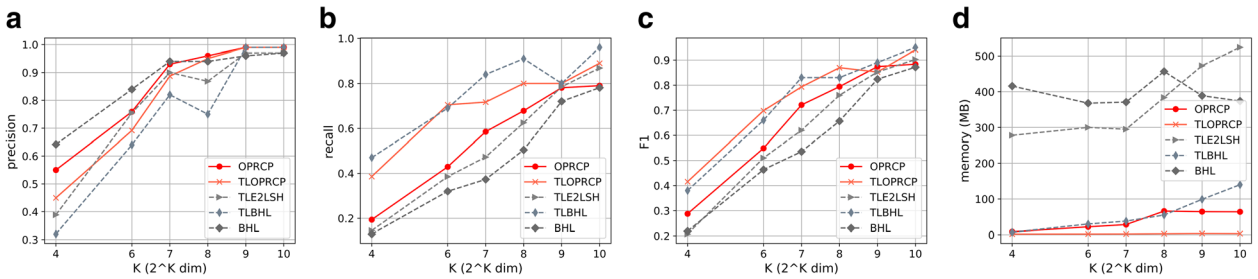


**Fig. 8** Comparison of algorithms in different dimensional data-2. **a** *k* versus the precision between different methods. **b** *k* versus the recall between different methods. **c** *k* versus the value of F1 between different methods. **d** *k* versus the cost of memory between different methods

functions in one hashing process. The more number of hash functions, the lower probability of collision but the more time of calculating hash value. In contrast, OPRCP uses less hash calculation time (Fig. 7a) to obtain fewer number of candidates (Fig. 7c). As a result, in (Fig. 7b), OPRCP and TLOPRCP achieve a modest speedup of 2–3 times compared to the other algorithm.

Precision and recall are important metrics for evaluating search algorithms; in order to make algorithm universal, OPRCP adopts a more balanced approach, which is not overly biased to one of the precision and recall. As the dimension increases, the precision and recall of each algorithm steadily increased (Fig. 8a). In particular, the precision of BHL is always higher than other algorithms; OPRCP is second only to BHL but has the least query time. The smaller number of candidates makes the recall of OPRCP not too high, but as the dimension increases, the gap in the recall of each algorithm is narrowing. Comparing with the F1-score, we found that the value of TLOPRCP is better than other methods. But for OPRCP, the low recall makes its F1-score only achieve the average value.

Figure 8d shows how the amount of space required to store hash tables changes as dimension increases. Due to the multi-probe query method being applied, OPRCP and TLOPRCP use less hash tables to achieve the specified $Pr_{fn}$, which makes them better than other methods in space occupied. Furthermore, as the dimension increases, the storage of OPRCP and TLOPRCP grows slowly, which makes the gap with other methods further widen.

Figure 9 compares metrics of each algorithm on MNIST dataset. TLOPRCP uses the least amount of hash calculation time (Fig. 9b), which stored the least amount of hash tables so as to minimize memory occupied (Fig. 9g). Basically, the OPRCP method presents as good as TLOPRCP in query time (Fig. 9a). The method ensures the highest precision (Fig. 9d) and has a few

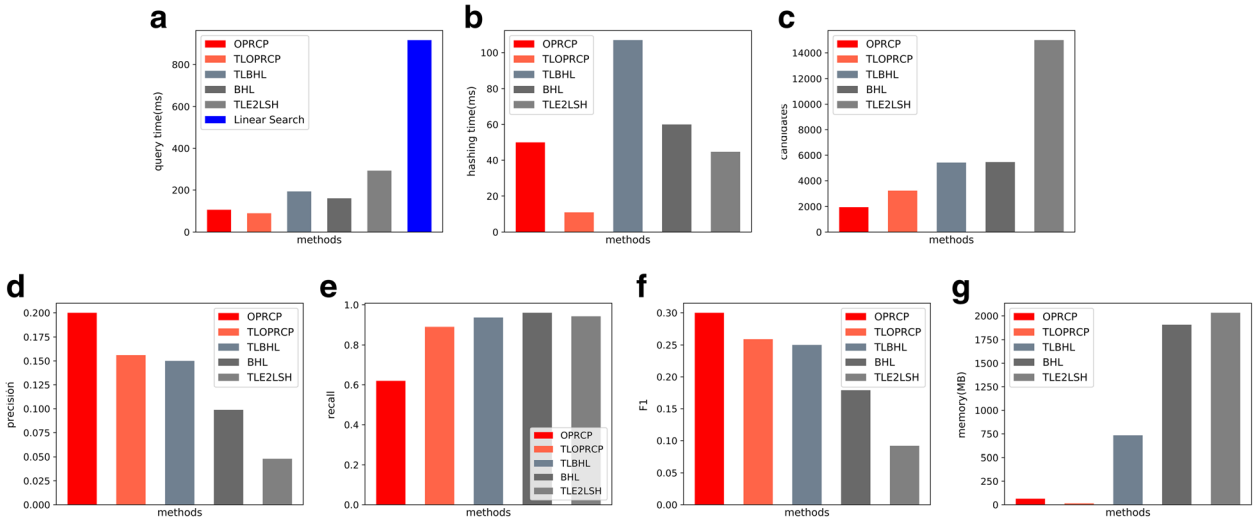


**Fig. 9** Comparison of algorithms in MNIST dataset. **a** *k* versus the find nearest time between different methods. **b** *k* versus the hashing time between different methods. **c** *k* versus the rate of candidates between different methods. **d** *k* versus the precision between different methods. **e** *k* versus the recall between different methods. **f** *k* versus the value of F1between different methods. **g** *k* versus the cost of memory between different methods

candidate sets (Fig. 9c); although it uses more hash tables than TLOPRCP, it is still less than other methods. Owing to the least number of candidates of OPRCP, OPRCP makes its recall lower than other methods (Fig. 9e), even though the F1-score of OPRCP is the best by comprehensively evaluating recall and precision.

In this section, we have carried out sufficient experiments on the proposed OPRCP method. These experimental results have shown that the OPRCP method is a very effective near neighbor hybrid search, and it can be applied to construct an effective WMSN data search intelligent contract within the WMSN blockchain application system.

## 6 Conclusions

In view of preventing sensitive data of WMSN IoT to be tampered, it is a feasible solution to build a blockchain application of WMSN IoT based on Ethereum and IPFS. Within the smart contract mechanism of this solution, it is of great significance to research on WMSN-based hybrid data query transactions. This paper aims at the search problem of high-dimensional, large-scale, and multi-type multimedia data out of WMSN IoT; we present a novel OPRCP method for binary hybrid approximate nearest neighbor search problem. Different from the existing work, we use a kind of LSH method to perform similarity search on the binary hybrid data and construct the query data structure; by preprocessing the dataset, we can obtain a more efficient dimension-insensitive query time. The curse of dimensionality of binary hybrid data can be effectively solved. Besides, multi-probe query method is used to solve the problem of massive space occupied in LSH. In particular, we further analyzed the relationship between the query time and the accuracy of query based on LSH for binary hybrid approximate nearest neighbor search. Therefore, we can choose different optimal parameters for different practical problems to achieve different effects required. A large number of experiments have proved that our method has lower query time and space consumption than previous methods, and it is universal for various datasets.

However, our method also has the disadvantage of a single query scenario, which is reflected in the inflexible data format. In the wireless multimedia sensor network application, engineering implementation still has a long way to go, and the application extension needs to be further studied. It is necessary to construct an index structure that is insensitive to data format to adapt to the query scenario of the WMSN blockchain application. In future works, we will try to construct a binary hybrid data search structure with missing values and extend the approximate nearest neighbor search to solve the problem of incomplete data.

### Abbreviations

CP: Cross-polytope LSH method; IoT: Internet of Things; IPFS: InterPlanetary File System; LSH: Locality-sensitive hashing; OPR: One Permutation with Rotation Hashing; OPRCP: One Permutation with Rotation and Cross-Polytope locality-sensitive hashing; TLOPRCP: OPRCP method of a two-layer filter structure; WMSN: Wireless multimedia sensor networks; WSN: Wireless sensor networks

### Acknowledgements

The authors would like to express their gratitude to the authors' institutions, for their many convenient conditions.

### Funding

This work was supported by the Great Project of Fujian Province Science and Technology Plan (grant number 2016H6007) and the City School Cooperation Project of Fuzhou Science and the Technology Bureau (grant number 2016-G-40).

### Availability of data and materials

- The MNIST dataset that support this study are available in MNIST repository: http://yann.lecun.com/exdb/mnist/
  The synthetic data that support this study was generated by the author based on experimental requirements and is available in github as follows,
- https://github.com/7thMar/synset/blob/master/OPRCP_synset.tar.gz.
- The data can be made available by sending a request via e-mail to xiaoruliang@fjnu.edu.cn.

### About the Authors

Huakun Liu was born in Fuyang City, Anhui, China, in 1996. He will be receiving his B.S. degree in software engineering from Fujian Normal University, Fujian, China, in 2019. Mr. Liu has received various scholarships. His current research interests include machine learning and search technology.

Xin Wei was born in Ningde City, Fujian, China, in 1998. She will be receiving her B.S. degree in software engineering from Fujian Normal University, Fujian, China, in 2019. Miss Wei has received various scholarships. Her current research interests include machine learning and search technology.

Ruliang Xiao was born in Loudi City, Hunan, China, in 1966. He is currently a Professor at the College of Mathematics and Informatics, Fujian Normal University (CN). He received his PhD in Computer Software and Theory from Wuhan University (CN) in 2007. His research interests include System Security Engineering (SSE) and Computing Intelligence (CI). He is the author of three books and more than 20 patents for invention and has published more than 50 papers in international journals and conference proceedings. He was awarded the prize of Fujian Provincial Science and Technology Progress Award in 2016.

Lifei Chen received his bachelor's degree in computer science from University of Electronic Science and Technology of China in 1993; his MSE from Tsinghua University, China, in 2004; and his PhD from Xiamen University, China, in 2008. He is a professor at the Department of Computer Science, Fujian Normal University, China. His research interests include machine learning and pattern recognition.

Du Xin is currently an Associate Professor at the College of Mathematics and Informatics, Fujian Normal University (CN). She received her PhD in Software Engineering from Wuhan University (CN) in 2010. Her research interests include Search-Based Software Engineering (SBSE) and evolutionary computation. She has published over 30 papers including GECCO and CEC international conference.

Zhang Shi is currently an Associate Professor at the College of Mathematics and Informatics, Fujian Normal University (CN). He received his PhD in Computer Science from Shanghai Jiaotong University (CN) in 2008. His research interests include Software Testing and Search-Based Software Engineering. He has published over 20 papers.

### Authors' contributions

HL, XW and RL are the main writers of this paper. HL, XW and RL proposed the main idea. HL and XW completed the experiments. XW analyzed the results. HL, XW, and RL discussed feasibility analysis. LC, XD, and SZ gave some important suggestions for this paper. All authors read and approved the final manuscript.

**Competing interests**
The authors declare that they have no competing interests.

## Publisher's Note



**Author details**
[1]College of Mathematics and Informatics, Fujian Normal University, Fuzhou 350117, China. [2]Digit Fujian Internet-of-Things Laboratory of Environmental Monitoring, Fuzhou 350117, China. [3]Fujian Provincial Key Laboratory of Network Security and Cryptology, Fuzhou 350007, China.

Received: 3 July 2018 Accepted: 2 August 2018
Published online: 20 August 2018

**References**

1. O. Novo, Blockchain meets IoT: an architecture for scalable access management in IoT. IEEE Internet Things J **5**(2), 1184–1195 (2018)
2. A. Reyna et al., On blockchain and its integration with IoT. Challenges and opportunities. Futur Gener Comput Syst **88**, 173–190 (2018)
3. Q. Xia et al., BBDS: blockchain-based data sharing for electronic medical records in cloud environments. Information **8**(2), 44 (2017)
4. A.C. Ekblaw, *MedRec: blockchain for medical data access, permission management and trend analysis* (Diss. Massachusetts Institute of Technology, Washington D.C., 2017)
5. M.S. Ali, K. Dolui, F. Antonelli, *IoT data privacy via blockchains and IPFS*, *International Conference on the Internet of Things* (ACM, New York, 2017)
6. H. Cheng et al., Data quality analysis and cleaning strategy for wireless sensor networks. EURASIP J Wirel Commun Netw **61**, 1–11 (2018)
7. Rifi, Nabil, et al. Towards using blockchain technology for IoT data access protection. *Ubiquitous Wireless Broadband (ICUWB), 2017 IEEE 17th International Conference on. IEEE*, 2017.
8. C. Böhm, S. Berchtold, D.A. Keim, Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. ACM Comput Surv **33**(3), 322–373 (2001)
9. A. Shrivastava, P. Li, Densifying one permutation hashing via rotation for fast near neighbor search. Int Conf Mach Learn **32**, I–557 (2014)
10. A. Andoni et al., Practical and optimal LSH for angular distance. Adv Neural Inf Proces Syst 1–9 (2015)
11. M. Datar et al., *Locality-sensitive hashing scheme based on p-stable distributions*, *Proceedings of the twentieth annual symposium on Computational geometry* (ACM, Brooklyn, 2004)
12. K. Weinberger et al., *Feature hashing for large scale multitask learning*, *Proceedings of the 26th Annual International Conference on Machine Learning* (ACM, Quebec, 2009)
13. Q. Lv et al., *Multi-probe LSH: efficient indexing for high-dimensional similarity search*, *Proceedings of the 33rd international conference on Very large data bases* (VLDB Endowment, Vienna, 2007)
14. A. Ouaddah, A.A. Elkalam, A.A. Ouahman, *Towards a novel privacy-preserving access control model based on blockchain technology in IoT*, *Europe and MENA Cooperation Advances in Information and Communication Technologies* (Springer, Cham, 2017)
15. IPFS – the permanent web. Retrieved May 18, 2017 from https://github.com/ipfs/ipfs. Accessed 16 Aug 2018
16. Y. Chen et al., *An improved P2P file system scheme based on IPFS and Blockchain*, IEEE International Conference on Big DataIEEE (2017), pp. 2652–2657
17. P. Indyk, R. Motwani, *Approximate nearest neighbors: towards removing the curse of dimensionality*, *Proceedings of the thirtieth annual ACM symposium on Theory of computing* (ACM, Dallas, 1998)
18. J.M. Kleinberg, *Two algorithms for nearest-neighbor search in high dimensions*, *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing* (ACM, Redmond, 1997)
19. E. Kushilevitz, R. Ostrovsky, Y. Rabani, Efficient search for approximate nearest neighbor in high dimensional spaces. SIAM J. Comput. **30**(2), 457–474 (2000)
20. A. Gionis, P. Indyk, R. Motwani, Similarity search in high dimensions via hashing. Vldb. **99**(6), 518–529 (1999)
21. S. Arya et al., An optimal algorithm for approximate nearest neighbor searching fixed dimensions. J ACM **45**(6), 891–923 (1998)
22. A.Z. Broder, *On the resemblance and containment of documents*, *Compression and Complexity of Sequences 1997. Proceedings* (IEEE), Salerno, p. 1997
23. C. Liu et al., *Cross domain search by exploiting wikipedia*, *Data Engineering (ICDE), 2012 IEEE 28th International Conference on* (IEEE, Washington D.C., 2012)
24. J. Fan et al., Seal: spatio-textual similarity search. Proc VLDB Endowment **5**(9), 824–835 (2012)
25. ZhuMing-dong, S. De-rong, N.T.-z. Kouyue, Y. Ge, A LSH-based method for similarity queries on binary hybrid data. Chin J Comput **40**, 1–15 (2017)
26. L. Chen et al., *Temporal spatial-keyword top-k publish/subscribe*, *Data Engineering (ICDE), 2015 IEEE 31st International Conference on* (IEEE, Seoul, 2015)
27. J. Lu, Y. Lu, C. Gao, *Reverse spatial and textual k nearest neighbor search*, *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data* (ACM, Athens, 2011)
28. S. Har-Peled, P. Indyk, R. Motwani, Approximate nearest neighbor: towards removing the curse of dimensionality. Theory Comput **8**(1), 321–350 (2012)
29. K. Terasawa, Y. Tanaka, *Spherical LSH for approximate nearest neighbor search on unit hypersphere*, Workshop on Algorithms and Data Structures (Springer, Berlin, Heidelberg, 2007)
30. A. Dasgupta, R. Kumar, T. Sarlós, *Fast locality-sensitive hashing*, *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining* (ACM, Sydney, 2011)
31. M. Slaney, Y. Lifshits, J. He, Optimal parameters for locality-sensitive hashing. Proc. IEEE **100**(9), 2604–2623 (2012)
32. UCI machine learning repository, MNIST, 1999. http://yann.lecun.com/exdb/mnist. Accessed 16 Aug 2018