Multi-Granularity Locality-Sensitive Bloom Filter

Jiangbo Qian, Qiang Zhu, *Senior Member, IEEE*, and Huahui Chen

Abstract—In many applications, such as homeland security, image processing, social network, and bioinformatics, it is often required to support an approximate membership query (AMQ) to answer a question like "is an (query) object *q* near to at least one of the objects in the given data set Ω?" However, existing techniques for processing AMQs require a key parameter, i.e., the distance value, to be defined in advance for the query processing. In this paper, we propose a novel filter, called multi-granularity locality-sensitive Bloom filter (MLBF), which can process AMQs with multiple distance granularities. Specifically, the MLBF is composed of two Bloom filters (BF), one is called basic multi-granularity locality-sensitive BF (BMLBF), and the other is called multi-granularity verification BF (MVBF). The BMLBF is used to store the data objects. It adopts an alignable locality-sensitive hashing (LSH) function family to support multiple granularities. The MVBF is used to reduce the false positive rate of the MLBF. The false negative rate of the MLBF is reduced by applying AND-constructions followed by an OR-construction. In addition, based on the MLBF structure, we suggest a more space-effective variant, called the MLBF*, to further reduce space cost. Theoretical analyses for estimating false positive/negative rates of the MLBF/MLBF* are given. Experiments using synthetic and real data show that the theoretical estimates are quite accurate, and the MLBF/MLBF* technique can handle AMQs with low false positive and negative rates for multiple distance granularities.

Index Terms—Approximate membership query, query processing, Bloom filter, locality-sensitive hashing, false positive/negative rates

1 Introduction

An approximate membership query (AMQ) is used to answer a question like "is an object *q* close to at least one of the objects in a given data set Ω?", where the closeness is measured by a given distance metric [1], [2]. There is an increasing demand to efficiently process such AMQs in numerous application domains including homeland security, network security, image processing, social network, and bioinformatics. For example, a homeland security officer may want to check if an unknown substance (with some observed characteristics) possibly belongs to a set of listed hazardous chemicals; a physician may wish to check if the unusual symptom of a patient shares some characteristics with a known disease in the past; a network manager would like to know if the behavior of a user demonstrates the characteristics of some intruders identified in the past; a judge for a photo competition may want to check if a submitted photo is similar to one of existing photos from a large image database.

The above application examples showcase several main characteristics of such AMQs: (1) the comparison of a query object *q* with an object in the given data set Ω is based on multiple dimensions/features; (2) a large number of such query objects have to be processed efficiently; (3) the answers to most of AMQs are negative. Clearly, it is desirable to have an efficient processing method that can quickly answer AMQs. This method can act as a filter to remove a large number of negative AMQs (i.e., query objects) from consideration. The survived AMQs (with a positive answer) can be further investigated by running a conventional query on data set Ω to identify the specific object(s) in Ω that is (are) close to the corresponding query object. Since a conventional query is typically expensive to run, filtering unnecessary queries is an important mechanism to significantly improve the overall performance of the system.

The AMQ problem has attracted research attention recently. The existing work for processing AMQs in a multidimensional space includes the distance-sensitive Bloom filter (DSBF) [2] and the locality-sensitive Bloom filter (LSBF) [1]. The DSBF is the first work to process AMQs by utilizing an integrated method of locality-sensitive hashing (LSH) [3] and Bloom filter (BF) [4]. The DSBF achieves improvement in both time and space, comparing to the naive method that performs costly comparisons of the given query object against the entire given data object set. The LSBF also uses LSH functions to construct a BF to answer an AMQ. Furthermore, it employs an additional verification BF to decrease the false positive rate.

One limitation for the DSBF and the LSBF is that they can handle only a predetermined distance value for a given set of AMQs on a given set Ω of data objects. This distance value is a key parameter that indicates how close a query object *q* is to at least one of the (data) objects in set Ω is acceptable. However, identifying such an appropriate distance value is a nontrivial task due to not only the challenges of capturing the application semantics but also the probabilistic properties of the LSH utilized in the proposed techniques. Too large or too small a distance value may result in unacceptable query results [1]. Unfortunately, the DSBF and the LSBF can support only one distance value. In other words, once a distance value is determined, it cannot be changed unless the BF structure is rebuilt.

To overcome the above problem, in this paper, we propose a new technique, called the multi-granularity locality-sensitive Bloom filter (MLBF), to support multiple distance

• *J. Qian and H. Chen are with the School of Information Science and Engineering, Ningbo University, Zhejiang 315211, China. E-mail: {qianjiangbo, chenhuahui}@nbu.edu.cn.*
• *Q. Zhu is with the Department of Computer and Information Science, University of Michigan, Dearborn, MI 48128. E-mail: qzhu@umich.edu.*

Manuscript received 12 Jan. 2014; revised 20 Sept. 2014; accepted 26 Jan. 2015. Date of publication 5 Feb. 2015; date of current version 11 Nov. 2015. Recommended for acceptance by A. El Abbadi.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TC.2015.2401011

values in processing AMQs without having to rebuild the filter structure for each distance value. The MLBF structure is composed of a basic multi-granularity locality-sensitive BF (BMLBF) and a multi-granularity verification BF (MVBF). The BMLBF is used to store data objects, while the MVBF is used to verify the membership of a data object by keeping the correlation information among the BMLBF locations used for the same data object. Both BMLBF and MVBF are designed to have multiple granularities that can be used to support AMQs with different distance values for their closeness evaluation. Since we adopt strategies to nicely embed information at a coarser granularity (corresponding to a larger distance value) into one at a finer granularity (corresponding to a smaller distance value), the space needed by our MLBF structure is only for keeping the information at the finest granularity without having to duplicate the space for each supported granularity. In addition, based on the MLBF structure, we suggest a more space-effective variant, called the MLBF*, to further reduce space cost.

To develop the MLBF/MLBF* technique, we encounter the following challenges: (1) how to select an LSH function family which can support multiple granularities; (2) how to incorporate a false positive verification scheme which can handle different granularities; (3) what schemes to adopt for controlling false negative rates; (4) how to derive theoretical models to estimate false positive and negative rates.

The main contributions of this paper are the following:

1. Novel MLBF and MLBF* structures using multiple granularities to support the efficient processing of AMQs with different distance values for their closeness evaluation are proposed. In particular, the relevant strategies to share storage space among different granularities and algorithms for structure construction and query processing are suggested.
2. Theoretical analyses for choosing proper LSH functions for the MLBF/MLBF* and estimating false positive and negative rates are presented.
3. Experiments on real and synthetic data were performed, which show that the theoretical estimates are accurate and the MLBF and MLBF* structures can efficiently and effectively process AMQs with high accuracy.

The rest of the paper is organized as follows. Section 2 gives the preliminaries. Section 3 briefly illustrates the proposed MLBF structure. Section 4 presents an alignable LSH function family that can handle multiple granularities. Section 5 introduces the verification BF with multiple granularities and the MLBF* structure. Section 6 provides theoretical analyses and models to estimate false positive and negative rates. Section 7 shows the experimental evaluation. Section 8 discusses some related work. Finally, Section 9 concludes the paper.

2 Preliminaries and Problem Description

In this section, we review some basic knowledge of the LSH and the Bloom filter that is essential to our proposed technique.

2.1 Locality Sensitive Hashing

An LSH [3] can keep the locality of objects in a data set by projecting close objects into the same hash bucket with a high probability. This is achieved by choosing a hash function(s) from a given LSH function family. The meaning of "locality sensitive" can be defined as follows.

Definition 1 (Locality Sensitive). *An LSH function family $\mathcal{H} = \{h : R^d \to U\}$ is called (r_1, r_2, P_1, P_2)-sensitive if, for any $\boldsymbol{o}, \boldsymbol{q} \in R^d$,*

- $\Pr[h(\boldsymbol{o}) = h(\boldsymbol{q})] \geq P_1$ when $\|\boldsymbol{o} - \boldsymbol{q}\| \leq r_1$,
- $\Pr[h(\boldsymbol{o}) = h(\boldsymbol{q})] \leq P_2$ when $\|\boldsymbol{o} - \boldsymbol{q}\| > r_2$,

where $1 > P_1 > P_2 > 0$, $r_2 > r_1 > 0$, $\|*\|$ denotes the Euclidean norm, and $\Pr[]$ denotes the probability of collision.

One such LSH family for the Euclidean distance is shown as follows [5]:

$$\mathcal{H}^{(\boldsymbol{a},b)} = \left\{ h^{(\boldsymbol{a},b)}(\boldsymbol{q}) | h^{(\boldsymbol{a},b)}(\boldsymbol{q}) = \left\lfloor \frac{\boldsymbol{a} \cdot \boldsymbol{q} + b}{w} \right\rfloor \right\}, \tag{1}$$

where $h^{(\boldsymbol{a},b)}(\boldsymbol{q})$ represents an LSH in family $\mathcal{H}^{(\boldsymbol{a},b)}$; $\boldsymbol{q}$ is the vector representation of an object in R^d; $\boldsymbol{a}$ is a d-dimensional vector whose component values are chosen independently from a p-stable distribution (e.g., the standard Gaussian distribution $N(0,1)$); w is a user-specified constant; and b is a real number drawn uniformly at random from $[0,w)$. Note that the dot product $\boldsymbol{a} \cdot \boldsymbol{q}$ represents the projection of $\boldsymbol{q}$ onto $\boldsymbol{a}$. The vector (line) $\boldsymbol{a}$ is split into equi-length segments with an appropriate size w, which intuitively preserves the locality, i.e., two close objects will be in the same segment with a high probability.

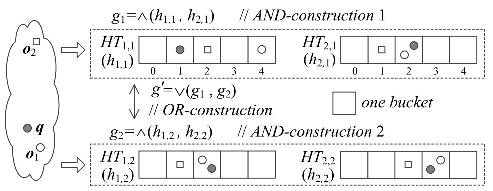
The collision probability, $p^{(\boldsymbol{a},b)}(c_i)$, for objects $\boldsymbol{o}_i$ and $\boldsymbol{q}$ under a hash function $h^{(\boldsymbol{a},b)}$ chosen uniformly and randomly from $\mathcal{H}^{(\boldsymbol{a},b)}$, can be computed as [5]:

$$p^{(\boldsymbol{a},b)}(c_i) = \Pr[h^{(\boldsymbol{a},b)}(\boldsymbol{o}_i) = h^{(\boldsymbol{a},b)}(\boldsymbol{q})] = \int_0^w \frac{1}{c_i} f_1\left(\frac{t}{c_i}\right)\left(1 - \frac{t}{w}\right) dt, \tag{2}$$

where $c_i = \|\boldsymbol{o}_i - \boldsymbol{q}\|$, $f_1(t)$ is the probability density function of the absolute value of the p-stable distribution, e.g., $f_1(t) = \sqrt{\frac{2}{\pi}} e^{\frac{-t^2}{2}}$, $t \geq 0$, for the Gaussian distribution.

From Formula 2, we can see that, for a fixed parameter w, the probability of collision decreases monotonically with the increase of c_i. In fact, $\mathcal{H}^{(\boldsymbol{a},b)}$ is (r_1, r_2, P_1, P_2)-sensitive for $P_1 = p^{(\boldsymbol{a},b)}(1)$, $P_2 = p^{(\boldsymbol{a},b)}(c_i)$ and $r_2/r_1 = c_i$ [5].

However, one cannot directly use just one function from $\mathcal{H}$ for an LSH, since the gap between probabilities P_1 and P_2 is usually quite small. To construct a useful LSH with desired collision probabilities P_1' and P_2', an "amplification" process, i.e., several AND-constructions followed by an OR-construction [6], is needed. The process can amplify the gap between the "high" probability P_1 and the "low" probability P_2 by concatenating several functions. Specifically, for given parameters k and L, kL hash functions $h_{t,j}(\boldsymbol{q})$ $(1 \leq t \leq k, 1 \leq j \leq L)$ are chosen independently and uniformly at random from $\mathcal{H}$. An AND-construction $g_j(\boldsymbol{q})$ $(j \in \{1, 2, \ldots, L\})$ is defined as $g_j(\boldsymbol{q}) = \wedge(h_{1,j}(\boldsymbol{q}), \ldots, h_{k,j}(\boldsymbol{q}))$ for an object $\boldsymbol{q}$, where $g_j(\boldsymbol{o}) = g_j(\boldsymbol{q})$ for two objects $\boldsymbol{o}$ and $\boldsymbol{q}$ if and only if $(h_{t,j}(\boldsymbol{o}) = h_{t,j}(\boldsymbol{q}))$ for any $t \in \{1, 2, \ldots, k\}$. As each hash function $h_{t,j}()$ in $g_j()$ defines a hash table, the AND-construction can be viewed as applying an AND operation to its k hash



$g_1=\wedge(h_{1,1}, h_{2,1})$ // AND-construction 1
$HT_{1,1}$ ($h_{1,1}$)
$HT_{2,1}$ ($h_{2,1}$)
0 1 2 3 4
$g'=\vee(g_1, g_2)$ // OR-construction
one bucket
$g_2=\wedge(h_{1,2}, h_{2,2})$ // AND-construction 2
$HT_{1,2}$ ($h_{1,2}$)
$HT_{2,2}$ ($h_{2,2}$)
o_2
q
o_1

Fig. 1. An example of the LSH approach.

tables. Two objects are considered to be close to each other under $g_j()$ if and only if each of its hash tables places them in the same bucket. An AND-construction increases the probability for identifying two true close objects, i.e., decreasing the false positive rate. However, the AND-construction may lead to a high false negative rate. To mitigate the problem, an OR-construction is used. Instead of relying on one AND-construction, an OR-construction is formed with L different AND-constructions, i.e., $g'(\boldsymbol{q}) = \vee(g_1(\boldsymbol{q}), \ldots, g_L(\boldsymbol{q}))$, such that $g'(\boldsymbol{o}) = g'(\boldsymbol{q})$ for two objects $\boldsymbol{o}$ and $\boldsymbol{q}$ if and only if $(g_j(\boldsymbol{o}) = g_j(\boldsymbol{q}))$ for some $j \in \{1, 2, \ldots, L\}$. With such an AND-and-OR construction, we can turn (r_1, r_2, P_1, P_2)-sensitive family into a (r_1, r_2, P'_1, P'_2)-sensitive family where $P'_1 = 1 - (1 - P_1^k)^L$ and $P'_2 = 1 - (1 - P_2^k)^L$ [6]. We could adjust k and L to push P'_1 closer to 1 and P'_2 closer to 0.

Fig. 1 illustrates an LSH $g' = \vee(g_1, g_2)$ using four Hash Tables (HT). Each table uses one hash function to map/store objects into one of the five buckets. For example, object $\boldsymbol{o}_1$ is mapped in bucket $h_{1,1}(\boldsymbol{o}_1) = 4$ of $\text{HT}_{1.1}$, and is mapped in bucket $h_{2,1}(\boldsymbol{o}_1) = 2$ of $\text{HT}_{2.1}$. $g_1 = \wedge(h_{1,1}, h_{2,1})$, i.e., AND-construction 1, is composed of two hash tables $h_{1,1}$ and $h_{2,1}$. $g_2 = \wedge(h_{1,2}, h_{2,2})$, i.e., AND-construction 2, is composed of another two hash tables $h_{1,2}$ and $h_{2,2}$. Since query object $\boldsymbol{q}$ is close to object $\boldsymbol{o}_1$, they are mapped into one bucket of a hash table with a high probability. In contrast, object $\boldsymbol{q}$ has a low chance to be mapped in one bucket with object $\boldsymbol{o}_2$ due to their large Euclidean distance. As $g_2(\boldsymbol{o}_1) = g_2(\boldsymbol{q})$, $\boldsymbol{q}$ is considered to be close to $\boldsymbol{o}_1$ under LSH g'.

2.2 Standard Bloom Filter

A standard Bloom filter is an m-bit array representing an n-element/object set. All bits in the array are initially set to 0. k hash functions, say, $h_1, h_2, \ldots, h_k$, are used to hash an object to k random locations of the *m-bit* array with a uniform distribution. For each object x of the set, locations $h_1(x)$, $h_2(x)$, $\ldots$, $h_k(x)$ of the array are set to 1. To check whether an object y is a member of the set, one needs to check whether all $h_i(y)$ are 1 ($1 \leq i \leq k$). If so, y is regarded as a member of the set. False positives are possible. However, a false negative is impossible. The false positives rate can be defined as follows.

Definition 2 (False positive rate). *For a given set of query objects, its false positive rate* FPR = *(*number of false positive objects*)/(*number of truly negative objects*).*

The false positive rate can be calculated by the following formula [7],

$$FPR = (1 - (1 - 1/m)^{kn})^k \approx (1 - e^{-kn/m})^k. \quad (3)$$

In general, the false positive rate can be controlled by the parameters n, m, and k. When m and n are given, the optimal number of hash functions is given as follows:

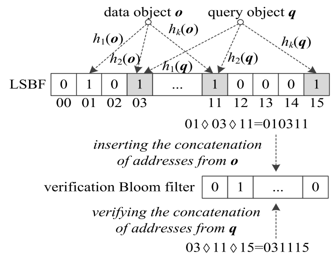


Fig. 2. The LSBF structure.

$$k = \ln 2 \times (m/n). \quad (4)$$

In this case, the false positive rate is: $FPR \approx 0.6185^{m/n}$.

2.3 Approximate Membership Query and Locality Sensitive Bloom Filter

Definition 3 (Approximate membership query). *Given a set Ω of objects, a query object $\boldsymbol{q}$, and a parameter (distance) $r > 0$, $\boldsymbol{q}$ is regarded as an approximate member of Ω if $\exists \boldsymbol{o} \in \Omega$ such that $\|\boldsymbol{o} - \boldsymbol{q}\| \leq r$, i.e., the answer to the corresponding AMQ (for $\boldsymbol{q}$) is positive [1].*

For a (r_1, r_2, P_1, P_2)-sensitive LSH family, we have the following definitions.

Definition 4 (False positive of AMQ). *A query object $\boldsymbol{q}$ is a false positive to object set Ω if the query gets a positive answer while in fact $\forall \boldsymbol{o} \in \Omega$, $\|\boldsymbol{o} - \boldsymbol{q}\| > r_2$ for a given parameter r_2.*

Definition 5 (False negative of AMQ). *A query object $\boldsymbol{q}$ is a false negative to object set Ω if the query gets a negative answer while in fact $\exists \boldsymbol{o} \in \Omega$, $\|\boldsymbol{o} - \boldsymbol{q}\| \leq r_1$ for a given parameter r_1.*

The false positive rate can be defined in the same way as Definition 2. The false negative rate is defined as follows:

Definition 6 (False negative rate). *For a given set of query objects (AMQs), its false negative rate* FNR = *(*number of false negative objects*)/(*number of truly positive objects*).*

Two techniques, i.e., the DSBF [2] and the LSBF [1], have been suggested to combine the LSH and the BF to process AMQs. The latter was an improvement over the former. Fig. 2 shows the basic structure for the LSBF technique. In total, k locality-sensitive hash functions $h_i (1 \leq i \leq k)$ are chosen to hash an object into k bits in a bit array. Once all the objects of a data set are inserted /hashed, the array is a summary vector to support AMQs on the set. When an AMQ for query object $\boldsymbol{q}$ is issued, it is processed like an insertion by hashing $h_i(\boldsymbol{q})$ $(1 \leq i \leq k)$ to k locations. If the bits at all k locations in the summary vector were set to 1, the answer to the AMQ for $\boldsymbol{q}$ is determined to be positive. However, this basic scheme may lead to many false positives. For example, the 1's in shaded locations in Fig. 2 are not set by a single data object. But in this case, query object $\boldsymbol{q}$ may be considered to have a positive answer because all h_i $(1 \leq i \leq k)$ are 1. Therefore, the LSBF employs an additional verification BF to decrease the false positive rate. The mapped addresses/locations from a single object are concatenated as a string, which is hashed into another standard BF for verification. Using such a scheme, an AMQ for object $\boldsymbol{q}$ needs to check the LSBF

TABLE 1
Important Notations

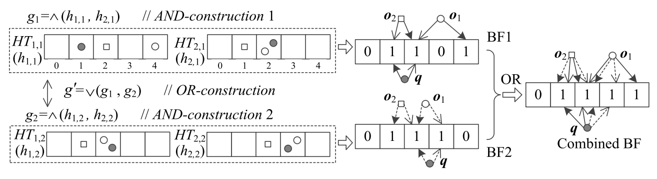
Ω: A set of data objects for Approximate Membership Query.
$\boldsymbol{o}, \boldsymbol{q}$: Two d-dimensional (data and query) objects.
$\|*\|$: The Euclidean norm.
c_i: Euclidean distance between two objects $\boldsymbol{o}_i$ and $\boldsymbol{q}$.
$\boldsymbol{a}$: A d-dimensional object whose component values are chosen randomly and independently based on the Gaussian distribution.
w: A user-specified constant which positively correlated with the query distance.
S: Number of different granularities.
θ: Granularity level ($0 \le \theta \le S-1$).
$\mathcal{H}^{(\boldsymbol{a},b)}$: LSH function family with functions of $h^{(\boldsymbol{a},b)}(\boldsymbol{q}) = \lfloor (\boldsymbol{a} \cdot \boldsymbol{q} + b)/w \rfloor$.
$\mathcal{H}^{(\boldsymbol{a})}$: Alignable LSH function family with functions of $h^{(\boldsymbol{a})}(\boldsymbol{q}) = \lfloor (\boldsymbol{a} \cdot \boldsymbol{q})/w \rfloor$.
$h_{t,j}$: An LSH function from $\mathcal{H}^{(\boldsymbol{a})}$, $t \in \{1, 2, \dots, k\}$, $j \in \{1, 2, \dots, L\}$.
n, m, k: Parameters for an m-bit BMLBF representing an n-element set using k LSH functions.
n', m', k': Parameters for an m'-bit MVBF representing an n'-element set using k' hash functions.
$g_j(\boldsymbol{q})$: The jth AND-construction $g_j(\boldsymbol{q}) = \wedge(h_{1,j}(\boldsymbol{q}), \dots, h_{k,j}(\boldsymbol{q}))$.
$\diamond$: The cancatenation operator for two binary string values.
$\xi_j(\boldsymbol{q})$: $j \diamond h_{1,j}(\boldsymbol{q}) \diamond \dots \diamond h_{k,j}(\boldsymbol{q})$ for the jth AND-construction.
p_i: The collision probability for hashing two objects $\boldsymbol{o}_i$ and $\boldsymbol{q}$ into one vector $\boldsymbol{a}$. That is, $p_i = \Pr[h^{(\boldsymbol{a})}(\boldsymbol{o}_i) = h^{(\boldsymbol{a})}(\boldsymbol{q})]$ for an LSH function $h^{(\boldsymbol{a})}(x)$ in $\mathcal{H}^{(\boldsymbol{a})}$.
p'_i: The collision probability for hashing two objects $\boldsymbol{o}_i$ and $\boldsymbol{q}$ into two vectors $\boldsymbol{a}'$ and $\boldsymbol{a}''$. That is, $p'_i = \Pr[h^{(\boldsymbol{a}')}(\boldsymbol{o}_i) = h^{(\boldsymbol{a}'')}(\boldsymbol{q})]$ for two LSH functions $h^{(\boldsymbol{a}')}(x)$ and $h^{(\boldsymbol{a}'')}(x)$ in $\mathcal{H}^{(\boldsymbol{a})}$.

basic array to see if all hashed bits are 1. If it is the case, the concatenation of its addresses is further hashed into the verification BF to check if the bits at the hashed locations for the concatenated addresses are also 1. If it is, object $\boldsymbol{q}$ is regarded to have a positive answer to its AMQ. Like the multi-probe LSH [8], the LSBF also performs the multiple probes on neighboring buckets/bits to improve the query accuracy. However, the LSBF have two shortcomings: (1) since OR-constructions are not used in the LSBF structure, the false negative rate may be high; (2) the LSBF structure cannot support AMQs with multiple distance values.

To overcome these shortcomings, we present a new method and its variant, i.e., the MLBF and the MLBF*, based on the BF and the LSH with multiple granularities to process AMQs, which are to be discussed in the subsequent sections.

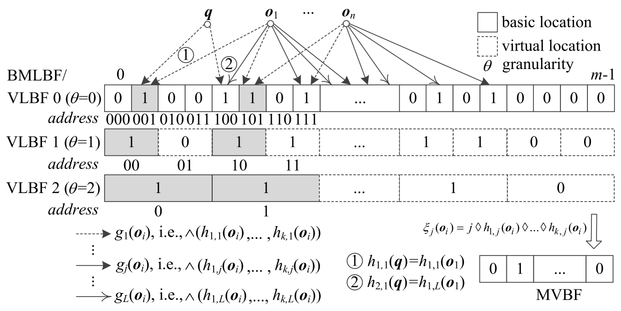
2.4 Important Notations

Table 1 lists some notations used in the paper.



$g_1=\wedge(h_{1,1}, h_{2,1})$ // AND-construction 1
$HT_{1,1}$ ($h_{1,1}$)
$HT_{2,1}$ ($h_{2,1}$)
$g'=\vee(g_1, g_2)$ // OR-construction
$g_2=\wedge(h_{1,2}, h_{2,2})$ // AND-construction 2
$HT_{1,2}$ ($h_{1,2}$)
$HT_{2,2}$ ($h_{2,2}$)
BF1
BF2
OR
Combined BF

Fig. 3. Transform LSH buckets to a BF structure.



basic location
virtual location
θ granularity
BMLBF/ VLBF 0 (θ=0)
VLBF 1 (θ=1)
VLBF 2 (θ=2)
address
$g_1(\boldsymbol{o}_i)$, i.e., $\wedge(h_{1,1}(\boldsymbol{o}_i),\dots,h_{k,1}(\boldsymbol{o}_i))$
$g_j(\boldsymbol{o}_i)$, i.e., $\wedge(h_{1,j}(\boldsymbol{o}_i),\dots,h_{k,j}(\boldsymbol{o}_i))$
$g_L(\boldsymbol{o}_i)$, i.e., $\wedge(h_{1,L}(\boldsymbol{o}_i),\dots,h_{k,L}(\boldsymbol{o}_i))$
① $h_{1,1}(\boldsymbol{q})=h_{1,1}(\boldsymbol{o}_1)$
② $h_{2,1}(\boldsymbol{q})=h_{1,L}(\boldsymbol{o}_1)$
$\xi_j(\boldsymbol{o}_i) = j \diamond h_{1,j}(\boldsymbol{o}_i) \diamond \dots \diamond h_{k,j}(\boldsymbol{o}_i)$
MVBF

Fig. 4. The MLBF structure.

3 MLBF Structure

For each AND-construction (e.g., in Fig. 1), we use the k locality-sensitive hash functions to build a BF with the buckets in hash tables being replaced with bit locations. For example, as object $\boldsymbol{o}_1$ is mapped in bucket 4 of $\text{HT}_{1.1}$ and bucket 2 of $\text{HT}_{2.1}$, the locations 4 and 2 of BF1 are set to 1. Here $k = 2$. Then, an AMQ can be answered by checking all such L different BFs. Once one of the BF results is positive, the answer is positive. However, as this scheme employs multiple BFs which require more space and calculation, we combine all the L BFs into a combined BF using an OR operation as shown in Fig. 3.

Because the locations in the above combined BF to which an AMQ (query object) is mapped may be set to 1 by different AND-constructions, i.e., causing a possible false positive, an additional BF for verifying each AND-construction is employed to reduce the false positive rate. Furthermore, a directly combined BF like the one in Fig. 3 cannot support multiple (distance) granularities. To solve these problems, we have carefully designed our new MLBF structure which is shown in Fig. 4.

The MLBF is composed of a BMLBF (which is similar to the combined BF in Fig. 3) with m physical bit locations for storing objects, and an MVBF for verifying set membership. The BMLBF contains $S-1$ Virtual LSH BFs (VLBF): VLBF 1, $\dots$, VLBF $S-1$, which are virtually built from the BMLBF rather than physically exist and are composed of $m/2^1$, $m/2^2, \dots, m/2^{S-1}$ virtual bit locations, respectively. The basic level BMLBF can also be considered as VLBF 0. Specifically, one (bit) location for VLBF θ ($1 \le \theta \le S-1$) covers 2^1 locations for VLBF $\theta - 1$, 2^2 locations for VLBF $\theta - 2, \dots, 2^\theta$ locations for VLBF 0. If one of the locations in the VLBF 0 is set to 1, its covering virtual location in each of the VLBFs is regarded as 1 too (see the shaded locations in Fig. 4).

The procedure of inserting an object $\boldsymbol{o}_i$ from set Ω into the MLBF is as follows. Assume that there are L AND-constructions and each has k hash functions $h_{1,j}$, $\dots$, $h_{k,j}$ ($j = 1, 2, \dots, L$) from a hash function family used for the BMLBF. For each j, we set the k locations of BMLBF (i.e., VLBF 0) at addresses $h_{1,j}(\boldsymbol{o}_i)$,$\dots$, $h_{k,j}(\boldsymbol{o}_i)$ to 1. We then store $\xi_j(\boldsymbol{o}_i) = j \diamond h_{1,j}(\boldsymbol{o}_i) \diamond \dots \diamond h_{k,j}(\boldsymbol{o}_i)$ into the MVBF, where '$\diamond$' denotes the concatenation of the binary string values, e.g., "1"$\diamond$"001"$\diamond$"101"$\diamond \dots =$ "1001101..." for AND-construction 1 of $\boldsymbol{o}_1$ in VLBF 0 in Fig. 4.

When an AMQ for query object $\boldsymbol{q}$ is issued, a VLBF θ (i.e., at granularity (level) θ) is determined based on the application requirement. For each $j = 1, 2, \dots, L$, we first check if all the k locations of the VLBF θ at addresses $h'_{1,j}(\boldsymbol{q}), \ldots, h'_{k,j}(\boldsymbol{q})$ are 1. If so, we further hash $j \diamond h'_{1,j}(\boldsymbol{q}) \diamond \ldots \diamond h'_{k,j}(\boldsymbol{q})$ to check whether it is a member of MVBF. If there exists one j such that both of its BMLBF and MVBF results are positive, the object $\boldsymbol{q}$ is regarded as an approximate member of Ω at granularity (level) θ.

It needs to be pointed out that two corresponding LSH functions for VLBFs at different granularities are different, e.g., $h'_{1,j} \neq h_{1,j}$, in the above discussion. In our MLBF structure, each bit location of a VLBF at one granularity merges two neighbor bit locations of the VLBF at the next finer granularity so that the space of the latter VLBF is reused by the former VLBF. To realize this strategy, we expect the LSH functions for the two VLBFs to be alignable in the sense of that objects hashed into two neighbor locations of the finer VLBF are hashed into the corresponding merged location of the coarser VLBF, and vice versa. For the classical LSH functions $h^{(\boldsymbol{a},b)}(\boldsymbol{o}) = \lfloor(\boldsymbol{a} \cdot \boldsymbol{o} + b)/w\rfloor$ where $b \in [0, w)$, parameter w is adjusted when handling a different distance value for AMQs. However, we cannot directly use this classical LSH function family since its functions are not alignable due to parameter b. If w is doubled for a specific VLBF, the parameter b may not be doubled as b is a random number chosen from [0, w). Therefore, we need to find an alignable LSH function family for our MLBF structure.

Another critical issue is how to reduce the false positive and negative rates. In traditional LSH methods, a collision between two objects $\boldsymbol{o}_i$ and $\boldsymbol{q}$ may occur when the results are the same after being processed by the same hash function (referring to Fig. 4 ①), i.e., $h_{t,j}(\boldsymbol{o}_i) = h_{t,j}(\boldsymbol{q})$. However, as we combine hash tables (BFs) into one BMLBF, another type of collision could occur (referring to Fig. 4 ②), that is $h_{t,j}(\boldsymbol{o}_i) = h_{t',j'}(\boldsymbol{q})$, where $t \neq t'$ and/or $j \neq j'$. Such a collision may increase the false positive rate. To reduce the false positive rate at different filter (distance) granularities, we use the MVBF to add a verification measure. The MVBF only stores $\xi_j(\boldsymbol{o}_i) = j \diamond h_{1,j}(\boldsymbol{o}_i) \diamond \ldots \diamond h_{k,j}(\boldsymbol{o}_i)$ at the finest granularity. However, such a concatenation can also represent verification information at other granularities. We will introduce the MVBF in Section 5. On the other hand, for each object $\boldsymbol{o}_i$ in set Ω, as we apply L AND-constructions followed by an OR-construction to map $\boldsymbol{o}_i$ into the BMLBF, the false negative rate will decrease.

4 Alignable LSH Function Family

In our MLBF structure, we modify the classical LSH family $\mathcal{H}^{(\boldsymbol{a},b)}$ in Formula 1 as $\mathcal{H}^{(\boldsymbol{a})} = \{h^{(\boldsymbol{a})}(\boldsymbol{q}) | h^{(\boldsymbol{a})}(\boldsymbol{q}) = \lfloor\frac{\boldsymbol{a} \cdot \boldsymbol{q}}{w}\rfloor\}$.

A function from such a new LSH family supports the merging operation of two neighbor locations from a finer VLBF to a coarser VLBF (i.e., alignable). However, omitting b from classical family $\mathcal{H}^{(\boldsymbol{a},b)}$may also cause a problem. For example, consider $\boldsymbol{o} = (1, 0, 0)$ and $\boldsymbol{q} = (-1, 0, 0)$ in R^3 and let $w = 1$. In $\mathcal{H}^{(\boldsymbol{a},b)}$, there is a chance that $\boldsymbol{o}$ and $\boldsymbol{q}$ are mapped into the same location because of the random variable b over $[0, w)$. While for our $\mathcal{H}^{(\boldsymbol{a})}$, such a probability is zero because $\boldsymbol{o} \cdot \boldsymbol{a}$ and $\boldsymbol{q} \cdot \boldsymbol{a}$ have different signs. This problem can be solved by making all the coordinates of an object/vector to be positive via adding (shifting) a constant vector with a suitable large positive number for each dimension, e.g., transforming $\boldsymbol{o}$ to (3, 2, 2) and $\boldsymbol{q}$ to (1, 2, 2). It can be easily verified that after the transformation, all the distances between each two objects are preserved.

4.1 Collision Probability under Same Hash Function

Theorem 1. *The collision probability of two objects $\boldsymbol{o}_i$ and $\boldsymbol{q}$ with distance $c_i = \|\boldsymbol{o}_i - \boldsymbol{q}\|$ under an LSH function $h^{(\boldsymbol{a})}(\boldsymbol{x}) = \lfloor\frac{\boldsymbol{a} \cdot \boldsymbol{x}}{w}\rfloor \in \mathcal{H}^{(\boldsymbol{a})}$ is $p(c_i) = p_i = \Pr[h^{(\boldsymbol{a})}(\boldsymbol{o}_i) = h^{(\boldsymbol{a})}(\boldsymbol{q})] = \int_0^w \frac{1}{c_i} f_1(\frac{t}{c_i})(1 - \frac{t}{w})dt$, where $f_1(t)$ denotes the probability density function of the absolute value of the Gaussian distribution, i.e., $f_1(t) = \sqrt{\frac{2}{\pi}} e^{\frac{-t^2}{2}}, t \geq 0$.*

Proof. Let t represent the distance of the two projected points on $\boldsymbol{a}$ from $\boldsymbol{o}_i$ and $\boldsymbol{q}$. As the random vector $\boldsymbol{a}$ whose component values are drawn from a Gaussian distribution, $|\boldsymbol{a} \cdot \boldsymbol{o}_i - \boldsymbol{a} \cdot \boldsymbol{q}|$ is distributed as $c_i|X|$ from the stable distribution theory, where X is a random variable drawn from a Gaussian distribution, i.e., $X \sim N(0, 1)$. That is, $t = |\boldsymbol{a} \cdot \boldsymbol{o}_i - \boldsymbol{a} \cdot \boldsymbol{q}| \sim c_i|X|$. When $t \in (0, w)$, the probability of falling into the same bucket is $(w - t)/w = 1 - t/w$.

Now let us consider the probability of $c_i|X| = t$. Assume $F()$ is the cumulative distribution function of $|X|$. Let $Y = c_i|X|$ be a random variable with cumulative distribution function $G(t)$. Then, $G(t) = \Pr(Y < t) = \Pr(c_i|X| < t) = \Pr(|X| < t/c_i) = F(t/c_i)$. Because $F'(t) = f_1(t)$, where $f_1(t)$ is the probability density function of $|X|$, the probability of $c_i|X| = t$ is $G'(t) = F'(t/c_i) = (1/c_i)f_1(t/c_i)$.

Therefore,

$$p_i = p(c_i) = \Pr[h^{(\boldsymbol{a})}(\boldsymbol{o}_i) = h^{(\boldsymbol{a})}(\boldsymbol{q})] = \int_0^w \frac{1}{c_i} f_1\left(\frac{t}{c_i}\right)\left(1 - \frac{t}{w}\right)dt$$

□

From Theorem 1 we can see that, for a fixed parameter w, the probability of collision decreases monotonically with c_i. We give the formal definition of our alignable LSH function family as follows:

Definition 7 (Alignable LSH family $\mathcal{H}^{(\boldsymbol{a})}$). *A function of the alignable LSH family $\mathcal{H}^{(\boldsymbol{a})}$ is defined as $h^{(\boldsymbol{a})}(\boldsymbol{q}) = \lfloor \boldsymbol{a} \cdot \boldsymbol{q}/w \rfloor$, where $\boldsymbol{q}$ is the vector representation of an object in R^d, $\boldsymbol{a}$ is a d-dimensional vector whose elements are chosen independently from a p-stable distribution; w is a user-specified constant.*

The above function is (r_1, r_2, P_1, P_2)-sensitive for $P_1 = p(1)$ and $P_2 = p(c_i)$ with $r_1 = 1$ and $r_2 = c_i$, respectively.

4.2 Collision Probability under Two Hash Functions

As Fig. 4 ② shows, two objects $\boldsymbol{o} = (o_1, o_2, \ldots o_d)$ and $\boldsymbol{q} = (q_1, q_2, \ldots q_d)$ would collide when they are mapped into the same bucket by two LSH functions with two random vectors $\boldsymbol{a}_1 = (a_{1_1}, a_{1_2}, \ldots, a_{1_d})$ and $\boldsymbol{a}_2 = (a_{2_1}, a_{2_2}, \ldots, a_{2_d})$. Therefore, the condition of collision is $\lfloor(\sum_{i=1}^d a_{1_i} o_i)/w\rfloor = \lfloor(\sum_{i=1}^d a_{2_i} q_i)/w\rfloor$. Let us consider the distribution of $t = |\sum_{i=1}^d a_{1_i} o_i - \sum_{i=1}^d a_{2_i} q_i|$. From the stable distribution theory, the distribution of $\sum_{i=1}^d a_{1_i} o_i$ and $\sum_{i=1}^d a_{2_i} q_i$ are the same as $\|\boldsymbol{o}\|X$ and $\|\boldsymbol{q}\|X$, respectively, where $X \sim N(0, 1)$.

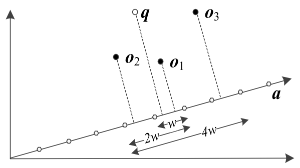


Fig. 5. Geometric illustration of LSH projecting.

Therefore, $\|o\|X \sim N(0, \|o\|^2)$, $\|q\|X \sim N(0, \|q\|^2)$, and $t = |\sum_{i=1}^{d} a_{1_i} o_i - \sum_{i=1}^{d} a_{2_i} q_i| \sim |N(0, \|o\|^2 + \|q\|^2)|$.

Suppose $f_2(t)$ is the probability density function of the absolute value of the above Gaussian distribution. When $t \in (0, w)$, the probability of falling into the same bucket is $(w-t)/w = 1 - t/w$. Therefore, we have,

Theorem 2. *The collision probability of the two objects o_i and q under two different LSH hash functions with random vectors a' and a'' from $\mathcal{H}^{(a)}$ (i.e., $h^{(x)}(q) = \lfloor x \cdot q/w \rfloor$ and $x = a'$ or a'') is $p'_i = \Pr[h^{(a')}(o_i) = h^{(a'')}(q)] = \int_0^w f_2(t)\big(1 - \frac{t}{w}\big)dt$, where $f_2(t)$ is the probability density function of the absolute value of Gaussian distribution $N(0, \|o_i\|^2 + \|q\|^2)$.*

4.3 Correctness of the Virtual Technique

Fig. 5 exhibits the geometric illustration of locality sensitive hash functions in a 2D space. Given a query object q, $\lfloor a \cdot q/w \rfloor$ maps q into one of segments/buckets /locations (with length w) along vector (line) a. In such a transformation, two close objects, e.g., q and o_1, have a high probability to be projected/mapped into the same segment. When enlarging the length of the segment, e.g., $2w$ and $4w$, o_2 and o_3 will also have a chance to be an approximate answer, respectively. That is, we can perform different closeness of AMQs by changing w. Specifically, we have the following Theorem 3.

Theorem 3. *Let p_i be the collision probability of two objects with distance c_i and parameter w, and p''_i be the collision probability of two objects with distance $2c_i$ and parameter $2w$. Then we have $p_i = p''_i$.*

Proof. From Theorem 1, the collision probability of two objects with distance c_i is $p_i = \int_0^w \frac{1}{c_i} f_1(\frac{t}{c_i})(1 - \frac{t}{w})dt$. We use a substitution $t' = t/c_i$. As $t \in (0, w)$, then $t' \in (0, w/c_i)$. Thus, $p_i = \int_0^{\frac{w}{c_i}} \frac{1}{c_i} f_1(t')(1 - \frac{t' c_i}{w})d(t' c_i) = \int_0^{\frac{w}{c_i}} f_1(t')(1 - \frac{t' c_i}{w})dt'$. As $2w/(2c_i) = w/c_i$, we have $p_i = p''_i$. □

The following Theorem 4 illustrates the correctness of merging neighbor locations.

Theorem 4. *If $x \in R$, $z \in Z^+$, then $\lfloor \frac{\lfloor x \rfloor}{z} \rfloor = \lfloor \frac{x}{z} \rfloor$.*

Proof. From $\lfloor x/z \rfloor \le x/z < \lfloor x/z \rfloor + 1$, we have $z\lfloor x/z \rfloor \le z(x/z) < z\lfloor x/z \rfloor + z$. Because $z\lfloor x/z \rfloor \in Z$, $(z\lfloor x/z \rfloor + z) \in Z$, we have $z\lfloor x/z \rfloor \le \lfloor x \rfloor < z\lfloor x/z \rfloor + z$, then $\lfloor x/z \rfloor \le \lfloor x \rfloor / z < \lfloor x/z \rfloor + 1$. Therefore, Theorem 4 holds. □

Let $x = a \cdot q/w$, $z = 2$. $\lfloor x \rfloor = \lfloor a \cdot q/w \rfloor$ is an LSH for VLBF 0, and $\lfloor \lfloor a \cdot q/w \rfloor /2 \rfloor$ is an LSH for VLBF 1 which is virtually built from the VLBF 0. From Theorem 4, we can see the VLBF 1 is the same as if a new VLBF 0 would be rebuilt with k LSHs of $\lfloor x/z \rfloor = \lfloor a \cdot q/(2w) \rfloor$ using k different a's for the coaser granularity.

5 Multi-Granularity Verification Bloom Filter

To reduce the false positive rate, we add an MVBF (see Fig. 4) to our structure to verify the concatenation of the addresses of object o_i generated from each AND-construction j, i.e., $j \diamond h_{1,j}(o_i) \diamond \ldots \diamond h_{k,j}(o_i)$. However, the BMLBF includes several VLBFs. All the concatenations of addresses from different VLBFs for object o_i need to be stored in the verification BF. As the number of verification elements (i.e., concatenations of addresses) in such a BF may be large, we have to carefully design its structure.

5.1 A Naive Method

A naive design for the verification BF is to store all the concatenations that are generated from all the different VLBFs of the BMLBF into a standard BF. Let us use $g_1(o_1)$ in Fig. 4 as an example. We store concatenations '0' $\diamond$ '1' $\diamond$ '1' $\diamond$ '5', '1' $\diamond$ '1' $\diamond$ '0' $\diamond$ '2', and '2' $\diamond$ '1' $\diamond$ '0' $\diamond$ '1' (in the binary format) into the BF. The beginning of each string, i.e., '0', '1', or '2', indicates the granularity level. The second letter of each string, i.e., '1', represents the AND-construction number (i.e., j). Thus, for each object, the above scheme stores the verification elements for all the granularities, i.e., storing $n \times L \times S$ elements, where n is the number of objects in the MLBF, L is the number of AND-constructions for one object, and S is the number of granularities supported in the MLBF. From Formula 3, we can see that the false positive rate increases with the increase of the number of set objects. One improving strategy is to only store the concatenations from the BMLBF and use them to represent the concatenations for all the granularities, which will be discussed below.

5.2 MVBF Structure

Now let us present our improved multi-granularity verification Bloom filter, as shown in Fig. 6. Different from the above naive method, we store the concatenations of addresses only for the BMLBF, i.e., $\theta = 0$, in the MVBF. However, the MVBF still supports multiple granularities. From the figure, we can see that, like the VLBFs in Fig. 4, two neighbor locations of basic verification BF, or virtual verification BF 0 (BVBF/VVBF 0) comprise a virtual location of VVBF 1, two neighbor locations of VVBF 1 comprise a virtual location of VVBF 2, and so on. With such a containment relationship between relevant locations at two consecutive granularity levels, we notice that the address of a location at the coarser level is a prefix of the two addresses of its contained locations at the next finer level. For example, in Fig. 4, the address "00" of a location in VLBF 1 is a prefix of the addresses "000" and "001" of its two contained neighbor locations in VLBF 0, while the address "0" of a location in VLBF 2 is a prefix of the addresses "00" and "01" of its two contained neighbor locations in VLBF 1. This property is utilized to construct our MVBF, as discussed below.

Let $\xi_j(o)$ be the concatenation of addresses from the jth AND-construction $g_j(o)$, i.e., $\xi_j(o) = j \diamond h_{1,j}(o) \diamond \ldots \diamond h_{k,j}(o)$. Assume that the MVBF has S granularities and adopts k' hash functions to insert $\xi_j(o)$ into the MVBF.

The insertion procedure works as follows: (1) split each binary address $h_{i,j}(o)$ $(1 \le i \le k)$ into two substrings $s_{ij}^{(1)}$

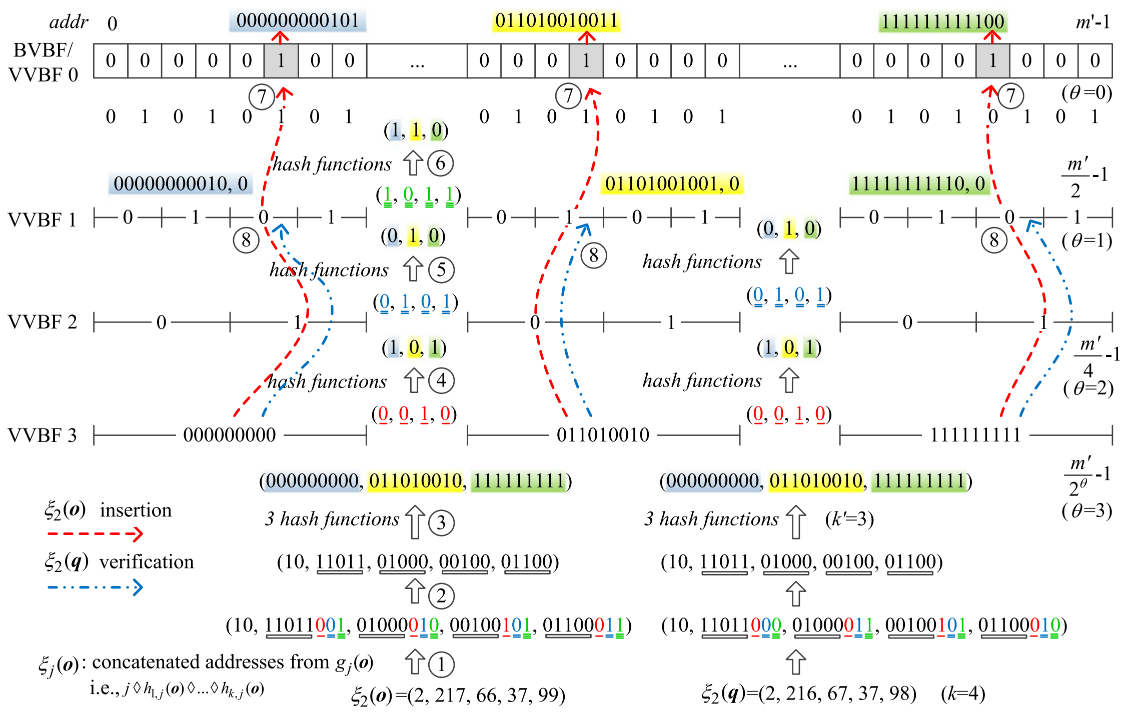


Fig. 6. The MVBF structure.

and $s_{ij}^{(2)}$ such that $h_{i,j}(\boldsymbol{o}) = s_{ij}^{(1)} \diamond s_{ij}^{(2)}$and the length of $s_{ij}^{(2)}$ is $S-1$; (2) use k' hash functions to hash the concatenation $j \diamond s_{1j}^{(1)} \diamond \ldots \diamond s_{kj}^{(1)}$ to k' addresses $A_{(S-1)1}, A_{(S-1)2}, \ldots, A_{(S-1)k'}$ for the coarsest VVBF $S-1$; (3) use a hash function to hash the concatenation $s_{1j}^{(2)}[v] \diamond \ldots \diamond s_{kj}^{(2)}[v]$ $(v = 1, \ldots, S-1$, respectively) of the next leading bit of each lower substring $s_{ij}^{(2)}$ to a k'-bit string: $B_{(S-v-1)1}B_{(S-v-1)2}\ldots B_{(S-v-1)k'}$; (4) the k' addresses for VVBF $S-v-1$ $(v = 1, \ldots, S-1)$ from the hashing are calculated as $A_{(S\text{-}v\text{-}1)\ 1} = A_{(S\text{-}v)\ 1} \diamond B_{(S\text{-}v\text{-}1)\ 1}$, $A_{(S\text{-}v\text{-}1)\ 2} = A_{(S\text{-}v)\ 2} \diamond B_{(S\text{-}v\text{-}1)\ 2}, \ldots, A_{(S\text{-}v\text{-}1)\ k'} = A_{(S\text{-}v)\ k'} \diamond B_{(S\text{-}v\text{-}1)\ k'}$; (5) for VVBF 0 (i.e., the finest granularity), set to 1 for all the locations at addresses $A_{0\ 1}, A_{0\ 2}, \ldots, A_{0\ k'}$.

Let us use the insertion of $\xi_2(\boldsymbol{o})$ in Fig. 6 as an example to illustrate the above procedure. The number of granularities in Fig. 6 is $S = 4$ (i.e., $\theta = 0, 1, 2, 3$). The MVBF in the figure uses $k' = 3$ hash functions for the element insertion. In step (1), each of the four addresses from the BMLBF for the object in $\xi_2(\boldsymbol{o})$ is split into two substrings (Fig. 6 ①). For example, the first address "11011001" is split into "11011" (higher substring) and "001" (lower substring). The length of the lower substring is 3 since $S - 1 = 3$. In step (2), three ($k' = 3$) hash functions are used to hash the concatenation of $j(= 2)$ and all the higher substrings of $h_{i,j}(\boldsymbol{o})$ into 3 addresses for VVBF 3 (Fig. 6 ②③), e.g., the first address for VVBF 3 from the hashing is "000000000". In step (3), the first bits of all the lower substrings are concatenated, and the concatenated string "0010" is hashed to a 3-bit string "101" (Fig. 6 ④) with each bit representing the suffix of one of the three addresses for VVBF 2. For example, "000000000" ◇ "1" = "0000000001" is the first address for VVBF 2, "011010010" ◇ "0" = "0110100100" is the second address for VVBF 2, etc. Similarly, the concatenation of the second bits of all the lower substrings is hashed to three suffix bits for the three addresses for VVBF 1 (Fig. 6 ⑤), and the concatenation of the third bits of all the lower substrings is hashed to three suffix bits for the three addresses for VVBF 0 (Fig. 6 ⑥). In step (4), the locations at the three addresses (e.g., "000000000101") for VVBF 0 from the hashing are set to 1 Fig. 6 ⑦).

Since the address of a coarser VVBF is a prefix of the relevant addresses of VVBF 0, we set to 1 only for the k' locations of VVBF 0 with the addresses obtained from the hashing. A location x in a coarser VVBF is considered to be set to 1 if any location in VVBF 0 whose address having the address of x as a prefix is set to 1. When verifying a query object with $\xi_j(\boldsymbol{q})$ at a granularity level θ ($\geq$1), we follow the same steps (1)-(4) in the insertion procedure to determine the k' addresses of the VVBF at granularity θ, then examine the contained locations in VVBF 0 to determine if the locations at these k' addresses at granularity level θ are all set to 1 or not. For example, to verify $\xi_2(\boldsymbol{q})$ at granularity level 1 in Fig. 6, we determine the addresses (e.g., "00000000010") for VVBF 1 (Fig. 6 ⑧) and then check the locations at the relevant addresses (e.g., "00000000010" ◇ "0" and "00000000010" ◇ "1") in VVBF 0 to see if any of them is set to 1. Since all the k' locations of VVBF 1 in Fig. 6 are considered to be 1, $\xi_2(\boldsymbol{q})$ passes the verification.

5.3 MLBF*: A Space-Efficient Variant of MLBF

The MLBF structure uses the BMLBF to store the hashing bits (from the LSH functions) of multiple AND-constructions followed by an OR-construction. When there are many objects in input set Ω, the BMLBF may produce many false positive answers. There are two reasons for this phenomenon: (1) the BMLBF/VLBFs may have too many 1's; (2) the LSH functions are based on the standard Gaussian distribution and do not scatter the 1's uniformly among all the available locations, especially when w is large. Hence, the filtering power of the BMLBF/VLBFs is very weak in such a case.

This phenomenon can also be observed in the experiments of Section 7.

Fortunately, the MLBF structure employs a verification step (via the MVBF) to reduce the false positives produced by the first step from the BMLBF. Since the MVBF uses the concatenation of the addresses from the same AND-construction as its input element, it avoids the false positives yielded by the addresses from different AND-constructions. On the other hand, the MLBF uses the LSH functions to map close objects into the same locations in the BMLBF. Such LSH functions are no longer needed for the MVBF. Hence, the MVBF can adopt hash functions following the uniform distribution to insert its elements. As a result, it can better utilize all its locations. Our experiments in Section 7 also demonstrate that the main filtering power of the MLBF comes from its verification step via the MVBF in the aforementioned scenario.

Based on the above observations, we propose a variant of the MLBF, called the MLBF*. The only difference between the MLBF* and the MLBF is the following. The MLBF* still applies the LSH functions to map an object into k addresses. However, it has no physical BMLBF to store the 1's in the locations at these addresses. Instead, the concatenation of these addresses is directly inserted into the MVBF. Therefore, the MVBF provides the sole filtering power for the MLBF*. The motivation behind such a structure is that, since the filtering power of the BMLBF is very weak in the aforementioned scenario, it is wise to remove it and use its space more effectively to enlarge the size of the MVBF.

In general, the MLBF is applicable for the cases in which the input set is not large and the interested distance values for AMQs are small, while the MLBF* is suitable for the applications demanding large input sets and big distance values for AMQs.

6 Theoretical Analysis

In this section, we analyze the properties of the proposed MLBF and MLBF* structures.

6.1 False Positive/Negative Rates of VLBF

Theorem 5 (False positive rate of VLBF). *Assume that objects $\boldsymbol{o}_i$ $(1 \leq i \leq n)$ in set Ω are represented by a BMLBF using LSH functions from $\mathcal{H}^{(\boldsymbol{a})}$ with parameters k and L. For a query object $\boldsymbol{q}$, the false positive rate for the VLBF with parameter w can be estimated as $1-(1-\alpha)^L$, where $\alpha = (1- \prod_{i=1}^{n}((1-p_i)(1-p_i')^{kL-1}))^k$, $p_i = \Pr[h^{(\boldsymbol{a})}(\boldsymbol{o}_i) = h^{(\boldsymbol{a})}(\boldsymbol{q})] = \int_0^w \frac{1}{c_i} f_1(\frac{t}{c_i})(1-\frac{t}{w})dt$, c_i is the distance between $\boldsymbol{o}_i$ and $\boldsymbol{q}$, $f_1(t)$ denotes the probability density function of the absolute value of the Gaussian distribution, i.e., $f_1(t) = \sqrt{\frac{2}{\pi}}e^{\frac{-t^2}{2}}$, $p_i' = \Pr[h^{(\boldsymbol{a}')}(\boldsymbol{o}_i) = h^{(\boldsymbol{a}'')}(\boldsymbol{q})] = \int_0^w f_2(t)(1-\frac{t}{w})dt$, and $f_2(t)$ is the probability density function of the absolute value of $N(0, \|\boldsymbol{o}_i\|^2 + \|\boldsymbol{q}\|^2)$.*

Proof. According to Theorems 1 and 2, the probability for two objects $\boldsymbol{q}$ and $\boldsymbol{o}_i$ to be mapped into one location (i.e., collision) by an identical vector $\boldsymbol{a}$ is p_i and the probability for the two objects to be collided by two different vectors $\boldsymbol{a}'$ and $\boldsymbol{a}''$ is p_i'. We first discuss the collision probability for one hashing, e.g., $h_{1,1}(\boldsymbol{q})$, with object $\boldsymbol{o}_i$. As we know, $\boldsymbol{o}_i$ has been hashed kL times, i.e., $h_{t,j}(\boldsymbol{o}_i)$ $(1 \leq t \leq k, 1 \leq j \leq L)$. In those kL functions, there is only one function, i.e., $h_{1,1}(\boldsymbol{o}_i)$, is the same as $h_{1,1}(\boldsymbol{q})$. Other $kL-1$ functions are different. Thus, the probability for $h_{1,1}(\boldsymbol{q})$ not to collide with any location mapped by $\boldsymbol{o}_i$ is $(1-p_i)(1-p_i')^{kL-1}$, and the probability for $h_{1,1}(\boldsymbol{q})$ not to collide with any object in set Ω is $\prod_{i=1}^{n}((1-p_i)(1-p_i')^{kL-1})$. As a result, for $h_{1,1}(\boldsymbol{q})$, the collision probability is $1-\prod_{i=1}^{n}((1-p_i)(1-p_i')^{kL-1})$. Because we use the AND-construction, for a $g_1(\boldsymbol{q})$, the collision probability is $\alpha = (1-\prod_{i=1}^{n}((1-p_i)(1-p_i')^{kL-1}))^k$. Finally, after the OR-construction, The false positive probability can be estimated as $1-(1-\alpha)^L$. □

Note that Theorem 5 is derived based on an assumption that all the locations to which the objects in Ω are mapped/hashed are different. In fact, there is a small probability that several objects are mapped to the same location. Therefore, the actual false positive rate may be a slightly smaller than the estimated one.

Theorem 6 (False negative rate of VLBF). *Assume that objects $\boldsymbol{o}_i$ $(1 \leq i \leq n)$ in set Ω are represented by a BMLBF using LSH functions from $\mathcal{H}^{(\boldsymbol{a})}$ with parameters k and L. For a query object $\boldsymbol{q}$ which is close to at least one of the objects in set Ω, the false negative rate for the VLBF with parameter w can be estimated as $(1-\alpha)^L$, where α is the same as that in Theorem 5.*

Proof. As the proof in Theorem 5, for a $g_j(\boldsymbol{q})$, the collision probability is α. Therefore, the false negative rate can be estimated as $(1-\alpha)^L$. □

Note that the actual false negative rate may be slightly larger than the estimated one due to the same reason mentioned after Theorem 5.

6.2 False Positive/Negative Rates of MLBF/MLBF* (Typical Case)

We first discuss the typical case, that is, the concatenations of addresses for the coarsest VLBF from objects in Ω are different. In this case, objects are scattered among the available locations, especially when the length of those addresses is long. An extended case will be analyzed in Section 6.3.

Theorem 7 (False positive rate of MVBF). *Assume that, the n' concatenations of addresses for the coarsest VLBF from objects in Ω are different. As VVBF θ has $m'' = m'/2^\theta$ locations and k' hash functions, the false positive rate of VVBF θ is $FPR = (1-(1-\frac{1}{m''})^{k'n'})^{k'} \approx (1-e^{-k'n'/m''})^{k'}$. When m' and n' are given, the optimal number of hash functions for VVBF θ is $k' = \ln 2 \times (m''/n')$. In this case the false positive rate is $FPR = (1/2)^{k'} \approx 0.6185^{m''/n'}$.*

Proof. Because the concatenations of addresses for the coarsest VLBF are different, n' elements will be uniformly distributed in the coarsest VVBF after being hashed by k' hash functions. Comparing to the standard Bloom filter, we use a special method to construct the VVBF. However, we can just regard the special construction as k' hash functions because each element is randomly mapped k' times in VVBF θ which has $m'' = m'/2^\theta$ locations. Therefore, all the theories for the standard Bloom filter still hold. Note that the optimal scenario occurs when the number of 1 bits equals to the number of 0 bits in a Bloom filter. □

Note that the MLBF/MLBF* uses AND-constructions followed by an OR-construction. However, a VVBF only verifies these AND-constructions. Theorem 8 reveals the effect of the OR-construction.

Theorem 8 (False positive rate of MLBF/MLBF*). *Assume that objects $\boldsymbol{o}_i$ ($1 \le i \le n$) in set Ω are represented by the MLBF/MLBF* using LSH functions from $\mathcal{H}^{(\boldsymbol{a})}$ with parameters k and L, VVBF θ has $m'' = m'/2^\theta$ locations and use k' hash functions. Then, the false positive rate of the MLBF can be estimated as $\alpha L(1 - e^{-k'nL/m''})^{k'}$, where α is the same as that in Theorem 5. The false positive rate of the MLBF* can be estimated as $L(1 - e^{-k'nL/m''})^{k'}$.*

Proof. From the proof of Theorem 5, we can see that, after the AND-construction, for a $g_j(\boldsymbol{q})$, the collision probability is α. From Theorem 7, the probability of the AND-construction, i.e., $g_j(\boldsymbol{q})$, to pass the verification of the MVBF is $\alpha(1 - e^{-k'nL/m''})^{k'}$. As being followed by an OR-construction, the collision probability of an object is $1 - (1 - \alpha(1 - e^{-k'nL/m''})^{k'})^L$. Because $\alpha(1 - e^{-k'nL/m''})^{k'}$ is small, after applying the Taylor series expansion, we have $1 - (1 - \alpha(1 - e^{-k'nL/m''})^{k'})^L \approx \alpha L(1 - e^{-k'nL/m''})^{k'}$. Similarly, the false positive rate of the MLBF* can be estimated as $L(1 - e^{-k'nL/m''})^{k'}$. □

Theorem 9 (False negative rate of MLBF/MLBF*). *Assume objects $\boldsymbol{o}_i$ ($1 \le i \le n$) in set Ω are represented by the MLBF using LSH functions from $\mathcal{H}^{(\boldsymbol{a})}$ with parameters k and L. If, among all the $\boldsymbol{o}_i$'s, only one object is close enough to the given query object $\boldsymbol{q}$, the false negative rate for $\boldsymbol{q}$ can be estimated as $(1 - p_i^k)^L$.*

Proof. Note that the probability of a hash collision should be no less than p_i in $h_{k,j}$ for objects $\boldsymbol{q}$ and $\boldsymbol{o}_i$. Thus, the collision probability of $\boldsymbol{q}$ and $\boldsymbol{o}_i$ under one g_j is p_i^k, and the false negative rate is $1 - p_i^k$. Then, the probability of $\boldsymbol{q}$ does not collide with $\boldsymbol{o}_i$ under any g_j can be estimated as $(1 - p_i^k)^L$. □

If there is more than one object (in set Ω) that is close to query object $\boldsymbol{q}$, we have,

Theorem 10. *Assume that, in Ω, there are n'' objects (constituting a subset Ω') close to query object $\boldsymbol{q}$. The false negative rate of the MLBF/MLBF* can be estimated as $\prod_{\boldsymbol{o}_i \in \Omega'} (1 - p_i^k)^L$.*

Proof. According to Theorem 9, Theorem 10 holds. □

Since a VVBF has false positives, the real false negative rate may be slightly smaller than the estimated one.

6.3 False Positive/Negative Rates of MLBF/MLBF* (Extended Case)

Since we use concatenations of higher bits of addresses to construct a VVBF at a coarser granularity level, it is possible that some concatenations of the higher bits are the same although the underlying full addresses are different. Although the probability for this situation to occur may be small (especially when the address length is long), it violates the assumption in Theorem 7 that such concatenations for the coarsest VVBF are different. In such a case, the accuracy of the estimation formulas derived in Section 6.2 would deteriorate. A new analysis is needed to drive better estimation formulas for the false positive and negative rates of the MLBF/MLBF* in the case of the assumption of Theorem 7 does not hold.

Let us consider the false positive rates of the VVBFs from the current granularity θ to the coarsest granularity S. We extend the notation of p_i and $\xi_j()$ as follows. Let $p_{i,2^\theta w}$ denote the collision probability of two objects $\boldsymbol{o}_i$ and $\boldsymbol{q}$ under parameter w at granularity θ, and $\xi_{j,\theta}(\boldsymbol{o})$ denote a concatenation of addresses from $g_j(\boldsymbol{o})$ for object $\boldsymbol{o}$ at granularity θ. A query object $\boldsymbol{q}$ passes the verification of VVBF θ if all the relevant k' bits are 1. For one of the k' bits, say, bit z, the probability for z to be set to 1 by object $\boldsymbol{o}_i$ in different scenarios can be estimated as follows:

- If $\xi_{j,\theta}(\boldsymbol{q}) = \xi_{j,\theta}(\boldsymbol{o}_i)$, i.e., the concatenations of addresses for $\boldsymbol{q}$ and $\boldsymbol{o}_i$ are the same at granularity θ, the probability is $p_{i,2^\theta w}^k$;
- If $\xi_{j,\theta+1}(\boldsymbol{q}) = \xi_{j,\theta+1}(\boldsymbol{o}_i)$ and $\xi_{j,\theta}(\boldsymbol{q}) \neq \xi_{j,\theta}(\boldsymbol{o}_i)$, i.e., the concatenations of addresses for $\boldsymbol{q}$ and $\boldsymbol{o}_i$ are the same at granularity $\theta + 1$, but different at granularity θ, the probability is $(p_{i,2^{\theta+1}w}^k - p_{i,2^\theta w}^k)/2^1$;
- ...
- If $\xi_{j,S-1}(\boldsymbol{q}) = \xi_{j,S-1}(\boldsymbol{o}_i)$ and $\xi_{j,S-2}(\boldsymbol{q}) \neq \xi_{j,S-2}(\boldsymbol{o}_i)$, the probability is $(p_{i,2^{S-1}w}^k - p_{i,2^{S-2}w}^k)/2^{S-\theta-1}$;
- If $\xi_{j,S-1}(\boldsymbol{q}) \neq \xi_{j,S-1}(\boldsymbol{o}_i)$, the probability is $(1 - p_{i,2^{S-1}w}^k)(1 - e^{-k'L/m''})$, where $m'' = m'/2^\theta$.

Therefore, the probability for bit z not to be set to 1 by $\boldsymbol{o}_i$ is

$$\begin{aligned}\tau_i = &\big(1 - p_{i,2^\theta w}^k\big)\big(1 - \big(p_{i,2^{\theta+1}w}^k - p_{i,2^\theta w}^k\big)/2^1\big)\\ &\big(1 - \big(p_{i,2^{\theta+2}w}^k - p_{i,2^{\theta+1}w}^k\big)/2^2\big)\ldots\big(1 - \big(p_{i,2^{S-1}w}^k - p_{i,2^{S-2}w}^k\big)/2^{S-\theta-1}\big)\\ &\big(1 - \big(1 - p_{i,2^{S-1}w}^k\big)\big(1 - e^{-k'L/m''}\big)\big).\end{aligned} \tag{5}$$

The probability for bit z not to be set by any $\boldsymbol{o}_i$ ($1 \le i \le n$) in Ω is $\prod_{i=1}^n \tau_i$. Thus, the probability for all the k' bits to be set to 1, i.e., the collision probability of an AND-construction, is $\varphi = (1 - \prod_{i=1}^n \tau_i)^{k'}$. After the OR-construction, we have,

Theorem 11. *The false positive rates of the MLBF and the MLBF* can be estimated as $1 - (1 - \alpha\varphi)^L$ and $1 - (1 - \varphi)^L$, respectively, where α is the same as that in Theorem 5, $\varphi = (1 - \prod_{i=1}^n \tau_i)^{k'}$ and τ_i is given in Formula 5.*

Theorem 11 is consistent with Theorems 7 and 9. When w is small, as $h_{k,j}(\boldsymbol{o}) = \lfloor \boldsymbol{a} \cdot \boldsymbol{o}/w \rfloor$, the concatenation of addresses from objects in Ω are different to each other with a high probability. On the other hand, if the distance between $\boldsymbol{o}_{i_1}$ and $\boldsymbol{o}_{i_2}$ ($1 \le i_1, i_2 \le n$) is large, the concatenations are also different. In those cases, as $p_{i,2^\theta w}^k$, $p_{i,2^{\theta+1}w}^k$, $\cdots$, are almost zero,τ_i is degenerated as follows:

$$\tau_i = \big(1 - \big(1 - p_{i,2^{S-1}w}^k\big)\big(1 - e^{-k'L/m''}\big)\big) = e^{-k'L/m''}.$$

Hence, $\varphi = (1 - \prod_{i=1}^{n} \tau_i)^{k'} = (1 - e^{-k'nL/m''})^{k'} = (1 - e^{-k'n'/m''})^{k'}$, which is consistent with Theorem 7.

Theorem 12. *The false negative rate of the MLBF/MLBF* can be estimated as* $(1-\varphi)^L$, *where* ϕ *is the same as that in Theorem 11.*

Proof. As explained in Theorem 11, for an AND-construction, the collision probability is φ. Therefore, the false negative rate can be estimated as $(1-\varphi)^L$. □

6.4 Selection of Parameters

Definition 3 uses a distance r to define an AMQ. However, for the MLBF/MLBF*, we use parameter w to filter AMQs. Thus, we need to use the given user requirements, i.e., r, $FNR < \delta_n$, and $FPR < \delta_p$, to determine the parameters w, k, L, k', and m' of the MLBF/MLBF*. Let us discuss the selection of the parameters for the BMLBF ($\theta = 0$) as an example.

First, let us consider the requirement of $FNR < \delta_n$. For a given set Ω, we assume that there is only one object that is close to the query object $\boldsymbol{q}$ and has a distance $= r$. If the distance is smaller or there are more than one close object in Ω, from Theorems 9 and 10, we can see that the FNR will decrease and still satisfy for the requirement of δ_n. From Theorem 9, we have

$$\left(1-P_1^k\right)^L \le \delta_n, \text{i.e., } L \ge \frac{\ln \delta_n}{\ln\left(1-P_1^k\right)}. \tag{6}$$

Second, let us consider the requirement of $FPR < \delta_p$. A false positive occurs in two scenarios: (1) The distance of two objects is larger than r_2 (refer to Definition 4), while they are regarded as close objects due to some AND-construction g_j. We denote this probability as δ_p'. (2) The query object passes the membership test of the BMLBF and also passes the verification of the MVBF although it is not close to any object in Ω. We denote this probability as δ_p''. Obviously, $\delta_p' + \delta_p'' = \delta_p$.

Thus, for scenario 1, we have

$$1-\left(1-P_2^k\right)^{nL} \le \delta_p', \text{i.e., } L \le \frac{\ln\left(1-\delta_p\right)}{n\ln\left(1-P_2^k\right)}. \tag{7}$$

Here parameter P_2 corresponds to the distance r_2. If the distance is larger, δ_p' will decrease and still satisfy the requirement.

From Formulas 6 and 7, we have

$$\frac{\ln \delta_n}{\ln\left(1-P_1^k\right)} \le \frac{\ln\left(1-\delta_p'\right)}{n\ln\left(1-P_2^k\right)}. \tag{8}$$

As r_1, r_2, and P_1 are defined, we can use numerical methods to determine w and P_2 according to Theorem 1. For example, suppose $r_1 = (20 \times 0.1^2)^{0.5}$, $P_1 = 0.80$, we can get $w = 1.78$. We then use w, $r_2 = 10r_1$, and Theorem 1 to calculate $P_2 = 0.16$.

More specifically, suppose $\delta_n = 0.1$, $\delta_p' = 0.09$, $n = 100$. From Formula 8, we can get $k \ge 5$. As a large k will cause much computation cost, we choose $k = 5$. Then from Formula 6, we have $L \ge 5.8$; and from Formula 7, we have $L \le 9.0$. As a large L also causes much computation cost, we choose $L = 6$.

If we choose appropriate parameters for scenario 2, i.e., k' and m', we can control the false positive rate δ_p'' of the MVBF to be sufficiently small. According to Theorem 7, the false positive rate of VVBF θ is $FPR \approx (1 - e^{-k'n'/m''})^{k'}$, where $m'' = m'/2^\theta$. Hence, the probability for scenario 2 is

$$\left(1-P_2^k\right)^{n'}\left(1-e^{-k'n'/m''}\right)^{k'} \le \delta_p''.$$

For different granularities, we can choose appropriate m' and k' to represent $n' = nL$ concatenations of addresses with a low false positive rate. For example, suppose $\delta_p'' = 0.001$, $n = 100$, $L = 6$, $k = 5$, we can get appropriate parameters to satisfy the requirement of δ_p'', e.g., $m' = 65{,}536$, $\theta = 0, 1, 2$ and $k' = 4$.

The above method is suitable for the case in which only a few objects (in Ω) that are close to query object $\boldsymbol{q}$. Sometimes, there may be many close objects in Ω. In that case, the above method may not be accurate because the FNR will decrease by many close objects in Ω. A sample method can be used in such a case. Specifically, the relevant parameters are determined by minimizing the FPR and the FNR for some sample query objects.

7 Experimental Results

We conducted extensive experiments to evaluate the accuracy of the MLBF and MLBF* structures using both synthetic and real data. All experimental results were measured from a large number of repeated executions ($\geq 10^4$) to achieve accurate mathematical expectations. The synthetic data were randomly generated with 20 dimensions that follow the uniform distribution over [1, 1000]. In the theoretical estimation, we applied the Monte Carlo method and Simpson's rule to calculate the integrals in the estimation models.

7.1 Collision Probability of LSH Function

To verify Theorems 1 and 2, we randomly generate synthetic object pairs ($\boldsymbol{o}_1$, $\boldsymbol{o}_2$) and ($\boldsymbol{o}_3$, $\boldsymbol{o}_4$). The values of each dimension in $\boldsymbol{o}_1$, $\boldsymbol{o}_2$, and $\boldsymbol{o}_3$ are uniformly distributed over [1, 1000], and the difference in each dimension between $\boldsymbol{o}_3$ and $\boldsymbol{o}_4$ is 0.1. Hence, the distance between $\boldsymbol{o}_1$ and $\boldsymbol{o}_2$ is expected to be larger than the distance between $\boldsymbol{o}_3$ and $\boldsymbol{o}_4$. In this way, we can test both scenarios in which the query object is close to or not close to the data object. We project ($\boldsymbol{o}_1$, $\boldsymbol{o}_2$) and ($\boldsymbol{o}_3$, $\boldsymbol{o}_4$) into one vector (i.e., $\boldsymbol{a}_1$ for one LSH function) and two vectors (i.e., $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ for two LSH functions) to investigate the relevant collision probabilities. In experiments, we use 10^7 randomly generated Gaussian-distribution-based vectors $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ for LSH functions, e.g., $\lfloor \boldsymbol{a}_1 \cdot \boldsymbol{o}_1 / w \rfloor$, and then count the number of collisions. The results are summarized in Tables 2 and 3, which demonstrate that Theorems 1 and 2 are quite accurate.

7.2 False Positive Rates of MVBF

We also compared the false positive rates of the MVBF and the standard BF to verify Theorem 7. Both the MVBF and the BF in the experiments had 65,536 locations. The BF

TABLE 2
Collision Probabilities of Hashing ($\boldsymbol{o}_1$, $\boldsymbol{o}_2$)

	$p\prime$ ($\boldsymbol{a}_1$ and $\boldsymbol{a}_2$)		p ($\boldsymbol{a}_1$)	
w	theoretical	experimental	theoretical	experimental
1.00	0.0001133	0.0001137	0.0002086	0.0002104
2.00	0.0002267	0.0002258	0.0004168	0.0004224
3.00	0.0003398	0.0003471	0.0006261	0.0006167
4.00	0.0004534	0.0004657	0.0008336	0.0008481
5.00	0.0005667	0.0005569	0.0010431	0.0010281
6.00	0.0006801	0.0006679	0.0012507	0.0012378
7.00	0.0007932	0.0007950	0.0014604	0.0014666
8.00	0.0009071	0.0008962	0.0016706	0.0016722
9.00	0.0010198	0.0010370	0.0018772	0.0018856

used the naive method presented in Section 5.1 to store (verification) elements. The elements stored in the MVBF were $\xi_j(\boldsymbol{o}) = j \diamond h_{1,j}(\boldsymbol{o}) \diamond \cdots \diamond h_{k,j}(\boldsymbol{o})$, where $k = 5$, $h_{k,j}(\boldsymbol{o})$ here is a random integer in [0, 2×10^5) to represent an address in the BMLBF, j is a random integer in [0, 5]. The tested MVBF had five granularities, i.e., $\theta = 0$, 1, 2, 3, and 4. Therefore, if the MVBF stored n' elements, the BF stored $5n'$ elements. We used Theorem 7 to estimate the false positive rates for all different granularities of the MVBF. To test the false positive rates, we used 5×10^5 additional elements, i.e., $\xi_j(\boldsymbol{o})$, as testing objects. We also used Formula 3 to estimate the false positive rates of the BF.

Fig. 7 shows the comparisons of the false positive rates of the VVBF and the BF for different parameters. The optimal number k'' of hash functions for the BF was calculated by Formula 4. Although different VVBFs have the same number n' of elements and use the same number k' of hash functions, they have different numbers of locations. Therefore, an optimal parameter k' is just for a specific VVBF. k' in Figs. 7a, 7c, 7d was optimized for VVBF 3, and k' in Fig. 7b was optimized for VVBF 4. In fact, which VVBF for optimization can be determined based on application requirements. For example, we can choose the VVBF which is used most frequently in the application for optimization. Because the BF stored much more elements than the MVBF, i.e., $5n'$ vs n', the false positive rates of the VVBFs 0-2 in Figs. 7a, 7c, 7d (i.e., $\theta = 0, 1, 2$) are much smaller than those of the BF. However, when the granularity becomes coarser, the number of locations

TABLE 3
Collision Probabilities of Hashing ($\boldsymbol{o}_3$, $\boldsymbol{o}_4$)

	$p\prime$ ($\boldsymbol{a}_1$ and $\boldsymbol{a}_2$)		p ($\boldsymbol{a}_1$)	
w	theoretical	experimental	theoretical	experimental
0.50	0.0000571	0.0000555	0.4047870	0.4048746
1.00	0.0001143	0.0001204	0.6471178	0.6473068
1.50	0.0001714	0.0001749	0.7621785	0.7620237
2.00	0.0002286	0.0002268	0.8215880	0.8216179
2.50	0.0002859	0.0002771	0.8572701	0.8573130
3.00	0.0003428	0.0003398	0.8810584	0.8811977
4.00	0.0004572	0.0004623	0.9107938	0.9108332
5.00	0.0005712	0.0005679	0.9286350	0.9286575
6.00	0.0006856	0.0006948	0.9405292	0.9406059
7.00	0.0008000	0.0007967	0.9490250	0.9490090
8.00	0.0009146	0.0009276	0.9553969	0.9554052
9.00	0.0010288	0.0010417	0.9603528	0.9604059

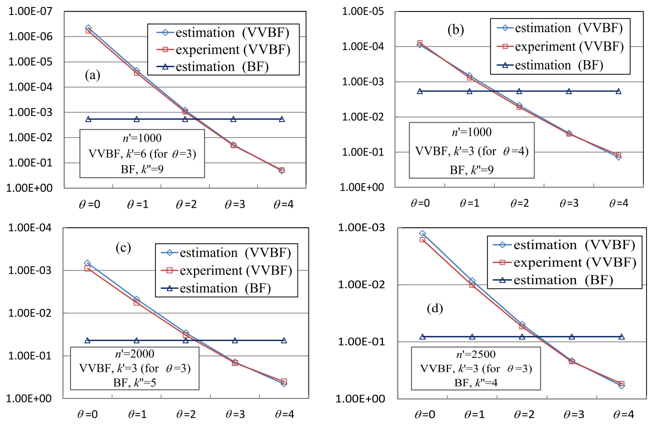


Fig. 7. Comparisons of false positive rates.

decreases and the false positive rates increases. The accuracy of the estimation given by Theorem 7 is also verified because the results from experiments and theoretical estimation match with each other well.

7.3 False Positive/Negative Rates without MVBF

To validate Theorems 5 and 6, we stored 500, 300, and 100 synthetic objects as three data sets Ω into the MLBF, respectively. To compare the false positive rates from experiments and theoretical estimation, we used additional 10^5 test objects which were not close to any of the objects in set Ω. In the experiments, we counted the numbers of false positives. Fig. 8 shows the results. As we know, every VLBF has a specific parameter w. The w in Fig. 8 is the parameter for BMLBF (i.e., VLBF 0). From the figure, we can see that the curve from theoretical estimation matches well with the curve from experiments for the same parameters. Note that, the more objects exist in the BMLBF, the more collisions will occur, which results in higher false positive rates. Furthermore, the theoretical estimation is based on an assumption that all hashed locations (i.e., bits) which store objects in set Ω are different. However, there may exist some objects hashed into identical locations with a small probability in reality. Therefore, the more objects exist in the VLBFs, the more hashing collisions will occur, which results in an overestimate than the observed result from experiments.

To compare the false negative rates from experiments and theoretical estimation, we randomly generated objects close to one of the objects in the BMLBF as test objects. The difference in each dimension of two close objects was set to 0.05. Fig. 9 shows the results. From the figure, we can see that the two curves from experiments and theoretical

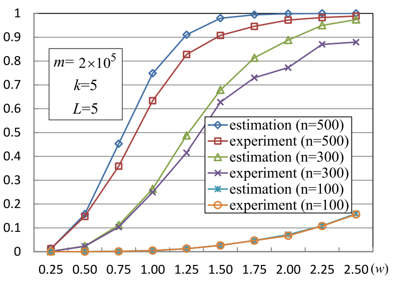


Fig. 8. FPRs without MVBF.

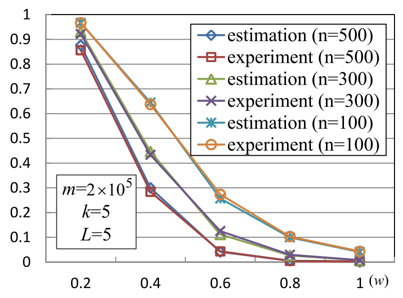


Fig. 9. FNRs without MVBF.

estimation for the same parameters match very well. In addition, the more objects exist in the VLBF, the more hashing collisions will occur, which results in lower false negative rates. Thus the false negative rates decrease with the increase of the number n of elements in set Ω.

From Figs. 8 and 9, we can see that the false positive rates increase with the increase of w, while the false negative rates decrease with the increase of w. When w is small, e.g., $w = 0.2$, although the false positive rates are low, the false negative rates may be high. When w is large, e.g., $w = 1$, the opposite results are observed. As a result, a VLBF without the MVBF verification cannot handle approximate membership queries well.

7.4 False Positive/Negative Rates of MLBF

To compare the false positive rates from experiments and theoretical estimation for granularities 0, 1, and 2, respectively, 500 synthetic objects were stored in the BMLBF, and additional 10^5 synthetic objects which are not close to any of the objects in the BMLBF were used as test objects. For each granularity, we counted the number of false positives. Comparing to Fig. 8, Fig. 10 shows that the MLBF technique can control the false positive rates to a low level for different granularities. Note that, in Fig. 10, the results for granularity level 0 ($\theta = 0$) are magnified 100 times and the results for granularity level 1 ($\theta = 1$) are magnified 10 times. It can be seen that, as the accuracy is highly correlated with the number of locations in a VVBF, namely, the finer the granularity, the higher the accuracy. The two curves from theoretical estimation and experiments for the same granularity show the similar pattern. In fact, the estimates for $\theta = 0$ are more accurate than those for $\theta = 1$ or 2.

To compare the false negative rates from experiments and theoretical estimation, we randomly generated query objects close to one of the objects in the MLBF as test objects. In other words, in each execution, there exists only one object in set Ω being close to the query object. The difference in each dimension of two objects was set to 0.05. Fig. 11 shows the results for different granularities $\theta = 0$, 1, and 2. From the figure, we can see that the two curves from theoretical estimation and experiments for the same parameters match very well. It can also be seen that, as the accuracy is highly correlated with the number of locations in a VVBF, the finer the granularity, the higher the accuracy. For the same granularity, if w is small, the false negative rates are high because of the LSH functions. Let us calculate the collision probability of two close objects with a distance $(20 \times 0.05^2)^{0.5}$. When $w = 0.25$ and $\theta = 0$, the collision probability for one LSH function is 0.4 from Theorem 1. After the AND-and-OR constructions, the collision probability is $1 - (1 - 0.4^5)^5 = 0.05$. In other words, the false negative rate is about 0.95, which is consistent with the experimental results.

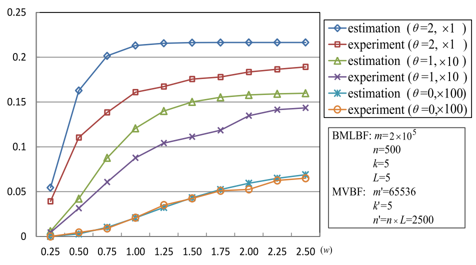


Fig. 10. FPRs of MLBF.

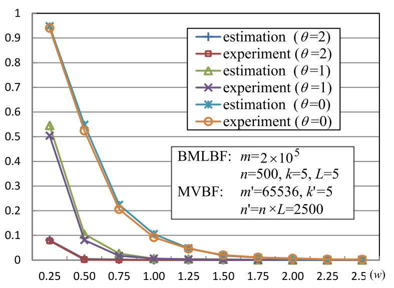


Fig. 11. FNRs of MLBF (one close object).

Fig. 12 shows the comparisons of the false negative rates for the scenarios with two objects in set Ω being close to the query object. As discussed in Theorem 10, the false negative rates of the MLBF do not change much with the number n of objects in the set. However, the false negative rates can decrease with the increase of the close objects in set Ω. Thus, the more close objects are in the set, the lower false negative rate is obtained.

From Figs. 10–12 we can see that, for different w's, the MLBF structure can control the false positive and false negative rates to a low level for different granularities, which implies that the structure can handle approximate membership queries with a high accuracy. For example, when $n = 500$ and $w = 2.5$, the false positive rate for granularity level 0 is 6.6×10^{-4}, while the false negative rate is 2.0×10^{-3}, both are very small.

7.5 False Positive/Negative Rates of MLBF*

We compared false positive rates between estimations by Theorem 11 and experimental results with $S = 4$, $\theta = 3, 2, 1$, and 0, respectively. Like the experiments in Fig. 9, we stored 500 synthetical objects in the MLBF/MLBF* (i.e., in Ω), and

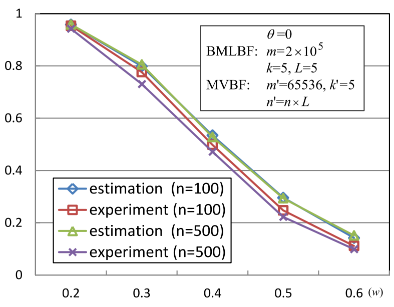


Fig. 12. FNRs of MLBF (two close objects).

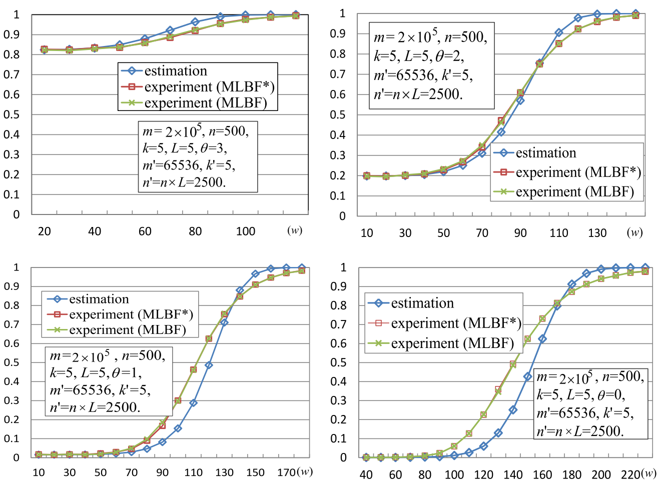


Fig. 13. FPRs for different granularities.

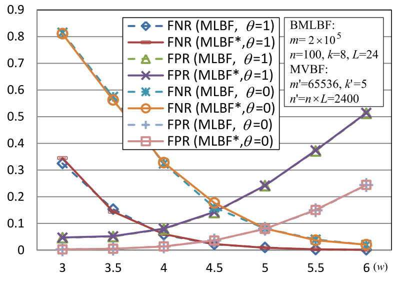


Fig. 14. FPRs/FNRs with real data.

used additional 10^6 synthetical objects which are not close to any of the objects (in Ω) as test objects. The difference is that, in this series of experiments, we set w much larger so that some objects (in Ω) are projected into identical locations of the relevant VVBF. From Fig. 13, we can see that the results of experiments of the MLBF and the MLBF* are almost same. This scenario can also be observed from Fig. 8: with the increase of w, the filtering ability of a VLBF is rather weak. Three curved lines in Fig. 13, one for estimation and the other two for experimental results, at the same granularity are close.

When w is small in Fig. 13, the least FPRs for granularities 3, 2, 1, and 0 can be estimated by Theorem 7, respectively, as Theorem 11 being consistent with Theorem 7. For example, if $\theta = 3$, FPR for one AND-construction is $(1 - e^{-k'n'/m''})^{k'} = (1 - e^{-5\times2500/(65536/8)})^5 = 0.293$. Thus, after OR-construction, the FPR is $1 - (1 - 0.293)^5 = 0.823$. Similarly, if $\theta = 2$, the FPR is 0.197; if $\theta = 1$, the FPR is 0.015; if $\theta = 0$, the FPR is 7.90×10^{-4}. These results testify the accuracy of Theorems 7 and 11.

We also used the real-world handwriting digit Letter Recognition Data Set[1] to evaluate the accuracy of the MLBF and MLBF* structures. The data set has 5,620 unique objects. Each objects has 64 features to represent one of the ten handwriting digit letter (i.e., '0', '1', ..., '9'). Each feature is an integer in the range of 0..16. We added 1 to each feature due to the positive dimension requirement of $\mathcal{H}^{(\boldsymbol{a})}$. We divided the set of objects for letter '0' into two groups: one as the input data set Ω, and the other as test data to show the false negative rates. We used the set of objects for letter '1' as test data to show the false positive rates. Fig. 14 shows the experimental results. From the figure, we can see that, with the increase of w, the false positive rates increase, and the false negative rates decrease. When $\theta = 0$, as the MVBF can obtain the best accuracy, the MLBF and MLBF* structures have the lowest false positive rates. Each pair of lines, one for MLBF and the other for MLBF*, at the same granularity are almost overlapping. For a specific application, we can choose an appropriate value of w according to sample data. For example, if $\theta = 0$, we can choose $w = 5$ with the false positive and negative rates are both close to 0.08. If $\theta = 1$, we can choose $w = 4$ with the false positive and negative rates are both close to 0.07.

1. http://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Hand written+Digits

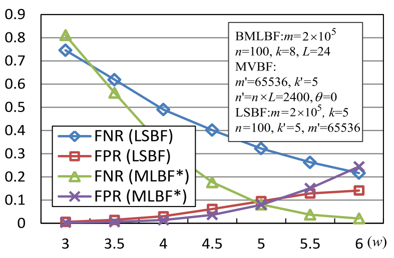


Fig. 15. FPR/FNR comparison.

7.6 MLBF* versus LSBF

We used the above real-world handwriting digit Letter data to compare performance between the MLBF* and the LSBF. The focused issues are the FPR/FNR, space cost and computation cost. The results are shown in Fig. 15. As the MLBF* adopts an OR-constructions to reduce the FNR, Fig. 15 shows that a more accurate result can be achieved from it. However, the accuracy is not achieved for free. LSBF only spends 0.116 ms to filter an object, while MLBF* costs 0.351 ms for the OR-constructions' computation. Fig. 16 shows the space cost comparison. Because the MLBF* only uses the MVBF with a virtual BMLBF, even for one granularity, the MLBF* is space-effective. With the increase of the number of different granularities, the space cost of the LSBF increases, while that of the MLBF* remains constant. From the comparison, we can see that, with extra computation cost, the MLBF* is more accurate and more space-effective than the LSBF.

8 Related Work

The Bloom filter was proposed by B. H. Bloom in 1970 [4], which has been playing an important role in numerous applications since then. A number of variants of the

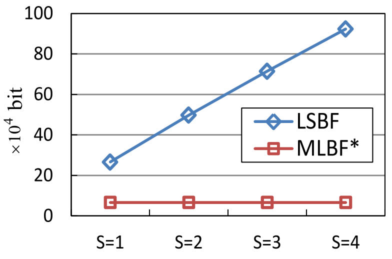


Fig. 16. Space cost comparison.

standard Bloom filter have been proposed. The hierarchical Bloom filter (HBF) [9] was designed for sub-string matching. To handle element deletions, the counting Bloom filter (CBF) [10] extends the 1-bit locations to 4-bit counters to avoid counter overflows. To process multi-attributes objects, Xiao and Hua [11] proposed auxiliary structures to capture the inherent dependency among the attributes of an object. To delete one attribute value according to another attribute value for a set of data objects with two correlated attributes, Qian et al. [12] proposed the improved associative deletion Bloom filter (IABF) to realize the associative deletion operation. A good survey of bloom filters can be found in [7], [13].

Similarity indices are desirable for building content-based search systems for multi-dimensional data such as audios, images, and sensor data. Indyk and Motwani [3] first proposed the well-known LSH technique in 1998. Nowadays, the LSH technique and its variants are the state-of-the-art indexing techniques for the approximate similarity search in a multi-dimensional space. Gionis et al. [14] proposed a method to process an in-memory data set in the Hamming space. Datar et al. [5] proposed the LSH functions based on the p-stable distribution. Motwani et al. [15] discussed the lower bounds for LSH functions. Andoni and Indyk [16] presented an algorithm which can almost achieve the lower bounds. The multi-probe LSH presented in [8] systematically probes multiple buckets that are likely to contain query results, instead of probing only the bucket that contains the query object, to improve both space and time efficiencies. The Collision Counting LSH (C2LSH) reported in [17] uses a base of several single LSH functions to construct dynamic composite hash functions with a pre-specified collision threshold. The BayesLSH from [18] is able to quickly prune away a large majority of the false positive candidate pairs, leading to significant speedups over the baseline approaches. To reduce the probability of false conflicts, Quislant et al. [19] proposed a novel design for sharing nearby locations that exploits the spatial locality in transactional memories. Tao et al. [20] improved the LSH by proposing an access method called the LSB-tree to enable fast, accurate, high-dimensional NN search in relational databases. Hua et al. [21] proposed SmartStore to exploit metadata semantics of files to judiciously aggregate correlated files into semantic-aware groups by using information retrieval tools.

Research on using a Bloom filter and LSH functions to process approximate membership queries has not been fully explored. The DSBF [2] can provide speed and space improvements for network and database applications, avoiding full nearest-neighbor queries or costly comparison operations against the entire set. The LSBF in [1] extends the standard Bloom filter by replacing hash functions with LSH functions to provide the AMQ processing service. The LSBF also uses a bit vector which verifies multiple attributes belonging to one object to reduce false positives. However, the key parameter, i.e., the closeness, in both techniques has to be defined in advance before the query processing. Our MLBF/MLBF* technique can process AMQs under multiple distance granularities with low false positive and negative rates. To our knowledge, no similar work has been reported in the literature.

9 Conclusion

Aiming for processing approximate membership queries for broad applications, we propose a novel filter structure MLBF, and its variant MLBF*, which can process AMQs under multiple distance granularities. The new structures use a modified version of a classical LSH function family to achieve an alignable property that is required for supporting multiple granularities in a BF. We show that the new LSH functions are also locality-sensitive for multi-dimensional objects with positive coordinates. The number of false negatives of the MLBF/MLBF* structure is reduced by using multiple AND-constructions followed by an OR-construction. Detailed theoretical analyses for the false positive/negative rates are given. Experiments on synthetic and real data sets show that the theoretical estimates for false rates are quite accurate, and the new techniques can handle AMQs with low false positive/negative rates for multiple distance granularities.

Further studies will be conducted in the future. For example, in the MLBF/MLBF* structure, the current distance parameter is restricted to a special form, i.e., $2^{\theta}w$. If a given distance parameter x is not of this form, we can set the distance parameter to $2^{\theta}w$ for the structure such that $x \in (2^{\theta-1}w, 2^{\theta}w)$. In this conservative way, we can still remove some useless AMQs with regards to distance parameter x ($< 2^{\theta}w$) in the query result set. However, more survived AMQs need to be further examined via the conventional query execution, comparing to a method that uses exact x. Extending the MLBF/MLBF* structure to allow an arbitrary distance granularity is an interesting open research issue. Furthermore, a trade-off among accuracy, computing overhead, and space cost will be further studied.

Acknowledgments

This work was supported in part by China NSF Grant No. 61472194, Zhejiang NSF Grant No. LY13F020040, Ningbo NSF Grants No. 2014A610023, No. 2013A610063 as well as programs sponsored by K.C.Wong Magna Fund in Ningbo University. They wish to thank the anonymous reviewers for their valuable time and suggestions to improve the paper.

References

[1] Y. Hua, B. Xiao, B. Veeravalli, and D. Feng, "Locality-sensitive Bloom filter for approximate membership query," *IEEE Trans. Comput.*, vol. 61, no. 6, pp. 817–830, Jun. 2012.
[2] A. Kirsch and M. Mitzenmacher, "Distance-sensitive Bloom filters," in *Proc. 8th Workshop Algorithm Eng. Exper.*, 2006, pp. 41–50.
[3] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *Proc. 30th Annu. ACM Symp. Theory Comput.*, 1998, pp. 604–613.
[4] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, 1970.
[5] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proc. 20th Annu. Symp. Comput. Geometry*, 2004, pp. 253–262.
[6] A. Rajaraman and J. D. Ullman, *Mining of Massive Databases*. New York, NY, USA: Cambridge Univ. Press, 2011.
[7] A. Broder and M. Mitzenmacher, "Network applications of Bloom filters: A survey," *Internet Math.*, vol. 1, no. 4, pp. 485–509, 2004.
[8] Q. Lv, W. Josephson, and Z. Wang, "Multi-probe LSH: Efficient indexing for high-dimensional similarity search," in *Proc. 33rd Int. Conf. Very Large Data Bases*, 2007, pp. 950–961.

[9] K. Shanmugasundaram, H. Bronnimann, and N. Memon, "Payload attribution via hierarchical Bloom filters," in *Proc. 11th ACM Conf. Comput. Commun. Security*, 2004, pp. 31–41.

[10] L. Fan, et al., "Summary cache: A scalable wide-area web cache sharing protocol," *IEEE/ACM Trans. Netw.*, vol. 8, no. 3, pp. 281–293, Jun. 2000.

[11] B. Xiao and Y. Hua, "Using parallel Bloom filters for multiattribute representation on network services," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 1, pp. 20–32, Jan. 2009.

[12] J. Qian, Q. Zhu, and Y. Wang, "Bloom filter based associative deletion," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 8, pp. 1986–1998, Aug. 2014.

[13] S. Tarkoma, C. E. Rothenberg, and E. Lagerspetz, "Theory and practice of Bloom filters for distributed systems," *IEEE Commun. Surveys Tutorials*, vol. 14, no. 1, pp. 131–155, Apr. 2011.

[14] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *Proc. 25th Int. Conf. Very Large Data Bases*, 1999, pp. 518–529.

[15] R. Motwani, A. Naor, and R. Panigrahy, "Lower bounds on locality sensitive hashing," *SIAM J. Discr. Math.*, vol. 21, pp. 930–935, 2005.

[16] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," in *Proc. 47th Annu. IEEE Symp. Found. Comput. Sci.*, 2006, pp. 459–468.

[17] J. Gan, J. Feng, Q. Fang, and W. Ng, "Locality-sensitive hashing scheme based on dynamic collision counting," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2012, pp. 541–552.

[18] V. Satuluri and S. Parthasarathy, "Bayesian locality sensitive hashing for fast similarity search," *Proc. VLDB Endowment*, vol. 5, pp. 430–441, 2012.

[19] R. Quislant, E. Gutierrez, and O. Plata, "LS-Sig: Locality-sensitive signatures for transactional memory," *IEEE Trans. Comput.*, vol. 62, no. 2, pp. 322–335, Feb. 2013.

[20] Y. Tao, K. Yi, C. Sheng, and P. Kalnis, "Efficient and accurate nearest neighbor and closest pair search in high dimensional space," *ACM Trans. Data. Syst.*, vol. 35, no. 3, pp. 1–46, 2010.

[21] Y. Hua, H. Jiang, Y. Zhu, D. Feng, and L. Tian, "Semantic-aware metadata organization paradigm in next-generation file systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 2, pp. 337–344, Feb. 2012.

Jiangbo Qian received the PhD degree in computer science from Southeast University, China, in 2006. He is currently a professor at the College of Information Science and Engineering, Ningbo University, China. He was a visiting scholar at the Department of Computer and Information Science, University of Michigan-Dearborn. His research interests include database management, streaming data processing, multidimensional indexing, query optimization, and hardware/software co-design.

Qiang Zhu received the PhD degree in computer science from the University of Waterloo, Canada, in 1995. He is currently a professor at the Department of Computer and Information Science, University of Michigan-Dearborn. He is also an ACM distinguished scientist, an IBM CAS faculty fellow, and a senior member of the IEEE. His current research interests include query optimization, streaming data processing, multidimensional indexing, self-managing databases, and web information systems.

Huahui Chen received the PhD degree in computer science from Fudan University, China, in 2009. He is currently a professor at the College of Information Science and Engineering, Ningbo University, China. His research interests include database management, streaming data processing, and query optimization.