2012 IEEE 14th International Conference on High Performance Computing and Communications

Data Independent Method of Constructing Distributed LSH for Large-Scale Dynamic High-Dimensional Indexing

Xiaoguang Gu[1,2,3], Lei Zhang[1,2,3], Dongming Zhang[1,3], Yongdong Zhang[1,3],Jintao Li[1,3], Ning Bao[4]
[1]*Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China*
[2]*Graduate University of Chinese Academy of Sciences, Beijing, China*
[3]*Beijing Key Laboratory of Mobile Computing and Pervasive Device*
(Institute of Computing Technology, Chinese Academy of Sciences)
[4]*School of Technical Physics, Xidian University,Xi'An, China*
xggu@ict.ac.cn

***Abstract*—Constructing effective and efficient indexes for explosive growing multimedia data is a very challenging problem. To solve the problem, Haghani et al. provide a distributed similarity search method in high dimensions using Locality Sensitive Hashing. However, their method needs to estimate a global parameter on the whole dataset beforehand. It is impractical for a large-scale dynamical dataset. This paper proposes a novel constructing method of distributed LSH which does not need any priori knowledge about the dataset. Through generating the hash function with consistent output distribution, we get a data independent predicting model in theory which can guarantee a well load balance even if the dataset dynamically changes. Furthermore, we modify the query algorithm of the basic LSH to make the proposed model more practical. The experimental results on two open large-scale high-dimensional datasets show that the proposed method is more robust, scalable and practical than state-of-the-art.**

***Keywords*-Distributed Similarity Search; Locality Sensitive Hashing; Peer-to-Peer; Data Independent;**

I. Introduction

Performing effective and efficient similarity search in large-scale high-dimensional feature sets is a common problem in many practical applications such as multimedia retrieval and multimedia database. Developing indexes is an effective solution. Locality Sensitive Hashing(LSH)[1][2] is the most successful and popular one. LSH has a truly sublinear dependence on the data size even for high-dimensional data. Euclidean LSH [3][4]is the most successful varition of the basic LSH because it uses Eucliden distance as the similarity metric.

A significant drawback of LSH is the requirement for a large number of hash tables in order to achieve a good search quality. In [1] and [2], a large number of hash tables are used. When the scale of the dataset is large, if the number of hash tables is also large, the index structure can not be loaded in the main memory to get the best performance. To reduce the memory overhead of LSH, some variations [5][6][7] based on multi-probe strategy are brought forward. Nevertheless, when using centralized indexing framework, the size of the index is limited by the physical resources of the center node.

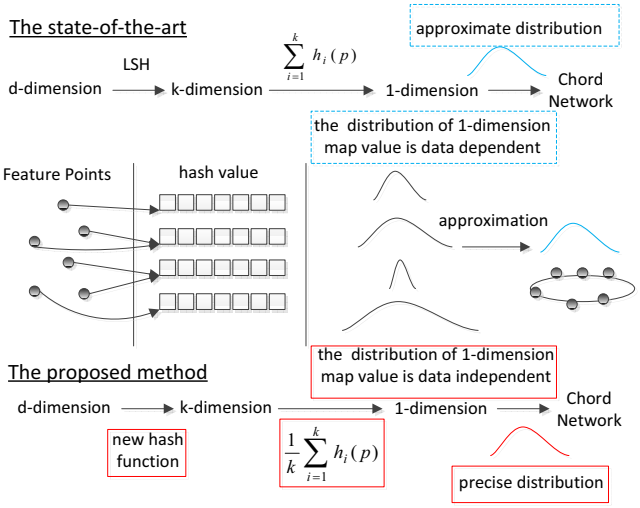


Figure 1. The principle of Distributed LSH and our contribution.

For the ever larger feature sets, constructing indexes in a distributed style is an effective way. In [8], an efficient, scalable and distributed index based on LSH is proposed. The key idea is to map multi-dimensional LSH buckets to the linearly ordered set of peers. Such mapping is locality preserving so that the buckets likely to hold the similar data are stored on the same or the neighbor peers and has a predictable output distribution to ensure a fair load. Although the framework of [8] is very successful, there are several fatal weaknesses when it is used in practical applications. To predict the output distribution, a global parameter of the dataset must be estimated in advance. However, it is difficult to determine an appropriate global parameter for a dynamic dataset beforehand in some actual applications. In this paper, we propose a solution to overcome these drawbacks. As shown in **Fig.1**, our main contributions include:

1.Propose a new hash function with a consistent output distribution;

2.Deduce a data independent predicting model which does

978-0-7695-4749-7/12 $26.00 © 2012 IEEE
DOI 10.1109/HPCC.2012.82

not need any priori knowledge about the dataset;

3.Provide an improved query algorithm to make the proposed model more practical.

II. Background

A. The framework of the distributed index based on LSH and Chord

The principle of LSH [1] is to project similar data points to the same bucket with a higher probability than dissimilar points. In [3], a more practical LSH scheme based on the p-Stable distribution is presented. For the l_2 metric, the typically used LSH function is defined as:

$$h(x) = \left\lfloor \frac{a \cdot x + b}{w} \right\rfloor \tag{1}$$

where a is a d-dimensional random vector with entries chosen independently from a Gaussian distribution and b is a real number chosen uniformly from the range $[0, w]$. E2LSH [4] is the more practical approach for the l_2 norm. For a query point q, its hash bucket is defined as $g(q) = (h_1(q), \ldots, h_k(q))$. To distribute the hash buckets to a Chord style Peer-to-Peer network[9], the mapping function defined as follows in [8].

$$\xi(g(q)) = \sum_{i=1}^{k} h_i(q) \tag{2}$$

The theoretical proof given in [8] points out that l_1 can capture the distance between the buckets in terms of the probability of holding the close data: Given the bucket labels b_1, b_2 and b_3 which are integer vectors of dimension k, if $\|b_1 - b_2\|_1 < \|b_1 - b_3\|_1$, then b_1 and b_2 have a higher probability to hold the similar data than b_1 and b_3. As a result, ξ can assign buckets likely to hold similar data to the same peer. Furthermore, if the elements of a are chosen from the Standard Normal Distribution $N(0, 1)$, $a \cdot x$ is distributed according to the Normal Distribution $N(0, \|x\|_2^2)$. For not too large w, $h(x)$ is distributed according to $N(\frac{w}{2w}, \frac{\|x\|_2^2}{w^2})$. Therefore, $\xi(g(q))$ is distributed according to the following Normal Distribution:

$$\xi(g(q)) \sim N(\frac{k}{2}, \frac{k\|q\|_2^2}{w^2}) \tag{3}$$

The global picture consisting of M data points $q_1, \cdots, q_M$ first projected using p-Stable LSH and then mapped by ξ, following the Normal Distribution:

$$\xi \sim N(\frac{k}{2}, \frac{k\Sigma_i\|q_i\|_2^2}{w^2 M}) \tag{4}$$

So far ξ has a predictable output distribution. After getting ξ and its distribution, we can place r peers at the positions known by all peers by sampling r values from (4) and mapping them to the buckets in the range of $\{1, \ldots, M\}$ using ψ. Here, M is the number of hash buckets.

$$\psi(value) = (\frac{value - (\mu_{sum} - 2 * \sigma_{sum})}{4 * \sigma_{sum}} * M) mod M \tag{5}$$

where μ_{sum} and σ_{sum} are the mean and the standard deviation of the values generated by ξ. These r peers and all the hash buckets are organized as a Chord style Peer-to-Peer network[9] to maintain one hash table. It is called "Local DHT" in [8]. By constructing multiple Local DHTs, multiple hash tables can be distributed.

B. The problem in practical applications

Using the framework reviewed above has two benefits. The first one is that the index can adapt to the large-scale datasets. The second one is that the load of the network and every node can be well balanced. However, there are two disadvantages when using the framework in practical applications.

1) The predicting model defined as (4) can only get an approximate value even if we can get the value of $\frac{\Sigma_i\|q_i\|_2^2}{M}$ precisely. According to the expression (3), k and w are constants, so the distribution is decided by the norms of the points. Different point will have different output distribution. $\frac{\Sigma_i\|q_i\|_2^2}{M}$ is used as an approximate representative value for all the points. When the points' norms change in a wide range, expression (4) may fail to predict the output distribution of all the points. Therefore, the predicting model may significantly deviate from the actual load distribution.
2) When the dataset dynamically changes, i.e. incrementally grows, the predicting model may become invalid. This is because the initial data may be very different from the appended data.

These disadvantages make the framework impractical for many applications. We propose a data independent method to overcome the framework's drawbacks.

III. The data independent method

A perfect predicting model should be independent to any point and keep consistent even if the dataset changes dynamically.

A. The hash function with a consistent output distribution

In the traditional LSH, the hash functions are same for all the points. However, this constraint makes the distribution of the hash value vary for the different point. To get a consistent distribution, a Normal Distribution $N(0, \delta^2)$ is selected as the basic distribution firstly. Given a d-dimensional point p, we generate $a_p = (a_{p1}, a_{p2}, \ldots, a_{pd})$ through drawing the values from the following Normal Distribution:

$$a_{pi} \sim N(0, \frac{\delta^2 k}{\|p\|_2^2}) \tag{6}$$

The hash function for the point p is defined as:

$$h(p) = \left\lfloor \frac{a_p \cdot p + b}{w} \right\rfloor \tag{7}$$

In the expression (7), every element of a_p is drawn from the Normal Distribution $N(0, \frac{\delta^2 k}{\|p\|_2^2})$. Normal Distribution is a 2-stable distribution according to [3]. Therefore, $a_p \cdot p$ follows the following Normal Distribution:

$$N(0, \frac{\delta^2 k}{\|p\|_2^2} \cdot \|p\|_2^2) = N(0, \delta^2 k) \tag{8}$$

This distribution is independent from the point p. For not too large w, $h(p)$ is distributed according to $N(\frac{w}{2w}, \frac{\delta^2 k}{w^2})$. Therefore, $\xi(g(p))$ has the following Normal Distribution:

$$\xi(g(p)) \sim N(\frac{k}{2}, \frac{k^2\delta^2}{w^2}) \tag{9}$$

Let $\xi_{avg}(g(p)) = \frac{\xi(g(p))}{k}$, then the following distribution can be get.

$$\xi_{avg}(g(p)) \sim N(\frac{1}{2}, \frac{\delta^2}{w^2}) \tag{10}$$

If we use the hash function (7) to construct the hash tables and use $\xi_{avg}(g(p))$ to map the k-dimensional hash value to the linear peer space, the distribution will be same for all the points, i.e., we can precisely predict the output distribution for the whole dataset.

However, it is obvious that if two similar points' norms are different, their hash functions will be different. In this case, two similar points may be hashed to the very different values, and two dissimilar points may be hashed to the same value. Therefore, the index structure will fail to answer the similarity queries. This problem can be seen in **Fig. 2**. Similar points, $P2$ and $P3$, are hashed to the different values. Dissimilar points, $P2$ and $P5$ are hashed to the same value. To overcome the problem, the query process must be modified.

When constructing the indexes, all appeared norm values should be recorded. Corresponding to every norm value, a group of hash functions generated according to the formulas (6) and (7) are also recorded. Let all the points with the same norm value use the same hash functions to compute their hash values. The norm values and their corresponding hash functions are organized into a search table. Suppose the search table has M records. Given a query point, every record of the search table is used to compute the query point's hash value. To find the similar points, all of M hash buckets need be searched. An example is shown in **Fig. 2**. In spite of some searches can be eliminated using the triangle inequality, the computation complexity is higher than the traditional LSH. Besides, the additional storage is required to maintain the search table. The root cause of the problem is that the proposed hash function is related to the norm of the point. The points with the equal norm values use the same hash functions. The points with different norm values use different hash functions. In theory, the proposed method can predict the hash value's distribution perfectly. But in practice, it introduces additional computing and storage complexity. In next section, an improved method to resolve the problem is presented. It can make the proposed method applicable in practical applications.

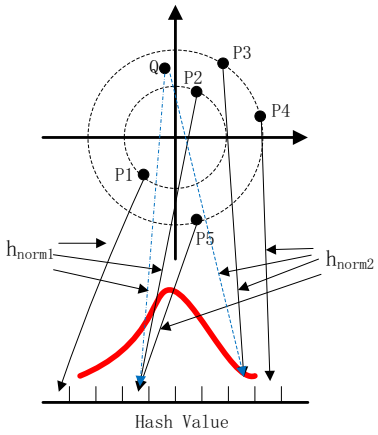


Figure 2. If we use different hash functions for the points with different norms, two similar points may be hashed to the very different values, and two dissimilar points may be hashed to the same value. $P1$ and $P2$ use h_{norm1}, $P3$, $P4$ and $P5$ use h_{norm2}. Similar points, $P2$ and $P3$, are hashed to the different values. Dissimilar points, $P2$ and $P5$ are hashed to the same value. To resolve this problem, the query point Q must use both h_{norm1} and h_{norm2} to search its similar points

B. The improved method for practical applications

In theory, the hash functions for different norm values should be generated independently. However, this constraint makes the hash values not reflect the similarity of the points when these points have different norm values. In the expression (6), we use the Normal Distribution $N(0, \delta^2)$ as the basic distribution to generate all the hash functions. Using the same strategy, we generate a group of basic random values beforehand, all a_p are generated from this basic random value set.

Suppose the points to be indexed is d-dimensional. d values are randomly drawn from $N(0, \delta^2)$. Repeat this process k times. Then a $k \times d$ basic random value matrix is generated. After doing that, the hash functions for every norm value can be easily generated. Let a_{bi} (i from 1 to k) represent a d-dimensional row vector. It corresponds to a row of the basic random value matrix. Given an arbitrary d-dimensional point, e.g., p, we redefine the hash function (7) as follows:

$$h_i(p) = \left\lfloor \frac{c_p a_{bi} \cdot p + b}{w} \right\rfloor \tag{11}$$

where $c_p = \sqrt{k}/\|p\|_2$. When $k \times d$ is big enough, the elements in the basic random value matrix will follow the distribution $N(0, \delta^2)$ well. Based on this precondition, the

elements of $c_p a_{bi}$ will follow the distribution described in (6). Therefore, the hash function (11) can be regarded as an approximation of the hash function (7). Specially, it does not need to be recorded because it can be computed at anytime. As a result, the additional storage space is saved. Next, we will detail how to optimize the query performance.

In section III-A, we point out that a query point must be computed M times to get M hash values and correspondingly M buckets will be searched. It is designed to avoid missing the similar points which have the different norm values. When M is big, although we can use the triangle inequality to reduce the computation, the computing cost is still much higher than the traditional LSH. Additionally, it can cause random access to multiple peers of the distributed index. Every random access to a peer in the Chord style Peer-to-Peer network needs $\frac{\log N}{2}$ network hops. N is the number of peers. In this case, the query efficiency may heavily decline. Using the hash function (11), the problem can be solved.

Given a query point q and its similar point p, according to the principle of LSH, $a_{bi} \cdot q$ and $a_{bi} \cdot p$ should be same or slightly different with a high probability. We record their relationship as follows.

$$a_{bi} \cdot q \simeq a_{bi} \cdot p \tag{12}$$

Consequently, c_p and c_q decide whether q and p can be hashed to the same value according to (11). If p and q have the equal norm values, they can be hashed to the same value, and vice versa. However, because q and p are similar points, their norm values can not have a large difference. Otherwise, according to the triangle inequality, their distance will be larger than the difference of their norm values. Therefore, the hash value of p should be close to the hash value of q. If we linearly probe the adjacent buckets of q, its similar points can be found in acceptable steps. So far the computing cost of getting M hash values for every norm value and the random network lookup for probing M buckets can be avoided. After mapping the hash buckets to the linear peer space by ξ_{avg}, the above property can be reserved. That is to say, the query point and its similar points will be mapped to the same or the adjacent peers with a high probability. Taking the peer to which the query point is mapped as the start, we can sequentially search its adjacent peers on both the left and the right directions to find the similar points. So far we can use the same algorithm as provided in [10] and [8] to construct a distributed index and execute a KNN query. For a more detailed description see [10] and [8].

IV. Experiments

A. Experiment setup

We implement the proposed method and a simulation system using standard C++. The simulation runs on a 2.27 GHz Quad-Core Intel Xeon E5520 CPU with 24G RAM and Redhat CentOS 5.0.

Table I
Datasets Summarizing

Type	Dim	Base Vectors	Query Vectors	$\sqrt{\frac{\Sigma_i \|q_i\|_2^2}{M}}$
SIFT	128	1,000,000	10,000	508
GIST	960	1,000,000	1,000	3

In the experiments, two open large-scale high-dimensional feature sets are used. They are provided by TEXMEX Research Team[11]. The detail is summarized in **Table I**.

We only simulate one local DHT of 100 peers to maintain a single LSH table. This is because the local DHTs are independent of each other. For both datasets, the Euclidean distance is used to measure the distances between the points, all the dimensions are treated equally and without being preprocessed.

For the LSH parameters, the default values are $k = 32$ and $w = 4$ for the SIFT set and $k = 128$ and $w = 0.1$ for GIST set.

B. Evaluation criterion

In [8], three evaluation criterions are used. They are Gini Coefficient, Number of Network Hops and Relative Recall. In our experiments, we continue to adopt them to validate our method.

Gini Coefficient: The Gini Coefficient is a measure of the inequality of a distribution, a value of 0 expressing total equality and a value of 1 maximal inequality. Pitoura et al[12] show that the Gini Coefficient is the most appropriate statistical metric for measuring load distribution fairness. If the Lorenz curve is represented by the function $Y = L(X)$, Gini Coefficient can be computed as follows:

$$G = 1 - 2 \int_0^1 L(X) dX \tag{13}$$

Number of Network Hops: The number of the network hops is one of the most critical criterion for distributed algorithms applicable in the large-scale wide-area networks. Because we only construct one local DHT, the global DHT lookups can be ignored. Besides, only one local DHT lookup is needed to locate the peer to which the query point is mapped. It can also be ignored because it is a constant. Hence, only the network hops for linear search in the local DHT are counted in the experiments.

Relative Recall: Relative Recall is the number of real KNN points among the returned KNN points. Since we rank all the candidate points and returning only the top K in KNN queries, the precision is equal to the relative recall and we do not report it separately. Real KNN points are determined by a full-scan over the entire dataset. The datasets used in our experiments provide the groundtruth sets for the Top-100 queries.

C. Comparison of load balance

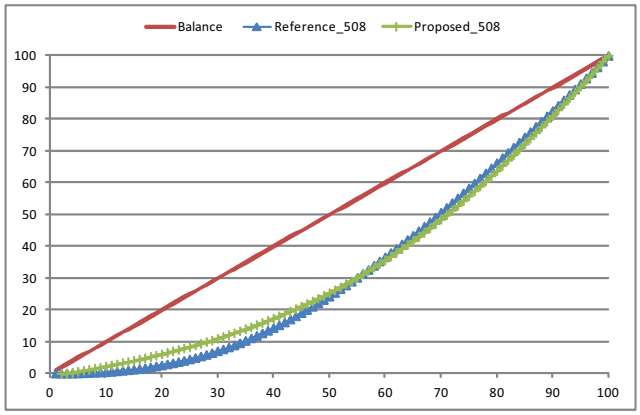
We compare the load distribution of the proposed method against that of Haghani[8] on both the static and dynamic datasets. For both the proposed method and the reference method, the best results of five repeated experiments are selected for comparison.

The experimental results on the static datasets: In this experiment, we use the whole test dataset as the initial dataset to construct the distributed indexes. That is to say, the initial dataset is same as the whole test dataset. Therefore, the dataset keep static in the experiment. In this case, the reference method can precisely compute the value of $\sqrt{\frac{\Sigma_i \|q_i\|_2^2}{M}}$ on the whole dataset beforehand. The values are listed in **Table I**. For the proposed method, δ can be arbitrarily selected. In this experiment, we let it equal to 508 for the SIFT dataset and 3 for the GIST dataset.

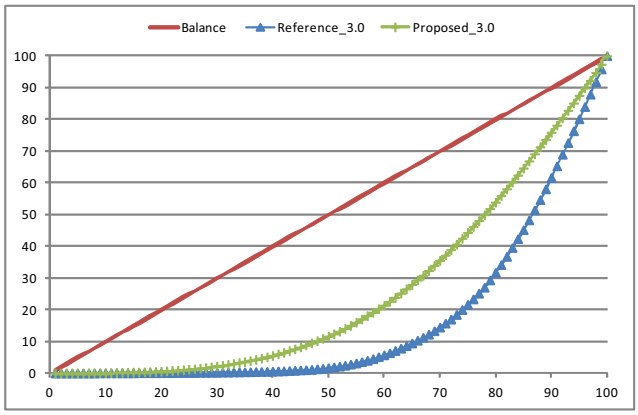
Fig.3 shows two groups of Lorenz curves which reflect the data distribution on 100 peers for the SIFT and GIST datasets. The curve is more close to the balance line, the distribution is more balanced. It means the load of every peer is more fair. The corresponding Gini coefficients are summarized in **Table II**. In **Fig.3(a)**, both methods have very similar distribution. Their Gini coefficients are all 0.32. In **Fig.3(b)**, the proposed method gets a better load balance than the reference method. The Gini coefficient of the proposed method is 0.48 which is significantly smaller than the value of the reference method which is 0.66. In fact, the norms of the points in the SIFT dataset are almost same, but the norms of the points in the GIST dataset change in a wide range. As a result, the reference method can precisely predict the distribution of the SIFT dataset to get the same good load balance as the proposed method, but it can not get a good load balance on the GIST dataset. The experimental results prove that the proposed method has a better adaptability than the reference method.

The experimental results on the dynamic datasets: To simulate the dynamic case, we generate two simulated datasets. Each dataset has 10000 points. One is 128-dimensional with $\sqrt{\frac{\Sigma_i \|q_i\|_2^2}{M}} = 1000$, the other is 960-dimensional with $\sqrt{\frac{\Sigma_i \|q_i\|_2^2}{M}} = 6$. The points in the simulated datasets are generated from random values. For example, we generate a 128-dimensional random point at first, and then we normalize it. Finally, we let its norm become 1000 though producing 1000 on every dimension of it. We take these two simulated datasets as the initial datasets and construct the distributed indexes on them using both methods. Then we insert the SIFT dataset and the GIST dataset into the constructed indexes. Therefore, the initial dataset is every different from the final dataset. For the reference method, the predicting model estimated on the initial dataset will fail to predict the actual distribution.

For the reference method, $\sqrt{\frac{\Sigma_i \|q_i\|_2^2}{M}}$ is equal to 1000 for the SIFT dataset and 6 for the GIST dataset. For the proposed method, we let δ equal to 1000 for the SIFT datset and 6 for the GIST dataset. **Fig.4** shows two groups of Lorenz curves which reflect the data distribution on 100 peers for the SIFT and GIST datasets. The curve is more close to the balance line, the distribution is more balanced. It means the load of every peer is more fair. The corresponding Gini coefficients are summarized in **Table II**. **Fig.4(a)** shows the results on the dynamic SIFT dataset. **Fig.4(b)** shows the results on the dynamic GIST dataset. The distributions of the reference method in both figures become very imbalanced. In contrast, the distributions of the proposed method keep well balanced in both figures. Specially, its Gini coefficients are almost same as the values on the static datasets. From the experimental results, we can conclude that the proposed method can get a better load balance even if the dataset



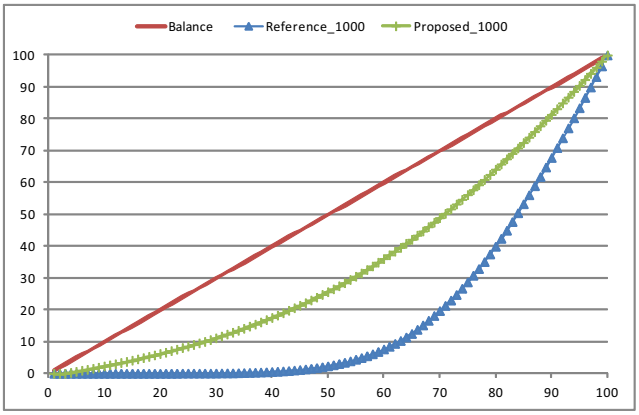
(a) SIFT



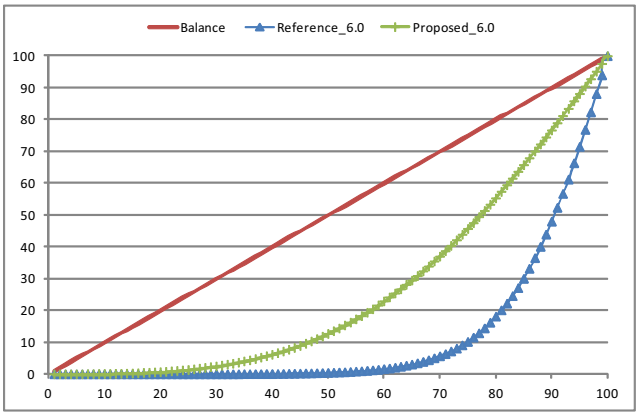
(b) GIST

Figure 3. The Lorenz curves of both methods on the static SIFT and GIST datasets. The horizontal axis is the cumulative share of the peers and the vertical axis is the cumulative share of the points distributed to those peers. (a)Both methods have very similar load balance. This is because the norms of the points in the SIFT dataset are almost same.(b)The proposed method gets a better load balance than the reference method. This is because the norms of the points in the GIST dataset change in a wide range.

changes significantly.



(a) SIFT



(b) GIST

Figure 4. The Lorenz curves of both methods on the dynamic SIFT and GIST datasets. The horizontal axis is the cumulative share of the peers and the vertical axis is the cumulative share of the points distributed to those peers. (a)The proposed method gets a better load balance than the reference method on the dynamic SIFT dataset(b)The proposed method gets a better load balance than the reference method on the dynamic GIST dataset

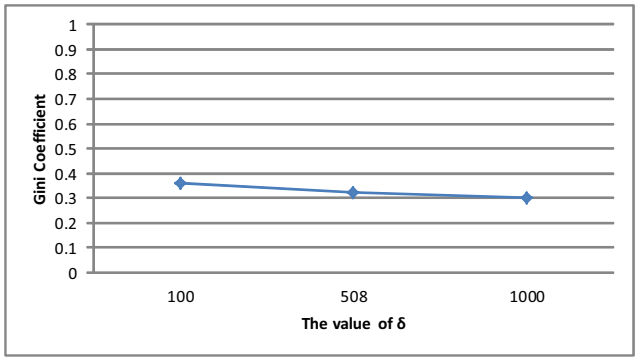
Table II
GINI COEFFICIENTS. THE SMALLER THE BETTER.

	SIFT		GIST	
	static	dynamic	static	dynamic
Reference	0.32	0.54	0.66	0.76
Proposed	0.32	0.30	0.48	0.46

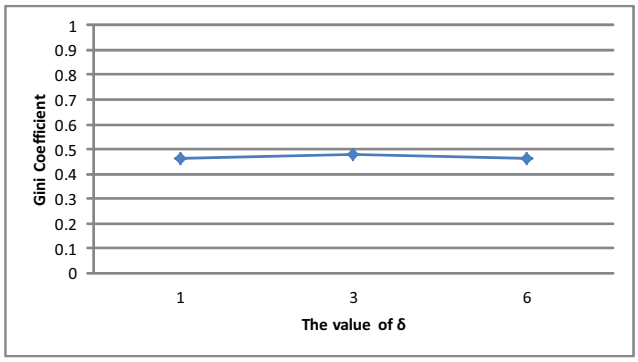
The impact of δ for the proposed method: The only parameter of the proposed method is δ. In fact, the selection of δ has only a slight impact on the load distribution. As shown in **Fig.5**, the proposed method can get the very similar Gini coefficients on the SIFT and GIST datasets even if δ varies in a wide range. Therefore, we can select δ freely.

D. Performance for KNN

We do Top-100 queries on the indexes constructed in section IV-C. The exactly same search algorithm and stop



(a) SIFT



(b) GIST

Figure 5. δ has only a slight impact on the load balance. (a)The proposed method can get the very similar Gini coefficients on the SIFT dataset when $\delta = 100, 508 and 1000$ (b)The proposed method can get the very similar Gini coefficients on the GIST dataset when $\delta = 1, 3 and 6$

conditions[8] are used for both methods. The relative recall and the number of network hops are computed and compared. Both the SIFT and the GIST datasets provide a query set and the corresponding groundtruth set. It is convenient for computing relative recall. All measures are averaged over 1000 queries which are randomly selected from the query set.

As shown in **Fig.6**, on the static SIFT dataset, the proposed method can get the relative recall of 21.3% with 46 network hops. The reference method gets the relative recall of 18.9% with 40 network hops. They have comparable query performance. In **Fig.8**, the proposed method can get the relative recall of 20.8% with 46 network hops on the dynamic SIFT dataset. The reference method gets the relative recall of 20.5% with 37 network hops. They also have comparable query performance. The same phenomenon can also be seen in **Fig.7** and **Fig.9**.

In **Fig.6 - 9**, the number of the network hops of the proposed method is more than that of the reference method. This is because the load distribution of the proposed method is more balanced. The similar points of the query may be distributed to more peers than the reference method. It leads

to more network hops. In fact, it proves that the proposed method can get a better trade-off between the load balance and the query performance in practical applications. From the experimental results, we can conclude that the query performance of the proposed method is comparable to that of the reference method while keeping a better load balance on both the static and the dynamic datasets.

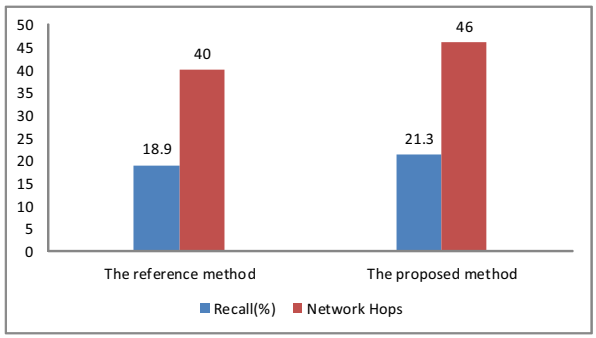


Figure 6. The KNN performance on the static SIFT dataset. Both methods have the comparable query performance.

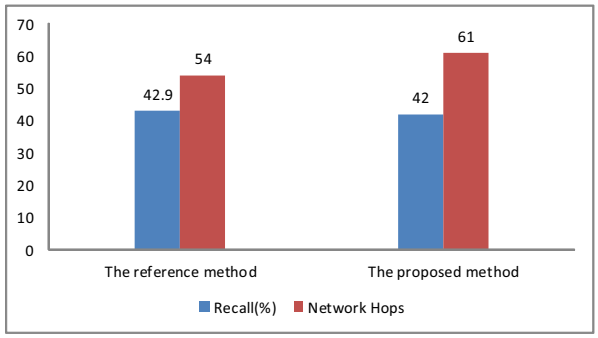


Figure 7. The KNN performance on the static GIST dataset.Both methods have the comparable query performance.

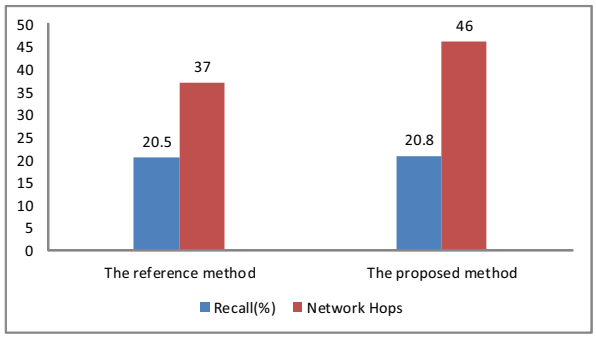


Figure 8. The KNN performance on the dynamic SIFT dataset.Both methods have the comparable query performance.

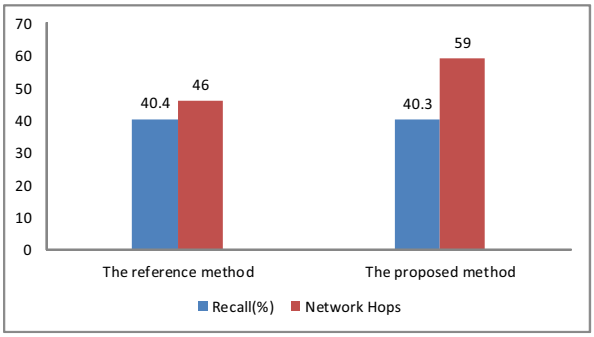


Figure 9. The KNN performance on the dynamic GIST dataset.Both methods have the comparable query performance.

V. Conclusions

In this paper, we propose a data independent method of constructing distributed LSH for the large-scale high-dimensional multimedia feature sets. The proposed method is inspired by the framework firstly provided in [8] which combines LSH and Chord style Peer-to-Peer network for the distributed similarity search problem. Through generating the hash function with a consistent output distribution and modifying the query algorithm of the basic LSH, we get three advantages. Firstly, it is avoided to compute the global parameter on the whole dataset beforehand as done in [8]. Secondly, there is no any estimated parameter in the expression. That is to say, the proposed model is more precise. Thirdly, the proposed model does not need to know any priori knowledge about the dataset. Consequently, even if the dataset dynamically changes, the proposed model can keep a well balanced load and a stable performance. The experiments conducted on two open large-scale high-dimensional datasets show that the proposed method is more robust, scalable and practical than state-of-the-art.

Acknowledgment

This work is supported by the National Nature Science Foundation of China (60802028); National High Technology and Research Development Program of China (863 Program, 2009AA01A403); Co-building Program of Beijing Municipal Education Commission.

References

[1] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *Proc. of 25th Intl. Conf. on Very Large Data Bases(VLDB)*, 1999, pp. 518–529.

[2] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," *Commun. ACM*, vol. 51, pp. 117–122, January 2008.

[3] M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proceedings of the twentieth annual symposium on Computational geometry*, ser. SCG '04. New York, NY, USA: ACM, 2004, pp. 253–262.

[4] A. Andoni and P. Indyk. (2004) E2lsh: Exact euclidean locality sensitive hashing. [Online]. Available: http://web.mit.edu/andoni/www/LSH/

[5] Panigrahy and Rina, "Entropy based nearest neighbor search in high dimensions," in *Proceedings Of The Seventeenth Annual Acm-siam Symposium On Discrete Algorithm*, ser. SODA '06. New York, NY, USA: ACM, 2006, pp. 1186–1195.

[6] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li, "Multi-probe lsh: Efficient indexing for high-dimensional similarity search," in *Proceedings Of The 33rd International Conference On Very Large Data Bases*, ser. VLDB '07. VLDB Endowment, 2007, pp. 950–961.

[7] A. Joly and O. Buisson, "A posteriori multi-probe locality sensitive hashing," in *Proceeding Of The 16th Acm International Conference On Multimedia*, ser. MM '08. New York, NY, USA: ACM, 2008, pp. 209–218.

[8] P. Haghani, S. Michel, and K. Aberer, "Distributed similarity search in high dimensions using locality sensitive hashing," in *Proceeding of ACM. EDBT 2009*, March 2009, pp. 744–755.

[9] I. Stoica, R. Morris, D. R. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications." in *In SIGCOMM*, 2001.

[10] P. Haghani, S. Michel, P. CudreMauroux, and K. Aberer, "Lsh at large-distributed knn search in high dimensions," in *Proceedings of the 11th International Workshop on Web and Databases (WebDB 2008)*, June 2008.

[11] *http://corpus-texmex.irisa.fr/*.

[12] T. Pitoura and P. Triantafillou, "Load distribution fairness in p2p data management systems," in *In ICDE*, 2007.