2009 Ninth IEEE International Conference on Data Mining

A Cost-Effective LSH Filter for Fast Pairwise Mining

Gang Zhao*, Yun Xiong*, Longbing Cao†, Dan Luo†, Xuchun Su† and Yangyong Zhu*
*School of Computer Science
Fudan University, Shanghai, China
Email: {061021069,yunx,yyzhu}@fudan.edu.cn
†Faculty of Engineering and Information Technology
University of Technology Sydney, NSW, Australia
Email: {lbcao,dluo,xsu}@it.uts.edu.au

***Abstract*—The pairwise mining problem is to discover pairwise objects having measures greater than the user-specified minimum threshold from a collection of objects. It is essential in a large variety of database and data-mining applications. Of late, there has been increasing interest in applying a Locality-Sensitive Hashing (LSH) scheme for pairwise mining. LSH-type methods have shown themselves to be simply implementable and capable of achieving significant performance gain in running time over most exact methods. However, the present LSH-type methods still suffer from some bottlenecks, such as "the curse of threshold". In this paper, we proposed a novel LSH-based method, namely Cost-effective LSH filter (Ce-LSH for short), for pairwise mining. Compared with previous LSH-type methods, it uses a lower fixed number of LSH functions and is thus more cost-effective. Substantial experiments evidence that our method gives significant improvement in running time over existing LSH-type methods and some recently reported method based on upper-bound. Experimental results also indicate that it scales well even for a relatively low minimum threshold and for a fairly small miss ratio.**

***Keywords*-pairwise mining; locality hashing function**

I. Introduction

The pairwise mining problem involves a collection of objects (e.g., documents [1]) that are represented as vectors in Euclidean or Hamming space. Given a measure function $\mathcal{D}$ defined on the collection of objects, we are required to find all the related pairs of objects that have the measure $\mathcal{D}$ greater than a pre-specified minimum threshold. The particularly interesting and well-studied case in data mining is Hamming space, which is related to the Market-basket Data model. The problem is of major importance to a variety of data mining applications, such as association rule mining [4], duplicate detection [1], and data analysis [11].

Typically, in the case that input data contains a huge number of objects, a brute-force enumeration algorithm is impractical because of expensive space/time cost. Performing a filtering process is always challenging due to the fact that the few measures are anti-monotonic. In many cases it is not necessary to insist on complete results that contain all above-threshold pairs; instead, determining an output consisting of *nearly all* related pairs should suffice. This principle underlies the large body of recent research in databases and data mining (e.g., [4][12]).

The above principle applies to the pairwise mining problem. Many studies have shown that locality-sensitive hashing (LSH) [7][3] has some desired advantages as a suitable candidate for pairwise mining. Such arguments rely on the assumption that an approximate pairwise-mining algorithm can be performed much faster than an exact one. This is indeed the case for the LSH algorithm in the majority of cases. However, LSH-type methods scale poorly as minimum threshold decreases, and it is likely to be slower than an exact algorithm when the minimum threshold is set relatively low [11]. In this work, we propose a novel LSh-based method, which performs more efficient filtering scheme involving far fewer LSH functions. We call the proposed scheme *Cost-effective* LSH filter (*Ce-LSH* for short) due to the fact that the main time/space costs of the LSH-type algorithm is dominated by the number of LSH functions [4]. It is provable that *Ce-LSH* uses $O(\epsilon^{-2}\log n)$ LSH functions to solve the pairwise mining problem, which states the considerable reduction of $O(n^{\frac{1}{1+\epsilon}}\log^2 n)$ LSH functions in previous works. As a result, our method is highly probably much faster in all cases than previous LSH methods and any exact algorithms. Furthermore, a remarkable property of the *Ce-LSH* filter is that the number of LSH functions is independent when other approximation parameters increase. It makes our method scale well as the minimum threshold decreases, which attacks the "curse of threshold".

We support our theoretical arguments by empirical evidence. Substantial experiments are conducted on widely used measures and both real and synthetic data sets to compare the performance of our algorithm with those of the previous LSH-based algorithm and TAPER, a recently reported upper-bound based algorithm [11]. Experimental results indicate that our technique is scalable for both low and high minimum threshold and significantly faster than other LSH-type methods and TAPER, in some cases by about an order of magnitude. Compared with other cases in which the previous LSH-type methods mainly work, the running time of ours is about 20%-30% of that of the state-of-the-art LSH technique. In addition, it also scales much better than existing LSH-type methods as the data size (the number of objects) and the accuracy of the result increase.

1550-4786/09 $26.00 © 2009 IEEE
DOI 10.1109/ICDM.2009.112

The rest of this paper is organized as follows. In Section II, we briefly review previous works. Section III states the problem. Section IV presents the *Ce-LSHn* filter and related analysis. A *Ce-LSH*-based algorithm and its complexity analysis are introduced in Section V. Experimental results are described in Section VI. The work is concluded in Section VII.

II. Related Work

Efficiently mining for related pairs plays an essential role in various applications such as data mining [4], database [1], statistics and data analysis [11]. When an approximation result suffices, the properties of the Locality-Sensitive Hashing (LSH) function make it a suitable candidate for pairwise mining [7]. To the best of our knowledge, the existing LSH type methods for pairwise mining can be summarized in the following forms:

One technique is to quantitatively estimate the measure of the pairwise objects by defining an estimator based on hash values [1][3]. The estimation scheme executes the comparisons of all possible pairs, which is quadratic to the total number of objects. Thus it is not scalable for mining a large scale data set.

The state-of-the-art technique based on LSH is a "filter scheme" that performs the randomly filtering for avoiding pairwise comparisons. Many studies have shown that this scheme yields an efficient solution for pairwise mining that considerably outperforms the estimation methods and other exact algorithms in most cases. However, such a method suffers from the "curse of the threshold" (see [12]) and is thus highly likely to run slower than an exact algorithm when the minimum threshold is fairly low.

We call the above filter scheme a Baseline LSH filter or *B-LSH* filter for short, and use it as a main benchmark algorithm in our analysis and evaluation, and later propose a more cost-effective filter scheme and thus remarkably outperforms previous techniques.

III. Problem Statement

We define pairwise mining and LSH function in terms of *Hamming space*. Our scheme is equally applicable to the various settings (e.g., the measures defined in Euclidean space [5]), since different LSH functions can be also used for other measures [3].

Let T be a binary data table with m rows and n columns where the column-vector represents the object. Let $\mathcal{D}(\cdot,\cdot) \in [0,1]$ is a measure function defined on the collection of objects (columns). The problem of pairwise mining is formulated as follows.

Definition 1. (Pairwise mining problem) *The problem of pairwise mining is to generate a set S of pairs of columns from T, and ensure that all and only pairs in S have $\mathcal{D}$ greater than a user-specified minimum threshold r.*

In this work, we decide whether two columns are output in terms of locality-sensitive hashing (LSH) functions, which is defined as follows:

Definition 2. (LSH scheme) *A function family $\mathscr{H} = \{h : \{0,1\}^m \rightarrow \mathbb{U}\}$ is locality-sensitive for measure $\mathcal{D}$ if for any $c_i, c_j \in \{0,1\}^m$,*

- *if $\mathcal{D}(c_i, c_j) > r$ then $Prob_{\mathscr{H}}\{h(c_i) = h(c_j)\} > p_1$*
- *if $\mathcal{D}(c_i, c_j) < (1-\epsilon)r$ then $Prob_{\mathscr{H}}\{h(c_i) = h(c_j)\} < p_2$ ($p_1 > p_2$ and $0 < \epsilon < 1$)*

To solving pairwise mining problem correctly and efficiently, we need to amplify the gap between p_1 and p_2. That is, we are required to use as few LSH functions as possible to ensure that the pairs with $\mathcal{D}$ above r is output with fairly high probability, while the pairs with $\mathcal{D}$ at most $(1-\epsilon)r$ is output with suitably low probability.

IV. Ce-LSH Filter: Design and Analysis

We define a filter as a randomized procedure that operated on a collection of hash values, in which two columns that have the measure above the minimum threshold r are accepted as a candidate pair with high probability (ideally with probability of "1"), while two columns that have the measure at most $(1-\epsilon)r$ are accepted as a candidate pair with low probability.

A. The Construction of Ce-LSH Filter

Suppose that there is a LSH family $\mathscr{H}$ for a given measure D. We construct our filter procedure as follows.

For an integer k to be specified later, compute k hash values for each column of table T by using k LSH functions from the family $\mathscr{H}$, and store those generated hash values in a table $\hat{T}[1 : k][i]$. Note that this phase produces a "summary" $\hat{T}$ of original table T. In the second phase, we will decide whether to accept two columns as a candidate pair on top of $\hat{T}$. For an integer t to be specified later, randomly pick t ($t < k$) row-coordinate positions $I_1, ..., I_t$ of $\hat{T}$. Let c_{jI} denote the projection of a column-vector c on the coordinate positions $I = \{I_1, ..., I_t\}$, i.e., we compute c_{jI} by selecting the coordinate positions as per I and concatenating the t bits (hash values) in those positions. We denote c_{jI} by a random function $g_{jk,t}(c) : \mathbb{U}^k \rightarrow \mathbb{U}^t$ that operates on table $\hat{T}$. Then two columns c_i and c_j are accepted as a candidate pairs if and only $g_{jk,t}(c_i) = g_{jk,t}(c_j)$; otherwise, they are filter out. To ensure that more above-threshold pairs are found by our filter, the above process of computing $g_{jk,t}(c)$ on top of $\hat{T}$ is repeated multiple, say l, times.

Example 1 illustrates how a candidate pair is generated in one trial.

Example 1. *Suppose that 6 hash values are generated for each of 4 columns by using LSH functions $\{h_1, ..., h_6\} \in \mathscr{H}$. Set $t = 3$. Then the function $g_{j6,3}(\cdot)$ for each columns can be computed as follows: randomly pick three row positions,*

say, 5, 2,*and* 2, *i.e.,* $I = \{5, 2, 2\}$. *We now obtain four 3-size "LSH-IDs" for all columns, which are represented by the columns of the following sub-matrix. In this case,* (c_1, c_3) *is the only accepted pair.*

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array}\!\!\begin{array}{c} \begin{array}{cccc} c_1 & c_2 & c_3 & c_4 \end{array} \\ \begin{pmatrix} 2 & 7 & 2 & 8 \\ 4 & 2 & 4 & 5 \\ 5 & 2 & 3 & 6 \\ 7 & 8 & 3 & 7 \\ 3 & 6 & 3 & 4 \\ 1 & 4 & 3 & 4 \end{pmatrix} \end{array} \Longrightarrow \begin{array}{c} \\ 5 \\ 2 \\ 2 \end{array}\!\!\begin{array}{c} \begin{array}{cccc} c_1 & c_2 & c_3 & c_4 \end{array} \\ \begin{pmatrix} 3 & 6 & 3 & 4 \\ 4 & 2 & 4 & 5 \\ 4 & 2 & 4 & 5 \end{pmatrix} \end{array}$$

One can observe that when we repeat the process of computing $g^w_{jk,t}(\cdot)$ $(1 \le w \le l)$, a coordinate position may be chosen more than once in some "LSH-ID" or be chosen for computing different $g^w_{jk,t}(c)$ in more than one trials. It differs from the *B-LSH* scheme, in which a coordinate position is chosen once in all trials, and the number of the LSH functions in the *B-LSH* scheme equals the product of t and l. In *Ce-LSH*, one can increase the values of t and l such that the value of $t \cdot l$ exceeds the value of k. Thus, the *Ce-LSH* filter comprises *three approximation parameters*, which are t, l and k.

B. Analysis

We now show the correctness of a *Ce-LSH* filter. Observe that the *Ce-LSH* filter *correctly* solve the pairwise mining problem if the following properties hold with high probabilities:

- P1 If $\mathcal{D}(c_i, c_j) > r$, then (c_i, c_j) is accepted.
- P2 If $\mathcal{D}(c_i, c_j) < r(1-\epsilon)$ $(\epsilon > 0)$, then (c_i, c_j) is rejected.

Note that the probability P_1 held by **P1** should be extremely high (ideally, 1), because it determine the occurrence of false negatives and thus the completeness of the results. The probability P_2 that **P2** holds is required to be relatively high, because a relatively small number of false-positives can be removed at small additional cost. The correctness of the *Ce-LSH* filter follows from the following fact.

Fact 1. *For a proper choice of* t, l *and* k, *properties* **P1** *and* **P2** *hold with high probabilities (*P_1 *is essentially* 1*) for a* Ce-LSH *filter.*

Proof: For the analysis, we refine the definition of LSH as the form in [3]; all the following results hold equally for the Definition 2. We first consider the probability that a pair having $\mathcal{D} = d$ is accepted in a *Ce-LSH* filter, which is given as follow:

$$\mathcal{F}_{t,l,k}(d) = \sum_{X=0}^{k} \binom{k}{X} r^X (1-d)^{k-X} \cdot f(\frac{X}{k})$$

We call the such a probability function a *filter function*, and observe that $\mathcal{F}_{t,l,k}(d)$ is the *Bernstein Polynomial* [8] of function $f_{t,l}(d) = 1-(1-d^t)^l$. That is, for $k > \|f\| \cdot \mu^{-1}\delta^{-2}$ and any $\mu, \delta > 0$ ($\delta > 0$ is constant such that $|f(x)-f(y)| < \frac{\mu}{2}$ whenever $|x-y| < \delta$),

$$\|\mathcal{F}_{t,l,k}(d) - f_{t,l}(d)\| < \mu.$$

The Fact 1 can follow from the properties of the functions $\mathcal{F}_{t,l,k}(d)$ and $f_{t,l}(d)$ ($f_{t,l}(d)$ is the filter function of *B-LSH*), since this kind of functions can approximate *a step function* translated to the threshold point $C = r$ for a proper choice of parameters. ■

According to *Berstien theorem*, the following property holds for *Ce-LSH* filter.

Remark 1. *The number* k *of LSH functions can be* fixed *when* t *and* l *are increased to provide a better approximation result.*

We now quantify the cost saving of the *Ce-LSH* filter by estimating the value of k, since the reduction in the number of LSH functions constitutes the main contribution of this work. We decide not to use directly the results of *Berstein theorem* [8]. Instead, the proof of *Bernstein Theorem* states that the condition $k > \frac{\|f\|}{\mu\epsilon^{-2}}$ is necessary only in the case that $|\frac{X}{k} - d| \ge \delta$. So a pair (c_i, c_j) is called *badly-estimated* if $|\frac{X}{k} - D(c_i, c_j)| \ge \delta$; otherwise, it is called *well-estimated*; for the latter, the only requirement to ensure that $\|\mathcal{F} - f_{t,l}\| < \mu$ is that the value of δ is suitably small such that $|f(x) - f(y)| < \frac{\mu}{2}$ whenever $|x-y| < \delta$ $(x, y \in [0, 1])$. Then the lower bound of k can be relaxed to ensure that *badly-estimated* pairs appear with extremely small probability.

Lemma 1. *By generating* $O(\delta^2 \log n)$ *hash values, the probability that a* badly-estimated *pair appears is* $O(\frac{1}{n^c})$ *(*c *is a arbitrary positive number).*

The quality of a *Ce-LSH* filter is satisfied if the following two properties hold:

1) $\|\mathcal{F}(r) - f(r)\| \approx \frac{1}{n^3}$. It indicates that the probability of missing pairs in *Ce-LSH* is same as that of *B-LSH*
2) $\mathcal{F}_{t,l,k}(d) \approx \frac{1}{n}$ when $d < (1-\epsilon)r$. It indicates that the costs for moving the unrelated pairs in both filter schemes are of same order of magnitude.

So the expected number of LSH functions in the *Ce-LSH* filter is given by the following theorem.

Theorem 1. *By computing* $O(\epsilon^{-2} \log n)$ *LSH functions, property* **P1** *holds with probability* $1 - O(\frac{1}{n^3})$ *and property* **P2** *holds with probability with* $1 - O(\frac{1}{n})$.

Obviously, the *Ce-filer* is more cost-effective, since it uses far fewer LSH functions to perform an efficient filtering (*B-LSH* scheme uses $O(n^{\frac{1}{1+\epsilon}} \log n)$ LSH functions).

V. The Algorithm for Pairwise Mining

We resort to a hash table to manage the buckets used in the filter procedure (refer to [10] for an alternative technique

based on sorting for managing the buckets). That is, we treat the $g_{jk,t}(c)$ as the hashing key of column c and store c on the bucket indexed by $g_{jk,t}(c)$. To obtain the candidate pairs, we search all the buckets of the hash table and store all the pairs that have been mapped into the same buckets in a candidate set $\mathcal{L}$. Finally, we determines for each candidate pair whether it indeed has measure greater than minimum threshold. Then the final output S does not include any false positives.

For quick reference, we summarize the *preprocessing algorithm* and *pairwise discovering algorithms* as follows.

Algorithm 1 Preprocessing

Input T, k (number of LSH functions), t, l
Output A hash table $\mathcal{HT}$
  for each column c_i in T **do**
    Generate k hash values for c_i by using k LSH functions
    Store all k hash value in $\hat{T}[1:k][i]$
    /* Produce a summary $\hat{T}_{k\times m}$ of T* /
  end for
  Initialize a hash table $\mathcal{HT}$
  for $i = 1$ to l **do**
    for $j = 1$ to n **do**
      Store c_j on bucket $g^i_{jk,t}(c_j)$ of hash table $\mathcal{HT}$
    end for
  end for

Algorithm 2 Pairwise Discovering

Input Minimum threshold r
Access To hash table $\mathcal{HT}$ generated by preprocessing algorithm
Output A set S consisting of the above-threshold pairs
  $S \leftarrow \emptyset$
  $\mathcal{L} \leftarrow \emptyset$
  for all bucket $B[i]$ of hash table $\mathcal{HT}$ **do**
    if $B(i) \notin \emptyset$ **then**
      $\mathcal{L} \leftarrow \mathcal{L} \cup$ {all pairs found in bucket $B[i]$ of $\mathcal{HT}$}
    end if
  end for
  for each pair (a, b) in $\mathcal{L}$ **do**
    if $\mathcal{D}(a, b) > r$ **then**
      $S \leftarrow S \cup (a, b)$
    end if
  end for

The total time for discovering all pairwise objects can be decomposed into two terms: the time for preprocessing and the the time for searching the buckets. The expected running time for solving the pairwise mining problem is then given in the next theorem.

Theorem 2. *The above solution solves the pairwise mining problem in expected time* $O(\epsilon^{-2} m \log n + (m + n^{\frac{1}{1+\epsilon}} \log n)$(*# of related pairs*) $+ m \log n$(*# of boundary pairs*)) *and space* $O(mn +$ (*# of related pairs*)).[1]

VI. Experiments

We test our algorithm on set resemblance $\mathcal{R}$ and Pearson correlation coefficient ϕ (refer to [1] and [11] for definitions). Note that both of the measures are representative and widely used in data-mining and have been handled by the *B-LSH* scheme in past studies. The LSH function can be generated by the Min-wise Independent Permutation scheme [2].

All the experiments were conducted on real life and synthetic data sets with various scales, and were performed on a PC with 2.66GHz CPU and 3.25G RAM. The real data here can be obtained from the public database[2]. The pumsb data set corresponds to binarized version of a census data set from IBM. The retail is an anonymous data set obtained from a large mail-order company. The LA1 data set is part of the TREC-5 collection (http://trec.nist.gov) and contains news articles from the Los Angeles Times.

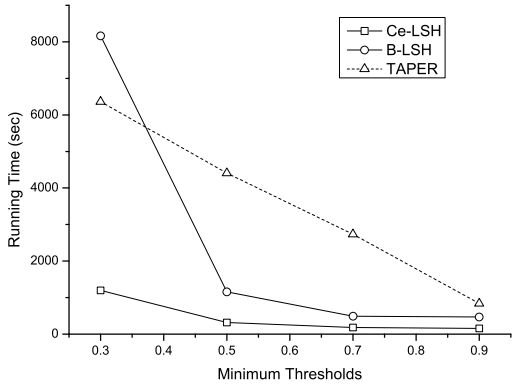
Synthesized data were used for evaluating the scalability of algorithms. We decided not to use randomly-chosen synthetic data in our experiments due to the observation that the overwhelming majority of pairs have fairly low measures in real data and this fact should also hold for a well-defined measure. To simulate the real distribution, we generated synthesized data by repeatedly applying sampling pairs from all possible pairs of LA1 until we produced the data sets with the expected sizes.

Comparisons of Running Times. We compared the performance of our algorithm with those of the *B-LSH* scheme, and TAPER [11], an *upper-bound based* algorithm for mining correlated pairs under the measure ϕ. We here report the results on pumsb data set and retail data set. The miss ratio is set to be smaller than 0.03. However, we observe that except for the very low threshold (say, below 0.3), an exact result set can be obtained by our algorithm. Figure 1 plots the running times of the algorithms against the minimum thresholds for the data sets *retail* (Figure 1(a)) and *pumb* (Figure 1(b)).

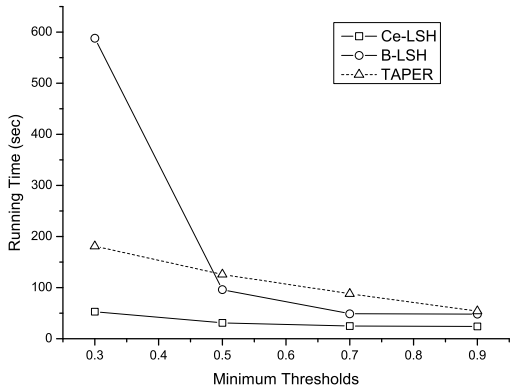
The results show that the *Ce-LSH* runs significantly faster than other methods and scales remarkably better than others for a relatively low minimum threshold. Observe that our technique is about an order of magnitude faster than the *B-LSH* scheme when minimum threshold is 0.3. In other cases, the average running time of ours is still a mere 20%-30% of those of *B-LSH* algorithm and TAPER. Notice that the running time of the *B-LSH* grows almost exponentially as the threshold decreases and runs even slower than TAPER when the minimum threshold is low. The reason is that the

[1] We assume that a LSH function is computed in $O(m)$, and call a pair having $\mathcal{D}$ between $(1-\epsilon)r$ and r a "boundary" pair.

[2] All these data sets are available at http://fimi.cs.helsinki.fi/data/.



(a) RETAIL



(b) PUMSB

Figure 1. Comparisons of Running Times on Measure ϕ.

number of LSH functions increases dramatically with the increase of t and l when the minimum threshold is decreased. However, the number of LSH functions in ours can be fixed with the decrease of the minimum threshold and is always much smaller than that of *B-LSH*. In our experiments, we obtain an exact result by using 10-20 LSH functions when the minimum threshold is not fairly low. When the minimum threshold is relatively low, the number of LSH functions is still about $50 - 80$.

The Scalability of Algorithm. We used synthesized data sets generated from LA1 to examine the scalability of our algorithm. The miss ratio is set to 0.01, and the minimum threshold is set to 0.3. The experimental results are reported on set resemblance $\mathcal{R}$ and data set *retail*. Figure 2 plots the running time against the number of items in the data set. Observe that both *Ce-LSH* and *B-LSH* show linear scalability with the number of items from about 15K to 30K. However, a close study shows that *Ce-LSH* is more scalable than *B-LSH*. It is noteworthy that we did not use extremely large-

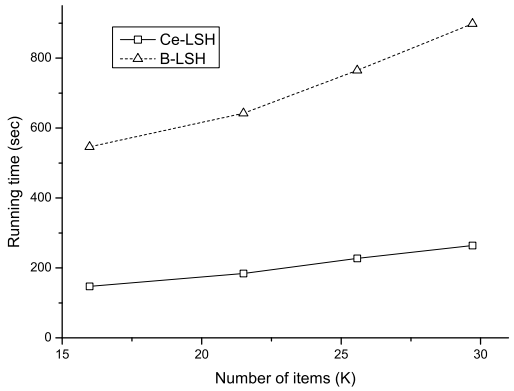


Figure 2. Scalability of the algorithms

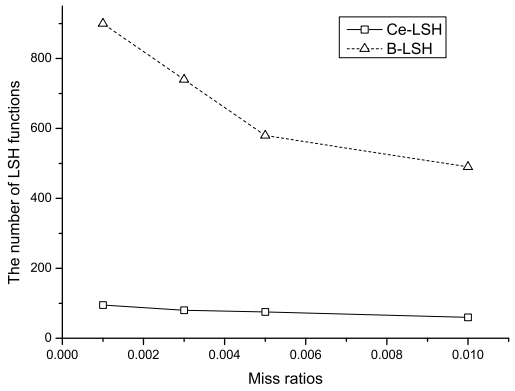
scale data in our tests; thus the I/O complexity is left out of consideration in our evaluations. The filtering stage of LSH-type schemes is the most main memory intensive part of the algorithm. It requires a linear scan over all the pairs represented by a group of hash values, assuming that all hash values for all columns fit in the main memory. If this is not the case, more passes over the pair file might be needed. One can imagine that for a fairly large scale data set, *Ce-LSH* requires much less disk access than *B-LSH* and is therefore more scalable than *B-LSH* for extremely large-scale data sets.

Execution Cost versus Error Tradeoff. Since our method is also based on LSH and produces an approximation result, we compared the execution cost versus error tradeoff with that of *B-LSH*. In order to perform an accurate evaluation, we report in Figure 3 both the tradeoff of the number of LSH functions versus the miss ratio (in Figure 3(a)) and the tradeoff the running time versus the miss ratio (in Figure 3(b)). We set the minimum threshold to 0.3 in the tests due to the observation that the miss ratio sensitively depends on execution cost when the minimum threshold is relatively low.

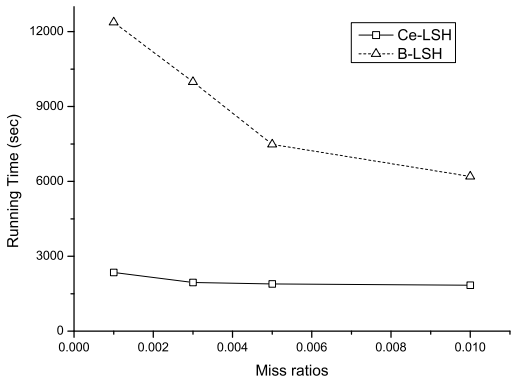
Figure 3 shows that the number of LSH functions in *Ce-LSH* exhibits little dependence on the miss ratio (the *Ce-LSH* curves in both figues are nearly flat) and so does the running time (shown in Figure 3(b)). The reason of the latter is that the only additional costs necessary for reducing miss ratio are the more iterations, which *do not exert much influence* over the performance of the algorithm. In summary, *Ce-LSH* scales considerably better than *B-LSH* for a fairly small miss ratio (say, 0.001).

VII. Conclusion

In this paper, we have proposed a novel LSH-based scheme, namely a Cost-effective LSH filter (Ce-LSH filter for short) for efficiently mining related pairs. Theoretical



(a) The number of LSH functions vs the miss ratio



(b) The running time vs the miss ratio

Figure 3. Tradeoff between Cost and Miss Ratio

analysis has shown that the *Ce-LSH* filter can guarantee good filtering accuracy at a low time/space cost. Substantial experiments conducted on both synthetic and real data sets have shown that the *Ce-LSH* filter can greatly outperform existing methods in terms of execution time, especially when the minimum threshold is low. An additional advantage of the *Ce-LSH* filter is that its scalability for a low minimum threshold and a low miss ratio. All these properties make our method a suitable candidate for efficient extraction of pairs in both large-scale and high dimensional data sets.

Besides the pairwise mining, the *Ce-LSH* filter has the potential for more complex settings, such as clustering. Furthermore, due to the cost-effectiveness of our scheme, it will be interesting to apply this technique to the settings in which space requirements and disk accesses are strictly limited, such as the stream computation. Our future work will further investigate these.

ACKNOWLEDGMENT

This work is sponsored in part by *National Grand Fundamental Research 973 Program of China under Grant No. 2005CB321905*, *Shanghai Leading Academic Discipline Project under Grant No. B114*, and *Australian Research Council Discovery Grants (DP0988016, DP0773412, DP0667060)*.

REFERENCES

[1] A. Z. Broder, *On the resemblance and containment of documents*. In Compession and Complexity of Sequences, 1998, pp. 21-29.

[2] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher, *Min-wise independent permutations*. Journal of Computer and System Sciences, (3), 2002, pp. 630-659.

[3] M. Charikar, *Similarity estimation techniques from rounding algorithm*. In Proceedings of the thiry-fourth annual ACM Symposium on Theory of Computing, 2002, pp. 380-388.

[4] E. Cohen and M. Datar, etc., *Finding interesting associations without support pruning*. In 16th International Conference on Data Engineering, 2000, pp. 489-499.

[5] M. Datar, N. Immorlica, P. Indyk, and V. S, Mirrokni. *Locality-sensitive hashing scheme based on p-stable distributions*. In Proceedings of the twentieth annual Symposium on Computational Geometry, 2004, pp. 253-262.

[6] A. Gionis, P. Indyk, and R. Motwani, *Similarity search in high dimensions via hashing*. In Proceedings of 25th Int'l Conference on Very Large Data Bases, 1999, pp. 518-529.

[7] P. Indyk and R. Motwani, *Approximate nearest neighbor: towards removing the curse of dimensionality*. In Proceedings of the 30th ACM Symposium on Theory of Computing, 1998, pp. 604-613.

[8] G. G. Lorentz, *Bernstein Polynomials*, Vol. 8 of Math. Expos., Univ. of Toronto Press, Toronto, 1953.

[9] R. Motwani and P. Raghavan, *Randomized Algorithms*. Cambridge University Press, New York, 1995.

[10] M. Narayanan and R. M. Karp, *Gapped local similarity search with provable guarantees*. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, Vol. 3240/2004, pp.74-86.

[11] H. Xiong, S. Shekhar, P. Tan, and V. Kumar, *TAPER: a two-step approach for all-strong-pairs correlation query in large databases*. IEEE Transactions on Knowledge and Data Engineering. Volume 18, Issue 4, 2006, pp. 493-508.

[12] J. Zhang and J. Feigenbaum, *Finding highly correlated pairs efficiently with powerful pruning*. In Conference on Information and Knowledge Management, 2006, pp. 152-161.