Hashing of Databases with the Use of Metric Properties of the Hamming Space

VLADIMIR B. BALAKIRSKY

Institute for Experimental Mathematics (IEM), University of Essen, Ellernstrasse 29, 45326 Essen, Germany
Email: v.b.balakirsky@tue.nl

Hashing of databases is considered from the point of view of information and coding theory. The records of a database are represented as binary vectors of the same length stored in the external memory of a computer. The task is formulated as follows: given a pattern and a fixed size of working memory, form the set of addresses of records that can disagree with the pattern in the number of positions smaller than the given threshold value. We use metric properties of the Hamming space and show that computational efforts needed to search for a pattern in databases can be essentially decreased by using the triangle inequality for the Hamming distances between binary vectors. Furthermore, an introduction of the Lee distance in the space containing the Hamming distances leads to a new metric space where the triangle inequality is effectively used.

Received 20 December 2002; revised 22 July 2004

1. INTRODUCTION

An important problem in computers is the following. Given a collection of items, we wish to store these items and upon demand retrieve those items whose key values match given key values. A particular approach to the storage and retrieval problem where the key value of an item determines the address for the storage of that item is known as hashing. Since the mapping between keys and addresses is not one-to-one, different keys can have the same address: these events are known as collisions [1, 2].

Similar ideas form the basis for so-called external hashing [3]. The cost of storage generally increases as access time decreases. Main memories usually have relatively short access time. Since main memory size is limited by increasing cost, databases are usually stored in a secondary memory with rather slow access [1]. One can reduce the number of required accesses if the values of a hash-function, applied to the key of each item of the database, are stored in the main memory. To find the item for a certain key, one computes the value of the hash-function for that key and accesses only items whose keys have this hashed value. We consider the external hashing where, for simplicity, the item and its key value coincide. The items in the collection are binary vectors of the same length, which are stored in an external memory. There is also a working memory for storing pre-computed values of the hash-function. For an input pattern, which is a binary vector of the same length as the items, we wish to find all items whose Hamming distance from the input is less than a fixed threshold value $T$.

We investigate the possibility of using the metric properties of the Hamming space for searching procedures. This direction relates to similar approaches to nearest neighbor search for different applications, like data compression and cryptography [4, 5]. Our hashing scheme exploits the Hamming metric and generalizes methods presented in [6] as an extension of ‘searching in random digital fields’ developed by Koshelev [7]. Implementation of the algorithm in practice also requires a certain organization of a computer memory oriented to specific requests for insertions and deletions keys in the hashtable. We expect that this implementation can be realized in a similar way as it is described in [8] where the well-known hashing algorithm by Fredman, Komlós and Szemerédi [9] was used. A theoretical analysis of the performance of hashing in our correspondence is carried out for the probabilistic ensemble of databases introduced in such a way that different records with high probability are close to each other in the Hamming sense. Namely, we assume that $L - N$ components of all records are equal to each other and that $N$ components are independently chosen at random, where $L$ is the length of the record and $N$ can be much smaller than $L$. Moreover, locations of randomly chosen components are unknown in advance, which makes any hashing algorithm difficult. An example where this model is of interest in biology and medicine concerns processing of data generated according to the so-called Gilbert–Elliott model [10] (in particular, such a model can be suggested as an extension of the analysis of biosequences presented in [11]). Suppose that there is a person. At each time instant he receives some input signal and generates a bit. If the person does not accept the
signal, then he generates a 0 with high probability. Otherwise, the person generates 0 or 1 with probabilities $\frac{1}{2}$. A typical binary sequence generated by the person contains long strings of 0s, which are sometimes interrupted with intervals where 0s and 1s are equally likely. A similar sequence has to be found in the database to classify the behavior of the person. The scheme described above is used for testing impulsive noise channels and has many applications in communication systems (e.g. [12]).

There are many papers on problems related to hashing in the computer science direction, and the reference list [13, 14, 15, 16] is far from being complete. Let us mention some points, developed in the present correspondence, that differ, in some sense, from a conventional representation of hashing in these papers. We consider hashing as a problem of coding theory where a pattern is generated as a result of transmission of some record of a database over a noisy channel. Moreover, records of our database are chosen according to a certain joint probability distribution. Thus, our analysis of the performance of the algorithm does not use independence of these vectors, which is essential for the mathematical analysis presented in [13, 14, 15, 16]. Our algorithm uses a specific deterministic hash-function, while the authors of [13, 14, 15] allow randomization over certain classes of hash-functions, and their statements should be understood as statements of an existence type (provided that hashing is defined as a deterministic algorithm). Notice also that ‘hashing’, from our point of view, is an arbitrary mapping understood as statements of an existence type (provided that the authors of [13, 14, 15] allow randomization over certain classes of hash-functions, and their statements should be understood as statements of an existence type). The authors of [13, 14, 15, 16] is far from being complete. Let us mention these purposes can differ from computing classes of hash-functions, and their statements should be understood as statements of an existence type (provided that hashing is defined as a deterministic algorithm). Notice also that ‘hashing’, from our point of view, is an arbitrary mapping understood as statements of an existence type (provided that the authors of [13, 14, 15] allow randomization over certain classes of hash-functions, and their statements should be understood as statements of an existence type). The authors of [13, 14, 15, 16] is far from being complete. Let us mention these purposes can differ from computing classes of hash-functions, and their statements should be understood as statements of an existence type (provided that hashing is defined as a deterministic algorithm). Notice also that ‘hashing’, from our point of view, is an arbitrary mapping understood as statements of an existence type (provided that the authors of [13, 14, 15] allow randomization over certain classes of hash-functions, and their statements should be understood as statements of an existence type).

The paper is organized as follows. After some notation and we write \[ a^{[q]} = a \mod q. \]

\[ |a - b| \geq d_{\text{Lee}}(a, b) \quad (2.4) \]

\[ d_{\text{Lee}}(a, b) = \min_{s \in \{0, \pm 1, \pm 2, \ldots\}} |a - (b[q] + sq)|. \quad (2.5) \]

The computation of the Lee distance using (2.5) is illustrated in Figure 1.

The binomial distribution will be denoted by \[ \Omega = \left( \Omega_d(n) = \binom{n}{d} 2^{-n}, \quad d = 0, \ldots, n \right). \quad (2.6) \]

The indicator function will be denoted by \( \chi \), i.e. \( \chi(S) = 1 \) if the statement \( S \) is true and \( \chi(S) = 0 \) otherwise. We will also assume that all logarithms are taken to the base 2.

3. MATHEMATICAL MODEL FOR HASHING

Let X be a binary \( M \times L \) matrix that will be referred to as the database (DB). The rows of \( X \), denoted by \( x_1, \ldots, x_M \in [0, 1]^L \), will be referred to as records. We write

\[ X = \left( x_1, \ldots, x_M \right)^T \triangleq \begin{bmatrix} x_1 \\ \vdots \\ x_M \end{bmatrix}, \quad (3.1) \]

where \( T \) denotes transposition, and consider the following problem:

\[ \mathcal{J}_T(x, X) \triangleq \left\{ j \in \{1, \ldots, M\}; d_H(x, x_j) \leq T \right\}. \quad (3.2) \]

Since we require that the algorithm for constructing the set \( \mathcal{J}_T(x, X) \) is chosen before the DB and the pattern become known, there is only one algorithm which requires

![Figure 1. Computation of the Lee distance between integers a and b for a given q.](image-url)
computations of the Hamming distances between all records and the pattern. If the DB is stored in an external memory and each access to this memory allows us to read only one record to working memory, then the number of accesses is equal to $M$ (the pattern is assumed to be available in the working memory). However this conclusion is changed if preprocessing of the DB is possible and $M\ell$ bits of working memory are available to store the results, where $\ell < L$ is given. In this case, we can assign two functions
\[ f : x_j \in \{0, 1\}^L \longrightarrow \{0, 1\}^\ell \]
\[ \varphi_T : (x, f(x_j)) \in \{0, 1\}^L \times \{0, 1\}^\ell \longrightarrow \{0, 1\} \]  \hspace{1cm} (3.3)
in such a way that
\[ d_H(x, x_j) \leq T \implies \varphi_T(x, f(x_j)) = 1, \]
for all $x, x_j \in \{0, 1\}^L$.  \hspace{1cm} (3.4)

In this paper, hashing will be understood as assignment of the particular pair of functions $(f, \varphi_T)$ satisfying (3.3), (3.4). The function $f$ will be referred to as hash-function and the function $\varphi_T$ will be referred to as rejection function.

By (3.2) and (3.4),
\[ J_T(x, X) \subseteq J(x, X|f, \varphi_T), \]  \hspace{1cm} (3.5)
where
\[ J(x, X|f, \varphi_T) \overset{\Delta}{=} \left\{ j \in \{1, \ldots, M\} : \varphi_T(x, f(x_j)) = 1 \right\}, \]  \hspace{1cm} (3.6)
so the searching problem can be solved in two steps:

(i) compute $\varphi_T(x, f(x_1)), \ldots, \varphi_T(x, f(x_M))$ using the data of working memory and construct the set $J(x, X|f, \varphi_T)$;

(ii) sequentially transmit records $x_j, j \in J(x, X|f, \varphi_T)$, to working memory and construct $J_T(x, X)$ as the set
\[ \left\{ j \in J(x, X|f, \varphi_T) : d_H(x, x_j) \leq T \right\}. \]

The procedure above can be also implemented as follows:

(1) Compute the values of the hash-function $f(x_1), \ldots, f(x_M)$ and store them in working memory.

(2) Enter a pattern $x$ for searching in the database.

(3) Set $j = 1$.

(4) Fetch $f(x_j)$ from working memory and compute the value of the rejection function, $\varphi_T(x, f(x_j))$.

(5) If $\varphi_T(x, f(x_j)) = 1$, then fetch $x_j$ from the database and compute the value of the Hamming distance, $d_H(x, x_j)$. If this value does not exceed $T$, then output $j$.

(6) Increase $j$ by $1$. If $j \leq M$, then go to 4.

(7) Go to 2.

Remarks. (i) Only the rejection function depends on $T$ in (3.3) since we assume that the value of $T$ is specified after the hash-function $f$ is fixed.

(ii) The ‘minimal’ rejection function $\varphi_T^{(\text{opt})}$ satisfying (3.4) is defined by
\[ \varphi_T^{(\text{opt})}(x, f(x_j)) = 1 \iff \exists x' \in \{0, 1\}^L : \begin{cases} f(x') = f(x_j), \\ d_H(x, x') \leq T. \end{cases} \]  \hspace{1cm} (3.7)

Thus, the rejection function $\varphi_T^{(\text{opt})}$ is completely determined by the hash-function $f$. Nevertheless, here we will introduce a rejection function $\varphi_T$ that can be different from the optimum one, since its simple implementation is required to assume that the effectiveness of the hashing is characterized by the number of transmissions of records from external memory to working memory (step 5 of the procedure above). This number is equal to the cardinality of the set $J(x, X|f, \varphi_T)$.

The analysis of any fixed hashing algorithm in a general form can be developed if we introduce probability distribution on a set of matrices $X$ and some conditional probability distribution on a set of patterns for any given $X$. Suppose that
\[ x_0 = (x_0^{(1)}, \ldots, x_0^{(L)}) \in \{0, 1\}^L \]
and $N \in \{0, \ldots, L\}$ are given. Furthermore, let $I$ be a given $N$-element subset of the set $\{1, \ldots, L\}$.
\[ I \subseteq \{1, \ldots, L\}, \quad |I| = N. \]  \hspace{1cm} (3.8)

Let us introduce the probability of the event (3.1) as the conditional probability
\[ \Pr \left\{ X = (x_1, \ldots, x_M)^T \mid x_0, I \right\} \overset{\Delta}{=} \prod_{j=1}^M P(x_j \mid x_0, I), \]  \hspace{1cm} (3.9)
where
\[ P(x_j \mid x_0, I) \overset{\Delta}{=} \prod_{i=1}^L p(x_j^{(i)} \mid x_0^{(i)}, i, I) \]  \hspace{1cm} (3.10)
and
\[ p(x_j^{(i)} \mid x_0^{(i)}, i, I) \overset{\Delta}{=} \begin{cases} 1, & \text{if } x_j^{(i)} = x_0^{(i)} \text{ and } i \notin I, \\ 0, & \text{if } x_j^{(i)} \neq x_0^{(i)} \text{ and } i \notin I, \\ 1/2, & \text{if } i \in I. \end{cases} \]  \hspace{1cm} (3.11)

Let
\[ G_{x_0, I}(x \mid f, \varphi_T) \overset{\Delta}{=} \sum_{x_1, \ldots, x_M} \Pr \left\{ X = (x_1, \ldots, x_M)^T \mid x_0, I \right\} \frac{J(x, X|f, \varphi_T)}{M} \]  \hspace{1cm} (3.12)
denote the expectation of the relative number of records included in the set $J(x, X|f, \varphi_T)$. Notice that


which is attained when none of the first $\ell$ positions belong to the set $\mathcal{I}$. It is easy to see that

\[
\mathcal{G}(f', \varphi_T') = \sum_{n=0}^{N} \left[ \sum_{t=0}^{T} \binom{n}{t} 2^{-n} \right] \left( \frac{L - \ell}{N - n} \right) \left( \frac{L}{N} \right)^{-1}.
\]

**Numerical example.** Suppose that the DB consists of $M$ records of length $L = 256$. Each record should be represented as a vector of length $\ell = 12$. For a given pattern of length 256 and a codeword of length 12, we want to know whether the Hamming distance between the pattern and the record does exceed $T$ or not. Let the DB be determined by an ‘opponent’ after the hash-function and the rejection function are fixed. The opponent proceeds as follows. He selects $N = 64$ positions and puts the same bit $x_0^{(i)}$ to the $i$-th position of all records if $i$ does not belong to the set of selected positions. Otherwise, if $i$ belongs to the set of selected positions, then the $i$-th bits of each record are generated by independent flippings of a fair coin, and the opponent does not know the outcomes of this random experiment. The opponent also specifies all $L$ bits of the pattern.

Some values of $G(f, \varphi_T)$ and $\mathcal{G}(f, \varphi_T)$ for $(f, \varphi_T) = (f, \varphi_T'), (f(q), \varphi(q), \varphi(q), (q, K) = (8, 4)$ and $T = 0, \ldots, 12$ are given in Table 1, where the functions $f_{4}^{(8)}$ and $\varphi_{T,4}^{(8)}$ are as follows.\footnote{We show the first 4 digits after the decimal point in Tables 1, 2 and do not include rounding to distinguish between 0.99999 and 1.} We represent each record $x_i$ of length 256 as a concatenation of $K = 4$ blocks of length 64. For each block, we compute its Hamming weight and reduce this weight modulo $q = 8$. As a result we obtain a vector $(w^{(8)}_1, \ldots, w^{(8)}_{256})$ with components belonging to the set $\{0, \ldots, 7\}$ and use $\ell = 12$ bits to store it in the memory. The pattern is also represented as a concatenation of 4 blocks of length 64 and their Hamming weights form the vector $(w_1, \ldots, w_4)$. The rejection function is defined as

\[
\varphi_{T,4}^{(8)}(x, f_{4}^{(8)}(x)) = \chi \left\{ \sum_{k=1}^{4} q_{T,\varphi_{K}}^{(8)}(w_k, w^{(8)}_{jk}) \leq T \right\},
\]

where (2.5) is used for the computation of the Lee distance. In the general case the proposed pair $(f^{(q)}, \varphi_{T,K}^{(q)})$ is introduced in the following sections.

The situation described above can be considered a very difficult one, because the compressed factor 256/12 is very big and the designer has no information about the content of the DB. Nevertheless, if $T = 0$ and hashing is specified by the pair of functions $(f_K^{(q)} , \varphi_{T,K}^{(q)})$, then we only leave $\sim 0.14\%$ of the total number of records in the list of records that can coincide with the pattern in the average case (the function $\mathcal{G}$) and leave about 12.65% of the total number of records in the worst case (the function $G$). These values increase with $T$, but much more slowly than the similar values for the pair $(f', \varphi_T')$. Notice also that the value of the function $\mathcal{G}$ is less than 1 even for $T = \ell = 12$.  

The function $G(f, \varphi_T)$ corresponds to the ‘worst’ assignment for $(x, x_0, \mathcal{I})$, while the function $\mathcal{G}(f, \varphi_T)$ corresponds to the ‘worst’ assignment for $(x, x_0)$ when $\mathcal{I}$ is uniformly chosen from the collection of sets satisfying (3.8). The values of $G(f, \varphi_T)$ and $\mathcal{G}(f, \varphi_T)$ can be accepted as criteria of the quality of hashing, and we represent our results as possible solutions to the optimization problems

\[
G(f, \varphi_T) \rightarrow \min, \quad \mathcal{G}(f, \varphi_T) \rightarrow \min,
\]

where the minimization domains consist of all pairs $(f, \varphi_T)$ satisfying (3.3), (3.4).

**Comments.** The probabilistic ensemble introduced by (3.9)–(3.11) is rather general. If $N = L$, then $\mathcal{I} = \{1, \ldots, L\}$. In this case, the product on the right-hand side of (3.9) is equal to $2^{-ML}$ for all $x_1, \ldots, x_M$, which means that each bit of the matrix $X$ is an i.i.d. random variable chosen from $\{0, 1\}$ with probability $\frac{1}{2}$. If $N = 0$, then $\mathcal{I} = \emptyset$. In this case, the product on the right-hand side of (3.9) is 1 if $x_1 = \cdots = x_M = x_0$ and 0 otherwise. If $N \in \{1, \ldots, L - 1\}$, then there is a freedom in the assignment of the set $\mathcal{I}$. In this case, the vectors $x_1, \ldots, x_M$ are chosen as the 1st, $\ldots$, as the $M$-th record of the DB with probability $2^{-MN}$ if and only if their bits at positions $i \notin \mathcal{I}$ coincide with corresponding bits of the vector $x_0$. Hence, we introduce a probability distribution with memory and consider it as a model for ‘dense’ DBs. Notice that we do not introduce a conditional probability distribution on patterns for a given $X$ and analyze the worst case instead.

Since we do not know the content of the DB while constructing the hashing algorithm (notice that both the hash-function and the rejection function do not depend on the pair $(x_0, \mathcal{I})$), a ‘direct’ hash-function can be assigned as

\[
f'(x_j) \triangleq (x_j^{(1)}, \ldots, x_j^{(L)}), \quad \text{for all } x_j \in [0, 1]^L,
\]

i.e. the first $\ell$ bits of the record $x_j$ of length $L$ are stored in working memory. In this case, the minimal rejection function satisfying (3.7) is defined as

\[
\psi_T(x, f'(x_j)) \triangleq \chi \left\{ d_4(f'(x), f'(x_j)) \leq T \right\}, \quad \text{for all } x \in [0, 1]^L,
\]

because the last $L - \ell$ bits of the record $x_j$ can coincide with the last $L - \ell$ bits of the pattern $x$. Then the inequality $N \leq L - \ell$ implies

\[
G(f', \psi_T) = 1,
\]
4. ESTIMATIONS OF THE HAMMING DISTANCE BETWEEN BINARY VECTORS USING INDIRECT OBSERVATIONS

4.1. Auxiliary results

The following properties of the binomial distribution $\Omega_n$ introduced in (2.6) will be referred to as properties (a)–(c).

(a) $\Omega_n(d)$ is a monotonically increasing function of $d \in [0, n/2]$;
(b) $\Omega_n(d)$ is a monotonically decreasing function of $d \in [n/2, n]$;
(c) $\Omega_n([n/2] - \Delta) = \Omega_n([n/2] + \Delta)$ for all $\Delta = 0, 1, \ldots$.

The distribution $\Omega_n$ can be estimated using Robbins’ improvement of the well-known Stirling’s approximation for the factorial ([17], Chapter 2):

$$\exp\left\{-\frac{1}{12(n+1)}\right\} \leq \frac{1}{\sqrt{2\pi n}} n^{-n} e^{n!} \leq \exp\left\{-\frac{1}{12n}\right\}.$$

As a result, we obtain

$$\Omega_n(\lfloor mp \rfloor) = \frac{1}{\sqrt{2\pi np(1-p)}} 2^{-(n-1-h(p))(1-c_n(p))}, \quad (4.1)$$

where

$$0 \leq c_n(p) \leq 1 - \exp\left\{-\frac{1}{12(n+1)} \frac{1 - p(1-p)}{p(1-p)} \right\} \to 0,$$

as $n \to \infty$,

and

$$h(p) \triangleq p \log \frac{1}{p} + (1-p) \log \frac{1}{1-p}$$

is the binary entropy function. In particular,

$$\Omega_n([n/2]) = (1 - c_n(1/2)) \sqrt{\frac{n}{\pi n}}, \quad (4.2)$$

where we omit rounding because the accuracy of the approximation is less than the correction, which should be introduced when $n$ is odd. The entropy of the distribution $\Omega_n$ is defined as

$$H(\Omega_n) \triangleq \sum_{d=0}^n \Omega_n(d) \log \frac{1}{\Omega_n(d)},$$

and (4.1) implies

$$\lim_{n \to \infty} \frac{H(\Omega_n)}{\log n} = \frac{1}{2}. \quad (4.3)$$

Equality (4.3) can be interpreted in the following way. Suppose that we compute the Hamming distances between a fixed binary vector of length $n$ and $M$ binary vectors generated at random according to a uniform distribution. Then we obtain a sequence of $M$ integers whose components are generated according to a binomial distribution. The total number of bits needed to store this sequence is equal to $M\lceil\log(n+1)\rceil$. Owing to (4.3) and Shannon’s data compression theorem [18], roughly $M \log \sqrt{n}$ bits are ‘effectively’ used in this case. Nevertheless, if any sequence that can be obtained in our experiment has to be stored, we cannot apply the data compression technique. The reason for the considerations below is the intention of building another sequence of length $M$ whose components have some distribution, which is closer to the uniform distribution than $\Omega_n$, and simultaneously conserving the possibility of using the triangle inequality axiom in the hashing algorithm.

Given an integer $q \geq 2$, let us denote

$$\mathcal{Q} \triangleq \{0, \ldots, q - 1\}$$

and, for all $\delta \in \mathcal{Q}$, introduce the probability

$$\Theta_n^{(q)}(\delta) \triangleq \sum_{d=0}^n \Omega_n(d) \chi\{d \equiv 1 \mod q = \delta\}. \quad (4.4)$$
are decreased in such a way that we get a balance. In our
of the ordinates of the pulses.

\[ \sum_{\delta \in Q} \Theta_n(q)(\delta) = 1 \quad (4.5) \]

and

\[ \Theta_n(q) = \left( \Theta_n(q)(\delta), \delta \in Q \right) \]

is the probability distribution on the set \( Q \).

The result presented below in Lemma 1 has the following
reasoning. One can easily see that if \( n \) is odd, then

\[ \sum_{d=0}^{n} \binom{n}{d} \chi \{ d \ is \ even \} = \sum_{d=0}^{n} \binom{n}{d} \chi \{ d \ is \ odd \}. \]

In our notations this equality is represented as

\[ \Theta_n^{(2)}(0) = \Theta_n^{(2)}(1) = \frac{1}{2} \]

and illustrated in Figure 2. Namely, we put a lattice
0, 2, 4, . . . on the x-axis and compute the sum of lengths of
pulses measured from the x-axis to the binomial distribution.
If we shift the lattice to the right by 1 and proceed as before,
then the lengths of all pulses at points \( d \leq (n - 1)/2 \) are
increased and the lengths of all pulses at points \( d \geq (n + 1)/2 \)
are decreased in such a way that we get a balance. In our
approach to hashing we would like to have such a balance
for \( q > 2 \), but this is not possible. Nevertheless, if \( q < \sqrt{n} \),
then an approximate balance can be reached, where the
approximation is understood in a way that the difference
between \( \Theta_n^{(q)}(\delta) \) and \( 1/q \) is bounded by a function, which
is proportional to \( 1/\sqrt{n} \), for all \( \delta = 0, \ldots, q - 1 \).

**LEMMA 1.** If \( q \) is even, then

\[ \frac{1}{q} - \Omega_n([n/2]) \leq \Theta_n^{(q)}(\delta) \leq \frac{1}{q} + \Omega_n([n/2]) \quad (4.6) \]

for all \( \delta \in Q \). Hence,

\[ \frac{1}{q} - (1 - c_n(1/2)) \sqrt{\frac{2}{\pi n}} \leq \Theta_n(q)(\delta) \leq \frac{1}{q} + (1 - c_n(1/2)) \sqrt{\frac{2}{\pi n}}, \]

and if \( q = q_n = \alpha \sqrt{n} \), where \( \alpha < \sqrt{2} \), then

\[ \lim_{n \to \infty} \frac{H(\Theta_n(q_n))}{\log q_n} = 1, \quad (4.8) \]

where

\[ H(\Theta_n(q)) = \sum_{\delta \in Q} \Theta_n(q)(\delta) \log \frac{1}{\Theta_n(q)(\delta)} \]

is the entropy of the distribution \( \Theta_n(q) \).

The proof of Lemma 1 is given in the Appendix. We
include numerical illustration of this proof for \( n = 10 \) and
\( q = 4 \). In this case,

\[ 2^{10} \Theta_{10}^{(4)}(0) = \left( \begin{array}{c} 10 \\ 0 \end{array} \right) + \left( \begin{array}{c} 10 \\ 4 \end{array} \right) + \left( \begin{array}{c} 10 \\ 8 \end{array} \right), \]

\[ 2^{10} \Theta_{10}^{(4)}(1) = \left( \begin{array}{c} 10 \\ 1 \end{array} \right) + \left( \begin{array}{c} 10 \\ 5 \end{array} \right) + \left( \begin{array}{c} 10 \\ 9 \end{array} \right), \]

\[ 2^{10} \Theta_{10}^{(4)}(2) = \left( \begin{array}{c} 10 \\ 2 \end{array} \right) + \left( \begin{array}{c} 10 \\ 6 \end{array} \right) + \left( \begin{array}{c} 10 \\ 10 \end{array} \right), \]

\[ 2^{10} \Theta_{10}^{(4)}(3) = \left( \begin{array}{c} 10 \\ 3 \end{array} \right) + \left( \begin{array}{c} 10 \\ 7 \end{array} \right). \]

Let us denote

\[ \sigma \triangleq \left( \begin{array}{c} 10 \\ 1 \end{array} \right) 2^{-10} + \left( \begin{array}{c} 10 \\ 3 \end{array} \right) 2^{-10} \]

and use the following inequalities in the expressions for
\( 2^{10} \Theta_{10}^{(4)}(\delta), \delta = 0, \ldots, 3 \),

\[ 0 < \left( \begin{array}{c} 10 \\ d \end{array} \right) \leq \left( \begin{array}{c} 10 \\ 1 \end{array} \right), \quad \text{if } d \in \{0, 1, 10\}, \]

\[ \left( \begin{array}{c} 10 \\ 1 \end{array} \right) \leq \left( \begin{array}{c} 10 \\ d \end{array} \right) \leq \left( \begin{array}{c} 10 \\ 3 \end{array} \right), \quad \text{if } d \in \{2, 3, 8, 9\}, \]

\[ \left( \begin{array}{c} 10 \\ 3 \end{array} \right) \leq \left( \begin{array}{c} 10 \\ d \end{array} \right) \leq \left( \begin{array}{c} 10 \\ 5 \end{array} \right), \quad \text{if } d \in \{4, 5, 6, 7\}. \]

Then, as it is easy to see,

\[ \sigma \leq \Theta_{10}^{(4)}(\delta) \leq \sigma + \left( \begin{array}{c} 10 \\ 5 \end{array} \right) 2^{-10}, \quad \text{for all } \delta = 0, \ldots, 3. \quad (4.9) \]

By (4.5), \( \Theta_{10}^{(4)}(0) + \cdots + \Theta_{10}^{(4)}(3) = 1 \), and (4.9) implies

\[ 4\sigma \leq 1 \leq 4 \left[ \sigma + \left( \begin{array}{c} 10 \\ 5 \end{array} \right) 2^{-10} \right]. \]

Thus,

\[ \frac{1}{4} - \left( \begin{array}{c} 10 \\ 5 \end{array} \right) 2^{-10} \leq \sigma \leq \frac{1}{4} \]

and using (4.9) again, we obtain inequalities (4.6):

\[ \frac{1}{4} - \left( \begin{array}{c} 10 \\ 5 \end{array} \right) 2^{-10} \leq \Theta_{10}^{(4)}(\delta) \leq \frac{1}{4} + \left( \begin{array}{c} 10 \\ 5 \end{array} \right) 2^{-10}. \]
TABLE 2. The values of $\phi_n^{(8)}(\delta)$ for $\delta = 0, \ldots, 7$ and $n = 16, 32, 64$.

<table>
<thead>
<tr>
<th>$\delta$</th>
<th>$\phi_{16}^{(8)}(\delta)$</th>
<th>$\phi_{32}^{(8)}(\delta)$</th>
<th>$\phi_{64}^{(8)}(\delta)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.1964</td>
<td>0.1448</td>
<td>0.1265</td>
</tr>
<tr>
<td>1</td>
<td>0.1748</td>
<td>0.1390</td>
<td>0.1261</td>
</tr>
<tr>
<td>2</td>
<td>0.1240</td>
<td>0.1250</td>
<td>0.1250</td>
</tr>
<tr>
<td>3</td>
<td>0.0752</td>
<td>0.1109</td>
<td>0.1238</td>
</tr>
<tr>
<td>4</td>
<td>0.0555</td>
<td>0.1051</td>
<td>0.1234</td>
</tr>
<tr>
<td>5</td>
<td>0.0752</td>
<td>0.1109</td>
<td>0.1238</td>
</tr>
<tr>
<td>6</td>
<td>0.1240</td>
<td>0.1250</td>
<td>0.1250</td>
</tr>
<tr>
<td>7</td>
<td>0.1748</td>
<td>0.1390</td>
<td>0.1261</td>
</tr>
</tbody>
</table>

Examples of the distribution $\phi_n^{(q)}$ are given in Table 2, where

$$\hat{\phi}_n^{(q)}(\delta) = \max_{\delta \in \mathbb{Q}} \phi_n^{(q)}(\delta). \quad (4.10)$$

4.2. Rejection rules derived from the triangle inequality for the Hamming distances

By the triangle inequality for the Hamming distances, we have

$$d_H(x, x_j) = |d_H(c, x) - d_H(c, x_j)|, \quad \text{for all } c \in \{0, 1\}^L.$$ 

If $c$ is the all-zero vector, then this inequality is equivalent to the lower bound

$$d_H(x, x_j) \geq |\text{wt}(x) - \text{wt}(x_j)| \quad (4.11)$$

and

$$|\text{wt}(x) - \text{wt}(x_j)| > T \implies d_H(x, x_j) > T. \quad (4.12)$$

Let $f^*(x_j)$ denote the binary vector of length $\lceil \log(L + 1) \rceil$ defined as the standard binary representation of the integer $\text{wt}(x_j) \in \{0, \ldots, L\}$. If $\ell \geq \lceil \log(L + 1) \rceil$ and

$$\phi^*_T(f^*(x_j), f^*(x_j)) \triangleq \chi \{ \text{wt}(x) - \text{wt}(x_j) \mid \leq T \},$$

then statement (4.12) can be used as the rejection rule for the index $j$ to be included in the set $\mathcal{J}(f^*(x), f^*, \phi^*_T)$ since the pair $(f^*, \phi^*_T)$ satisfies (3.3), (3.4).

Notice that if some bits of the record $x_j$ are chosen at random, then $\text{wt}(x_j)$ becomes a random variable having the binomial distribution. To attain a more effective tradeoff between the expected accuracy of a lower bound on $d_H(x, x_j)$ and the required size of memory, let us proceed as follows. Suppose that

$$L = Kr, \quad \ell = K \lceil \log q \rceil. \quad (4.13)$$

where $K, r \geq 1$, and $q \geq 2$ are integers, and represent given vectors $x, x_j \in \{0, 1\}^L$ as concatenations of $K$ blocks of length $r$,

$$x = (x_1, \ldots, x_K), \quad x_j = (x_{j1}, \ldots, x_{jK}). \quad (4.14)$$

where $x_k, x_{jk} \in \{0, 1\}^r$ for all $k = 1, \ldots, K$. Then, similarly to (4.11), we derive the inequalities

$$d_H(x_k, x_{jk}) \geq |\text{wt}(x_k) - \text{wt}(x_{jk})|, \quad k = 1, \ldots, K.$$ 

Hence

$$d_H(x_k, x_{jk}) = \sum_{k=1}^{K} d_H(x_k, x_{jk}) \geq \sum_{k=1}^{K} |\text{wt}(x_k) - \text{wt}(x_{jk})|. \quad (4.15)$$

Let us use (2.2) with $a = \text{wt}(x_{jk})$ and denote

$$\text{wt}^q(x_{jk}) \triangleq \text{wt}(x_{jk}) \mod q.$$ 

Then, by (2.4), (2.5), we have

$$|\text{wt}(x_k) - \text{wt}(x_{jk})| \geq d_L^q(\text{wt}(x_k), \text{wt}^q(x_{jk})) \quad (4.16)$$

and continue (4.15) as follows:

$$d_H(x_k, x_{jk}) \geq \sum_{k=1}^{K} d_L^q(\text{wt}(x_k), \text{wt}^q(x_{jk})). \quad (4.17)$$

Notice that statement (4.17) does not contain statement (4.12) as a special case when $K = 1$ and $\ell = \lceil \log(L + 1) \rceil$, $q = L + 1$, because

$$d_L^{(L+1)}(\text{wt}(x), \text{wt}(x_j)) \leq (L + 1)/2,$$

and (4.16) cannot be replaced with the equality. However, we will assigns $K$ and $q$ differently, and the introduction of the Lee distance makes sense for those assignments.

5. HASHING ALGORITHM

DEFINITION. Suppose that $L$ and $\ell$ are expressed by (4.13). For all

$$x_j = (x_{j1}, \ldots, x_{jK}) \in \{0, 1\}^K,$$

let $f^{(q)}_K(x_j)$ be the concatenation of the standard binary representations of integers

$$\text{wt}^q(x_{j1}), \ldots, \text{wt}^q(x_{jK}) \in \{0, \ldots, q - 1\},$$

THE COMPUTER JOURNAL VOl. 48 NO. 1, 2005
where each of \( K \) entries is represented with \([\log q]\) bits. For all
\[
\mathbf{x} = (x_1, \ldots, x_K) \in \{0, 1\}^K,
\]
let
\[
\psi_{T,K}^{(q)}(\mathbf{x}, f_K^{(q)}(\mathbf{x}_j)) \triangleq \mathcal{X} \left\{ \sum_{k=1}^K d_{\text{Lee}}^{(q)}(wt(x_k), wt^{(q)}(x_{j,k})) \leq T \right\}.
\]

It is easy to see that the pair \( (f_K^{(q)}, \psi_{T,K}^{(q)}) \) satisfies (3.3). By (4.17), condition (3.4) is also satisfied.

The rejection function \( \psi_{T,K}^{(q)} \) is not a minimal rejection function for the hash-function \( f_K^{(q)} \) in the general case. For example, if \( k \) is the all-zero vector and \( x_j \) is chosen in such a way that \( wt^{(q)}(x_{j,k}) = q - 1 \) for all \( k = 1, \ldots, K \), then \( d_{\text{Lee}}(\mathbf{x}, x_j) \geq (q - 1)K \), while the sum of the Lee distances on the left-hand side of (4.17) is equal to \( K \). Thus, \( T \in [K + 1, \ldots, (q - 1)K - 1] \) leads to the conclusion that the value of the minimal rejection function is equal to 0 (see (3.7)), while the value of the rejection function \( \psi_{T,K}^{(q)} \) is equal to 1. Notice that if the value of the parameter \( N \) is taken into account, then one can also receive different values of these decoding functions. Nevertheless, a simple implementation of searching procedure is important, and we stay with the rejection function defined in (5.1).

Our mathematical model for DBs and the hashing algorithm are shown in Figure 3. Any hashing procedure can be represented as partitioning of the space \([0, 1]^L\) in \(2^\ell\) subsets. In our case the DB is a collection of records having a ‘dense’ structure, in the sense that there are many positions where contents of all records coincide. If we partition the space according to values of the projection of the record on a fixed set of positions, as in the ‘direct’ hashing scheme, then one can easily construct an example when almost all records have the same hashed values. We present a different scheme where the same values of the hash-function have all vectors located at surfaces of spheres centered in the all-zero vector. The radii of spheres differ by \( q \). If the values of all these radii are increased or decreased by \( \Delta q \in [1, \ldots, q - 1] \), then the value of the hash-function is changed.

6. PERFORMANCE OF THE HASHING ALGORITHM

6.1. Computation of the values of the functions
\[
G(f_K^{(q)}, \psi_{T,K}^{(q)}) \text{ and } \overline{G}(f_K^{(q)}, \psi_{T,K}^{(q)})
\]
The performance of the hashing algorithm for a fixed pair \((q, K)\) can be analyzed by the generating functions technique [14, Chapter 11], with the exact values of the entries of the probability distributions \( \Theta_n^{(q)}, n = 2, \ldots, N \). Let us construct the probability distribution
\[
\Gamma_n^{(q)} = \left( \Gamma_n^{(q)}(d), d = 0, \ldots, q/2 \right)
\]
in such a way that
\[
\Gamma_n^{(q)}(d) = \begin{cases} 
\Theta_n^{(q)}(\delta_n), & \text{if } d = 0, \\
+\Theta_n^{(q)}(\delta_n \oplus d), & \text{if } d \in [1, \ldots, q/2 - 1], \\
\Theta_n^{(q)}(\delta_n \oplus q/2), & \text{if } d = q/2, \\
0, & \text{if } d \not\in [0, \ldots, q/2] 
\end{cases}
\]
where
\[
\delta_n \triangleq \arg \max_{\delta \in \Theta_n^{(q)}} \Theta_n^{(q)}(\delta)
\]
and
\[
\delta_n \oplus d \triangleq \delta_n - d \mod q, \quad \delta_n \oplus d \triangleq \delta_n + d \mod q.
\]
It is easy to see that if a vector \( \hat{x} \in \{0, 1\}^n \) is fixed in such a way that \( wt^{(q)}(\hat{x}) = \delta \), then \( \Gamma_n^{(q)}(d) \) is the probability that the Lee distance between \( wt(\hat{x}) \) and \( wt^{(q)}(\hat{x}) \) is equal to \( d \), where \( \hat{x}_j \in \{0, 1\}^n \) is a vector whose components are i.i.d. random variable chosen from \([0, 1]\) according to a uniform distribution.

Let us denote
\[
\Phi_n^{(q,1)}(d) \triangleq \left\{ \begin{array}{ll}
\Gamma_n^{(q)}(d), & \text{if } d \in [0, \ldots, q/2], \\
0, & \text{otherwise}
\end{array} \right.
\]
and, for all \( k = 2, \ldots, K \), recurrently compute
\[
\Phi_n^{(q,k)}(D) \triangleq \sum_{n_k=0}^r \psi_n^{(k)} + \sum_{d=0}^{D-d} \Phi_n^{(q,k-1)}(D-d) \Gamma_n^{(q)}(d),
\]
\[
D = 0, \ldots, T.
\]
where

\[ \psi_{n-n_k,n_k}^{(k)} \triangleq \binom{(k-1)r}{n-n_k} \binom{r}{n_k} \binom{kr}{n}. \]

Then \( \Phi_{(q,K)}^{(q)}(D) \) is the upper bound on the probability that the Lee distance between the Hamming weight of \( N \) randomly chosen bits of the record \( x_j \) and the Hamming weight of corresponding \( N \) bits of the pattern \( x \) is equal to \( D \), i.e.

\[ \overline{G}(f_k^{(q)} \cdot \varphi_{T,K}^{(q)}) \leq \sum_{D=0}^{T} \Phi_{N}^{(q,K)}(D). \]

Some technical difficulties in the calculations above appear because we have to analyze all possible distributions of the numbers of bits of the records and the pattern that are chosen at random in each of \( K \) blocks. These difficulties are essentially less if we assume the uniform distribution and restrict ourselves to the case that there are exactly \( N/K \) bits in each block having this property. As a result, we arrive at the following approximation of the function \( \overline{G}(f_k^{(q)} \cdot \varphi_{T,K}^{(q)}) \):

\[ \overline{G}(f_k^{(q)} \cdot \varphi_{T,K}^{(q)}) \approx \sum_{D=0}^{T} \Phi_{N}^{(q,K)}(D), \]

where

\[ \Phi_{N}^{(q,K)}(D) \triangleq \sum_{d=0}^{D} \Phi_{(q,k-1)}^{(q,K)}(D-d) \Gamma_{N}^{(q,K)}(d). \quad \text{(6.2)} \]

Recurrence formulae (6.1), (6.2) were used to compute data in Table 1.

The analysis of the function \( G(f_k^{(q)} \cdot \varphi_{T,K}^{(q)}) \) for the values of \( T, K, q, L \) and \( \ell \) under consideration is simpler because the worst case for the pattern is attained when all randomly chosen bits are located in the same block. In this case,

\[ G(f_k^{(q)} \cdot \varphi_{T,K}^{(q)}) = \sum_{D=0}^{T} \Phi_{N}^{(q,K)}(D). \]

### 6.2. Evaluation of the asymptotic performance

In this section we want to convince a reader that if the parameters \( L \) and \( N \) are big enough and

\[ N \sim L, \quad \ell \sim \log L, \quad \text{(6.3)} \]

then the proposed hashing algorithm can be very effective.

**Theorem 1.** If

\[ \ell \leq \log \sqrt{N} + \log \sqrt{\pi/2} \quad \text{(6.4)} \]

and

\[ q = 2^\ell, \quad \text{(6.5)} \]

then

\[ \overline{G}(f_1^{(q)} \cdot \varphi_{T,1}^{(q)}) \leq \overline{G}(f_1^{(q)} \cdot \varphi_{T,1}^{(q)}) \leq (4T + 2)2^{-\ell}. \quad \text{(6.6)} \]

The proof is given in the Appendix.

The proof for the case \( K = 1 \) proceeds as follows. We construct an upper bound on the probability of choosing the record \( x_j \) with \( d_{\text{lee}}^{(q)}(\text{wt}(x), \text{wt}^{(q)}(x_j)) = t \) for a given pattern \( x \), set \( I \) and \( t \) in \([0, \ldots, q-1]\). It turns out that this bound is asymptotically tight in a sense that the result is proportional to \( 2^{t-\ell} \) for all \( x \), \( I \) and \( t \). Our upper bound on \( \overline{G}(f_k^{(q)} \cdot \varphi_{T,1}^{(q)}) \) and \( G(f_1^{(q)} \cdot \varphi_{T,1}^{(q)}) \) is obtained by multiplying this result by \( 2T + 1 \) since, for any fixed \( x \),

\[ \left| \{ t : \exists x_j \text{ with } d_{\text{lee}}^{(q)}(\text{wt}(x), \text{wt}^{(q)}(x_j)) = t, \right| \leq 2T + 1. \]

If \( K > 1 \), then we take the \( K \)-th power of the obtained universal bound and optimize the final result over auxiliary parameters.

**Remark.** An anonymous referee proposed an interesting hashing algorithm where the same hashed value \( f_{\text{rel}}(x_j) \triangleq w \in \{0, \ldots, q/q_0 - 1\} \) is assigned to all records \( x_j \) with

\[ \text{wt}^{(q)}(x_j) \in \{q_0w, \ldots, (w + 1)q_0 - 1\}, \]

where the parameters \( q, q_0 \) are chosen in such a way that \( 2^\ell = q/q_0 \). Then there are values of \( T > 0 \) such that, for all \( x \), the cardinality of the set \( t \in \{0, \ldots, q/q_0 - 1\} \) with \( f_{\text{rel}}(x, x_j) = t \) and \( d_{\text{rel}}(x, x_j) \leq T \) for some \( x_j \) is independent of \( T \). Therefore there are approaches which allow one to replace linear dependence of our bounds on \( T \) by a constant. Nevertheless, the dependence on \( \ell \) as a function \( 2^{t+\ell} \) remains.

Notice that inequality (6.4) also implies

\[ \ell < \log L, \]

since \( N \leq L \). If (6.4) is not satisfied, but the proportions (6.3) are valid, then the possibility of partitioning the records in \( K \) blocks of length \( L/K \) allows us to adopt given values of the parameters to the data required for a good performance of the hashing algorithm.

We will assume that the record and the pattern are represented by (4.14) and denote

\[ T_k \triangleq (k-1)r + 1, \ldots, kr \cap I, \quad k = 1, \ldots, K, \quad \text{(6.7)} \]

i.e. we can represent the set \( I \) as the union of pairwise disjoint subsets \( I_1, \ldots, I_K \). Denote also

\[ S_{T,K}^{(q)} \triangleq \{ t_K = (t_1, \ldots, t_K) : t_1, \ldots, t_K \in [0, \ldots, q-1], \]

\[ t_1 + \cdots + t_K = T \}. \quad \text{(6.8)} \]

Let us restrict our attention to the following approximations of the functions \( G(f_k^{(q)} \cdot \varphi_{T,K}^{(q)}) \) and \( \overline{G}(f_k^{(q)} \cdot \varphi_{T,K}^{(q)}) \) defined in (3.13), (3.14) when \( f = f_k^{(q)} \) and \( \varphi = \varphi_{T,K}^{(q)} \).

(1) Suppose that \( N \) is divisible by \( r \) and that there are \( N/r = KN/L \) blocks where all bits of \( M \) records are chosen at random. All bits of the records whose indices
do not belong to the set of selected blocks coincide with corresponding bits of the vector $x_0$. Thus, there is a set $[k_1, \ldots, k_{N/r}] \subseteq [1, \ldots, K]$ of cardinality $N/r$ such that

$$|\mathcal{I}_k| = \begin{cases} r, & \text{if } k \in [k_1, \ldots, k_{N/r}], \\ 0, & \text{if } k \not\in [k_1, \ldots, k_{N/r}] \end{cases}$$

(6.9)

for all $k = 1, \ldots, K$. If the maximum on $\mathcal{I}$ in (3.13) is replaced with the maximum on $k_1, \ldots, k_{N/r}$, then we write $\hat{G}(f^{(q)}_k, \psi^{(q)}_{T,K})$ instead of $G(f^{(q)}_k, \psi^{(q)}_{T,K})$.

(2) Suppose that $N$ is divisible by $K$ and that each block has exactly $N/K$ positions where bits of the records are chosen at random. Thus,

$$|\mathcal{I}_k| = N/K, \quad k = 1, \ldots, K.$$  

(6.10)

If the sum on $\mathcal{I}$ in (3.14) is replaced with the sum on the sets having this property and the normalizing factor $(L/N)^{-1}$ is replaced with the factor

$$(L/K)^{-K} (N/K)^{-K},$$

then we write $\hat{\mathcal{G}}(f^{(q)}_k, \psi^{(q)}_{T,K})$ instead of $\mathcal{G}(f^{(q)}_k, \psi^{(q)}_{T,K})$.

The reason for introducing the approximations above is the point that an investigation of the functions $G$ and $\mathcal{G}$ requires rather complicated formalism related to the analysis of large deviations because the accuracy of the upper bound (4.6) does not allow us to use it for blocks with a small number of randomly chosen bits. Therefore we cannot prove that the maximum on $\mathcal{I}$ in (3.13) is attained exactly on the set satisfying (6.9) for some $k_1, \ldots, k_{N/r}$. We also cannot claim that the largest contribution to the sum on $\mathcal{I}$ in (3.14) is due to the sets satisfying (6.10). Nevertheless, if we proceed with the analysis, then conclusions about the behavior of the functions $G$ and $\mathcal{G}$ are essentially the same as the ones presented below for the functions $\hat{G}$ and $\hat{\mathcal{G}}$.

**Theorem 2.** Let

$$q = 2^\ell/K.$$  

(6.11)

1. If the parameter $K$ is chosen in such a way that

$$\ell \leq K \log \sqrt{L/K} + K \log \sqrt{\pi/2}$$  

(6.12)

then

$$\hat{G}(f^{(q)}_k, \psi^{(q)}_{T,K}) \leq 4^{KN/L} \left| S^{(q)}_{T,KN/L} \right| 2^{-\ell N/L}.$$  

(6.13)

2. If the parameter $K$ is chosen in such a way that

$$\ell \leq K \log \sqrt{N/K} + K \log \sqrt{\pi/2}$$  

(6.14)

then

$$\hat{\mathcal{G}}(f^{(q)}_k, \psi^{(q)}_{T,K}) \leq 4^K \left| S^{(q)}_{T,K} \right| 2^{-\ell}.$$  

(6.15)

The proof is given in the Appendix.

Obvious upper bounds on the cardinalities of the sets $S^{(q)}_{T,KN/L}$ and $S^{(q)}_{T,K}$ can be given as

$$\left| S^{(q)}_{T,KN/L} \right| \leq (T+1)^{KN/L}, \quad \left| S^{(q)}_{T,K} \right| \leq (T+1)^K.$$  

(6.16)

since each $t_k, k = 1, \ldots, K$, in (6.8) can take at most $T + 1$ different values. Thus, if (6.3) holds, $K$ is a constant and

$$T \sim L^\alpha, \quad \alpha \in (0, 1),$$

then upper bounds on $\hat{G}$ and $\hat{\mathcal{G}}$ decrease with $L$ as a function $1/L^{1-\alpha}$; the value of $T$ can be certainly greater than $\ell$. It is rather peculiar that the expression on the right-hand side of (6.15) does not depend on $N$, and we only require the inequality (6.14) to be satisfied. Notice also that if we use upper bounds (6.16), then the function on the right-hand side of (6.13) is the one in (6.15) to the power $N/L$. This ratio can be viewed as the probability that the bit of a record located at some fixed position is chosen at random according to a uniform distribution. This probability affects only the upper bound on the function $\hat{G}$ that estimates the ‘worst case performance’ of the hashing algorithm and does not affect the upper bound on the function $\hat{\mathcal{G}}$ that estimates the ‘average case performance’ of the algorithm.

The above conclusions confirm that our hashing scheme allows one to attain a good performance of hashing for a wide class of DBs.

### 7. CONCLUDING REMARKS

We presented a hashing algorithm, which can be effectively used when the database is a collection of records that are close to each other in the Hamming sense. The basic model for these databases was chosen in such a way that relatively few positions of records have different contents, while all the others coincide. It seems to be that this model is much closer to practical needs than the model where each position of each record is an i.i.d. random variable chosen from $\{0, 1\}$ with probability $\frac{1}{2}$. The introduction of the Hamming weight of the record as the value of the hash-function is reasonable since it makes the algorithm independent of the locations of non-coinciding positions. Note that similar effects can be observed in the theory of integral transformations where accumulated parameters, like the results of the Fourier transform, are effectively used. In our case, the Hamming weight of a binary vector serves as an accumulated parameter. Moreover, the Hamming weight is the distance between a given vector and the all-zero vector in topological sense, and we use the triangle inequality to bound the distance between the records and the given pattern from below on the basis of their Hamming weights. This possibility is important for the problem where the task of the decoder is formulated as construction of the set of addresses of records that can differ from the pattern in at most $T$ positions. The information obtained from the values of the hash-function allows us to estimate the Hamming distance between binary vectors and essentially reduce the number of comparisons equal to $\sum_{\ell=0}^{T} \binom{L}{\ell}$, which is needed for the search for all.
items generated as corrupted versions of the pattern in at most \( T \) positions. Notice that the above considerations are also valid for any distance function.

The next step of the analysis was an estimation of the performance of the hashing algorithm. We proceeded using the properties of the binomial distribution obtained when some bits of each record are assigned at random. It turns out that simple reduction of the Hamming weight of the record modulo \( q \), where \( q \) is assigned in a certain way, creates another random variable having almost uniform distribution over the set \( \{0, \ldots, q - 1\} \) and simultaneously conserves the triangle inequality. Another interesting possibility, coming with the reduction, is partitioning the vectors in \( K \) blocks having the length \( L/K \). As a result, we can independently form lower bounds on the Hamming distances between corresponding blocks and determine the value of the total bound as the sum of \( K \) terms. This scheme allows us to adopt given parameters of the model to the desired values of \( q \).

An accurate estimate of the performance of the algorithm for specific data requires calculations with the exact values of the binomial distribution because the accuracy of the bounds established in Lemma 1 is not sufficient for this purpose. These calculations can be performed rather easily using generating functions technique, and they lead to results which are similar to the results given in Table 1. We think that conclusions about the behavior of the functions, which were used as criteria of quality of hashing, are interesting.

ACKNOWLEDGEMENT

The author is grateful to Prof. James L. Massey for the help in preparing the manuscript. Comments of anonymous referees are also highly appreciated.

REFERENCES


APPENDIX

A.1 Proof of Lemma 1

Suppose that \( n \) is even. If we prove that

\[
\sigma \leq \Theta_n(q)(\delta) \leq \sigma + \left(\frac{n}{n/2}\right)^{2^n} \tag{A.1}
\]

for all \( \delta \in \mathbb{Q} \) and some \( \sigma \geq 0 \), then inequalities (4.6) follow. Really, (4.5) and (A.1) imply

\[
q \sigma \leq 1 \leq q \sigma + q \left(\frac{n}{n/2}\right)^{2^n}.
\]

Thus

\[
\frac{1}{q} - \left(\frac{n}{n/2}\right)^{2^n} \leq \sigma \leq \frac{1}{q}
\]

and we derive (4.6) from (A.1).

Let us express (4.4) as

\[
\Theta_n(q)(\delta) = \sum_{s \geq 0} \Omega_n(\delta + sq) \tag{A.2}
\]

and prove (A.1) for

\[
\sigma = \sum_{s \geq 1} \Omega_n(n/2 - sq/2). \tag{A.3}
\]

(i) Suppose that \( \delta \) is chosen in such a way that the lattice \( \delta, \delta + q, \delta + 2q, \ldots \) contains integers \( n/2 - q/2 \).
and $n/2 + q/2$. Then this lattice can be represented as

$$\sum_{s \geq 0} \Omega_n(\delta + sq) = \sum_{s \geq 0} \left[ \Omega_n(n/2 - (2s + 1)q/2) + \Omega_n(n/2 + (2s + 1)q/2) \right].$$

Thus, using (A.2), (A.3) and properties (a), (c) of the distribution $\Omega_n$, we obtain the following inequalities:

$$\Theta_n(q) = \sigma + \Omega_n(n/2)$$

for all $s \geq 1$. Therefore, using (A.4) and property (c) we write

$$\Theta_n(q) \leq \sum_{s \geq 0} \Omega_n(n/2 - (2s + 1)q/2) + \sigma + \Omega_n(n/2)$$

The case $d_0 \in (n/2, n/2 + q/2)$ can be analyzed in a similar way. One can also easily extend the above considerations to the case when $n$ is odd.

**A.2. Proof of Theorem 1**

Since

$$|J(x, X|f_1^{(q)}, \varphi_{T,1}^{(q)})| = \sum_{j=1}^{M} \chi \left\{ j \in J(x, X|f_1^{(q)}, \varphi_{T,1}^{(q)}) \right\}$$

using (3.9)–(3.12) we have

$$G_{x_0,T}(x|f_1^{(q)}, \varphi_{T,1}^{(q)}) = \sum_{x_j} P(x_j|x_0, T) \times \chi \left\{ j \in J(x, X|f_1^{(q)}, \varphi_{T,1}^{(q)}) \right\}.$$
Therefore, for any vector $x_j \in \{0, 1\}^L$ with $p(x_j | x_0, I) > 0$, we have

$$\text{wt}(x_j) = w_j + w_0,$$

where

$$w_j \triangleq \left\{ i \in I : x_j^{(i)} = 1 \right\},$$

$$w_0 \triangleq \left\{ i \in \{1, \ldots, L\} \setminus I : x_0^{(i)} = 1 \right\}.$$

Then

$$d_{\text{Lee}}^{(q)}(\text{wt}(x), w_0^{\{q\}}(x)) = d_{\text{Lee}}^{(q)}(\text{wt}(x) + w_0^{\{q\}}, w_0^{\{q\}}) = d_{\text{Lee}}^{(q)}((\text{wt}(x) - w_0^{\{q\}}), w_0^{\{q\}})$$

and

$$d_{\text{Lee}}^{(q)}(\text{wt}(x), w_0^{\{q\}}(x_j)) = \tau$$

$$\iff w_j^{\{q\}} \in \left\{ (\text{wt}(x) - w_0^{\{q\}} + \tau)^{\{q\}}, (\text{wt}(x) - w_0^{\{q\}} - \tau)^{\{q\}} \right\}.$$

The random variable $w_j^{\{q\}}$ has the distribution $\Theta(I)$ in the probabilistic ensemble under considerations. Hence, using notation (4.10), we write

$$\Delta_{x_0, I}(\tau | x) = \sum_{\delta \in \mathcal{Q}} \Theta(I)(\delta)$$

$$\times \chi \left\{ \delta \in \left\{ (\text{wt}(x) - w_0^{\{q\}} + \tau)^{\{q\}}, (\text{wt}(x) - w_0^{\{q\}} - \tau)^{\{q\}} \right\} \right\}$$

$$\leq \hat{\Theta}(q)|I| \cdot \begin{cases} 1, & \text{if } \tau = 0, \\ 2, & \text{if } \tau > 0, \end{cases}$$

(A.6)

and

$$G_{x_0, I}(x | f^{(q)}_K, \varphi^{(q)}_{T, K}) \leq (2T + 1) \hat{\Theta}(q)|I|,$$

(A.7)

as it follows from (A.5). Using (6.5) and upper bound (4.7) with $n = N$ we have

$$\hat{\Theta}(q)|I| \leq 2^{-\xi} + \sqrt{\frac{2}{\pi N}},$$

(A.8)

and (6.6) follows from (A.7), (A.8) if inequality (6.4) is satisfied.

### A.3. Proof of Theorem 2

For any vector $t_K \in \{0, 1, \ldots, q - 1\}^K$, the conditional probability of choosing a record $x$ with $f^{(q)}_K(x) = (t_1, \ldots, t_K)$ (we assume that each component of this vector is represented with log $q$ bits) such that

$$d_{\text{Lee}}^{(q)}(\text{wt}(x_k), w_0^{\{q\}}(x_{jk})) = t_k, \ k = 1, \ldots, K,$$

can be expressed as

$$\prod_{k=1}^K \Delta_{x_0, I}(t_k | x_k),$$

where

$$\Delta_{x_0, I}(t_k | x_k) \triangleq \sum_{x_j} P(x_j | x_0, I)$$

$$\times \chi \left\{ d_{\text{Lee}}^{(q)}((\text{wt}(x_k), w_0^{\{q\}}(x_{jk})) = t_k \right\}.$$

Thus, using the definition of the set $S_{T, K}^{(q)}$ in (6.8) we have

$$G_{x_0, I}(x | f^{(q)}_K, \varphi^{(q)}_{T, K}) = \sum_{t_K^* \in S_{T, K}^{(q)}} \prod_{k=1}^K \Delta_{x_0, I}(t_k | x_k).$$

Notice that

$$\Delta_{x_0, I}(t_k | x_k) \begin{cases} 0, & \text{if } t_k > 0, \\ 1, & \text{if } t_k = 0, \end{cases}$$

and repeating arguments of the proof of Theorem 1 obtain

$$G_{x_0, I}(x | f^{(q)}_K, \varphi^{(q)}_{T, K}) \leq \sum_{t_K^* \in S_{T, K}^{(q)}} \prod_{k=1}^K \left[ 2\hat{\Theta}(q)|I| \right].$$

(A.9)

The upper bound (A.9) does not depend on $x$. Thus, using (6.9), (6.10) and notations $\hat{G}, \hat{G}$ introduced in Section 6 we have

$$\hat{G}_{x_0, I}(f^{(q)}_K, \varphi^{(q)}_{T, K}) \leq \sum_{t_K^* \in S_{T, K}^{(q)}} \prod_{k=1}^K \left[ 2\hat{\Theta}(q)|I| \right].$$

(A.10)

and

$$\hat{G}_{x_0, I}(f^{(q)}_K, \varphi^{(q)}_{T, K}) \leq \sum_{t_K^* \in S_{T, K}^{(q)}} \prod_{k=1}^K \left[ 2\hat{\Theta}(q)|I| \right].$$

Notice that we have replaced the set $S_{T, K}^{(q)}$ in (A.10) with the set $S_{T, N/r}^{(q)}$ because $t_k = 0$ for all indices $k \notin \{k_1, \ldots, k_{N/r}\}$ (if $\mathcal{T}$ is fixed, then the indices $k_1, \ldots, k_{N/r}$ are known). Using the upper bound (4.7) with $n = L$ and $n = N/r$, respectively, and inequalities (6.12), (6.14) we derive (6.13), (6.15) from (A.10), (A.11).