BRICS RS-99-13 R. Pagh: Hash and Displace: Efficient Evaluation of Minimal Perfect Hash Functions

BRICS

Basic Research in Computer Science

Hash and Displace: Efficient Evaluation of Minimal Perfect Hash Functions

Rasmus Pagh

BRICS Report Series **RS-99-13**

ISSN 0909-0878 **May 1999**

See back inner page for a list of recent BRICS Report Series publications. Copies may be obtained by contacting:

BRICS
Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
DK–8000 Aarhus C
Denmark

Telephone: +45 8942 3360
Telefax: +45 8942 3255
Internet: BRICS@brics.dk

BRICS publications are in general accessible through the World Wide Web and anonymous FTP through these URLs:

`http://www.brics.dk`
`ftp://ftp.brics.dk`
This document in subdirectory `RS/99/13/`

Hash and Displace:
Efficient Evaluation of Minimal Perfect Hash Functions

Rasmus Pagh*

May 1999

Abstract

A new way of constructing (minimal) perfect hash functions is described. The technique considerably reduces the overhead associated with resolving buckets in two-level hashing schemes. Evaluating a hash function requires just one multiplication and a few additions apart from primitive bit operations. The number of accesses to memory is two, one of which is to a fixed location. This improves the probe performance of previous minimal perfect hashing schemes, and is shown to be optimal. The hash function description ("program") for a set of size n occupies $O(n)$ words, and can be constructed in expected $O(n)$ time.

1 Introduction

This paper deals with classes of hash functions which are *perfect* for the n-subsets of the finite universe $U = \{0, \ldots, u-1\}$. For any $S \in \binom{U}{n}$ – the subsets of U of size n – a perfect class contains a function which is 1-1 on S ("perfect" for S). We consider perfect classes with range $\{0, \ldots, a-1\}$.

A perfect class of hash functions can be used to constuct static dictionaries (data structures storing sets $S \in \binom{U}{n}$ and supporting membership queries, "$u \in S$?"): Store a function h which is 1-1 on the set, S, and for each element $s \in S$, store s in entry $h(s)$ of an a-element table. A membership query on s is processed by comparing s to entry $h(s)$ in the table. The attractiveness of using perfect hash functions for this purpose depends on several characteristics of the class.

1. The efficiency of evaluation, in terms of computation and the number of probes into the description.

2. How hard is it to find a perfect function in the class?

3. How close to n can we choose a, the range of the functions?

*Supported in part by the ESPRIT Long Term Research Programme of the EU under project number 20244 (ALCOM-IT)

4. How much space is required to store a function?

It turns out that, for suitable perfect classes of hash functions, the answers to all of these questions are satisfactory in the sense that it is possible to do (more or less) as well as one could hope for. Nevertheless, theoretically optimal schemes have seen limited use in practice, being substituted by heuristics which typically work well. It is thus still of interest to find classes with good properties from a theoretical as well as from a practical point of view.

This paper presents a simple perfect class of hash functions which, in a unit-cost word RAM model with multiplication, matches the best results regarding items 2 and 3 above, and at the same time improves upon the best known efficiency of evaluation. The (constant) amount of computation required is about halved, and the number of non-oblivious probes into the function description (probes depending on the argument to the function and/or previous probes) is reduced from two to one, which is optimal. The latter is particularly important if the perfect hash function resides on secondary storage where seek time is the dominant cost. The class is surpassed in space usage (by rigorously analyzed schemes) only by classes which are quite complicated and inefficient with respect to the evaluation of the function. Altogether, the class is believed to be attractive in practice.

1.1 Previous Work

Czech, Havas and Majewski provide a comprehensive survey of perfect hashing [3]. We review some of the most important results.

Fredman, Komlós and Szemerédi [8] showed that it is possible to construct space efficient perfect hash functions with range $a = O(n)$ which can be evaluated in constant time. Their model of computation is a word RAM where an element of U fits into one machine word [1], and with unit cost arithmetic operations and memory lookups. Henceforth we will concentrate on hash functions with range $a = O(n)$ which can be *evaluated in constant time* in this model of computation.

An FKS hash function description occupies $O(n)$ machine words. At the expense of an extra level of indirect addressing, it can be turned into a *minimal* perfect hash function, that is, one with range $a = n$. The description size is then $6n$ words (in a straightforward implementation), and three words are probed during evaluation. Schmidt and Siegel [12] utilize compact encoding techniques to compress the description to the optimal $O(n + \log\log u)$ bits. The scheme is hard to implement, and the evaluation time is prohibitive (although constant). Their scheme is thus mainly of theoretical interest. In the following, we will consider classes whose functions can be stored in $O(n)$ machine words.

The fastest algorithm for deterministically constructing a perfect hash function with constant lookup time runs in time $O(n \log n)$ [11]. *Randomized* construction algorithms offer better expected performance. An FKS perfect hash function can be constructed in expected time $O(n)$ if the algorithm has access to a source of random bits.

Some work on minimal perfect hashing has been done under the assumption that the algorithm can pick and store truly random functions [2, 9]. Since the space requirements

[1] This is not a serious limitation, since universal classes provide (efficient) perfect hash functions with $O(\log n + \log\log u)$ bit description and range $O(n^2)$. This means that the universe in question can be reduced to size $O(n^2)$.

for truly random functions makes this unsuitable for implementation, one has to settle for pseudo-random functions in practice. Empirical studies show that limited randomness properties are often as good as total randomness. Fox et al [6, 7] studied some classes which share several features with the one presented here. Their results indicate convincing practical performance, and suggest that it is possible to bring down the storage requirements further than proved here. However, it should be warned that doing well in most cases may be much easier than doing well in the worst case. In [3, Sect. 6.7] it is shown that three published algorithms for constructing minimal perfect hash functions, claiming (experimentally based) expected polynomial running times, had in fact expected *exponential* running times. But bad behavior was so rare that it had not been encountered during the experiments.

1.2 The Tarjan-Yao Displacement Scheme

The class of hash functions presented here can be seen as a variation of an early perfect class of hash functions due to Tarjan and Yao [13]. Their class requires a universe of size $u = O(n^2)$ (which they improve to $u = O(n^c)$ for constant $c > 2$). The idea is to split the universe into blocks of size $O(n)$, each of which is assigned a "displacement" value. The ith element within the jth block is mapped to $i + d_j$, where d_j is the displacement value of block j. Suitable displacement values can always be found, but in general displacement values (and thus hash table size) larger than n may be required. A "harmonic decay" condition on the distribution of elements within the blocks ensures that suitable displacement values in the range $\{0, \ldots, n\}$ can be found, and that they can in fact be found "greedily" in decreasing order of the number of elements within the blocks. To achieve harmonic decay, Tarjan and Yao first perform a displacement "orthogonal" to the other.
The central observation of this paper is that a reduction of the universe to size $O(n^2)$, as well as harmonic decay, can be achieved using universal hash functions. Or equivalently, that buckets in a (universal) hash table can be resolved using displacements.

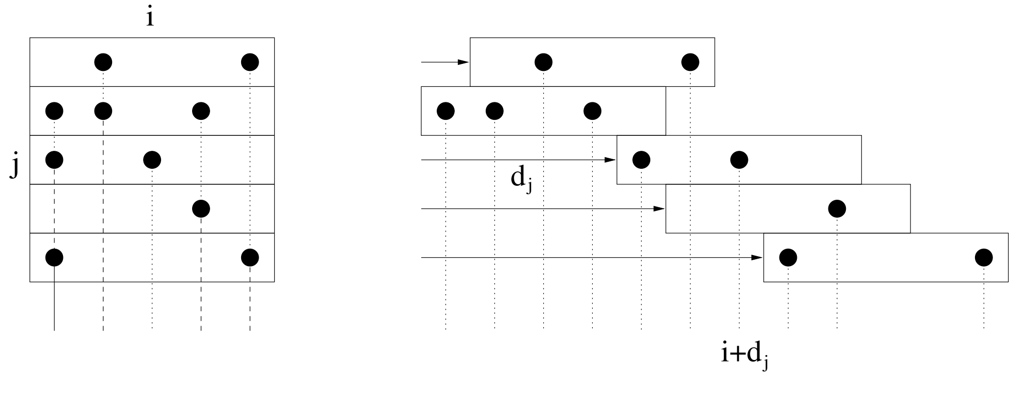

Figure 1: Tarjan-Yao displacement scheme

2 A Perfect Class of Hash Functions

The concept of universality [1] plays an important role in the analysis of our class. We use the following notation.

Definition 1 *A class of functions* $H_r = \{h_1, \ldots, h_k\}$, $h_i : U \to \{0, \ldots, r-1\}$, *is* c-universal *if for any* $x, y \in U$, $x \neq y$,

$$\Pr_i[h_i(x) = h_i(y)] \leq c/r \ .$$

It is $(c,2)$-universal *if for any* $x, y \in U$, $x \neq y$, *and* $p, q \in \{0, \ldots, r-1\}$,

$$\Pr_i[h_i(x) = p \text{ and } h_i(y) = q] \leq c/r^2 \ .$$

Many such classes with constant c are known, see e.g. [4]. For our application the important thing to note is that there are universal classes that allow efficient storage and evaluation of their functions. More specifically, $O(\log u)$ (and even $O(\log n + \log\log u)$) bits of storage suffice, and a constant number of simple arithmetic and bit operations are enough to evaluate the functions. Furthermore, c is typically in the range $1 \leq c \leq 2$. Henceforth we will refer to such efficient universal families only.

The second ingredient in the class definition is a *displacement* function. Generalizing Tarjan and Yao's use of integer addition, we allow the use of any group structure on the blocks. It is assumed that the elements $\{0, \ldots, a-1\}$ of a block correspond to (distinct) elements in a group $(G, \boxplus, e)$, such that group operations may be performed on them. We assume the group operation and element inversion to be computable in constant time.

Definition 2 *A* displacement function *for a group* $(G, \boxplus, e)$ *is a function of the form* $x \mapsto x \boxplus d$ *for* $d \in G$. *The element* d *is called the* displacement value *of the function.*

In the following it may be helpful to simply think of the displacement functions as addition modulo a. We are ready to define the class.

Definition 3 *Let* $(G, \boxplus, e)$ *be a group,* $D \subseteq G$ *a set of displacement values, and let* H_a *and* H_b *be* c_f*- and* c_g*-universal, respectively. We define the following class of functions from* U *to* G*:*

$$\mathcal{H}(\boxplus, D, a, b) = \{h \mid h(x) = f(x) \boxplus d_{g(x)}, f \in H_a, g \in H_b, d_i \in D \text{ for } i = 0, \ldots, b-1\} \ .$$

Evaluation of a function in $\mathcal{H}(\boxplus, D, a, b)$ consists of using a displacement function, determined by $g(x)$, on the value $f(x)$. In terms of the Tarjan-Yao scheme, g assigns a block number to each element of U, and f determines its number within the block. The next section will show that when $a > c_f n/4$ and $b \geq 2c_g n$, it is possible to find f, g and displacement values such that the resulting function is perfect. The class requires storage of b displacement values. Focusing on instances with a reasonable memory usage, we from now on assume that $b = O(n)$ and that elements of D can be stored in one machine word.

The range of functions in $\mathcal{H}(\boxplus, D, a, b)$ is $\{x \boxplus d \mid x \in \{0, \ldots, a-1\}, d \in D\}$. The size of this set depends on the group in question and the set D. Since our interest is in functions with range n (or at most $O(n)$), we assume $a \leq n$. In Sect. 3.2 the issue of classes with larger range is briefly treated.

Our class (with group operator addition modulo n and $D = \{0, \ldots, n-1\}$) is a special case of the class defined in [5, Section 4]. There, several interesting properties of randomly chosen functions from the class are exhibited, but the issue of finding perfect functions in the class is not addressed.

2.1 Analysis

This section gives a constructive (but randomized) method for finding perfect hash functions in the class $\mathcal{H}(\boxplus, D, a, b)$ for suitable D, a and b. The key to the existence of proper displacement values is a certain "goodness" condition on f and g. Let $B(g,i) = \{x \in S \mid g(x) = i\}$ denote the elements in the ith block given by g.

Definition 4 *Let $r \geq 1$. A pair $(f,g) \in H_a \times H_b$, is r-*good *(for S) if*

1. *The function $x \mapsto (f(x), g(x))$ is 1-1 on S, and*
2. *$\sum_{i,|B(g,i)|>1} |B(g,i)|^2 \leq n/r$.*

The first condition says that f and g successfully reduce the universe to size ab (clearly, if $(f(x_1), g(x_1)) = (f(x_2), g(x_2))$ then regardless of displacement values, x_1 and x_2 collide). The second condition implies the harmonic decay condition of Tarjan and Yao (however, it is not necessary to know their condition to understand what follows). We show a technical lemma, estimating the probability of randomly finding an r-good pair of hash functions.

Lemma 5 *Assume $a \geq c_f n/4r$ and $b \geq 2c_g rn$. For any $S \in \binom{U}{n}$, a randomly chosen pair $(f,g) \in H_a \times H_b$ is r-good for S with positive probability, namely more than $(1-\frac{2c_g rn}{b})(1-\frac{c_f n}{4ra})$.*

Proof. By c_g-universality, the expected value for the sum $\sum_i \binom{|B(g,i)|}{2} = \sum_{\{u,v\} \in \binom{S}{2}, g(u)=g(v)} 1$ is bounded by $\binom{n}{2}\frac{c_g}{b} < \frac{c_g n^2}{2b}$, so applying Markov's inequality, the sum has value less than $n/4r$ with probability $> 1 - \frac{2c_g rn}{b}$. Since $|B(g,i)|^2 \leq 4\binom{|B(g,i)|}{2}$ for $|B(g,i)| > 1$, we then have that $\sum_{i,|B(g,i)|>1} |B(g,i)|^2 \leq n/r$. Given a function g such that these inequalities hold, we would like to bound the probability that $x \mapsto (f(x), g(x))$ is 1-1 on S. By reasoning similar to before, we get that for randomly chosen f the expected number of collisions among elements of $B(g,i)$ at most $\binom{|B(g,i)|}{2} c_f/a$. Summing over i we get the expected total number of collisions to be less than $c_f n/4ra$. Hence there is no collision with probability more than $1 - \frac{c_f n}{4ra}$. By the law of conditional probabilities, the probability of fulfilling both r-goodness conditions can be found by multiplying the probabilities found. □

We now show that r-goodness is sufficient for displacement values to exist, by means of a constructive argument in the style of [13, Theorem 1]:

Theorem 6 *Let $(f,g) \in H_a \times H_b$ be r-good for $S \in \binom{U}{n}$, $r \geq 1$, and let $|D| \geq n$. Then there exist displacement values $d_0, \ldots, d_{b-1} \in D$, such that $x \mapsto f(x) \boxplus d_{g(x)}$ is 1-1 on S.*

Proof. Note that displacement d_i is used on the elements $\{f(x) \mid x \in B(g,i)\}$, and that any $h \in \mathcal{H}(\boxplus, D, a, b)$ is 1-1 on each $B(g,i)$. We will assign the displacement values one by one, in non-increasing order of $|B(g,i)|$. At the kth step, we will have displaced the $k-1$ largest sets, $\{B(g,i) \mid i \in I\}$, $|I| = k-1$, and want to displace the set $B(g,j)$ such that no collision with previously displaced elements occurs. If $|B(g,j)| \leq 1$ this is trivial since $|D| \geq n$, so we assume $|B(g,j)| > 1$. The following claim finishes the proof:

Claim 7 *If $|B(g,j)| > 1$, then with positive probability, namely more than $1 - \frac{1}{r}$, a randomly chosen $d \in D$ displaces $B(g,j)$ with no collision.*

Proof. The displacement values which are *not* available are those in the set

$$\{f(x)^{-1} \boxplus f(y) \boxplus d_i \mid x \in B(g,j), y \in B(g,i), i \in I\} \ .$$

It has size $\leq |B(g,j)| \sum_{i \in I} |B(g,i)| < \sum_{i, |B(g,i)|>1} |B(g,i)|^2 \leq n/r$, using first non-increasing order, $|B(g,j)| > 1$, and then r-goodness. Hence there must be more than $(1 - \frac{1}{r})|D|$ good displacement values in D. □

Lemma 5 implies that when $a \geq (\frac{c_f}{4r} + \epsilon)n$ and $b \geq (2c_g r + \epsilon)n$, an r-good pair of hash functions can be found (and verified to be r-good) in expected time $O(n)$. The proof of Theorem 6, and in particular Claim 7, shows that for a $1+\epsilon$-good pair (f,g), the strategy of trying random displacement values in D successfully displaces all blocks with more than one element in expected $1/\epsilon$ attempts per block. When $O(n)$ random elements of D can be picked in $O(n)$ time, this runs in expected time $O(n/\epsilon) = O(n)$. Finally, if displacing all blocks with only one element is easy (as is the case in the examples we look at in Sect. 2.2), the whole construction runs in expected time $O(n)$.

For a 1-good pair (f,g), Theorem 6 just gives the *existence* of proper displacement values.

2.2 Instances of the Class

We defined our class at a rather abstract level, so let us look at some specific instances. In particular, the choice of group and set of displacement values is treated. For simplicity, we use universal classes with constant 1. The number of displacement values, b, must be at least $2n$ to ensure existence of perfect functions, and at least $(2+\epsilon)n$, $\epsilon > 0$, for expected linear time construction. For convenience, self-contained definitions of the classes are given.

Addition

The set of integers with addition is a very natural choice of group. For $D = \{0, \ldots, n-1\}$, we get the class:

$$\mathcal{H}_{\mathbb{Z}} = \{h \mid h(x) = f(x) + d_{g(x)}, f \in H_{n/4}, g \in H_b, 0 \leq d_i < n \text{ for } i = 0, \ldots, b-1\}\}$$

The range of hash functions in the class is $\{0, \ldots, \frac{5}{4}n - 2\}$, so it is not minimal.

Addition Modulo n

The previous class can be made minimal at the expense of a computationally more expensive group operator, addition modulo n:

$$\mathcal{H}_{\mathbb{Z}_n} = \{h \mid h(x) = (f(x) + d_{g(x)}) \bmod n, f \in H_n, g \in H_b, 0 \le d_i < n \text{ for } i = 0, \dots, b-1\}\}$$

Note that since the argument to the modulo n operation is less than $2n$, it can be implemented using one comparison and one subtraction.

Bitwise Exclusive Or

The set of bit strings of length $\ell = \lceil \log n \rceil$ form the group $\mathbb{Z}_2^\ell$ under the operation of bitwise exclusive or, denoted by $\oplus$. We let $\{0, \dots, n-1\}$ correspond to ℓ-bit strings of their binary representation, and get

$$\mathcal{H}_{\mathbb{Z}_2^\ell} = \{h \mid h(x) = f(x) \oplus d_{g(x)}, f \in H_{2^{\ell-1}}, g \in H_b, d_i \in \{0,1\}^\ell \text{ for } i = 0, \dots, b-1\}$$

The range of functions in this class is (corresponds to) the numbers $\{0, \dots, 2^\ell - 1\}$. It is thus only minimal when n is a power of two. However, for $b \ge 4n^2/2^\ell$, displacement values can be chosen such that elements of the set are mapped to $\{0, \dots, n-1\}$: All displacement values for sets with more than one element can be chosen from $0\{0,1\}^{\ell-1}$, and single elements can be displaced to any value desired. Since some elements not in the set might map outside $\{0, \dots, n-1\}$, a check for values larger than $n-1$ must be inserted.

2.2.1 Construction Time

The discussion in Sect. 2.1 implies that perfect functions in the above classes can be found efficiently. Also note that it is possible to "pack" all displacement values into $b\lceil \log n \rceil$ bits.

Theorem 8 *Let $S \in \binom{U}{n}$ and take $\epsilon > 0$. A perfect hash function in the classes $\mathcal{H}_{\mathbb{Z}}$, $\mathcal{H}_{\mathbb{Z}_n}$ and $\mathcal{H}_{\mathbb{Z}_2^\ell}$, with storage requirement $(2+\epsilon)n$ words (or $(2+\epsilon)n\lceil \log n \rceil$ bits), can be found in expected time $O(n)$.*

Pseudo-code for the construction algorithm and evaluation function of $\mathcal{H}_{\mathbb{Z}_n}$ is given in Appendix A.

3 Variants

This section describes some variants of the basic scheme presented in the previous section.

3.1 Two Hash Functions for the Price of One

In the introduction we promised a class that was not only efficient with respect to the number of probes into the description, but also with respect to the amount of computation involved. It would seem that the evaluation of hash functions f and g makes the computational cost of the scheme presented here comparable to that of e.g. the FKS scheme. However, we have one advantage that can be exploited: The hash functions used are *fixed* in advance, as opposed to other schemes where the choice of second hash function depends on the value of the first. We will show how to "simulate" our two universal hash functions with a single $(c, 2)$-universal hash function.

Definition 9 *Let $V = \{0, \ldots, ab-1\}$. Functions $p_f : V \to \{0, \ldots, a-1\}$ and $p_g : V \to \{0, \ldots, b-1\}$ are said to* decompose *V if the map $x \mapsto (p_f(x), p_g(x))$ is 1-1 on V.*

Let H_{ab} be a class of $(c, 2)$-universal hash functions with range $\{0, \ldots, ab-1\}$, and let p_f and p_g decompose $\{0, \ldots, ab-1\}$. It is not hard to show that $\{p_f \circ h \mid h \in H_{ab}\}$ and $\{p_g \circ h \mid h \in H_{ab}\}$ are $(c, 2)$-universal, and hence also c-universal. However, because of possible dependencies, this will not directly imply that $(p_f \circ h, p_g \circ h)$ is r-good with positive probability, for random $h \in H_{ab}$. But we now show that, with a small penalty in the number of displacement values, this is the case.

Lemma 5b *Let H_{ab} be a $(c, 2)$-universal class, and let p_f and p_g decompose $\{0, \ldots, ab-1\}$. Assume $ab \geq cn(n + 4ra)/2$. For any $S \in \binom{U}{n}$, a pair $(p_f \circ h, p_g \circ h)$ with randomly chosen $h \in H_{ab}$, is r-good for S with positive probability, namely more than $1 - \frac{cn(n+4ra)}{2ab}$.*

Proof. When choosing $h \in H$ uniformly at random, the expected number of collisions between pairs of elements in S is $\leq \binom{n}{2} c/ab < \frac{cn^2}{2ab}$. Hence a collision occurs with probability less than $\frac{cn^2}{2ab}$. Using injectivity of the decomposition functions, the same holds for $x \mapsto ((p_f \circ h)(x), (p_g \circ h)(x))$. Since $\{p_g \circ h \mid h \in H_{ab}\}$ is c-universal, the argument in the proof of Lemma 5 shows that $\sum_{i,|B(p_g \circ h, i)|>1} |B(p_g \circ h, i)|^2 > n/r$ occurs with probability less than $\frac{2crn}{b}$. Since $\frac{cn^2}{2ab} + \frac{2crn}{b} = \frac{cn(n+4ra)}{2ab}$, the stated probability of r-goodness follows. □

For $a = n$ this means that $b = \frac{5c}{2} n$ is enough to ensure the existence of a 1-good pair $(p_f \circ h, p_g \circ h)$. For $b = (\frac{5c}{2} + \epsilon)n$ a $1 + \epsilon'$-good pair, where $\epsilon' < \epsilon/2c$, can be found in an expected constant number of attempts.

Decomposition of $\{0, \ldots, ab-1\}$ can be done efficiently when a or b is a power of two. Then p_f and p_g simply pick out appropriate bits. More generally, $p_f(u) = u \text{ div } b$, $p_g(u) = u \bmod b$, and $p_f(u) = u \bmod a$, $p_g(u) = u \text{ div } a$ are natural choices for decomposing the range of h.

Returning to the claim in the introduction, we use the $(1, 2)$-universal class of Dietzfelbinger [4], which requires just one multiplication, one addition and some simple bit operations when the range has size a power of two. Choose a and b as powers of two satisfying $ab \geq (n(n + 4a) + \epsilon)/2$, and it is easy to compute p_f and p_g. Since, by Sect. 2.2, the complexity of the displacement function can be low, we have argued that (apart from a few less expensive operations) one multiplication suffices.

3.2 Larger Range

We have focused on perfect hash functions with range $O(n)$, since these have the most applications. It is, however, straightforward to generalize the classes we have seen to ones with range $a = \omega(n)$. The number of displacement values necessary is in general $b = O(n^2/a)$

(note that for $a \geq n^2$ a universal class in itself is perfect). The differences to the theory seen so far are:

- r-goodness is generalized so that the second requirement is $\sum_{i,|B(g,i)|>1} |B(g,i)|^2 \leq a/r$.
- The set of displacement values must have size at least a.

4 Proof of Probe Optimality

This section serves to prove that one cannot in general do better than two probes into the description of a perfect hash function. Of course, for large word length, w, one probe *does* suffice: If $w \geq \lceil \log n \rceil + \lceil u/b \rceil$ then a bitmap of the whole set, as well as rank information in each word, can be put into b words, and one-probe perfect minimal hashing is easy. This bound is close to optimal.

Theorem 10 *Let $\mathcal{H} = \{h_1, \ldots, h_k\}$, $h_i : U \to \{0, \ldots, a-1\}$, where $u \geq 2a$, be a class of perfect hash functions for $\binom{U}{n}$, $n \geq 2$. Assume that the functions can be described using b words of size $w \geq \log u$, and can be evaluated using one word-probe. Then $w \geq \frac{cn^2/a}{1+ab/u} - 1$, where $c > 0$ is a constant.*

Proof. Denote by U_i the subset of U for which word i is probed. We will choose $B \subseteq \{0, \ldots, b-1\}$ such that $U_B = \cup_{i \in B} U_i$ has size at least $2a$. By the pigeon hole principle, B can be chosen to have size $\lceil \frac{2ab}{u} \rceil$. The crucial observation is that the words given by B must contain enough information to describe a perfect hash function for any n-subset of U_B. By [10, Theorem III.2.3.6] such a description requires at least $\frac{n(n-1)}{2a \ln 2} - \frac{n(n-1)}{2|U_B| \ln 2} - 1 \geq \frac{n(n-1)}{4a \ln 2} - 1$ bits. Therefore $(1 + \frac{2ab}{u})w \geq |B|w \geq \frac{n(n-1)}{4a \ln 2} - 1$, from which the stated bound follows. □

Corollary 11 *In the setting of Theorem 10, if $a = O(n)$ then $b = \Omega(u/w)$ words are necessary.*

5 Conclusion and Open Problems

We have seen that displacements, together with universal hash functions, form the basis of very efficient (minimal) perfect hashing schemes. The efficiency of evaluation is very close to the lowest one conceivable, in that the number of probes into the description is optimal and only one "expensive" arithmetic operation is needed.

The space consumption in bits, although quite competitive with that of other evaluation-efficient schemes, is a factor of $\Theta(\log n)$ from the theoretical lower bound [10, Theorem III.2.3.6]. As mentioned in the introduction, some experiments suggest that the space consumption for this kind of scheme may be brought close to the optimum by simply having fewer displacement values. It would be interesting to extend the results of this paper in that direction: Can low space consumption be achieved in the worst case, or just in the average case? Which randomness properties of the hash functions are needed to lower the number of displacement values? Answering such questions may help to further bring together the theory and practice of perfect hashing.

Acknowledgments: I would like to thank my supervisor, Peter Bro Miltersen, for encouraging me to write this paper. Thanks also to Martin Dietzfelbinger, Peter Frandsen and Riko Jacob for useful comments on drafts of this paper.

A Construction Algorithm and Evaluation Function

Pseudo-code for the $\mathcal{H}_{\mathbb{Z}_n}$ construction algorithm and evaluation function follows below. Parameter ϵ must be greater than 0, the expected running time is inversely proportional to ϵ. The permutation P can be found in linear time using bucket-sort. Note that the modulo operation is assumed to compute the smallest *non-negative* remainder.

function Construct(S, b, ϵ)
  $n := |S|$;
  if $b < (2+\epsilon)(1+\epsilon)n$ **then return** 'Too few displacement values';
  repeat
    Pick f at random in a 1-universal class with range $\{0, \ldots, n-1\}$;
    Pick g at random in a 1-universal class with range $\{0, \ldots, b-1\}$;
    for $i := 0$ **to** $b-1$ **do** $B[i] := \{s \in S \mid g(s) = i\}$;
    Find permutation P such that $|B[P[i]]| \geq |B[P[i+1]]|$ for $i = 0, \ldots, b-2$;
    $m := \max\{i \mid |B[P[i]]| > 1\}$;
  until $(|B[P[0]]|^2 + \cdots + |B[P[m]]|^2 \leq \frac{n}{1+\epsilon})$ and (f is 1-1 on each B[i]);

  for $i := 0$ **to** $n-1$ **do** $T[i]$:=false; /* Marks occupied table entries */

  for $i := 0$ **to** m **do**
    repeat
      Pick $d \in \{0, \ldots, n-1\}$ at random;
    until ($T[(f(s)+d) \bmod n]$=false for each $s \in B[P[i]]$);
    $D[P[i]] := d$;
    for each $s \in B[P[i]]$ **do** $T[(f(s)+d) \bmod n]$:=true;
  end;

  Set $E[1], \ldots, E[k]$ to the k distinct values where $T[E[i]]$=false;
  for $i := m+1$ **to** $m+k$ **do**
    Take (the unique) $s \in B[P[i]]$;
    $D[P[i]] := (E[i-m] - f(s)) \bmod n$;
  end;

  return (f, g, D, n);
end;

function Evaluate(u, f, g, D, n)
  return $(f(u) + D[g(u)]) \bmod n$;
end;

References

[1] J. Lawrence Carter and Mark N. Wegman. Universal classes of hash functions. *J. Comput. System Sci.*, 18(2):143–154, 1979.

[2] Zbigniew J. Czech, George Havas, and Bohdan S. Majewski. An optimal algorithm for generating minimal perfect hash functions. *Information Processing Letters*, 43(5):257–264, October 1992.

[3] Zbigniew J. Czech, George Havas, and Bohdan S. Majewski. Perfect hashing. *Theoretical Computer Science*, 182(1–2):1–143, 15 August 1997. Fundamental Study.

[4] Martin Dietzfelbinger. Universal hashing and k-wise independent random variables via integer arithmetic without primes. In *STACS 96 (Grenoble, 1996)*, pages 569–580. Springer, Berlin, 1996.

[5] Martin Dietzfelbinger and Friedhelm Meyer auf der Heide. A new universal class of hash functions and dynamic hashing in real time. In *Automata, languages and programming (Coventry, 1990)*, pages 6–19. Springer, New York, 1990.

[6] Edward A. Fox, Qi Fan Chen, and Lenwood S. Heath. A faster algorithm for constructing minimal perfect hash functions. In *Proceedings of the Fifteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Data Structures, pages 266–273, 1992.

[7] Edward A. Fox, Lenwood S. Heath, Qi Fan Chen, and Amjad M. Daoud. Practical minimal perfect hash functions for large databases. *Communications of the ACM*, 35(1):105–121, January 1992.

[8] Michael L. Fredman, János Komlós, and Endre Szemerédi. Storing a sparse table with $O(1)$ worst case access time. *J. Assoc. Comput. Mach.*, 31(3):538–544, 1984.

[9] George Havas and Bohdan S. Majewski. Graph-theoretic obstacles to perfect hashing. In *Proceedings of the Twenty-fourth Southeastern International Conference on Combinatorics, Graph Theory, and Computing (Boca Raton, FL, 1993)*, volume 98, pages 81–93, 1993.

[10] Kurt Mehlhorn. *Data structures and algorithms. 1.* Springer-Verlag, Berlin, 1984. Sorting and searching.

[11] Rasmus Pagh. Faster deterministic dictionaries. Manuscript, 1999.

[12] Jeanette P. Schmidt and Alan Siegel. The spatial complexity of oblivious k-probe hash functions. *SIAM J. Comput.*, 19(5):775–786, 1990.

[13] Robert Endre Tarjan and Andrew Chi Chih Yao. Storing a sparse table. *Communications of the ACM*, 22(11):606–611, November 1979.

Recent BRICS Report Series Publications

RS-99-13 **Rasmus Pagh.** ***Hash and Displace: Efficient Evaluation of Minimal Perfect Hash Functions.*** **May 1999. 11 pp. A short version to appear in *Algorithms and Data Structures: 6th International Workshop*, WADS '99 Proceedings, LNCS, 1999.**

RS-99-12 **Gerth Stølting Brodal, Rune B. Lyngsø, Christian N. S. Pedersen, and Jens Stoye.** ***Finding Maximal Pairs with Bounded Gap.*** **April 1999. 31 pp. To appear in *Combinatorial Pattern Matching: 10th Annual Symposium*, CPM '99 Proceedings, LNCS, 1999.**

RS-99-11 **Ulrich Kohlenbach.** ***On the Uniform Weak König's Lemma.*** **March 1999. 13 pp.**

RS-99-10 **Jon G. Riecke and Anders B. Sandholm.** ***A Relational Account of Call-by-Value Sequentiality.*** **March 1999. 51 pp. To appear in *Information and Computation*, LICS '97 Special Issue. Extended version of an article appearing in *Twelfth Annual IEEE Symposium on Logic in Computer Science*, LICS '97 Proceedings, 1997, pages 258–267. This report supersedes the earlier report BRICS RS-97-41.**

RS-99-9 **Claus Brabrand, Anders Møller, Anders B. Sandholm, and Michael I. Schwartzbach.** ***A Runtime System for Interactive Web Services.*** **March 1999. 21 pp. Appears in Mendelzon, editor, *Eighth International World Wide Web Conference*, WWW8 Proceedings, 1999, pages 313–323 and *Computer Networks*, 31:1391–1401, 1999.**

RS-99-8 **Klaus Havelund, Kim G. Larsen, and Arne Skou.** ***Formal Verification of a Power Controller Using the Real-Time Model Checker*** **UPPAAL. March 1999. 23 pp. To appear in Katoen, editor, *5th International AMAST Workshop on Real-Time and Probabilistic Systems*, ARTS '99 Proceedings, LNCS, 1999.**

RS-99-7 **Glynn Winskel.** ***Event Structures as Presheaves—Two Representation Theorems.*** **March 1999. 16 pp.**

RS-99-6 **Rune B. Lyngsø, Christian N. S. Pedersen, and Henrik Nielsen.** ***Measures on Hidden Markov Models.*** **February 1999. 27 pp. To appear in *Seventh International Conference on Intelligent Systems for Molecular Biology*, ISMB '99 Proceedings, 1999.**