

5X-10

Ordered minimal perfect
hashing function について

和田 雅美

(東京理科大学 理学部 応用数学科)

1. はじめに

これまで、キー番地変換の一手法として様々なハッシュ関数が提案されてきた [9]。しかし、それらは一般にキー (整数) の集合 $W = \{w_1, w_2, \dots, w_N\}$ から番地の集合 $I = \{0, 1, 2, \dots, M-1\}$ への多対1変換となるため、相異なるキーが同じ番地に変換される現象、いわゆる衝突、を生じるという問題点を持っている。従って衝突が起こった場合の処理についての研究もいろいろと行われて来たが、一方で、衝突の全く起こらない、キーから番地への1対1変換となるハッシュ関数を構成出来ないかという研究が、主に静的および探索と格納を含む準静的ファイルに対して行われてきた。そうしたハッシュ関数を、完全ハッシュ関数 (Perfect Hashing Function 以下 PHF) と呼び、その中でもキーの集合の大きさと番地の集合の大きさが等しい、つまりテーブルサイズ最小のものを、最小完全ハッシュ関数 (Minimal PHF 以下 MPH F)、その上、データの格納番地の順序がキーの順序に従い、そのシーケンシャル性を保つものを、順配置最小完全ハッシュ関数 (Ordered MPH F 以下 OMPHF) と呼ぶ。こうしたPHFの研究の中から、幾つかのPHFおよびその構成法が提案されてきた [1, 4, 5, 6, 7]。

Jaeschke [6] は、 $h(w) = \lfloor C/(Dw+E) \rfloor \bmod N$ なるMPHFを提案し、与えられたWに対し、hがMPHFになるような、整数C、D、そしてEを決めるアルゴリズムを与えた (なお、 $\lfloor \cdot \rfloor$ は切り捨てを表す)。井上 [8] は、Jaeschke の関数がOMPHFになるためのWに関する必要十分条件を求め、それを用いて、hがOMPHFとなるようなC、D、そしてEの決定アルゴリズムを与えた。Chang [1, 2, 3] は、 $h(w) = C \bmod p(w)$ なる素数関数pを用いたOMPHFを提案し、与えられたWに対し、hがOMPHFになるように、整数Cを決めるアルゴリズムを与えた。Chang の関数は、Jaeschke の関数に比較し、必ずOMPHFが求まると言う利点があるが、その素数関数の与え方や、整数Cの大きさがかなり大きくなることなど議論の残るところである。以上2つのハッシュ関数が、整数論の中国剰余定理の応用であるのに対し、Sprugnoli [7] は、違った観点から、 $h(w) = \lfloor (w+C)/D \rfloor$ なるPHFを提案し、整数C、Dを決めるアルゴリズムを与えた。Sprugnoli のそれは、データの値に偏りがある場合に、最小性が得られないという弱点はあるが、MPHFとならない場合でも、順配置の性質を持っており、(つまり、必らず Ordered PHFになる)、また先の2つと異なって、新しいデータの追加においてもそれが保持されるという点で、検討する価値はかなり大きいと思われる。

そこで我々は、ここでは、検討の第1段階として、Sprugnoli の関数が、OMPHFとなるためのWに関する必要十分条件を考察する。そして、その結果を用いて

OMPHFになる場合の整数C、Dの決定アルゴリズムを与える。また、OMPHFとならない場合に、Sprugnoli が導入したカッティングの手法を拡張、適用してみる。

2. 必要十分条件

Sprugnoli の関数 (Quotient reduction function) $h(w) = \lfloor (w+C)/D \rfloor$ $C \in \mathbb{Z}$ $D \in \mathbb{Z}^+$ は、キーの集合 $W = \{w_1, w_2, \dots, w_N\} \subset \mathbb{Z}$ (以降、考察を容易にするため $w_1 < w_2 < \dots < w_N$ を仮定する) から番地の集合 $I = \{0, 1, 2, \dots, M-1\}$ への関数で、その形から分かるように、データについてのシーケンシャル性を保持している。つまり、 $x \leq y$ に対し、 $h(x) \leq h(y)$ となる。

ここではまず、 $D_{\min}$ $D_{\max}$ を定義し、それを用いてSprugnoli の関数hがOMPHFになるための、Wに関する必要十分条件を与える。

【定義 1】

与えられたキーの集合 $W = \{w_1, w_2, \dots, w_N\}$ に対し、 $D_{\min}$ $D_{\max}$ を次のように定義する。

$$D_{\min} = \max_{j>i} \lceil (w_j - w_i + 1) / (j - i + 1) \rceil$$

$$D_{\max} = \min_{j>i+1} \lfloor (w_j - w_i - 1) / (j - i - 1) \rfloor$$

なお、 $\lceil \cdot \rceil$ は切り上げを表す。

【定理 2】

与えられたキーの集合 $W = \{w_1, w_2, \dots, w_N\}$ に対し、 $D_{\min} \leq D_{\max}$ である時、そしてその時のみ、以下のようなC、Dを選ぶことで、ハッシュ関数： $h(w) = \lfloor (w+C)/D \rfloor$ は、WについてOMPHFとなる。

$D : D_{\min} \leq D \leq D_{\max}$ なる整数、

$C : C_{\min} = \max_j \{(j-1) * D - w_j\}$

$C_{\max} = \min_j \{j * D - w_j - 1\}$ によって決まる

$C_{\min} \leq C \leq C_{\max}$ なる整数

3. アルゴリズム

ここでは、2. で与えられた必要十分条件を用いたOMPHF構成アルゴリズム (Algorithm OMPHF1) を与える。アルゴリズムは、キーN個とその個数Nをデータとして与え、それに対して関数 $h(w) = \lfloor (w+C)/D \rfloor$ がOMPHFになるような整数C、Dの対の1つ ($C_{\min}, D_{\min}$) を返す (line 18)。データが必要十分条件を満たさない場合は、アルゴリズムはそのメッセージを返して止まる (line 10)。

Algorithm OMPHF1

```
0 procedure OMPHF1(N, w1, w2, ..., wN)
1 if N=1 then return (1, -w1)
2 else begin
```

```

3  Dmin←1; Dmax←∞;
4  for j←2 until N do begin
5    dmin←1; dmax←∞;
6    for i←1 until j-1 do begin
7      dmin←max(dmin, ⌈(wj-wi+1)/(j-i+1)⌋);
8      if j>i+1 then
9        dmax←min(dmax, ⌊(wj-wi-1)/(j-i-1)⌋)
10     end;
11   if dmin>Dmax or dmax<Dmin then
12     return ('non-OMPHF')
13   else begin
14     Dmin←max(Dmin, dmin);
15     Dmax←min(Dmax, dmax)
16   end
17   Cmin←-w1;
18   for j←2 until N do
19     Cmin←max(Cmin, (j-1)*Dmin-wj);
20   return (Dmin, Cmin)
21 end

```

4. カッティング

与えられたキーの集合 $W = \{w_1, w_2, \dots, w_N\}$ においてキーに偏りがあった場合、Sprugnoli の関数 h は、OMPHF からかなり程遠くなってしまう。そこで、彼は、あるキー w_j を区切り点として、2つのハッシュ関数を接続して、全体でより OMPHF に近づくことを考え、この操作をカッティングと呼んだ [7]。つまり、次のような、整数 C_1 、 C_2 、そして D を決定するわけである。

$$h(w) = \lfloor (w+C_1)/D \rfloor, (w_1 \leq w \leq w_j)$$

$$h(w) = \lfloor (w+C_2)/D \rfloor, (w_j < w \leq w_N)$$

そこで、我々はここで、この操作を次のように複数の区切り点に拡張し、全体で OMPHF を構成するようなアルゴリズム (Algorithm OMPHF2) を与える。

$$h(w) = \lfloor (w+C_t)/D_t \rfloor, (a_{t-1} < w \leq a_t), (t = 1, 2, \dots, k)$$

アルゴリズムは、キー N 個とその個数 N をデータとして与え、 (a_t, D_t, C_t) の対を返してくる。

Algorithm OMPHF2

```

0 procedure OMPHF2(N, w1, w2, ..., wN)
1 if N=1 then return (w1, 1, -w1)
2 else begin
3   t←0;
4   while t<N do begin
5     s←t+2; Dmin←1; Dmax←∞;
6     for j←s until N do begin
7       dmin←1; dmax←∞;
8       for i←s-1 until j-1 do begin
9         dmin←max(dmin, ⌈(wj-wi+1)/(j-i+1)⌋);
10        if j>i+1 then
11          dmax←min(dmax, ⌊(wj-wi-1)/(j-i-1)⌋)
12      end;
13      if dmin>Dmax or dmax<Dmin then begin
14        t←j-1; goto *SKIP
15      end
16    else begin
17      Dmin←max(Dmin, dmin);
18      Dmax←min(Dmax, dmax)
19    end
20  end;
21  t←N;

```

```

20 *SKIP:
21   Cmin←(s-2)*Dmin-ws-1;
22   for j←s until t do
23     Cmin←max(Cmin, (j-1)*Dmin-wj);
24   print (wt, Dmin, Cmin);
25   if t=N-1 then begin
26     t←N; print (wN, 1, N-1-wN)
27   end
28 end

```

5. 例題

1) Sprugnoli の例

$$W = \{ 17, 138, 173, 294, 306, 472, 540, 551, 618 \}$$

$$h(w) = \lfloor (w-2)/76 \rfloor \quad (16 < w \leq 306)$$

$$h(w) = \lfloor (w-287)/37 \rfloor \quad (306 < w \leq 618)$$

2) 12ヶ月の英名の3、2文字目の ASCII コード

$$W = \{ 20033, 16965, 21057, 21072, 22849, 20053, 19541, 18261, 20549, 21571, 22095, 17221 \}$$

$$h(w) = \lfloor (w-16447)/774 \rfloor \quad (16964 < w \leq 20033)$$

$$h(w) = \lfloor (w-17512)/445 \rfloor \quad (20033 < w \leq 22849)$$

6. まとめ

我々は、Sprugnoli が提案した PHF が、OMPHF となるための必要十分条件を求め、カッティングを用いて、OMPHF を構成するアルゴリズムを与えた。

MPHF や OMPHF はあくまで静的で少ないデータに対してのみ構成が実現的ではあるが、一般の動的な大量データ管理に利用できる良いハッシュ関数の選択に対し、それら研究の成果の持つ価値は、大きいものと思われる。

参考文献

- (1) C.C.Chang: The study of an ordered minimal perfect hashing scheme. Comm. ACM 27,4 (April 1984), 384-387.
- (2) C.C.Chang: The study of a letter oriented minimal perfect hashing scheme. Proc. of the Int. Conf. on Foundations of Data Organization (May 1985), 61-65.
- (3) C.C.Chang and R.C.T.Lee: A letter-oriented minimal perfect hashing scheme. The Comput. J. 29, 3 (June 1986), 277-281.
- (4) R.J.Cichelli: Minimal perfect hash functions made simple. Comm. ACM 23,1 (Jan. 1980), 17-19.
- (5) S.P.Ghosh: Data Base Organization for Data Management. Academic Press, New York, 1977.
- (6) G.Jaeschke: Reciprocal hashing: A method for generating minimal perfect hashing functions. Comm. ACM 24,12 (Dec. 1981), 829-833.
- (7) R.Sprugnoli: Perfect hashing functions: A single probe retrieving method for static sets. Comm. ACM 20,11 (Nov. 1977), 841-850.
- (8) 井上久美子: A method for generating ordered minimal perfect hashing function. 修士論文, 東京理科大学 (March 1986), 1-12.
- (9) 西原清一: ハッシングの技法と応用. 情報処理 21, 9 (Sept. 1980), 980-991.