数理解析研究所講究録
296巻 1977年 146-153

自己組織的に構成される逐次検索表と Open Scatter Table

山口大 工学部　高浪 五男
中西 紫朗
藤井 宏久

1. 自己組織化逐次検索表

“最初から始めて，正しいキーが見つかるまで探し続けて停止する”という逐次検索は最も簡単な検索法である．多くのより複雑なアルゴリズムがこれに基づいているため，検索法を考察する際の有効な出発点である．

このアルゴリズムは次のように定式化される．

〔逐次検索アルゴリズム〕 K_1, K_2, ・・・, K_Nをそれぞれ対応するキーとするようなレコードのテーブル R_1, R_2, ・・・, R_Nが与えられたとき，キーKを探す．$N \geq 1$とする．

S1. Set $i \leftarrow 1$

S2. If $K = K_i$, アルゴリズムは成功して終る．

S3. Set $i \leftarrow i+1$

S4. If $i \leq N$, go back to S2. otherwise, アルゴリズムは不成功に終る』．

一般的にキー K_i が確率 p_i で生起するものとする．$p_1 + p_2 + \cdots + p_N = 1$ である．成功に終る検索の時間はキーの比較回数に大体比例し，その平均比較回数 $\overline{C_N}$ は

$$\overline{C_N} = p_1 + 2p_2 + \cdots + Np_N \qquad (1)$$

である．レコードをテーブルに任意の順に置くことができるならば，$p_1 \geq p_2 \geq \cdots \geq p_N$ のとき $\overline{C_N}$ は最小となる．

一方，多くの場合各レコードの生起確率は不明である．各レコードがアクセスされる回数をカウントしてゆき，このカウントに基づいてレコードを再配列する方法が考えられるが，カウント領域に多くの記憶スペースを割当てる必要がある．そこで，カウントすることなくレコードを自己組織的に再配列する方法について，従来次のようなアルゴリズム（始源は不明とされている）[1]が知られていた．

『レコードが成功して見つかったときはそのレコードをテーブルの先頭にもってくる』．

このアルゴリズムによると，レコードの平均比較回数は $1 + 2\sum_{1 \leq i < j \leq N} p_i p_j / (p_i + p_j)$ となることが知られている．[1]

我々は，前にもってくる割合（MTFR）によって自己組織的に形成されるテーブルに対する検索でキーの平均比較回数ならびにこれの試行回数にともなう収束性が異なるものと直観し，さらには平均比較回数が最小という意味での最適な

MTFR が存在するものと考えた．これらのことを数学的に求めるのはかなり困難なようであるので，ここでは計算機によるシミュレーションを行なった．シミュレーションを行なうには確率分布 $P_1, P_2, \cdots, P_N$ をどのように与えるかを定める必要がある．いろいろ考えられるが典型的分布として Zipf's law や 80－20 rule of thumb がある．これらは一般的につぎのように表わされる．

$$p_1 = c/1^{1-\theta},\ p = c/2^{1-\theta},\ \ldots,\ p = c/N^{1-\theta}.$$

$$1/c = 1/1^{1-\theta} + 1/2^{1-\theta} + \ldots + 1/N^{1-\theta} \equiv H_N^{(1-\theta)} \qquad (2)$$

$\theta = 0$ のときが Zipf's law，$\theta = \log 0.8 / \log 0.2 = 0.1386$ のときが 80－20 rule of thumb，$\theta = 1$ のときが等確率分布である．

MTFR γ による自己組織化は次のように行なわれる．レコード R がファイルの先頭から m 番目の所で見つかったとき，$m = 1$ なら変化なし，$m \geq 2$ なら「γm」だけ前にもってきて挿入する．この操作を $F(\gamma m) \leftarrow R$ と書く．

ファイルの先頭から第 i 番目のレコードを $R(i)$，その対応するキーを $K(i)$ とする．

〔MTFR γ による等長レコードからなる自己組織化ファイルの構成アルゴリズム S_γ〕

$S_\gamma 1$. If a record R with key K is sought, go to $S_\gamma 2$. Otherwise go to $S_\gamma 1$.

$S_\gamma 2$. Set $i \leftarrow 0$.

$S_\gamma 3$. Set $i \leftarrow i+1$.

$S_\gamma 4$. If $R(i)=\emptyset$, $R(i) \leftarrow R$ and go to $S_\gamma 1$.

$S_\gamma 5$. If $K \neq K(i)$, go to $S_\gamma 3$.

$S_\gamma 6$. $F(\gamma i) \leftarrow R(i)$, and go to $S_\gamma 1$. ⌟

$\gamma = 1$ の場合が従来の自己組織化ファイルの構成法であり，$\gamma = 0$ の場合が文献(2)の *transposition heuristics* である．式(2)の分布においては，式(1)による最小平均比較回数 $C_{opt}(N, \theta)$ は

$$C_{opt}(N, \theta) = H_N^{(1-\theta)} / H_N^{(1-\theta)} \qquad (3)$$

であり，特に $C_{opt}(N, 0) = N / H_N^{(1)}$．一方，アルゴリズム S_γ による平均比較回数を $C(\gamma, N, \theta)$ で表わすと，

$$C(1,N,0) = 1/2 + 1/H_N^{(1)} \left((2(N+1)H_{2N}^{(1)} - 2N - 2(N+1)H_N^{(1)} + 2N \right)$$

$$\approx 2N / \log_2 N$$

$$C(1,N,0) / C_{opt}(N,0) \approx 1.386$$

なることが知られている．

$C(\gamma, N, \theta)$ のシミュレーション結果の概略を Fig. 1 に示す．このことから，(1) $C(\gamma, N, \theta)$ は試行回数 T とともに $\gamma \approx 0.25$ を境にして γ がこれより小さいときは“オーバシュート”現象を起し，γ が 0.25 より大きいときは T とともにだ

らだらと増大する．(2) $C(\gamma, N, \theta)$の最小値を与える$\gamma=\gamma_{min}$が試行回数Tとともに次第に減少し，$T\to\infty$で$\gamma_{min}\to 0$となる傾向を示している．自己組織化の収束を早めるにはγをTとともに変化させるとよい．(3) 従来の先頭にもってくるアルゴリズム（$\gamma=1$ に相当）は最も良くない．

2. 自己組織化 Open Scatter Table

スキッター・テーブル手法（ハッシュ検索法）のうちで最も簡単な Linear Probing Open Scatter Table（以下LPSTと略す）法を1で述べた自己組織化を用いて構成し，その計算機実験の結果を示す．

LPST法ではキーKの検索開始番地は$h(K)$で与えられる．ここにhはハッシュ関数でその値域は$\{0, 1, \cdots, M-1\}$である．レコードの個数Nに対し，$N \leq M$であり，$\alpha = N/M$をロードファクタ，Mをテーブルのサイズといい，これにレコードが格納される．検索は$h(K)$から逐次検索で行なわれ，キーKが見つかるか（検索成功），空番地が見つかる（検索不成功）まで続けられる．その結果サイズMの1次元テーブルは空番地によって複数個の線形リストに分割され，これらの各々はキーによって異なりうる開始番地をもっている．各々の線形リストに1と同様の手法により自己組織化を行なう．

テーブルの第i番地の内容を$T(i)$とする．

〔MTFR γによる自己組織化LPSTの構成アルゴリズム〕

$H_\gamma 1$. If a record R with K is sought, go to $H_\gamma 2$. Otherwise go to $H_\gamma 1$.

$H_\gamma 2$. $i \leftarrow h(K)$, $c \leftarrow 0$.

$H_\gamma 3$. $c \leftarrow c+1$. If $T(i)=\emptyset$, $T(i) \leftarrow R$ and go to $H_\gamma 1$.

$H_\gamma 4$. If $T(i) \neq K$, $i \leftarrow i-1 \bmod M$ and go to $H_\gamma 3$.

$H_\gamma 5$. Move R to front by γc and go to $H_\gamma 1$. ⌟

自己組織化を行なわず通常のやり方で確率の大きい順にキーをテーブルに挿入すれば最適なテーブルが得られ，逆順に挿入すると最悪のテーブルが得られる．シミュレーションで用いるレコードの生起確率は式 (2) によるものとする。ハッシュ関数は一般には算術関数が用いられているが，その種類もいろいろあり，その性能についても一長一短があるので，ここでは次のように定めた。まず，一様乱数を用いて次のテーブル T_1 を作る．

	1	2	·····	k	·····	M-1	M
T_1 :	i_1	i_2	·····	i_k	·····	i_{M-1}	i_M

$T_1(k)=i_k$ と書く．$i_1\, i_2 \cdots i_M$ は $1\, 2 \cdots M$ の置換になっている。ついで，正規乱数を用いて次のテーブル T_2 を作る。

	1	2	·····	N-1	N
T_2 :	i_1	i_2	·····	i_{N-1}	i_N

$T_2(\ell) = i_\ell$ と書くと，この乱数の第 ℓ 回目（$1 \leq \ell \leq N$）に k（$1 \leq k \leq M$）が発生したとき $T_2(\ell) = T_1(k) = i_k$ とする．

例えば，$M = 5$，$N = 4$ で

	1	2	3	4	5
T :	2	1	5	4	3

で，正規乱数により，２３２４の順で乱数が発生したとき

	1	2	3	4
T :	1	5	1	4

T_2 をハッシュ関数表として用いる．上の例では

$h(1)=1$，$h(2)=5$，$h(3)=1$，$h(4)=4$．

キー１と３でコリジョンが生じている。

シミュレーション結果の一部は Fig.2 の通りである。このことから (1) 通常の方法に対して約７％程度の改良がなされる。(2) Fig.2b からわかるように，確率の変動に対しそれに追随する（自己組織化効果）特性が顕著に見られる。

最後に，このような特性を示す自己組織化の数学的解析が望まれる．

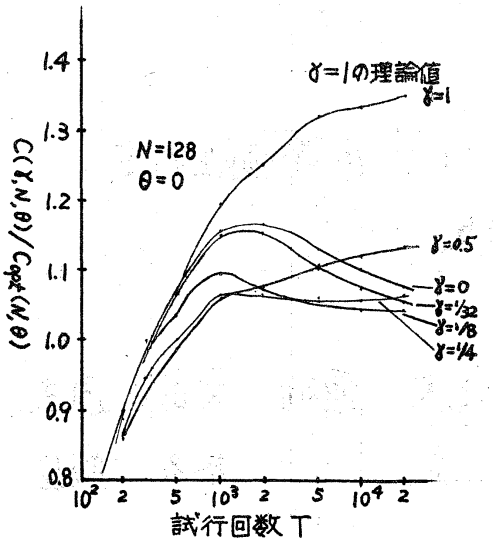
文献

(1) D.E.Knuth; The Art of Computer Programming, Vol.3, pp506-542

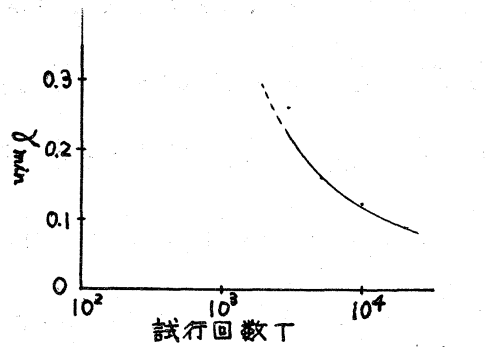
(2) R.L.Rivest; On Self-organizing sequential Search Heuristics, C.A.C.M. Feb. 1976.

(3) 高浪，藤井；逐次検索における自己組織化ファイルの発見的構成，信学論誌 D掲載予定

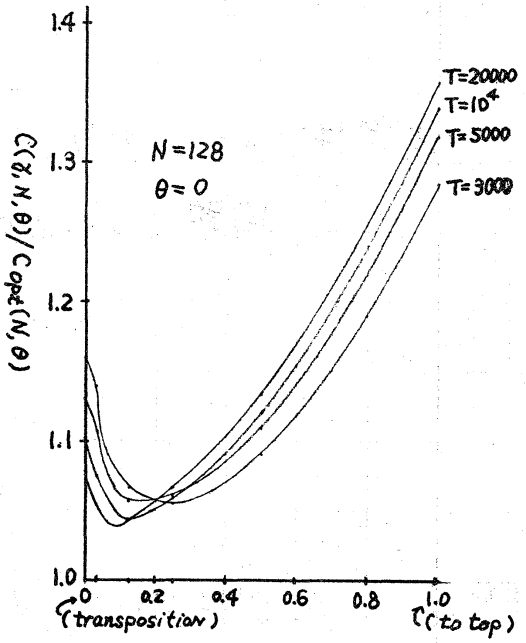
(4) 中西，高浪；自己組織的に構成される Open Scatter Table の効率，信学EC研究会 1976,12



(a) 自己組織化の収束性

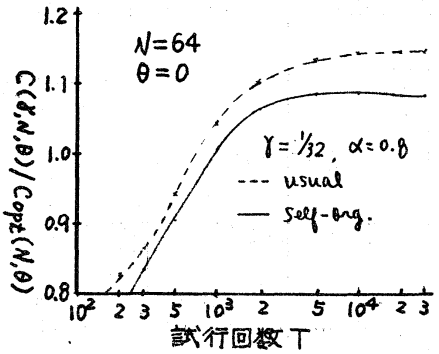


(c) 最小比較回数を与えるγの変化

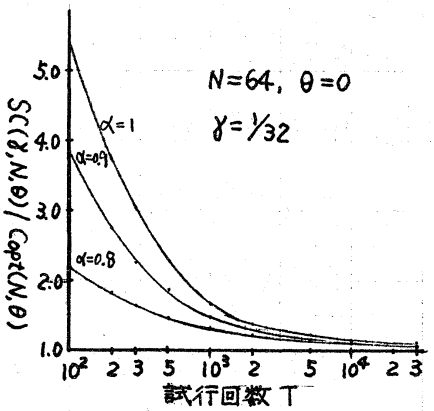


(b) γに対する比較回数の変化

Fig.1 自己組織化逐次検索表



(a) 自己組織化の収束性



(b) 最悪配列のテーブルに
自己組織化を行なったときの収束性

Fig.2 自己組織化LPST