論　文

クラスタ分析によるセグメント編成*

西　垣　通** 野木秀子** 宮本伸也***

Abstract

An efficient Segments Organization plays an important role in the design of high performance data base.

This parer describes a new method to organize a lot of data-items into proper segments which diminishes data access overhead remarkably. In the process of organization, Cluster Analysis which is widely used in the field of multivariate analysis, is employed to cluster data-items.

Simulation results showed this method reduced data access overhead 35～45%.

1. まえがき

データ・ベース関連技術の発展に伴い，膨大な件数のデータを蓄積，検索するための技術が開発されつつある．特に，従来各業務ごとに存在した個別専用ファイル群を統合して共通ファイル・システムを設計し，これにデータ・ベースとしての機能を持たせることが一般的に行われている．この際，個別専用ファイル群から抽出された多種類のアイテム群に対するデータ構造の決定と，それらを効率的に補助記憶媒体に蓄積することが，設計の中心課題となる．特に，データの蓄積構造の決定は，データ・ベースの性能を決定する最大の問題点である．

蓄積構造決定の過程は，2段階に分けることができる．

第1は，データの理論的最小単位であるアイテムからセグメントを編成する「セグメント編成」の過程であり，第2は，セグメントを記憶媒体上に配置し，データを物理的実体に対応させる「マッピング」の過程である．

ここで，セグメントとは，記憶媒体の性質によらず，常に1つの集合単位として蓄積され，かつ検索される理論的な単位のことであり，その編成はマッピングとともにデータ・ベースの性能に大きな影響を与える．

一般に，セグメントの検索に伴うオーバヘッドを減らすためには，1つのアプリケーション・プログラムにより同時に参照されるアイテムは極力1つのセグメント内に存在することが望ましい．反面，共通ファイルは複数のアプリケーション・プログラムで使用されるものであるから，参照される多くのアイテムを1つのセグメントに併合すると，アイテム参照における冗長度が高くなる．したがって，セグメントを編成する場合，両者の得失を考慮する必要がある．

このような背景のもとに，本報告では，共通ファイル・システムの設計時におけるセグメント編成手法，ならびにその実験結果についてのべる．アイテム群を複数のセグメントに分類するために，クラスタ分析を用いたことが本提案の特徴である．適用実験の対象としては，34種類のアプリケーション・プログラム群により参照される，29種類，1,000,000件のアイテム群から構成される共通ファイル・システムを採択した．

セグメント編成は，通常経験的に行われることが多いが，記憶媒体の物理的条件を考慮した効率的蓄積構造決定に関しては，いくつかの報告がある．たとえば，データの検索時間やアクセス回数等の統計情報をもとに，データ・ベースの蓄積構造を決定した例がある[5]．また，記憶媒体として磁気ディスクを用いる場合，データの検索効率を上げるための手法，ならびに解析に関する報告もある[6)-9)]．これらの報告と比較す

* Segments Organization by Cluster Analysis by Tohru NISHIGAKI, Hideko NOGI (Systems Development Laboratory, Hitachi, Ltd.) and Shinya MIYAMOTO (Systems Engineering Division, Hitach, Ltd.).
** (株)日立製作所システム開発研究所
*** (株)日立製作所システム技術本部

ると，本報告で提案する手法は，記憶媒体の物理的条件に依存する部分が小さく，共通ファイル・システム設計において広く応用が可能である．

2. データ・ベースのセグメント編成

まず，本報告で用いる用語を以下に定義する．

(1) アイテム：データの論理的最小単位．セグメントの構成要素となる．

(2) セグメント：データの論理構造を蓄積構造にマッピングする際の最小単位．データの蓄積および検索の最小単位でもある．一般に複数のアイテムから構成される．

(3) アプリケーション：一般に複数のアイテムを同時に参照するプロセス．

達成すべき目的は，アプリケーション群によるアイテム検索の効率を最良にするようなセグメントを編成することにある．以下，このための定式化を行い，さらに検索効率の良さを表わす評価式を設定する．

いま，共通ファイル・システムを参照する r 種類のアプリケーションと，その参照アイテム群とが与えられたとする．

アプリケーションの集合G：

$$G=\{G_1, G_2, \cdots, G_p, \cdots, G_r\}. \tag{1}$$

アプリケーション G_p の参照するアイテムの集合を I_{G_p} とかくと，アイテムの集合Iは式(2)により与えられる．

$$I=I_{G_1}\cup I_{G_2}\cup\cdots\cup I_{G_p}\cup\cdots\cup I_{G_r}. \tag{2}$$

この結果，Iが n 種類のアイテム群から構成されたとする．

$$I=\{\alpha_1, \alpha_2, \cdots, \alpha_q, \cdots, \alpha_n\}. \tag{3}$$

アプリケーションと，その参照するアイテムとの関係は，参照マトリックス R: $\{r_{pq}\}$ により表わすことができる．

$$r_{pq}=\begin{cases}1: G_p\text{で}\alpha_q\text{を参照する．}\\ 0: G_p\text{で}\alpha_q\text{を参照しない．}\end{cases}\quad\begin{pmatrix}p=1,2,\cdots,r\\ q=1,2,\cdots,n\end{pmatrix} \tag{4}$$

いま，Iを適当な部分集合に分割し，m 種のセグメント s_k $(k=1,2,\cdots,m)$ を編成するとする．

セグメントの集合S：

$$S=\{s_1, s_2, \cdots, s_k, \cdots, s_m\} \tag{5}$$

セグメント s_k を編成するアイテムの集合を I_{S_k} とかく．このとき，式(6)(7)を仮定する．

$$I_{S_1}\cup I_{S_2}\cup\cdots\cup I_{S_k}\cup\cdots I_{S_m}=I \tag{6}$$

$$I_{S_u}\cap I_{S_v}=\phi\ (u\neq v;\ u,v=1,2,\cdots,m) \tag{7}$$

すなわち，アイテムは必ずいずれかのセグメントに属し，また2つ以上のセグメントに属すことはないとする．

このとき，α_q のアイテム長を l_q とかくと，s_k のセグメント長 L_k は式(8)で定められる．

$$\left.\begin{aligned}&L_k\triangleq\sum_{q=1}^{n} l_q\cdot\delta_{qk}\qquad(k=1,2,\cdots,m)\\ &\text{ただし}\quad\delta_{qk}=\begin{cases}1: \alpha_q\in I_{S_k}\\ 0: \alpha_q\notin I_{S_k}\end{cases}\end{aligned}\right\} \tag{8}$$

次に，セグメント集合Sの検索効率の良さを表わす評価式を式(9)により設定する．

$$\left.\begin{aligned}&\phi\triangleq\sum_{p=1}^{r}\left[w_p\cdot\sum_{k=1}^{m}\{(\Delta T(p,k)+\Delta A(p,k))\cdot\delta_{kp}\}\right]\\ &\text{ただし}\quad\delta_{kp}=\begin{cases}1: I_{S_k}\cap I_{G_p}\neq\phi\\ 0: I_{S_k}\cap I_{G_p}=\phi\end{cases}\end{aligned}\right\} \tag{9}$$

式(9)において，w_p はアプリケーション G_p の重みであり，処理頻度に対応する．$\Delta A(p,k)$ は G_p が s_k をアクセスするためのオーバヘッドを表わす．また，$\Delta T(p,k)$ は G_p が s_k を処理可能な記憶域と蓄積記憶域との間で転送するためのオーバヘッドを表わす．したがって，$(\Delta T(p,k)+\Delta A(p,k))$ は，G_p が s_k を検索するためのオーバヘッドに対応する．なお，式(9)において，セグメントの種類mも変数である．セグメント長 L_k と m との間には次の関係が成り立つ．

$$\max_k L_k\geq\frac{1}{m}\sum_{q=1}^{n} l_q. \tag{10}$$

$\Delta T(p,k)$ は L_k の増加関数であり，したがって陰に m の関数である．

結局，我々の目的は，アプリケーション集合Gとアイテム集合Iとの関係が参照マトリックスRにより与えられたとき，式(9)で定義される ϕ を最小にするセグメント集合Sの編成に帰着される．

3. クラスタ分析によるセグメント最適編成

3.1 クラスタ分析手法

G, I, R が与えられたとき，ΔT および ΔA を定め，$\phi\rightarrow\min$ をみたす最適なSを，直接，たとえば整数計画法により求めることは，アイテムの種類nが大きい場合，非実用的である．本報告では，多変量解析の分野で用いられるクラスタ分析を利用して，ϕ の小さいSを求める方式を採用した．

クラスタ分析は，ある変量で特徴づけられた複数の個体を，一定の算法にしたがって分類する手法である．本報告で用いたものは，特に階層的クラスタ分析

手法とよばれ，互いに近い距離にある複数のクラスタを同一のクラスタに併合するという過程の反復により，クラスタが形成される．

階層的クラスタ分析手法は，次の3つが与えられれば，完全に定まる．

(1) クラスタ間の距離の初期値である個体間の距離．

(2) クラスタ間の距離の定義．

(3) 複数のクラスタから新たなクラスタを作る反復過程におけるクラスタリング・アルゴリズム．

本報告では，上記(1)(2)(3)として以下の方式を採用した．

(1) アイテム α_i と α_j との距離 d_{ij}:

アプリケーションによる α_i, α_j の共通同時参照頻度の逆数

(2) クラスタ C_x と C_y との距離 D_{xy}:

$\min_{\alpha_i \in C_x, \alpha_j \in C^4} d_{ij}$. 本定義は，最近隣 (Nearest Neighbor) とよばれる．

(3) クラスタリング・アルゴリズム:

Single Linkage 法および階層的モード法．

ここで，Single Linkage 法は最も基本的な方法であり，クラスタの形式は $\min_{x,y} D_{xy}$ をみたす2つのクラスタの融合過程の反復により行われる．また，階層的モード法は，Single Linkage 法に改良を加えた方法であり，この場合，各点（個体）は密な点と疎な点に分けられ，まず密な点についてクラスタが作られた後，疎な点が併合されていく（詳細なアルゴリズムについては，文献 2),3) を参照されたい）．

上記のようなクラスタ分析手法により，ϕ の小さなSを求める基本的考え方は次の通りである．

クラスタリングを実行した結果，アプリケーションにより同時に参照される頻度の多いアイテム同志は，同一のクラスタに属するように分類される．各セグメントは，基本的に，それぞれ同一のクラスタに属するアイテム群から編成される．ところで，アプリケーション G_p により同時に参照されるセグメント s_k は，k が次の条件をみたすものである．

$$k \ s.t. \ I_{S_k} \cap I_{G_p} \neq \phi$$

このようなセグメントの個数を K_p とかくと，クラスタリングの効果として，$\sum_{p=1}^{r} w_p \cdot K_p$ は小さくなる．したがって，式(9)において ϕ の減少効果が期待できる．

3.2 セグメント編成の実施

クラスタ分析によるセグメント編成の具体的手順を以下にのべる．

(1) 参照マトリックスRの作成

業務分析を実行し，アプリケーション集合G: $\{G_1, G_2, \cdots, G_r\}$ と，その参照アイテム集合 $I_{G_1}, I_{G_2}, \cdots, I_{G_r}$ とから，共通ファイル・システムを構成するアイテム集合I: $\{\alpha_1, \alpha_2, \cdots, \alpha_n\}$ を求める．さらにGによるIの参照関係を表わす参照マトリックスRを作成する．

(2) 関連マトリックスF，距離マトリックスDの作成

参照マトリックスRにもとづき，アイテム相互の関連度を表わす関連マトリックスF: $\{f_{ij}\}$ を作成する（**Fig. 1** 参照）．

ここで，f_{ij} はアイテム α_i と α_j との関連度を表わし，式(11)により定められる．

$$\left.\begin{aligned} & f_{ij} = f_{ji} \triangleq \sum_{p=1}^{r} w_p \cdot \delta_{pij} \quad (i, j = 1, 2, \cdots, n) \\ & \text{ただし} \quad \delta_{pij} = \begin{cases} 1: \ \alpha_i, \alpha_j \in I_{G_p} \\ 0: \ \alpha_i \notin I_{G_p} \text{ or } \alpha_j \notin I_{G_p} \end{cases} \end{aligned}\right\} \quad (11)$$

式(11)において，アプリケーション G_p の重み w_p を次のように与える．

(a) G_p がバッチ・プログラムの場合

w_p: (単作時間あたりの平均実行頻度)×(1回実行あたりの平均処理件数)

(b) G_p がリアルタイム・プログラムの場合

w_p: 単位時間あたりのトランザクション平均処理件数

上記のように作成した関連マトリックスFを，距離マトリックス D: $\{d_{ij}\}$ におきかえる．

$$d_{ij} = c/f_{ij} \quad (c: \text{定数}; \ i, j = 1, 2, \cdots, n) \qquad (12)$$

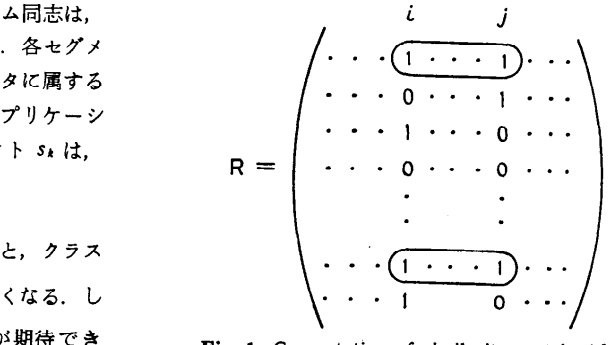

Fig. 1 Computation of similarity matrix $\{f_{ij}\}$ based on reference matrix R

ただし，式(12)において，$f_{ij}=0$ すなわち関連度ゼロのときは，d_{ij} を∞とする．

(3) クラスタ分析実行

Dにもとづき，**3.1** でのべたアルゴリズムにしたがってアイテム群のクラスタリングを実行する．階層的クラスタ分析手法の出力は，樹形図(dendrogram) のかたちで得られる．樹形図は，横軸にアイテムを並べ，縦軸にアイテム間ないしアイテム群からなるクラスタ間の距離を表わしたものである．

ただし，階層的クラスタ分析においては，すべてのアイテムは最終的にはひとつのクラスタに融合される．したがって，分類のためには，樹形図を適当に切断することが必要となる．

(4) セグメントの編成

樹形図を縦軸上の適当な高さで横に切断したとき，そこまでで互いに接続しているアイテムのグループがクラスタである．切断する高さは，編成するセグメントの数 m によって異なる．

各々のクラスタに含まれるアイテム群をまとめて，それぞれセグメントとして編成する．

4. 適用実験例

4.1 適用の対象としたシステム

3. でのべた手法を実際のアプリケーション・システムに適用し，セグメント編成の実験を行った．対象としたシステムについて以下にのべる．

(1) アプリケーション集合G

$$G=\{G_1, G_2, \cdots, G_{34}\}$$

G_p $(p=1,2,\cdots,34)$ は，対象システムの日常業務に対応するものであり，データの検索，抽出，更新，簡単な加工などがその機能である．処理形態は，すべてバッチである．

(2) アイテム集合I

$$I=\{\alpha_1, \alpha_2, \cdots, \alpha_{29}\}$$

α_q $(q=1,2,\cdots,29)$ のアイテム長 l_q は，1～30バイトであり，合計長 $\sum_{q=1}^{29} l_q$ は 178 バイトである．また，アイテムの件数は約1,000,000件である．

一般に，Gは何らかのキー・アイテムを用いてIを参照するわけであるが，本アプリケーションでは 15 種類のアイテムがキー・アイテムとして使用される．

なお，アイテムの種類 n の比較的少ないシステムを選んだのは，クラスタリング結果の妥当性を直観的に確かめやすくするためである．アイテムの種類 n は少ない反面，その件数はかなり膨大であり，これらのデータを検索効率よく記憶媒体に蓄積することが必要とされる．

4.2 セグメント編成と蓄積構造の決定

適用対象システムの業務分析を行い，G,Iの関係を表わす参照マトリックスRを作成した．次に式(11)にもとづいて関連マトリックスFの要素を計算し，さらに距離マトリックスDを求めた．Dを **Fig. 2**(次頁参照) に示す (Fig. 2 では，見やすくするため，$\log d_{ij}$ を出力してある)．なお，式(12)における定数 c は 10^6 と定めた．

Dにもとづき，Single Linkage 法および階層的モード法によるクラスタリングを実行した．階層的モード法における density threshold としては2を採用した．

Fig. 3 (次頁参照)は，Single Linkage 法による結果の樹形図であるが，階層的モード法による結果もほぼ同様であった．その原因としては，クラスタの「核」がひとつ*になったため，Single Linkage 法の欠点である鎖効果が明確に表われなかった点を挙げることとができる．

クラスタ分析の結果として得られた樹形図を切断して，セグメントを編成する．樹形図より，ただちに $\psi\rightarrow\min$ をみたすSを求めることはできない．今回の実験では，まず樹形図を2分割することにより，**Table 1** に示す S_A, S_B 2つのセグメント編成を試行した．両者の比較ならびに3分割以上による編成との比較については，**5.** にのべる．

以上でセグメント編成が完了する．編成したセグメントの検索効率を表わす評価式(9)において，$\Delta T(p,k)$ および $\Delta A(p,k)$ の具体的数値を求めるためには，データの蓄積構造が決定されねばならない．本実験例では，複雑なファイル管理プログラムによるオーバヘッドの影響を最小限に抑えるため，以下のような比較的基本的なファイル編成法ならびにアクセス法を採択した．

Table 1 Examples of segments organized by Cluster Analysis.

Organization \ Segment	number of items belong to s_1	number of items belong to s_2
S_A	14	15
S_B	21	8

* Nearest Neighbor では，こうなりやすい．

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
2	4.5																												
3	4.7	3.9																											
4	4.5	3.7	3.9																										
5	4.5	3.6	3.9	3.7																									
6	∞	5.0	5.0	5.0	5.0																								
7	5.0	3 9	4.0	4.0	4.0	4.7																							
8	4.7	3.7	3.9	3.8	3.7	5.0	3.9																						
9	4.7	3.7	3.9	3.8	3.7	5.0	3.9	3.6																					
10	4.7	3.7	3.9	3.7	3.7	5.0	4.0	3.6	3.5																				
11	5.0	4.1	4.7	4.3	4.2	∞	4.7	4.0	4.0	4.0																			
12	∞	∞	∞	5.0	∞	∞	∞	∞	∞	5.0	5.0																		
13	∞	∞	∞	5.0	∞	∞	∞	∞	∞	5.0	5.0	5.0																	
14	∞	5.0	∞	∞	5.0	∞	∞	5.0	5.0	5.0	∞	∞	∞																
15	4.7	3.7	4.0	3.9	3.8	∞	4.0	3.7	3.7	3.7	4.1	∞	∞	∞															
16	∞	4.5	∞	∞	4.5	∞	5.0	4.5	4.5	4.7	4.5	∞	∞	∞	4.5														
17	∞	5.0	5.0	5.0	5.0	∞	∞	5.0	5.0	4.7	5.0	∞	∞	∞	5.0	∞													
18	∞	∞	∞	∞	∞	∞	∞	∞	∞	5.0	∞	∞	∞	∞	∞	∞	5.0												
19	∞	∞	∞	∞	∞	∞	∞	4.7	4.7	4.7	∞	∞	∞	∞	5.0	∞	∞	∞											
20	∞	5.0	∞	∞	∞	∞	5.0	5.0	4.7	4.7	5.0	∞	∞	∞	5.0	∞	∞	∞	∞										
21	∞	5.0	∞	∞	5.0	∞	5.0	5.0	4.7	5.0	5.0	∞	∞	∞	5.0	5.0	∞	∞	∞	∞									
22	∞	∞	∞	∞	∞	∞	∞	4.7	4.7	4.7	∞	∞	∞	∞	5.0	∞	∞	∞	∞	∞	∞								
23	∞	∞	∞	∞	∞	∞	∞	5.0	5.0	5.0	∞	∞	∞	∞	5.0	∞	∞	∞	∞	∞	∞	∞							
24	∞	4.7	4.7	4.7	4.7	∞	∞	4.7	4.7	4.7	4.7	∞	∞	∞	4.7	∞	5.0	∞	∞	∞	∞	∞	∞						
25	∞	4.7	∞	4.7	4.7	∞	∞	4.7	4.7	4.7	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞					
26	∞	4.7	∞	4.7	4.7	∞	∞	4.7	4.7	4.7	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	4.7				
27	∞	4.7	∞	4.7	4.7	∞	∞	4.7	4.7	4.7	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	4.7	4.7			
28	5.0	5.0	5.0	5.0	5.0	∞	5.0	5.0	5.0	5.0	∞	∞	∞	∞	5.0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞		
29	∞	∞	∞	∞	∞	∞	∞	∞	4.7	4.7	∞	∞	∞	∞	∞	∞	∞	∞	∞	5.0	5.0	∞	∞	∞	∞	∞	∞	∞	

Fig. 2 Dissimilarity matrix D: the degree of similarity between items

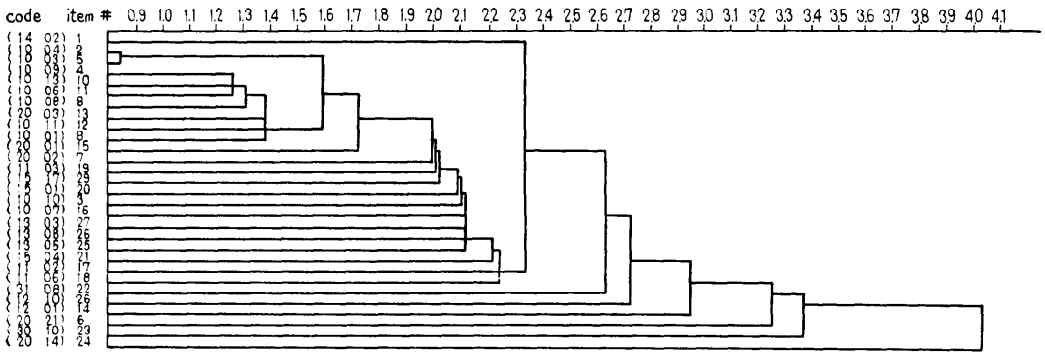

Fig. 3 Dendrogram example: The result of clustering items by Single Linkage Method

(1) 記憶媒体としては，大容量磁気ディスク(100 Mバイト/スピンドル) とする.

(2) アプリケーション G_1〜G_{34} において，キーとして使用される頻度の最も高いアイテムをプライマリ・キーとする，索引付順編成 (Indexed Sequential Organization) を，ファイル編成法として採択する.

(3) プライマリ・キー以外のアイテムをキーとするアプリケーションにおいては，必要に応じて2次インデクス (Secondary Index) を設け，マルチ・キーによるアクセスを可能にする.

5. 適用効果の測定

5.1 評価式 ϕ による効果測定

クラスタ分析結果にもとづいて編成したセグメント集合 S_A, S_B について，その検索効率を表わす評価式 ϕ の値を求めた.

評価式(9)において，$\Delta T(p,k)$ としては，1セグメント当たりの，ディスク・トラックと主記憶との間の平均転送時間を採用した. $\Delta A(p,k)$ としては，2次イ

ンデクスならびに ISAM (Indexed Sequential Access Method) を用いる平均アクセス時間を採用した．$\Delta A(p,k)$ は，プライマリ・キーによるシーケンシャル・アクセス，プライマリ・キーによるランダム・アクセス，2次インデクスを使用するランダム・アクセスなど，場合に応じて様々の値をとる．

Table 2 は，評価式(9)による φ の値の比較結果である．Table 2 において，ϕ_A は S_A に対応し，ϕ_B は S_B に対応する．ϕ_0, ϕ_0' は，ともにクラスタ分析結果によらず，無作為にアイテム集合Iを2分割し，各々をセグメントに編成した S_0, S_0' の評価値である．

クラスタ分析を用いると，無作為に編成した場合に比べ，約 35～45% ϕ の値が小さくなっていることがわかる．また，ϕ_A は ϕ_B に比べ約6%小さいので，本実験例では S_A の方が S_B より検索効果の良いセグメント編成であることがわかる．

次にセグメント数 $m=1$ および $m\geq 3$ と，上記 S_A $(m=2)$ との比較を行った．**Fig. 4** において，$m=1$ はすべてのアイテムを1セグメントに編成した場合であり，$m\geq 3$ は樹形図を多分割して編成した場合である．本実験では，$m=2$ が最小値を与えることがわかる．この理由として，$m=1$ のときは転送オーバヘッド ΔT の増大を，$m\geq 3$ のときはアクセス・オーバヘッド ΔA の増大をあげることができる．

上記のごとく，S_A は $\phi\rightarrow\min$ をみたす最適解ではないとしても，これに近い値を与えるセグメント集合と考えることができる．

5.2 シミュレーションによる効果測定

共通ファイル・システムの検索効率を表わす評価式として，式(9)を設定することにより議論を進めてきた．評価式(9)の妥当性を確かめ，かつクラスタ分析によるセグメント編成の効果を．より現実的な環境のもとで検証するため，シミュレーションを実行した．

Table 2 Performance comparison estimated by evaluation value ϕ, in the case $m=2$.

ϕ_0/ϕ_A	ϕ_0'/ϕ_A	ϕ_B/ϕ_A
1.86	1.61	1.06

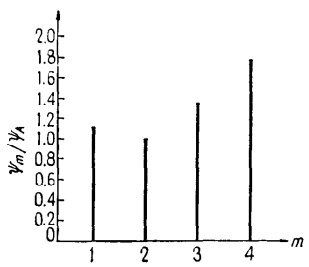

Fig. 4 Variation of performance with number of segments m.

Table 3 Performance comparison estimated by simulated I/O total time, in the case $m=2$.

T_0/T_A	T_0'/T_A	T_B/T_A
1.71	1.55	1.02

Table 3 は，アプリケーション集合Gについて，シミュレーションより求めた共通ファイル検索時間累計を比較した結果を示す．Table 3 において，T_A, T_B, T_0, T_0' は，それぞれセグメント集合 S_A, S_B, S_0, S_0' の場合の共通ファイル検索時間累計である．

Table 2 と Table 3 とを比較すると，評価式(9)により，共通ファイル・システムの検索効率を，かなりの精度で表現できることがわかる．

すなわち，クラスタ分析により編成した S_A や S_B は，これによらず無作為に編成した S_0 や S_0' に比べ，検索効率のよいセグメント集合であることが，シミュレーション結果からも明らかにされた．

6. むすび

業務別個別専用ファイル群を統合してデータ・ベース・システムの共通ファイルを設計する際，検索効率の良い蓄積構造を得るため，蓄積および検索の論理的最小単位であるセグメントの編成をクラスタ分析により行う手法を提案した．

実際のアプリケーション・システムを対象にして，本手法を実験的に適用した結果，クラスタ分析によるセグメント編成を行った共通ファイル・システムの検索効率は，無作為に編成した場合に比べ 35～45% 向上することが明らかになった．

多くの場合，経験と試行錯誤により行われる蓄積構造の設計において，本手法は有用な手段となりうると考えられる．

問題点としては，本報告で用いたクラスタ分析手法は，クラスタの分離が難しいという欠点がある．数多くあるクラスタ分析手法のうちで，アイテム群の分割手段としてどれが最適かは今後解決すべき問題である．

終りに，クラスタ分析プログラム適用に関して助言を与えて下さった当社中央研究所津田主任研究員，同システム開発研究所益田主任研究員に深く感謝いたします．

参 考 文 献

1) 西垣，井上，宮本：クラスタ分析によるデータ編成，昭 48 情報処理学会大会，講演番号 215.
2) 奥野忠一他：多変量解析法，日科技連 (1971).
3) Wishart, D.: Numerical Classification Method for Deriving Natural Classes, Nature, Vol. 221, No. 5175, pp. 97～98 (1968).
4) 益田隆司他：仮想メモリ向きの最適プログラム構成方式と実験，情報処理，Vol. 15, No. 9, pp. 662～669 (1974).
5) 窪田八州洋他：データベースの実際(1)，造船業における生産管理用データベースの一例，情報処理，Vol. 14, No. 7, pp. 522～530 (1973).
6) 近藤秀文他：推移行列を用いたディスク・デバイスへのファイル実体配置法，昭 48 情報処理学会大会，講演番号 221.
7) 近藤秀文他：大容量ディスクへのファイルの適正配置法，昭 49 情報処理学会大会，講演番号 265.
8) Margan, H. L.: Optimal Space Allocation on Disk Storage Devices, Comm. ACM, Vol. 17, No. 3. pp. 139～142 (1974).
9) Teorey, T. J., et al.: A Comparative Analysi of Disk Scheduling Policies, Comm. ACM, Vol. 15, No. 3, pp. 177～184 (1972).
10) CODASYL Systems Committee: Future Analysis of Generalized Data Base Management Systems, May 1971.

(昭和 50 年 4 月 2 日受付)
(昭和 50 年 10 月 22 日再受付)