

《論文》

仮想メモリシステム向きの最適プログラム構成方式と実験*

益田 隆 司** 塩田 博 行**

Abstract

It is known that program reference patterns exert a deep influence upon paging performance in virtual storage systems.

This paper describes new techniques which achieve good locality of reference of a program under virtual storage systems by rearranging the program. A cluster analysis technique, which is widely used in the field of multivariate analysis, is employed to cluster relocatable program units. Also employed is the working set approach for the definition of the strength of connection between program units.

Experimental results show that these techniques have a remarkable effect on the reduction of the working set size, compared with methods already reported.

1. まえがき

仮想メモリ方式を採用した計算機システムが普及しつつある。仮想メモリシステムは実メモリシステムに比較し、機能的に新しい点をいくつも含むが、この中でも最大の利点は、いうまでもなく、主メモリ・サイズよりも大きな領域を、あたかもそれだけ大きな主メモリが存在するかのような感覚で使用できることである。しかしながら、逆にこの機能は性能に大きな影響を与える要因を含んでいる。

その第1は、主メモリと2次メモリの間の情報のやりとりの管理を、ページ単位にハードウェア、オペレーティング・システムが一括して行なうことによる。システムが効率よく動作するためには、仮想メモリ上での使用ページ数と主メモリ・ページ数の間のバランスがとれていなければならない。また、主メモリと2次メモリの間のページ出し入れのアルゴリズムも性能に大きな影響を及ぼす。これらに関しては従来数多く

の報告がある^{5),6),7)}。

第2の要因は、各命令実行ごとの論理アドレスから実アドレスへのアドレス変換である。この変換は主メモリ上(または仮想メモリ上)に存在する数段階のアドレス変換テーブルを参照すれば可能であるが、各命令実行ごとにこれを行なうことは性能に大きな影響を与えるので、最近に利用したいいくつかのページに対し、論理ページアドレスと実ページアドレスの対応をASR (Associative Register) 上に記憶し、主メモリ上(または仮想メモリ上)のテーブル参照のひん度を減らす。ASRの適切な数に関する報告がある⁸⁾。

仮想メモリシステムの性能に大きな影響を与えるこれらの要因は、そのもとで動作するプログラムの構造に大きく依存する。あるプログラムにおいて、それが実行に際して一定時間内に使用または参照するページの集合を、そのプログラムのワーキング・セットと呼んでいる²⁾。平均的なワーキング・セットの大きさが小さいプログラムを局所性の良いプログラムと呼ぶならば、局所性の良いプログラムほど仮想メモリシステム下では効率よく動作するプログラムであることが知られている。

この報告では、仮想メモリシステム下で動作するす

* Optimization of Program Organization in Virtual Storage Systems, by Takashi MASUDA and Hiroyuki SHIOTA (Systems Development Laboratory, Hitachi, Ltd.)

** (株)日立製作所システム開発研究所

で開発済みのプログラムを、再配置可能な単位で並べかえることにより、そのプログラムの局所性を高めるいくつかの方式を提案し、それらを比較検討した結果を報告する。再配置可能なプログラム単位間の結び付きの強さの定義にワーキング・セットの考え方を導入したこと、再配置のための方式として多変量解析の分野で広く利用されているクラスタ分析手法を利用したことが、提案した方式の特徴である。再構成の対象としては、仮想メモリ方式による HITAC 8700 システムの下で動作する FORTRAN コンパイラを取り上げた。

この種の方式による性能改善法としては、Hatfield 等の報告がある³⁾。同報告では、また、プログラムの局所性を高めることが、性能改善に非常に有効であることを確かめている。本論文では、我々の提案したいくつかの再構成方式と、Hatfield 等による方式とを詳細に比較検討する。また、Morrison はユーザの立場から、ワーキング・セットの大きさが小さくなるようなプログラムの書き方を報告している⁴⁾。

2. ワーキング・セット・アプローチ

我々の目的は、開発済みのプログラムを再配置可能なプログラム単位（以下、これをルーチンと呼ぶ）で並べかえることにより、そのプログラムの平均的なワーキング・セットの大きさをできるだけ小さくすることである。

基本的な考え方は、各々のルーチンの大きさをページ・サイズの数分の1と仮定し、1つのページ内にできるだけ関係の強いルーチン（プログラムの実行時に時間的に近い範囲で用いられるルーチン）を集めることにより、そのプログラムの平均的なワーキング・セットの大きさを小さくする効果があると考え、ルーチン間の関係の強さを求めるために、そのプログラムを実行したときのアドレス・トレース・データを利用する。そこに含まれるルーチンが並べかえの対象となる。

アドレス・トレース・データからルーチン間の結び付きの強さを表わす対称行列 R を以下のように作成する。

$$R = \{r_{ij}\}, \quad i, j = 1, 2, \dots, n; \quad i \neq j.$$

(n はルーチンの数)

総トレース・ステップ数を N とする。 $W_r(kT, t)$ をトレース・データにおいて、 $(kT-t)$ ステップ目から kT ステップ目までに利用された異なったルーチン

の集合とする。 k, T, t はいずれも正の整数である。このとき、 R の要素 r_{ij} を次のように定義する。

$$r_{ij} = \sum_{k=1}^{[N/T]} \delta_{ijk}$$

ここで、 $[x]$ は x をこえない最大の整数を表わし、また、

$$\delta_{ijk} = \begin{cases} 1 & \text{when } i, j \in W_r(kT, t) \\ 0 & \text{otherwise.} \end{cases}$$

すなわち、 r_{ij} はトレース・データを最初から T ステップ単位に順々にみたとき、各 T ステップのところで、そこから過去の t ステップ内にルーチン i とルーチン j が共に利用されている場合の総頻度を表わす。

このように R を定義することにより、 r_{ij} には、ルーチン i と j の間の時間的要素までを含めた結び付きの強さ、すなわち、ワーキング・セットの考えを含んだ結び付きの強さを反映することができる。 R を求めるときに、 t は実システムでのページ・フォールト間の平均間隔のオーダーにとるのが妥当であり、また T は、ワーキング・セット・サイズの時間的平均を小さくするということから、小さいほど良いはずである。ただ、 T が小さすぎると R を求めるための計算時間が大きくなる。 t として数 1000 ステップ、 T として数 100 ステップ程度が後述の実験データからも適当である。

Hatfield, Gerald³⁾ (以下 H & G と記す) が定義している nearness matrix C は、 R において $T=t=1$ という特殊な場合に対応する。マトリックス C の要素 c_{ij} は、トレース・データの最初から最後までにおけるルーチン i と j の間の制御の移動または参照の絶対的な回数を表わしており、結び付きの強さに時間的要素は全く含んでいない。たとえば、トレース・データの 100,000 ステップの範囲で、ルーチン i からルーチン j への参照が時間的に、

(i) 10,000 ステップ目ごとに1回ずつ、計10回生じた場合

(ii) 10,000 ステップ目から10,100 ステップ目までに集中的に10回生じた場合

のいずれの場合にも $c_{ij}=10$ であるが、メモリ負荷の点からルーチン i と j の結び付きを考えると、(i)の方をずっと重く評価すべきである。マトリックス R では、 r_{ij} にワーキング・セットの考えを取り入れることにより、この点を十分に評価することができる。

3. 整数計画法による定式化

マトリックス R に基づき、互いに関連の強いルーチン同志を同一のページに集める問題は、以下のように定式化できる。

再構成の評価基準として、ページ内におけるルーチン間の結びつきの強さの和を全ページについて加え合わせたもの、すなわち、

$$\sum_{k=1}^m \sum_{i,j \in P_k} r_{ij}$$

を選び、これを最大にする並べかえを求める。ここで m はページ数、 P_k は k 番目のページに入っているルーチンの集合を表わす。

ルーチン i の大きさを s_i 、ページ・サイズを p 、さらに、

$$x_i^{(k)} = \begin{cases} 1 & \text{when } i \in P_k \\ 0 & \text{otherwise,} \end{cases}$$

とすると、最も単純には、

$$\begin{aligned} & \max \left[\sum_{k=1}^m \sum_{i>j} \sum r_{ij} x_i^{(k)} x_j^{(k)} \right] \\ & \text{s. t. } \begin{cases} \sum_i x_i^{(k)} = 1 & \text{for } \forall k \\ \sum_i s_i x_i^{(k)} \leq p & \text{for } \forall k \end{cases} \end{aligned}$$

と定式化が可能である。このままでは目的関数が線形でないで、通常の整数計画法のプログラムにのせるには、それを線形化する必要がある。

詳細な検討は略するが、このアプローチは、整数変数の数と計算時間の兼ね合いから、そのままでは実用化はむずかしい。ある単純なモデル・プログラムについて解いた結果では、ルーチン数 28、ページ数 4 のプログラムに対し、IBM SYSTEM 370/MODEL 165 を用い、また、整数計画法のプログラムとしては、IBM の MPSX-MIP を利用して、最適解を得るのに 14 分を要した。したがって、ルーチンの数が 100 を越すような現実の問題に対しては、混合整数計画法をそのまま利用するのは無理である。これを階層的に解くことは可能であるが、結果の妥当性を評価することがかなり面倒になる。

4. クラスタ分析手法の利用

4.1 幾何学的解釈

整数計画法に代わる再構成方式として、多変量解析の分野で広く利用されているクラスタ分析手法が、我々の問題に対して有力な武器となることが判明した。

本論にはいる前に若干の幾何学的説明をする。ルーチン間の距離行列 D を次のように定義する。

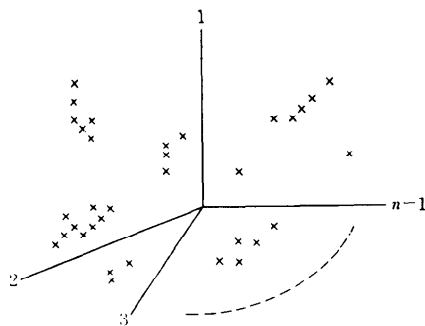
$$D = \{d_{ij}\}, \quad i, j = 1, 2, \dots, n; \quad i \neq j$$

ここで、

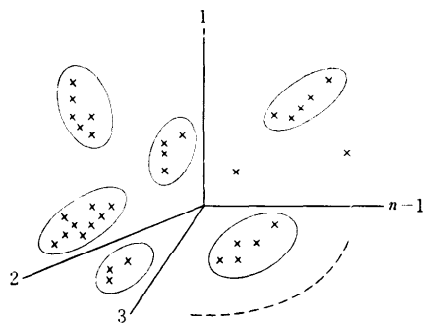
$$d_{ij} = \begin{cases} 1/r_{ij} & \text{for } i \neq j \text{ and } r_{ij} \neq 0 \\ \text{undefined,} & \text{otherwise.} \end{cases}$$

この距離の関係を満たす n 個のルーチンは、全く形式的には、 $(n-1)$ 次元空間上に散らばっていると考えることができる。たとえば、 $n=3$ の場合には、3 つのルーチン間の相互関係は 2 次元平面上に表わされる。無論、上記の距離は、ユークリッド距離の性質は満たしていないが、形式的にこのように考えることにより、プログラム再構成の問題に対し、クラスタ分析法を適用することの妥当性が明確になる。

Fig. 1 (a) は、 $(n-1)$ 次元空間上の n 個のルーチンを表わしている。この空間上での n 個の点の散らばり具合が、そのプログラム実行に際してのメモリ参照の局所性と密接に関連している。点の散らばり具合が、いくつかの点ごとに密なクラスタを構成しているようなプログラムは、再配置により局所性を高める



(a) n routines in the $(n-1)$ dimensional space



(b) Clustering of n routines

Fig. 1 A conceptual representation of n routines

ここで、 n_{fj} は、 $\{r_{ij} | i \in C_f, j \in C_g\}$ における $r_{ij} \neq 0$ の要素の数を表わす。

次に、これら3つの関連の強さの基準のほかに、ルーチンのページへの割当てを特に考慮に入れた今1つの関連の強さの基準を定義した。上記3つの定義のごとく、通常のクラスタ分析におけるクラスタ間の距離あるいは関連の強さの定義には、各個体の大きさの影響が全く考慮されていない。しかしながら、我々の問題では、後述するように各クラスタの大きさにページ・サイズの制限がある。そこで、各ルーチンの大きさをクラスタ間の関連の強さの定義に加えた方が妥当と考えられる。クラスタ間の関連の強さを次のように定義する。

(iv) 修正平均法 (Modified Average: MA)

$$R_{fg} = \frac{1}{S_f + S_g} \sum_{i \in C_f} \sum_{j \in C_g} r_{ij}$$

ここで、 S_f, S_g は各クラスタの大きさ、すなわち、

$$S_f = \sum_{i \in C_f} s_i, \quad S_g = \sum_{j \in C_g} s_j$$

である。 R_{fg} は、2つのクラスタに属するルーチン間の関連の強さの総和を2つのクラスタの大きさの和で基準化したものであり、クラスタ間の関連の強さへの1語当りの貢献度を表わしているといえる。この場合には、各ルーチン間の関連の強さも r_{ij} ではなく、 $r_{ij}/(s_i + s_j)$ とする。

クラスタリングの実行過程では、最も近い2つのクラスタ同志を順々に結びつけ、新しいクラスタを作成していく。その過程は、Fig. 2 のような樹形図 (Dendrogram) に表現する。Fig. 2 の縦方向はルーチン名であり、横方向の長さはクラスタ間の関連の強さの逆数 (距離) に対応する。我々の問題は通常のクラスタ分析と異なり、各クラスタの大きさにページ・サイズの制限がある。無論、ページ・サイズを意識せずに数ページにわたるクラスタを作成してもよく、ワーキング・セットの立場からはその方がよいとも考えられるが、今回の試みでは、アルゴリズムの比較に重点をおくため、各クラスタの大きさを単純にページ・サイズ以下に制限した。新たに作られるクラスタがページ・サイズを越えるときには、それらは互いに結びつけない。Fig. 2 で下方にクラスタリングされずに残っているルーチンの大部分は、トレース・データに含まれないルーチンの集合であるが、ごく一部上記のクラスタの大きさの制限から、どのページにも割り当てられなかったルーチンが存在する。これらは後に人手で適当な割付けを行なう。

また、再配置の結果、各ページの最後にできる若干の空きエリアはそのまま残しておくことにする。

5. 実験結果

5.1 モデル・プログラム

本節の目的は、前節に述べたクラスタ間の関連の強さについての4つの定義のうち、プログラムの局所性を高めるのに最も適しているものを見出し、さらに、それらと H&G の提案している “automatic reordering” の方式を、実際のプログラムに基づいて比較検討することである。そのためのモデル・プログラムとして、仮想メモリ方式を有する HITAC 8700 システムのもとで動作する FORTRAN コンパイラを取り上げた。丁度、コンパイラの性能向上が各側面から実施されている時期であったので、コンパイラの再構成もその性能向上のための1手段として実施した。

最適化機能を含めぬ OPT=0 と、IBM 社 FORTRAN H 相当の最適化機能を有する OPT=2 に対して実験を試みた。プロシージャ部の大きさが、OPT=0 が約 250 K バイト、OPT=2 が約 340 K バイト、ルーチン数が、OPT=0 が 533、OPT=2 が 792 で、その内 388 個のルーチンが両者で共通に利用されている。FORTRAN コンパイラの動作特性を表わす1つの科学計算向けの標準問題を設定し、そのコンパイル過程の全軌跡をトレースした。トレース・ステップ数が OPT=0 で 87 万ステップ、OPT=2 で 194 万ステップである。なお、HITAC 8700 のページ・サイズは 4 K バイトであり、1 ページに平均 8~10 個のルーチンが入ることになる。

5.2 再構成アルゴリズムの比較

上記のモデル・プログラムに対し、前節の4つのクラスタリング・アルゴリズム、および H&G 法の比較をする。まず、 $T=1, t=1$ として、マトリックス R を作成する。この場合には、 R は、H&G の定義する nearness matrix C に完全に一致する。H&G 法としては、彼等の報告にあるルーチンのページへの割当てを力学的モデルで取り扱う方法を採用した。すなわち、マトリックス M を次のように定義する。

$$M = \{m_{ij}\} \quad i, j = 1, 2, \dots, n.$$

ここで

$$m_{ij} = \begin{cases} -c_{ij} & \text{when } i > j \\ \sum_{k=1, k \neq i}^n c_{ik} + 2n & \text{when } i = j \\ -c_{ji} & \text{when } i < j. \end{cases}$$

Table 1 Average working set size (pages) of the algorithms investigated (page size=4 K bytes)

(a) OPT=0

window size (instructions)	original ordering	H&G	NN	FN	AV	MA	WS&MA
1000	5.82	4.77	4.27	4.36	4.38	3.51	3.41
2000	6.80	5.89	5.10	5.26	5.32	4.28	3.88
5000	8.36	7.04	6.25	6.44	6.47	5.44	4.64
10000	9.60	7.76	7.15	7.26	7.30	6.37	5.37
20000	10.35	8.36	7.90	7.93	8.07	7.18	6.13
50000	11.93	8.95	8.59	8.68	8.88	8.03	7.20

(b) OPT=2

window size (instructions)	original ordering	H&G	NN	FN	AV	MA	WS&MA
1000	5.24	4.28	3.69	3.68	3.69	3.45	3.29
2000	6.51	5.30	4.54	4.50	4.51	4.28	3.94
5000	8.64	6.90	6.00	5.94	5.89	5.75	5.10
10000	10.02	7.99	7.12	7.08	6.95	6.90	6.08
20000	11.48	9.08	8.38	8.34	8.14	8.20	7.21
50000	13.37	10.35	10.08	10.14	9.83	9.96	9.03

M の逆行列を求め、それに基づきルーチンのページへの割当てを行なう (詳細は参考文献(3)).

以上5つの方法を, OPT=0, OPT=2 のマトリックス R に適用し, ルーチンのページへの割付けを行なった. そして, 各割付け結果に基づき, それぞれのワーキング・セット・サイズの平均 $w_p(t)$ を求めた. $w_p(t)$ は次式により定義する.

$$w_p(t) = \frac{1}{\left\lfloor \frac{N}{T} \right\rfloor} \sum_{k=1}^{\left\lfloor \frac{N}{T} \right\rfloor} w_p(kT, t)$$

$w_p(kT, t)$ は, トレース・データにおいて, $(kT-t)$ ステップ目から kT ステップ目までに利用された異なったページ数を表わす. t を window size と呼ぶ.

$w_p(t)$ を求めるに際して, T としては $T=100$ を選んだ. T の値は小さいほど $w_p(t)$ の精度はよくなるが, t に比較してかなり小さな値であれば実際的には十分である. t はページ・フォールト間の平均間隔のオーダーを選ぶのが適当である. t としては, $t=1000 \sim 50000$ ステップの間の6種類を選んだ. 結果を Table 1 に示す. 並べかえ前に比較し, $w_p(t)$ の減少率は, OPT=0, OPT=2 で大きな差異がないことがわかる. $w_p(t)$ の減少率は, 並べかえ前に比較し, H&G 法で 15~20%, NN 法, FN 法, AV 法ではいずれもほとんど差がなく, ほぼ 25~30%, クラスタ間の関連の強さに各ルーチンの大きさまでを考慮に入れた MA 法では 30~40% まで減少している. H&G 法の減少率 15~20% は, 彼等の報告に "automatically reordered" としてある結果, 16~18% と

よく一致している.

この結果から, プログラム再配置のためのアルゴリズムとしてクラスタ分析手法が非常にすぐれていることがわかる. 特に, クラスタ間の関連の強さの定義に, ルーチンの大きさまで考慮に入れた MA 方式では, 平均的なワーキング・セットの大きさが, 約 40% 近くも減少することが判明した.

さらに, 3. で述べたルーチン数 28, ページ数 4 の小さなプログラムに上記5種類のアルゴリズムを適用した結果, MA 法によるクラスタリング結果が, 整数計画法から得られた解と全く一致した.

5.3 ワーキング・セット・アプローチの効果

ルーチン間の関連の強さの定義に対して, 2. で述べたワーキング・セット方式の効果を求める. $T=100$, $t=1000$ としてマトリックス R を作成し, それに基づき MA 法によりクラスタリングを実施した. その結果から, $w_p(t)$ を求めた結果を Table 1 に WS&MA として示した. OPT=0 では $w_p(t)$ の減少率は 40~45% に達している. OPT=2 でもほぼ 40% に近い. 他の実験として, $t=5000$ の R に基づいて MA 法によるクラスタリング結果から求めた $w_p(5000)$, $t=10000$ の R からの $w_p(10000)$, $t=20000$ の R からの $w_p(20000)$, $t=50000$ の R からの $w_p(50000)$ などを求めたが, いずれも $t=1000$ に基づいた R からの $w_p(5000)$, $w_p(10000)$, $w_p(20000)$, $w_p(50000)$ の結果とはほとんど差はない. したがって, $T=100$, $t=1000$ に基づいた再構成結果は, window size t が 1000~50000 の範囲で満足すべき結果を与えていると

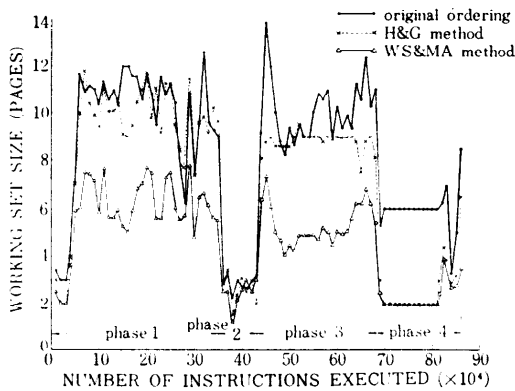


Fig. 3 Working set size of compiling process (OPT=0, window size=5000)

Table 2 Average working set size (pages) of OPT=0 for each phase (page size =4 K bytes, window size=5000)

phase	original ordering	H&G	WS&MA
1	9.47	9.01	5.76
2	2.89	2.67	2.54
3	10.26	8.81	5.36
4	5.98	2.65	2.36
total	8.36	7.04	4.64

いえる。

最後に、OPT=0 に対し、再構成前、H&G 法による再構成後、これまでの検討で最適であった WS&MA 法による再構成後の 3 種類について、コンパイル実行に伴うワーキング・セット・サイズの時間的変化を Fig. 3 に示した。 $w_p(100k, 5000)$, $k=1, 2, \dots, [N/100]$ の結果を 10000 ステップ単位に平均をとってプロットしたものである。ワーキング・セットの大きさに急激な変化があるところは、そこでコンパイラのフェーズが変わったことを意味している。WS&MA 法はいずれのフェーズでも再構成前に比較し、ワーキング・セット・サイズを大幅に小さくする効果を有しているが、H&G 法による再構成結果はやや特殊な傾向を示している。すなわち、フェーズ 1 では再構成の効果は少なく、再構成により逆にワーキング・セットが大きくなっている場合もある。フェーズ 3 ではワーキング・セットが再構成によりほぼ一定の割合で減少しており、フェーズ 4 では 1/2 以上にも大幅に減少している。

これら 3 つの並べ方について、各フェーズごとにま

とめた平均的なワーキング・セットの大きさを、Table 2 にまとめておく。

6. むすび

仮想メモリスistem向けのプログラム再構成法として、クラスタ分析手法を用いた方法を提案した。クラスタ間の関連の強さの定義にルーチンの大きさまでも含ませた MA 法を用いると、平均的なワーキング・セットの大きさが再構成前に比較し 40% 近くも減少することが確かめられた。さらにルーチン間の結びつきの強さの定義にワーキング・セットの考えを導入すると、その減少率は 45% にも及ぶことが判明した。これらの結果は、従来報告されている方式による結果よりもはるかによい。再構成結果は、FORTRAN コンパイラの性能向上に積極的に利用されている。

最後に本研究の遂行に当たり、方式的に種々の貴重など意見を賜った東京大学穂坂衛教授、大須賀節雄助教授、クラスタ分析法につきご討論戴いた農技研奥野忠一、山陽パルプ芳賀敏郎、日科技連矢島敬二、埼玉大学古林隆の諸氏、筆者と共に細部の検討を戴いた当社中研高橋延匡、ソフトウェア工場野口健一郎、大木尚の諸氏、本研究の機会を与えて下さった同永井部長、高須、堂免両課長に厚く感謝致します。

参考文献

- 1) 益田隆司：ページングシステム向けのプログラム再構成の方式、システム評価シンポジウム報告集, pp. 213~222, 情報処理学会 (1972).
- 2) P. J. Denning: The Working Set Model for Program Behavior, Comm. ACM, Vol. 11, No. 5, pp. 323~333 (1968).
- 3) D. J. Hatfield & J. Gerald: Program Restructuring for Virtual Memory, IBM Sys. J., Vol. 10, No. 3, pp. 168~192 (1971).
- 4) J. E. Morrison: User Program Performance in Virtual Storage Systems, IBM Sys. J., Vol. 12, No. 3, pp. 216~237 (1973).
- 5) 益田隆司, 他: ページング・マシンにおけるスワッピング・アルゴリズムの比較とプログラムの動作解析, 情報処理, Vol. 13, No. 2, pp. 81~88 (1972).
- 6) L. A. Belady: A Study of Replacement Algorithms for a Virtual Storage Computer, IBM Sys. J., Vol. 5, No. 2, pp. 78~101 (1966).
- 7) R. L. Mattson, et al.: Evaluation Techniques for Storage Hierarchies, IBM Sys. J., Vol. 9, No. 2, pp. 78~117 (1970).

- 8) M. D. Schroeder: Performance of the GE-645 Associative Memory while Multics is in Operation, Proc. of ACM Workshop on System Performance Evaluation, pp. 227~245 (1971).
- 9) 奥野忠一, 他: 多変量解析法, 日科技連(1971).
- 10) 矢島敬二, 他: クラスター・アナリシス, オペレーションズ・リサーチ, Vol. 16, No. 7~11 (1971).
- 11) 古林 隆: デンドログラムの1つの指標, 経営科学, Vol. 17, No. 2, pp. 88~97 (1973).
(昭和49年2月1日受付)
(昭和49年7月11日再受付)