第６章　言語処理系の性能解析

第６章　言語処理系の性能解析

6.1　概　　要

あるプログラミング言語の言語仕様が与えられた時，そのコンパイラの機能や性能が一意に決まるわけではない。特に本研究のように従来言語と異なる言語機能を導入した場合は，前章で述べたような新しいコンパイル技法を考案する必要がある。そのため，そのような新技法を導入した時の言語処理系の性能を事前に適確に把握することは難しい。そこで，本章では，言語処理系の主たる性能評価基準であるコンパイル効率とオブジェクト効率のうち，前者に関する性能解析を行う。なお，後者については次章で述べる。

ＳＰＬコンパイラの処理性能に影響すると思われる要因のうち，特に従来のコンパイラの場合と異なるものは次のようなものである。

(1)　ＳＰＬライブラリを用いた分割コンパイル方式

(2)　中間語形式に変換された手続きのインライン展開方式

そこで，まずこれらの影響度を調べるために，各処理フェイズ毎のコンパイル時間比と各々の処理内容について分析する。そして，本システムの稼動マシンが１６ビット計算機であることに伴なう主記憶容量の制約下で，その影響がとりわけ大きいと思われるＳＰＬライブラリの入出力処理については，主記憶バッファと補助記憶装置の間のページング動作特性を明らかにすると共に，その改良方式を検討する。

このページング処理に関する研究は，既に仮想記憶システムの分野で詳しく行われてきているが，その成果をＳＰＬライブラリの入出力処理にそのまま適用するには２つの問題があった。即ち，第１には，これまでの実データに基づくページング動作解析の多くはプログラムの手続き部を対象にしており，データ参照に対する特性は明らかでないこと，第２には，実用的なページングアルゴリズム間の性能比較が主に実測データによって行われており，未だ十分には解明されていないことである。そこで，本研究ではこれらの問題を解決することが重要な課題である。

なお，このページング処理の問題は，今回のように主記憶容量が６４Ｋ語程度の中小型計算機においては厳しいものであるが，１語３２ビットの大型計算機において，主記憶バッファエリアを十分に確保できる場合は問題は少ない。

6.2　コンパイル効率に関する実験と評価[C14)]

6.2.1　評価の方法

まず，コンパイル効率の実測評価に用いるＳＰＬプログラムは，その評価目的から，

(1)　open 型手続きを含むこと

(2)　環境モジュールの階層構造を有すること

という条件を満たすものとして，ＳＰＬの教育用に開発したものを選んだ。これは，２つの環境モジ

ュールと1つの処理モジュールから構成され，後者には，1つのmain 型手続きと5つの open 型手続きを含むもので，全体で185行のプログラムである。

コンパイラの稼動環境としては，SPLのターゲットマシンであるHIDIC80をプログラム開発に用いる場合の基本的なシステム構成を用いた。それは，プログラムカードリーダから入力し，ラインプリンタに出力するもので，補助記憶装置としては磁気ドラムを用いた。各々の性能を以下に示す。

(1) 主記憶容量　64KW

(2) カードリーダ　750枚／分

(3) ラインプリンタ　600行／分

(4) 磁気ドラム　転送速度150KW／秒
　　平均アクセス時間10 msec

6.2.2　評価の結果

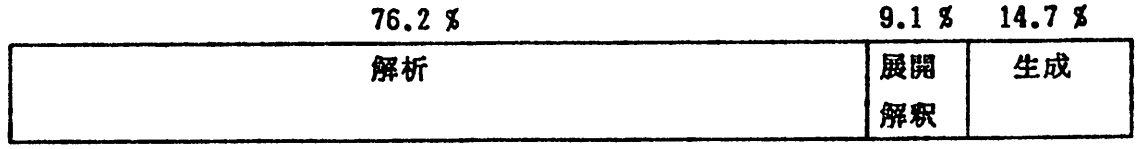
この評価用のプログラムのコンパイルに要した時間は，全体では191.27秒であり，毎分58行のコンパイル速度ということになる。この詳細な結果は，図6.1に示す。まず，図の(a)からわかるように，各処理フェイズ間のコンパイル時間比では，解析処理が全体の3／4を占めており，SPLコンパイラに固有の展開・解釈処理は9%程度と少ない。他の評価結果でも展開・解釈処理は10%前後のものが多く，中間語形式の手続きのインライン展開処理が比較的高速処理されていることが確認された。

一方，各処理時間に占めるCPU時間や各種入出力処理時間の割合は，図6.1(b)に示すような構成になっており，全体ではCPUと入出力処理の割合はほぼ1：1である。解析処理に占める補助記憶装置との入出力処理時間比が他の処理フェイズの2倍近くになっているのは，SPLの特徴である環境モジュールの独立化と階層化により，SPLライブラリへのアクセスが頻繁に行われることを示している。

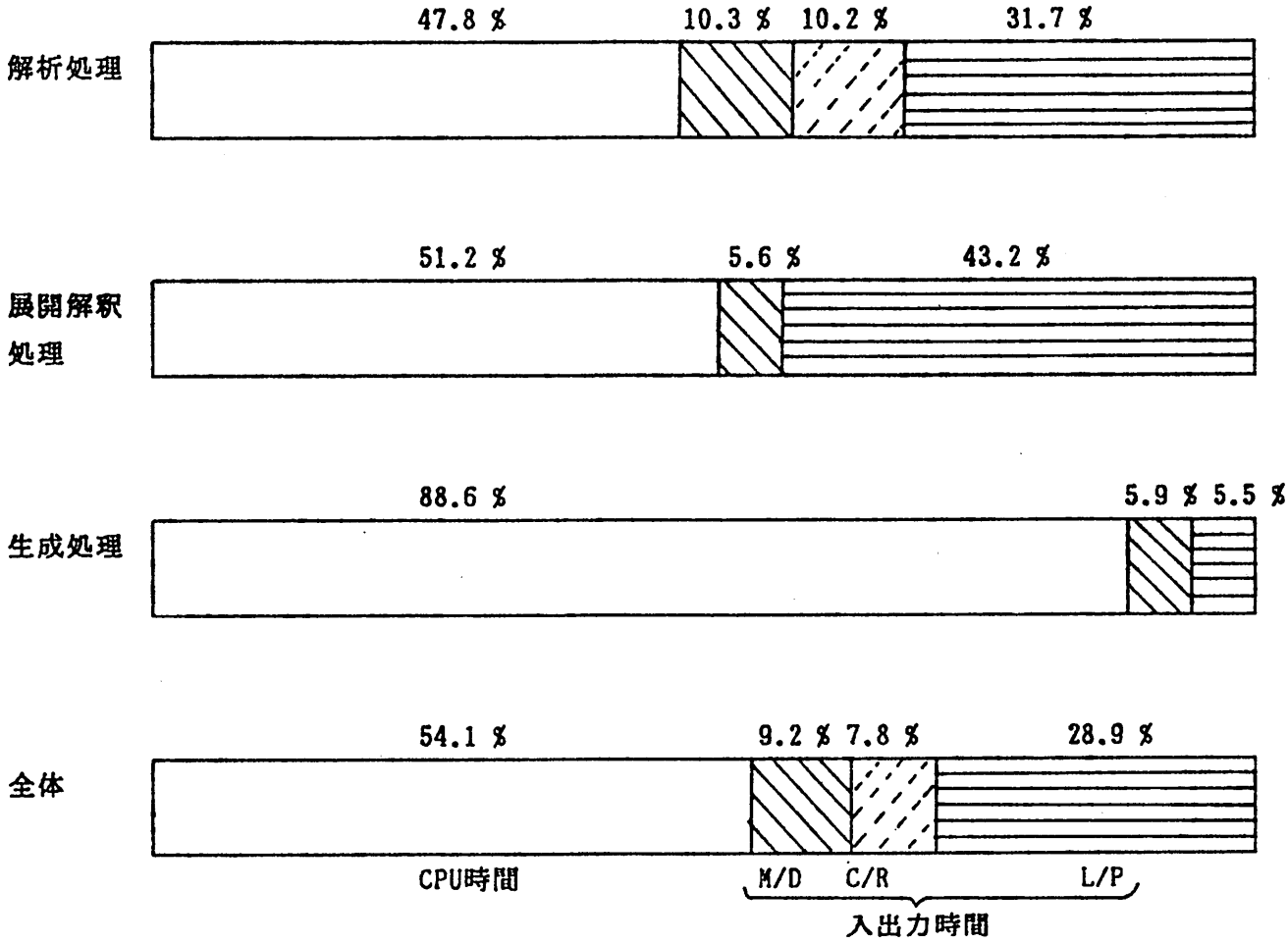
この補助記憶装置へのアクセス時間は，全体のコンパイル時間の中では1割に満たない程度であるが，次に述べる3つの理由からコンパイル効率に与える影響は大きいと考えられる。

(1) 今回のシステム構成では，最大の主記憶容量を装備しているため，SPLライブラリのための主記憶バッファも比較的に余裕があった。しかしながら，実際のシステムでは，むしろ主記憶容量はこれより少ない場合が一般的である。そして，その減少分はそっくり主記憶バッファの減少となるため，それに伴い，補助記憶装置への入出力処理が急激に増加することが予想される。例えば，後に述べる実験結果では，バッファサイズを16KBから4KBに減じた場合，その処理時間は5～20倍になる。

(2) 今回用いた評価用プログラムは185行の小規模なものであったが，実際の制御用応用プログラムは数万行に及ぶ大規模なものが普通である。そのため，SPLライブラリも大規模なものに



（a）　各処理フェイズのコンパイル時間比



（b）　ＣＰＵおよび入出力時間比

図６．１　ＳＰＬコンパイラの処理性能

なるにもかかわらず，その主記憶バッファの容量は一定であることから，補助記憶装置への入出力処理が大巾に増加することが予想される。例えば，ＳＰＬの最初の適用プロジェクトである化学システムの場合，ＳＰＬライブラリは約２４０ＫＢになっている。

(3) 今回の処理時間の中には，カード読取機からのカード入力時間および行プリンタへの出力時間が全体の１／３を占めている。しかしながら，これらの処理は，補助記憶装置内に入出力バッファを設けて，カード読取機や行プリンタとの入出力処理を並行処理することにより，大巾削減が可能であるため，ページング処理が全体に占める比重はより大きくなる。

そこで，次節では，ＳＰＬライブラリのページング動作解析を行うと共に，それに適したページングアルゴリズムについて述べる。

6.3 共通ライブラリ用ページングアルゴリズムの分析[C8,C13)]

6.3.1 ページング動作解析の方法

ＳＰＬライブラリへのアクセス時のページング動作解析を行うために，図6.2に示すようなモジュール階層構造を有するＳＰＬプログラム（約６００行）を選んだ。そして，このプログラムのコンパイル時に生じたライブラリの参照アドレス系列を補助記憶装置に収集し，後で解析するという方法をとった。なお，評価用に選んだ図6.2のプログラムはこの解析プログラム自身であり，そのソースリストを付録３に示す。また，このプログラムの開発過程は他の文献[C6)]に詳しい。

ページング動作解析の基礎データとなる参照アドレス系列は，コンパイラの処理フェイズ毎の分析を可能とするため，次のジョブ単位に収集した。

(1) 環境モジュール解析　　（ＥＮＶ）

(2) 処理モジュール解析　　（ＰＲＯＣ）

(3) オブジェクト生成　　（ＧＥＮ）

なお，括弧内は以後の図表に用いる略号である。

また，これらの収集データの解析プログラムには次の機能を備えた。

(1) 各ページ参照頻度の測定

(2) ＬＲＵ（Least Recently Used ）スタックの各深さへの参照頻度の測定

(3) 各種のページングアルゴリズムの適用と性能比較

ここで，評価対象とした５種類のページングアルゴリズムについて述べておく。

(a) ＬＲＵ方式

ページフォールト時には主記憶バッファ内の最も長い間参照されていないページを追い出す。

(b) ＦＩＮＵＦＯ[M2)]方式（First In Not Used First Out ）

主記憶バッファ内の各ページにＬＲＵフラグ１ビットを設け，参照されるとオンにする。ページフォールト時には，バッファを円環状につながったものとみなし，最後にロードされたページの次からサーチし，フラグがオフのページのうちで最初に出会ったものを追い出す。このサーチ

の途中で出会ったフラグがオンのページに対しては，そのフラグをオフにしてスキップする。これはMultics[14]，ＯＳ７[01]などで採用されている。

(c) ＦＩＦＯ方式（First In First Out）

ページフォールト時には，主記憶バッファ内に最初にロードされたページを追い出す。

(d) ＦＩＶＥ方式

ＬＲＵフラグとして５ビット設け，参照されると左端ビットをオンにする。ページフォールト時には，このフラグを０～３１の整数値と見て最小値のページを追い出す。この時，すべてのフラグを１ビット分右へシフトする。なお，略称ＦＩＶＥは本論文で新たに定義して用いるものである。

(e) 部分的ページ常駐方式

例えば参照頻度が上位のページなど，１部分のページを主記憶バッファに常駐化して，ページング対象からはずす。

6.3.2 実験結果

(1) ページサイズ

まず，ページサイズの影響を見るために．ＬＲＵ方式の場合を例にとって，主記憶バッファ容量が４，８，１６キロバイトの時のページサイズとページフォールト率の関係を調べた。その結果，図6.3に示すように，ページサイズが１２８，２５６，５１２バイトの場合では，ページサイズが小さいほどフォールト率が小さくなることがわかった。

(2) 局所参照性

ＬＲＵ方式の場合，主記憶バッファ内にロードされているページの間でどのページがより最近に参照されたかという情報を保存するためにスタックが用いられる。このスタックはより最近に参照されたページほど上位になるように処理される。従って，スタックの上位のページほど頻繁に参照される場合はＬＲＵ方式が最適である。この性質は局所参照性と言われ，スタックの深さｉのページの参照確率をＬ（ｉ）とすると，

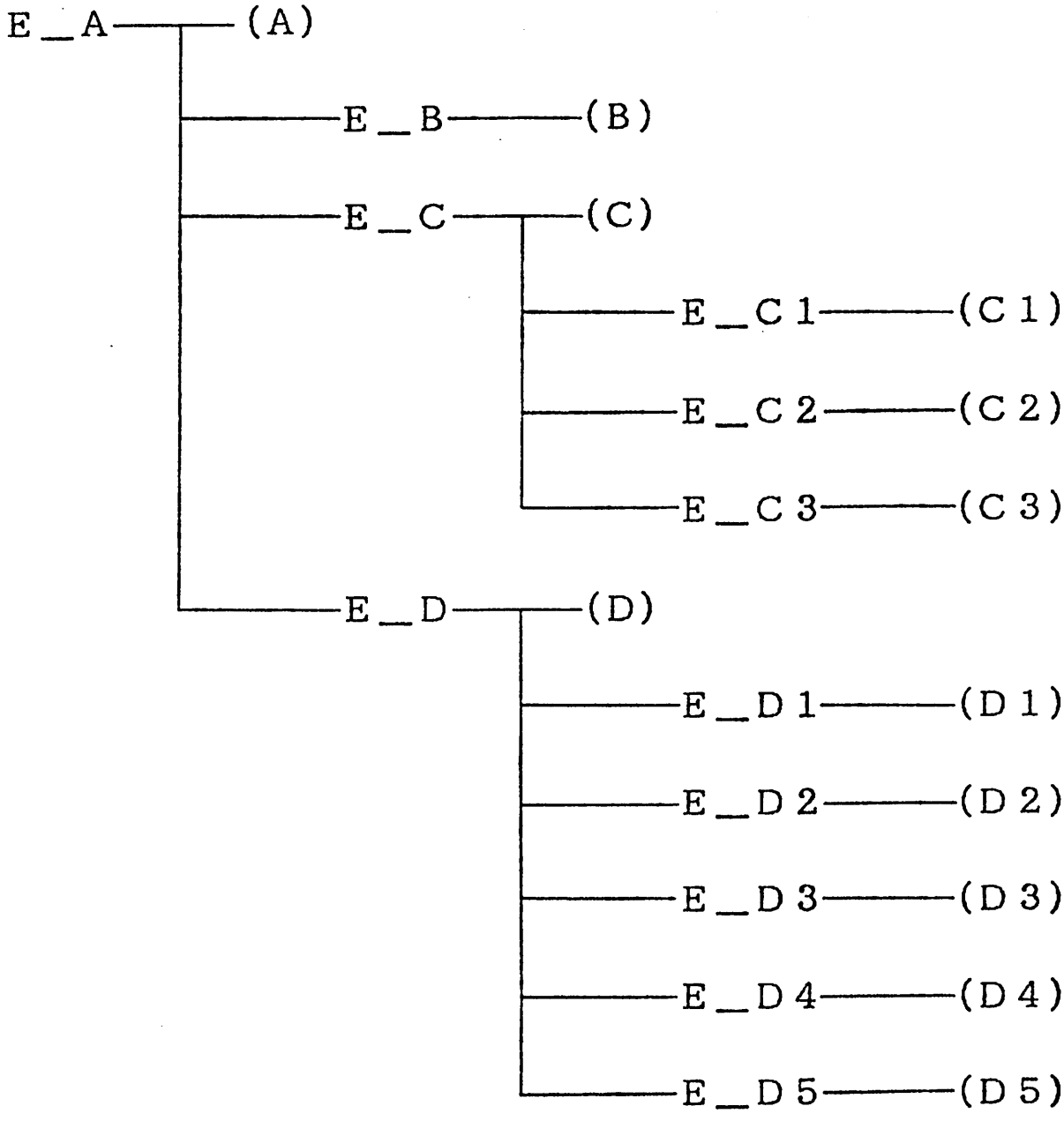
$$L(i) > L(j),\ i < j \qquad (6.1)$$

と表現される。

この局所参照性に関する実測データの例を表6.1(1)に示す。これは，ページサイズが２５６バイトの時のＰＲＯＣジョブの例であり，その参照アドレス系列のはじめの３００００回分を表にしたものである。この表からわかるように，スタックの先頭のページへの参照が全体の６０％を越えるなど，明らかな局所参照性を示し，スタックの深さに従って参照頻度は急速に減少している。

(3) 静的参照頻度

一部のページをページングの対象から除き，主記憶に常駐させる方式が研究されているが，常



E_aa：環境モジュール
（aa）：処理モジュール

図６．２　評価用プログラムのモジュール階層構造

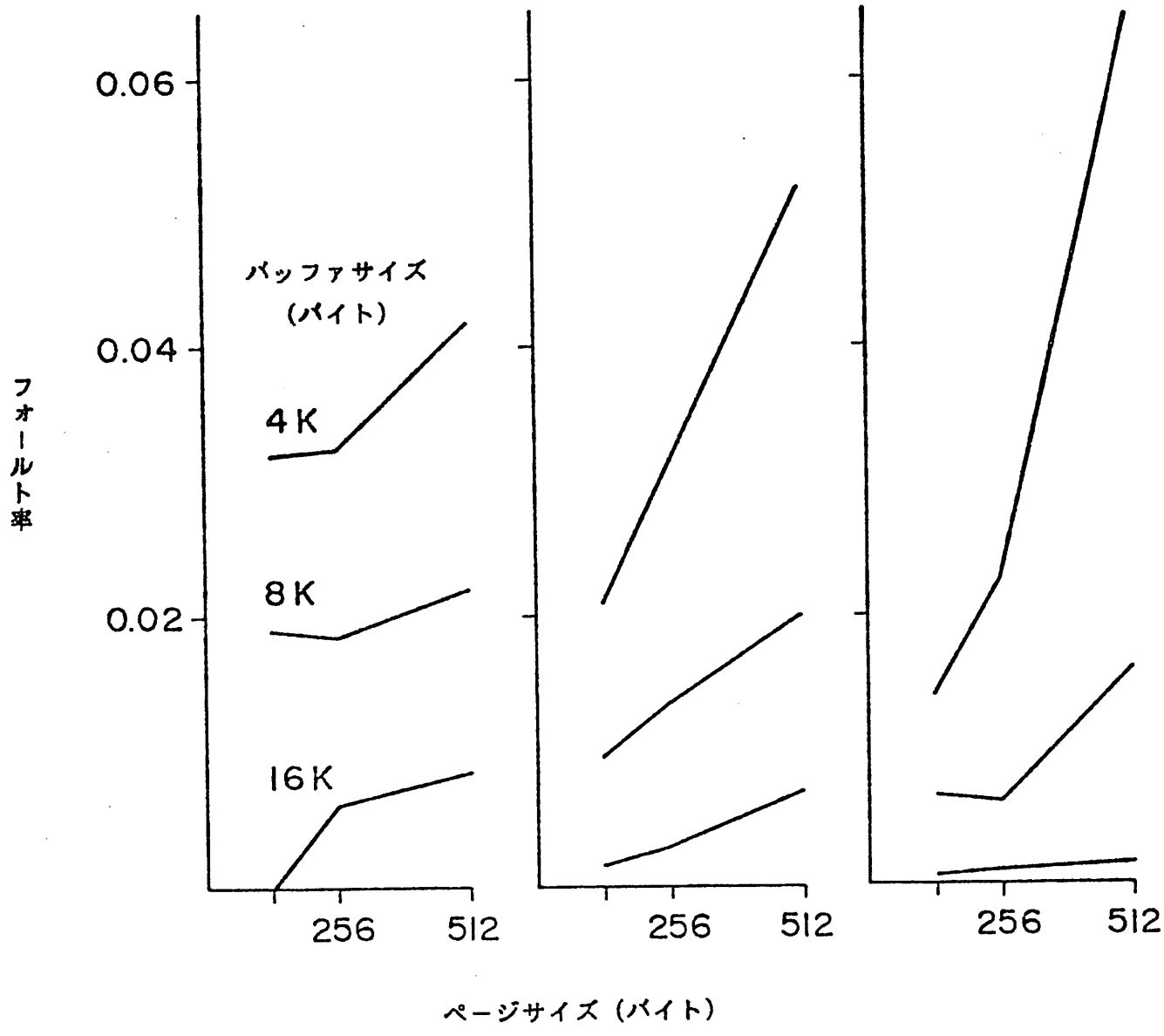


図6．3　ページサイズとフォールト率の関係

表６．１　ＬＲＵスタックと各ページの参照頻度分布

（１）　ＬＲＵスタック（上位）

スタック深さ	参照回数	比率(%)
1	18164	60.5
2	5360	17.9
3	1267	4.2
4	1263	4.2
5	1034	3.4
6	447	1.5
7	347	1.2
8	236	0.8
9	212	0.7
１０	127	0.4

（２）　参照頻度上位のページ

順位	参照回数	比率(%)
1	4903	16.3
2	2271	7.6
3	2148	7.2
4	2055	6.9
5	1616	5.4
6	1527	5.1
7	1036	3.5
8	783	2.6
9	682	2.3
１０	572	1.9

（注）　いずれもＰＲＯＣジョブ（ページサイズ２５６バイト）の場合の
３００００回の参照データ使用

駐ページの選択が難しい問題である。本解析においても，各ページの参照頻度が表6.1(2)の例のように偏っていたので，参照頻度の上位のものを常駐にする方式を適用してみたが，特に良い結果は得られなかった。その理由は，表6.1の(1)と(2)を比較すれば明らかなように，ＬＲＵスタック内の局所参照性が静的参照頻度の偏りよりもかなり大きいためと思われる。なお，表の(2)は(1)の場合と同じ基礎データを用いている。

(4) ページングアルゴリズムの性能比較

実測によって得た参照アドレス系列に対して，6.3.1で述べたページングアルゴリズムを適用してフォールト率を調べた結果，汎用アルゴリズムの間では，ＬＲＵ，ＦＩＮＵＦＯ，ＦＩＶＥ，ＦＩＦＯの順に良い性能を示した。その一部を図6.4に示すが，これは，3種類のジョブに対してページサイズ２５６バイトの条件下で各ページングアルゴリズムを適用した例である。なお，図の見易さのためにＦＩＶＥのグラフは省略している。

この結果を元にして，ＬＲＵ方式を基準にした時のＦＩＦＯ，ＦＩＮＵＦＯ，ＦＩＶＥの各方式によるページフォールト増加率を求めると図6.5のようになる。この図の縦軸は，図6.4の各測定点，即ち特定の主記憶バッファサイズにおける次の式の値を示す。

$$g(f_{LRU}) = \frac{\text{比較対象方式のフォールト率}}{\text{ＬＲＵ方式のフォールト率}} - 1 \qquad (6.2)$$

これらの結果から次のことが確められる。

(1) ＦＩＦＯ方式は明らかに他方式より悪く，その度合はフォールト率の低い所ほど大きい。

(2) ＦＩＮＵＦＯ方式は比較的安定した性能を示しており，ＬＲＵ方式に比べて概ね１０％以下の悪化にとどまっている。

(3) ＦＩＶＥ方式もＬＲＵ方式と比較して１０％以下の悪化のものが多いが，フォールト率の低い所ではＦＩＮＵＦＯ方式よりも劣る。

なお，これらの方式間の性能差の普遍性については次項で理論的に解析する。

6.3.3 理論的解析

(1) 最適ページサイズ

6.3.2(1)の実験結果では，主記憶バッファサイズ一定の条件下ではページサイズが小さいほどフォールト数が少ないという結果が得られた。その理由としては，ロケーションｘのデータ参照直後にロケーションｙのデータを参照する確率をｐ（ｙ｜ｘ）とすると，コンパイラのテーブル参照では，

$$p(x \mid x) \gg p(x \pm d_1 \mid x) > p(x \pm d_2 \mid x),$$
$$0 < d_1 \ll d_2, \qquad (6.3)$$

即ち，同一データの参照確率がその近傍の参照確率よりもかなり高いためと考えられる。一方，手続き部の場合は近傍の参照確率の方が高いのでページサイズはある程度大きいものが良い。

一般に仮想記憶システムでは２または４キロバイトのものが多い。

この実験結果からページサイズを小さくするとフォールト率は減少するが，それに反比例して各ページの物理アドレスを保存するアドレス変換テーブルは大きくなる。従って，主記憶容量一定の条件下での実際の最適ページサイズは次のようにして決まる。まず，参照データエリア（仮想記憶領域）の大きさをＤ，ページサイズをｓとした時，アドレス変換テーブルの大きさは，１エントリ長をａとして，全体ではａ（Ｄ／ｓ）となる。これはページング処理のために使用可能な主記憶容量Ｃの一部が用いられるため，主記憶バッファサイズをｂとすると，

$$b + a(D/s) = C \qquad (6.4)$$

となる。一方，１回のページ参照による主記憶と補助記憶の間の平均入出力時間ｔは，補助記憶装置の平均アクセス時間をｕ，転送速度をｖとした時の１ページの入出力時間ｕ＋ｓ／ｖと追い出すページの内容が書き換えられている確率ｗとページフォールト率ｆとから，

$$t(s,b) = (1+w)\cdot(u+s/v)\cdot f(s,b) \qquad (6.5)$$

で表される。従って，最適ページサイズは，式（6.4）の条件下で式（6.5）のｔを最小にするｓである。

本解析においては，関数ｆ（ｓ，ｂ）として図6.3の実測値を用いてｔ（ｓ，ｂ）を求めた結果，最適ページサイズは２５６バイトであることがわかった。図6.6はアドレス変換テーブルのメモリ容量を考慮して図6.3を補正したものである。

(2) バッファサイズとフォールト率の関係

図6.4の実験結果からもわかるように，フォールト率はバッファサイズの増加と共に急速に減少している。これは，6.3.2(2)で既に述べたようにデータの局所参照性が強いためである。この実験結果を詳しく調べると，バッファサイズｂとフォールト率ｆの間の関係は，

$f \geq f_0$ の時，$f(b) = A\,b^{-k}$ （6.6）

$f < f_0$ の時，$f(b) = B\,e^{-\alpha b}$ （6.7）

の近似式になっている。ここで，定数（Ａ，ｋ，Ｂ，α，f_0）の値は，図6.4，図6.7などから各ジョブ毎に，

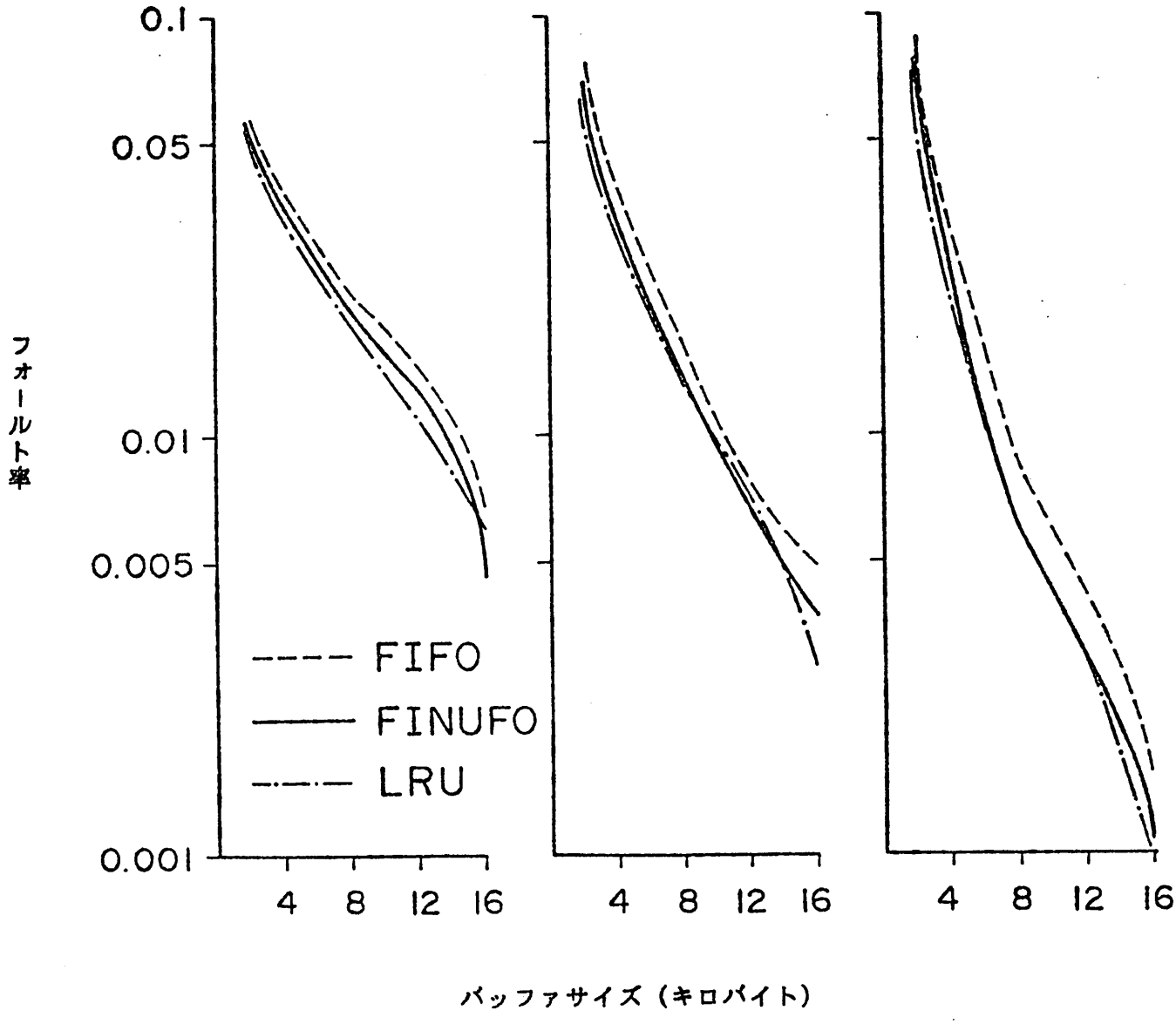
ＥＮＶ　：（0.14，0.48，0.0058，0.035，0.04）

ＰＲＯＣ：（0.40，0.90，0.068，0.050，0.03）

ＧＥＮ　：（0.34，0.58，0.065，0.066，0.18）　（6.8）

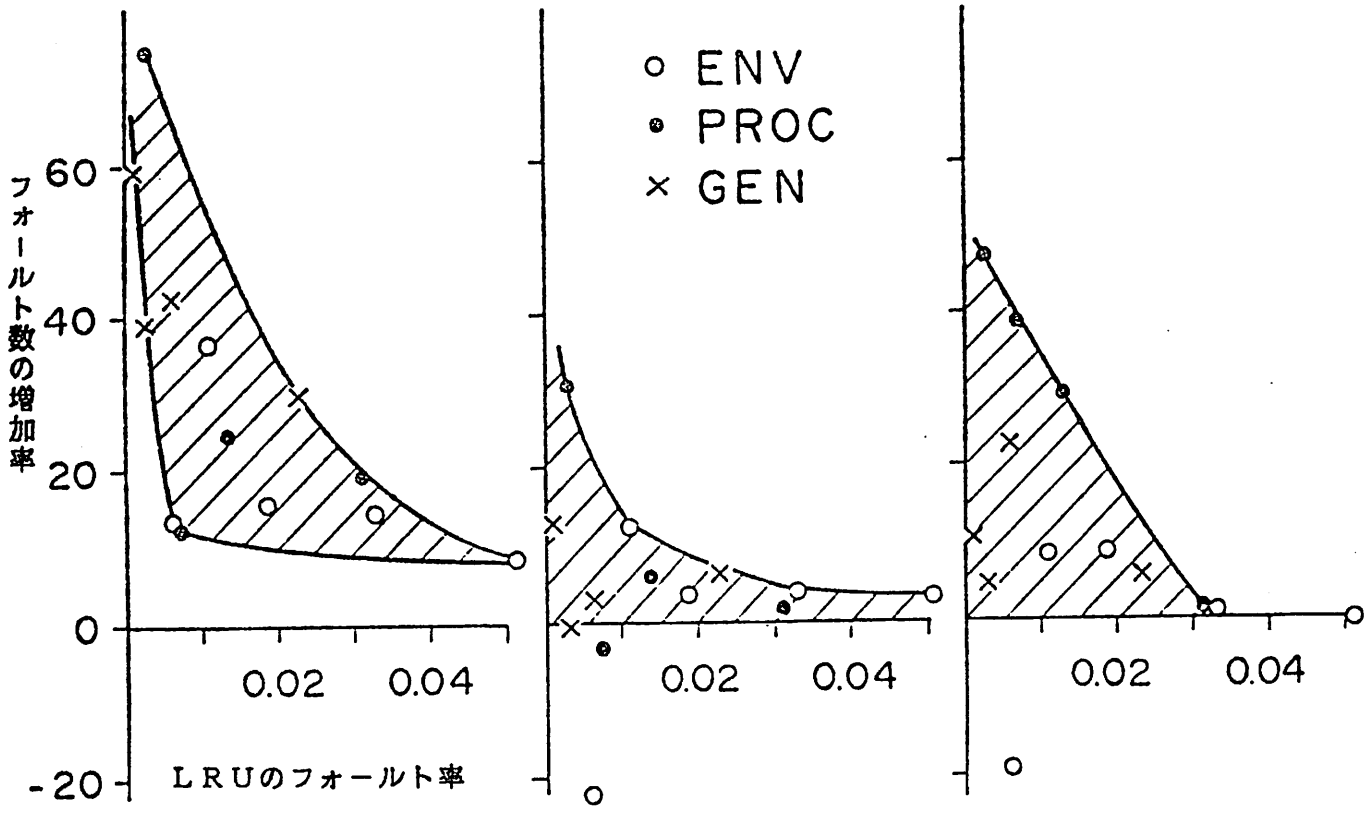
であった。

これまで，手続き部のページング動作解析に関するL.A.Belady 他の幾つかの研究において，一般に主記憶バッファサイズの小さい所では式（6.6）が成立つという報告[B6]がなされているが，本実験でも図6.7に示すように同様の結果が得られた。しかしながら，ｋの値はいずれもｋ＜１であり，一般値（ｋ≒２）よりも小さい。これはデータの局所参照性が手続きの場合ほどは強くないことを示している。



（注）　ページサイズ：２５６バイト

図６．４　ページングアルゴリズムの性能



(a) FIFO 対 LRU　(b) FINUFO 対 LRU　(c) FIVE 対 LRU

(注)　ページサイズ：256バイト

図6．5　LRU方式に比した他方式の性能低下度

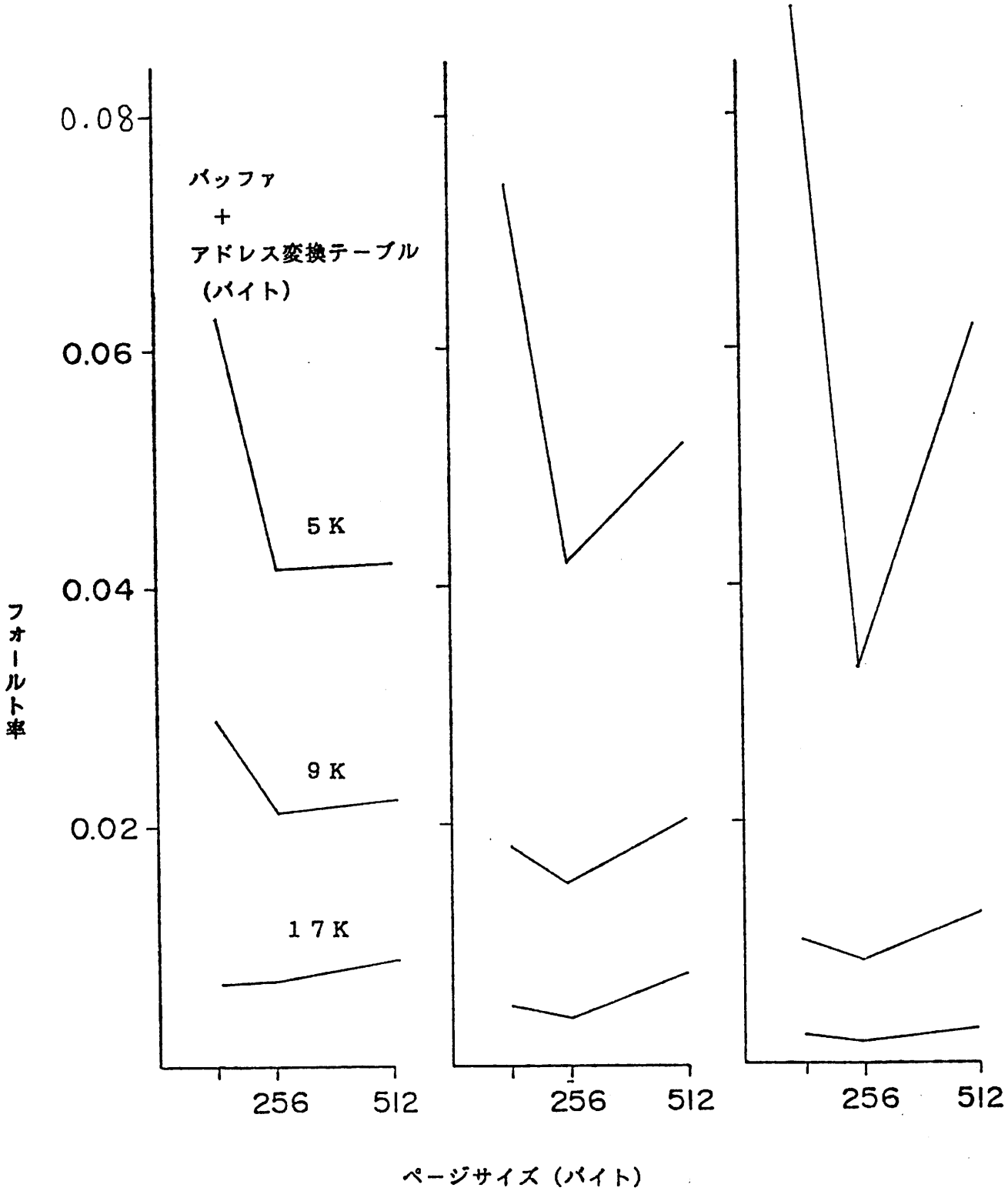
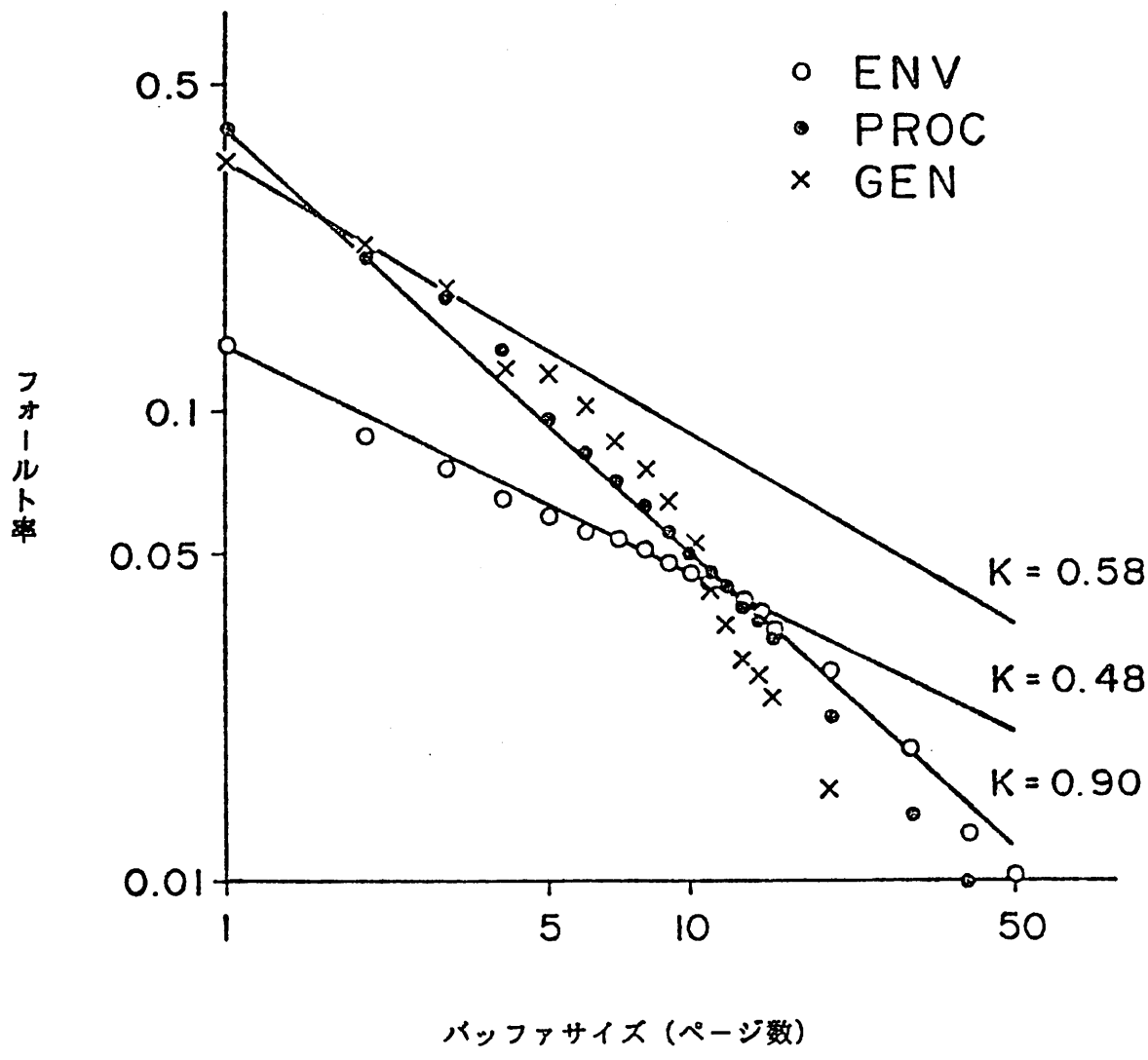


図6．6　ページサイズとフォールト率の関係（補正後）



（注）　ページサイズ：２５６バイト

図６．７　ＬＲＵ方式におけるフォールト率とバッファサイズの関係

(3) ＬＲＵとＦＩＦＯの性能差

ページング動作に関する理論的解析は既に幾つか行なわれているが，ＬＲＵ方式とＦＩＦＯ方式の性能差を解析したものは見当らない。その理由の１つは，式（6.1）で表現されるような局所参照性を有するデータにＦＩＦＯを適用した時の数学的な扱いが難しいためである。一方，実システムによる実験例としては，簡易ＬＲＵ方式であるＦＩＮＵＦＯとＦＩＦＯのフォールト率の比が１：2.2[14]あるいは１：1.3[01]であったという報告などがある。

ここでは，性能差の理論式を求める代りに，その上界式と下界式を導くことにより，両者の性能差が図6.5(a)のようにフォールト率の低い所で大きくなることを示す。

今，同じページ参照列に対してＬＲＵおよびＦＩＦＯによるページング処理をしているとして，ある時点でのバッファ上のページ集合G_{LRU}，G_{FIFO}の間に，

$$G_{FIFO} = G_{LRU} - \{ u_1, \cdots, u_k \} + \{ v_1, \cdots, v_k \} \qquad (6.9)$$

の関係があるとする。ここでu_iはG_{LRU}にのみ存在するページ，v_iはG_{FIFO}にのみ存在するページである。この時点でページフォールトが生じたとすると，追い出すページは，ＬＲＵではバッファページ数をｂとしてＬＲＵスタックのｂ番目のページであるが，ＦＩＦＯでは最初にロードされたページであり，それをｑとする。この時，ＬＲＵおよびＦＩＦＯのフォールト率f_{LRU}とf_{FIFO}の差ｄは，各々のバッファ内のページの参照確率の和の差に等しいので，ページｘの参照確率をｐ（ｘ）とすると，

$$d = f_{FIFO} - f_{LRU}$$

$$= \sum_{i=1}^{k} p(u_i) - \sum_{i=1}^{k} p(v_i) + p(q) - L(b+1) \qquad (6.10)$$

となる。Ｌ（ｂ＋１）は，ＬＲＵの方で今追い出されたばかりの，スタックのｂ＋１番目のページの参照確率である。ｑは，ＬＲＵスタック内の位置とは無関係に選ばれることから，G_{FIFO}内のページの平均参照確率を持つと考えられるので，

$$p(q) = \frac{1}{b} \left\{ \sum_{i=2}^{b+1} L(i) - \sum_{i=1}^{k} p(u_i) + \sum_{i=1}^{k} p(v_i) \right\} \qquad (6.11)$$

となる。一方，式（6.1）より，

$$p(u_i) > p(v_j), \quad 1 \leq i, j \leq k \qquad (6.12)$$

が成立つので，式（6.10），（6.11），（6.12）から，ｄの下界式として，

$$d \geq \frac{1}{b} \sum_{i=2}^{b+1} L(i) - L(b+1) \qquad (6.13)$$

が導かれる。

次に上界を求める。ＦＩＦＯにおいて，あるページｘがロードされ，ｂ回のページフォールト後に追い出されるまでにこのページｘが参照される回数をn_x，全参照回数をｎとすると，

$$f_{FIFO} = b / n, \qquad (6.14)$$

$$p(x) = n_x / n, \qquad (6.15)$$

となる。ここでページｘに注目すると，１回のページフォールトに対してn_x回はフォールトが生じないため，

$$f_{FIFO} = 1 / (n_x + 1) \qquad (6.16)$$

と考えて良い。そこで，式（6.14），（6.15），（6.16）から，

$$f_{FIFO} = 1 - bp(x) \qquad (6.17)$$

となる。これは，バッファ内のページの平均参照確率をｐ（ｘ）とした時のフォールト率を意味している。

ここで，ｐ（ｘ）の下界を求めるために，ページｘがバッファ内に存在する間にＬＲＵスタックのどこまで下がり得るかを考える。まず最悪の場合として，最初にｘがロードされた時にバッファ内にあった他のページはすべて，追い出される前に参照されたとすると，これらのページが上にくるので（ｂ－１）段下がる。さらに，（ｂ－１）回のフォールトで（ｂ－１）段下がる。結局，最大２（ｂ－１）段下がる。そこで，ｐ（ｘ）はこの場合の平均参照確率以上と考えられるので，

$$p(x) \geq \frac{1}{2b-1} \sum_{i=1}^{2b-1} L(i) \qquad (6.18)$$

となる。一方，f_{LRU}は，

$$f_{LRU} = 1 - \sum_{i=1}^{b} L(i) \qquad (6.19)$$

である。そこで，ＬＲＵとＦＩＦＯのフォールト率の差の上界式は，式（6.17），（6.18），（6.19）から，

$$d \leq \sum_{i=1}^{b} L(i) - \frac{b}{2b-1} \sum_{i=1}^{2b-1} L(i) \qquad (6.20)$$

となる。

以上，式（6.13）および（6.20）で求めた上下界式を式（6.2）に適用すると，ＬＲＵとＦＩＦＯの性能差ｇの上下界式は共にフォールト率の低い所で単調減少になることが導かれるが，その詳細は付録１に述べる。この結果は，フォールト率の低い所ほど相対的性能差が大きくなっている図6.5(a)の実験結果を裏づけている。

(4) 簡易ＬＲＵ方式間の性能差

ＬＲＵ方式はページ参照系列に局所参照性がある場合に適した方式であるが，スタック処理に時間を要するためにあまり実用的ではない。そこで実際にはＦＩＮＵＦＯやＦＩＶＥなどの簡易方式が採用されている。これらの簡易方式は，ページ参照系列の履歴情報の保存量を限定することにより実行効率を良くしたものであるが，この限定は，履歴情報を収集する期間の短縮と情報量の削減という形で行われる。従って，ここではこれらの保存量の限定度合と簡易方式間の性能差に関する図6.5の実験結果との関係について解析する。

まず，保存する情報量についてみると，ＬＲＵでは，主記憶バッファ内のすべてのページについて，どれがより最近に参照されたものかという順序が記憶されている。一方，ＦＩＮＵＦＯでは，ある一定の期間内に参照されたか否かの情報だけ記憶している。また，ＦＩＶＥでは，最も最近に生じた5回のフォールトの各々の間で参照されたか否かの情報を記憶している。従って，情報量は，ＬＲＵ，ＦＩＶＥ，ＦＩＮＵＦＯの順に多い。

次に，保存する情報の収集期間Ｔを求める。Ｔはページ参照回数で表現するのが直観的で良いが，ここでは比較容易性を考慮してその間に生じたフォールト数で表現する。Ｔはフォールト率の関数になるので，参照回数の代りにそれにフォールト率を乗じたフォールト数を用いても一般性は失われない。

まずＬＲＵの場合，ある時点でのＬＲＵスタックの状態は，主記憶バッファページ数をｂとして，ｂ回前のページフォールト以降のページ参照系列によって決まる。なお，このことは，それ以前に参照されてその後は参照されていないページは既にバッファ内に存在しないこと，および最も最近のｂ回のページ参照がすべてフォールトの場合にはｂ回前のページフォールトでロードされたページは深さｂのスタックの底に記憶されていることなどから証明できるが，詳細は略す。従って，

$$T_{LRU} = b \qquad (6.21)$$

となる。

次に，ＦＩＮＵＦＯの参照フラグは，6.3.1（ｂ）項で述べた周期的なスキャンの間に参照されたものだけ記憶している。そして，その周期の間に生じるフォールト数はバッファ内のページでフラグがオフのものの数に等しいので，そのオフのフラグの割合をｒとすると，

$$T_{FINUFO} = rb \qquad (6.22)$$

となる。ｒは，定常状態ではオンからオフに変わるフラグとオフからオンに変わるフラグの数が等しいことから，次式を解いて求めることができる。

$$\frac{b(1-r)}{br} \cdot f \leq r(1-f) \qquad (6.23)$$

左辺は，フォールト時のスキャンでオフのフラグに出会うまでにオンからオフに変えられるフラグの数にフォールト率を乗じたものである。右辺は，参照されてオンにするフラグがその前にオ

フである確率δをrで近似し，それにフォールトの生じない確率を乗じたものである。なお，δは実際には局所参照性のためにrより小さいので不等式になっている。結局，式（6.21），（6.22），（6.23）から，

$$T_{FINUFO} \geq T_{LRU} \cdot \frac{\sqrt{f(4-3f)}-f}{2(1-f)} \qquad (6.24)$$

となる。

最後に，ＦＩＶＥの場合は，6.3.1（ｄ）項で述べたように最も最近に生じた５回のフォールトの各々の間で参照されたか否かの情報を記憶しているので，

$$T_{FIVE} = 5 \qquad (6.25)$$

となる。

以上で３方式の履歴情報収集期間の一般式が導かれたので，相互比較を行う。まず，T_{LRU}とT_{FINUFO}はデータの局所参照性に依存するので，今回の実験式（6.6）と（6.7）に（6.8）のＥＮＶジョブの定数パラメータを適用すると，図6.8が得られる。なお，ＦＩＮＵＦＯの場合は不等式なので，同じＥＮＶジョブの場合の実測値も求めておいた。

この図6.8において，ＬＲＵと他の簡易方式の履歴情報収集期間の差はフォールト率の小さい所ほど大きく，図6.5(b)，(c)の実験結果と合致している。一方，ＦＩＮＵＦＯとＦＩＶＥを比べると，ＬＲＵフラグとしては前者が１ビット，後者が５ビット用いているにもかかわらず，フォールト率の小さい所で履歴情報収集期間は前者が上回る傾向にあり，図6.5の実験結果と合っている。これは，簡易ＬＲＵ方式の性能は，単に履歴情報保持のための記憶容量だけでなく，ＬＲＵフラグのリセットのタイミングに大きく影響されることを示している。

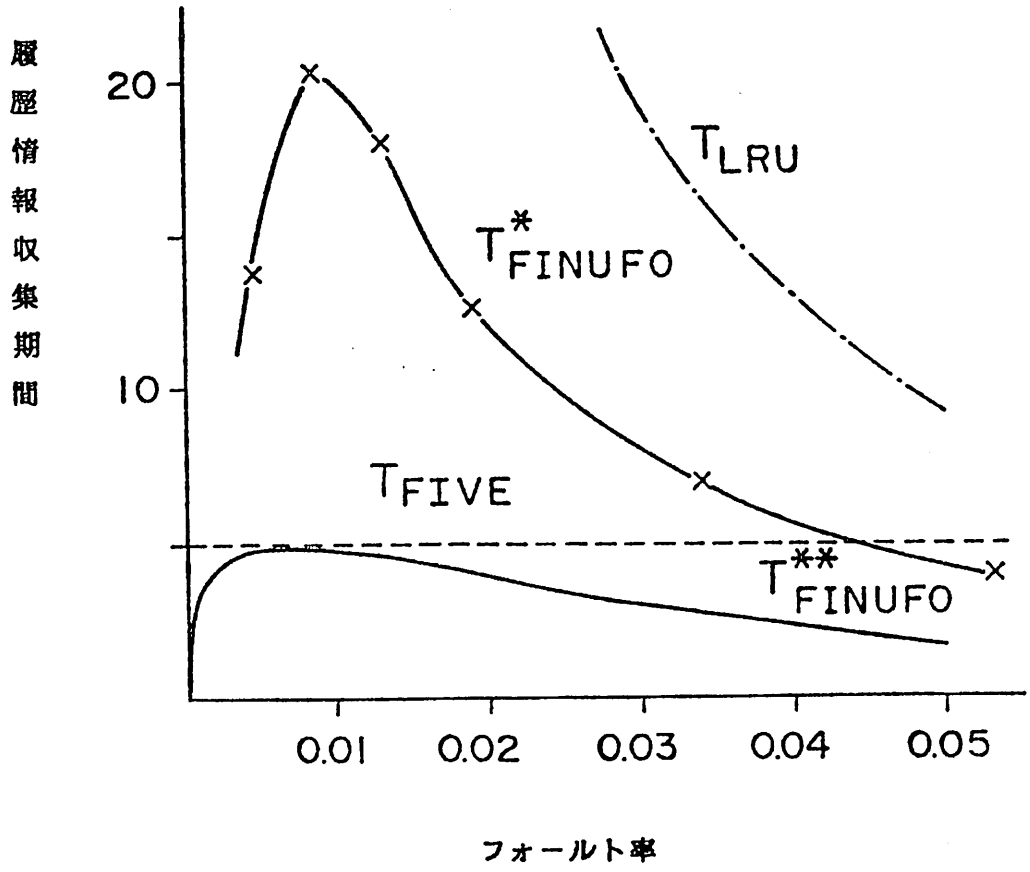
6.3.4　改良効果

最初に開発したＳＰＬコンパイラにおいては，ＳＰＬライブラリを対象としたページング処理にはインプリメンテーションの容易なＦＩＦＯ方式を採用し，ページサイズは５１２バイトとしていた。しかしながら，本節で述べたページングアルゴリズムの分析結果に基づき，次のような改良方式が導かれた。

(1)　ページングアルゴリズムをＦＩＦＯ方式からＦＩＮＵＦＯ方式に変更する。

(2)　ページサイズを５１２バイトから２５６バイトに変更する。

この方式の定量的な効果をみるために，6.2節でのコンパイラの処理性能を実測した時の稼動環境を考えると，主記憶容量６４ＫＷのうち８ＫＷ（１６ＫＢ）が主記憶バッファに割当てられた。この場合，コンパイル処理全体を通じてのページフォールト率は，(1)のページングアルゴリズムの変更により２０.２％の削減，(2)のページサイズの変更により２８.７％の削減がなされ，全体では，約５０％の削減効果が得られた。



＊ ： 実測値

＊＊ ： 理論式の下界値

図６．８　各アルゴリズムの履歴情報収集期間

6.4 結　言

ＳＰＬに特徴的な言語機能に対応して導入されたコンパイル技法がコンパイル効率に与える影響を分析するために，ＳＰＬコンパイラの性能解析を行った。

まず初めに，評価用プログラムを用いて各処理フェイズ毎のコンパイル時間を測定したところ，解析処理が全体の３／４を占めている一方で，展開・解釈処理は１０％程度と少なく，中間語形式の手続きのインライン展開処理が比較的高速処理されていることが確認された。

次に，各処理時間に占めるＣＰＵ時間や各種入出力処理時間の割合を分析した結果，解析処理に占める補助記憶装置との入出力時間比が他の処理フェイズの２倍近くになっていることから，分割コンパイルのために導入したＳＰＬライブラリへのアクセスが頻繁に行われていることが確認された。

そこで，ＳＰＬライブラリアクセス時のページング動作解析とそれに対するページングアルゴリズムの性能評価を行い，次のような実験結果を得た。

(1) ページサイズが小さいほどページフォールト率は低くなる。

(2) 手続き部を対象とした場合ほどではないが，明らかな局所参照性がある。

(3) 汎用ページングアルゴリズムの間では，ＬＲＵ，ＦＩＮＵＦＯ，ＦＩＶＥ，ＦＩＦＯの順に良い性能を示す。

そして，これらの結果に理論的解析を加え，次の結論を得た。

(1) 使用可能な主記憶容量一定の条件下での最適ページサイズは２５６バイトである。

(2) ＬＲＵとＦＩＦＯの性能差はフォールト率の低い所ほど大きい。

(3) ＬＲＵと簡易ＬＲＵ方式（ＦＩＮＵＦＯとＦＩＶＥ）の性能差はページ参照に関する履歴情報の収集期間に対応している。

以上の結果から，初版のＳＰＬコンパイラに対して，ページングアルゴリズムをＦＩＦＯからＦＩＮＵＦＯに変えること，およびページサイズを５１２バイトから２５６バイトに変えるという改良方式を得た。なお，今回の理論的解析では，一部に実験結果を用いて近似したところがあり，その部分の数学的処理は今後の課題である。