仮想メモリにおけるプログラム再構成の解析と最適化†

木下俊之†† 吉澤康文††

仮想メモリの性能向上方式のひとつであるプログラム再構成方式は，対象とするプログラムのモジュールを仮想メモリ上で再配置して，頻繁に参照されるものを同じページに割付けることにより，ページ・フォールトの発生を減少させる．そのため，プログラムの実行過程からモジュール参照の時系列を取得し，これから2つのモジュールの結びつきの強さ（コスト）を算出する．

本論文では，コストの算出法として有力なもののひとつである Critical Working Set 法（CWS 法）について確率モデルによる解析を行う．また，CWS 法を現実のプログラムに適用し，シミュレーションによりコスト算出法としての妥当性を検討すると共に，CWS 法で使用するパラメータの最適値を推定する一方法を考察する．

また，プログラム再構成方式における実用上の問題点として，1ページ・サイズより大きいモジュールには効果が上がらないという問題がある．この解決策として，各モジュールの中をさらに再構成する2段階プログラム再構成方式を提案した．これはサイズの大きいモジュール内の参照部分の偏りを改善するもので，シミュレーションにより適用結果を従来方式と比較して，その有効性を確認した．

1. まえがき

仮想メモリを用いたOSが一般的になった現在，その性能は計算機システム全体にとって大きな問題である．デマンド・ページング*による仮想メモリでは，ページング・オーバヘッドによるCPU利用率の低下や応答性の劣化，ならびにワーキング・セット**,[1]の増大に伴う実メモリ利用率の低下が，性能上の問題点としてあげられる．この仮想メモリの性能は，その上で動作するプログラムの性質にも依存し，実行中の一時期にはプログラム内の特定部分を主に参照するという，いわゆるプログラム局所性の高いプログラム程仮想メモリの性能は上る．したがって，プログラムの作成時に仮想メモリ上で動作することを考慮し，局所性の高いプログラムを作成することが最も望ましい．しかし，すでに開発済みのプログラムや，作成段階ではプログラム動作の予測ができないものなどについては，プログラムの作成後に仮想メモリ上でプログラムを再配置することにより局所性を高める方法が必要となる．これが，プログラム再構成方式である．

一般に性能向上の対象となるプログラムは，頻繁に参照されかつ容量的に大きなもの（例えば，コンパイラ，オンライン・プログラムなど）であり，複数個のモジュールから成り立っている．プログラム再構成は，これらのモジュールをひとつの単位として取扱う．そして，プログラムの実行過程を時系列的に表現するモジュール参照列を取得し，これから各モジュールの対ごとに同じページに割付けることの望ましさを定量的に表わすモジュール間結合度（以下，コスト[3]と呼ぶ）を算出する．次に1ページ・サイズの範囲内で，コストの大きいモジュール群をクラスタリング（グループ分け）し，このグループごとに新しく1ページに割付ける．このように，プログラム再構成はコストの算出とクラスタリングの2つの段階に分けることができる．

プログラム再構成の以上の手順は，Hatfield & Gerald[2], Ferrari[3], 益田・塩田[4]などによりすでに用いられている．このうち，Hatfield と Gerald はモジュール参照列中の隣り合ったモジュールを同じページに割付けることを目的とした 'nearness' と呼ばれるコストを提唱した．しかし，現実には直前に参照された以外のモジュールも主記憶に読込まれており，これらと次に参照されるモジュールを同じページに割付けてもページ・フォールトの発生は防げるはずである．このような観点から，Ferrari はコストの算出法としてモジュール参照列中のさらに過去に参照されたモジュールと次参照モジュールの関係に着目した Critical Working Set 法（以下，CWS 法と略す）を提案した．この方法は，Hatfield と Gerald の方法を特殊な場合として含み，さらにOSのもつページ・リプレー

† An Analysis and Optimization of Program Restructuring in Virtual Storage by TOSHIYUKI KINOSHITA and YASUFUMI YOSHIZAWA (Systems Development Laboratory, Hitachi, Ltd.).

†† (株)日立製作所システム開発研究所

* 仮想メモリの分割単位であるページを，必要に応じて主記憶と二次記憶間で転送する仮想メモリ方式．

** プログラムが同時に主記憶上に必要とするページの集合．

スメント・アルゴリズムの働きも考慮に入れた一般的な方法である．本論文では，このCWS法を確率的にモデル化して解析すると共に，適用例における数値データを基に，CWS法がコストの算出法としてどの程度妥当かを検証し，使用するパラメータの最適値を推定する一方法を提案する．

次に，プログラム再構成方式の実用上の問題点として，1ページ・サイズより大きいモジュールを含む場合の問題をとり上げる．現在の仮想メモリの1ページ・サイズは2〜4 KBで，この大きさを超えるモジュールは現実に数多く存在する．しかし従来のプログラム再構成方式では，これらのモジュールには効果が上らなかった．そこで，この解決策としてモジュールの内部をさらに再構成する2段階プログラム再構成方式を提案し，その根拠を述べ適用効果について解析する．

2. モジュール間結合度の解析

2.1 プログラム再構成とCWS法

解析に先だち，CWS法について説明する．ただし，用いる記号は以下の通りとする．

(1) $\{1, 2, \cdots, n\}$：モジュールの集合
(2) $(m_1, m_2, \cdots, m_t, \cdots, m_T)$：モジュール参照列
(3) T：モジュール参照列の長さ
(4) t：モジュール参照時点
(5) c_{ij}：モジュール i と j の間のコスト
(ただし，$1 \leq i < j \leq n$ とする．)
(6) $C=(c_{ij})$ $(i=1, 2\cdots, j-1 : j=2, 3, \cdots, n)$
：コスト行列

CWS法では，ウィンドウ・サイズと呼ばれるパラメータ τ を設定し，時間 $[t-\tau+1, t]$ の間に参照されるモジュールの集合をモジュール・ワーキング・セットと呼び $W(t, \tau)$ で表わす．そして，もし次参照モジュール m_{t+1} が $W(t, \tau)$ に含まれていなければ，この $W(t, \tau)$ をCritical Working Set (CWS) と呼ぶ．$W(t, \tau)$ がCWSである場合には，次の m_{t+1} の参照時にページ・フォールトが発生する可能性がある．しかし，もし m_{t+1} が $W(t, \tau)$ 内のモジュールと同じページに割付けられていれば，ページ・フォールトは発生しない．つまり，モジュール m_{t+1} と $W(t, \tau)$ に含まれるモジュールを同じページに割付けることは，ページ・フォールトの発生を抑制する効果がある．そこで，モジュール i と j の間のコスト c_{ij} は，$W(t, \tau)$ がCWSである場合に，$m_{t+1}=i$ かつ $j \in W(t, \tau)$ であるかまたは $m_{t+1}=j$ かつ $i \in W(t, \tau)$ となる回数として定義する．これにより，c_{ij} はモジュール i と j を同じページに割付ける望ましさを表わす尺度となることが分かる．

Hatfield & Gerald[2] による nearness matrix では，モジュール参照列中で2つのモジュール i と j が隣り合って参照される回数を c_{ij} と定義した．これは，CWS法において $\tau=1$ とした場合と同一である．すなわち，CWS法は nearness matrix のひとつの拡張であり，パラメータ τ によりモジュール参照列中の過去に参照されたモジュールとの関係を観察する範囲を広げたものである．(この効果については後述する．) しかも，単に時間的に近い範囲で参照されたモジュールにコストを計上するのではなく，CWS を導入してページ・フォールトをモジュールのレベルで模擬することにより，OS のページ・リプレースメント・アルゴリズムの働きを十分に考慮していることが大きな特徴である．

2.2 確率モデルによる解析

CWS法のコストの算出でもとになるものはモジュール参照列であるが，その背景には命令の参照がある．そこで，命令参照列が定常マルコフ連鎖をなすものと仮定して，CWS法を確率的にモデル化して解析することを試みる．この仮定と現実のプログラム動作の違いについては後述する．

モジュール参照列中のモジュール i の出現確率を p_i，モジュール i から j への遷移確率を p_{ij} とおく．すなわち，任意の時点 t について，

$$p_i = P\{m_t = i\}$$
$$p_{ij} = P\{m_{t+1} = j \mid m_t = i\}$$

とおく．すると，次参照モジュール m_{t+1} が i であるようなCWSの発生確率 $r_i(\tau)$ は，CWSの定義により

$$\begin{aligned} r_i(\tau) &= P\{i \notin W(t, \tau),\ m_{t+1} = i\} \\ &= P\{m_{t-\tau+1} \neq i,\ m_{t-\tau+2} \neq i, \\ &\quad \cdots, m_t \neq i,\ m_{t+1} = i\} \\ &= P\{m_{t-\tau+1} \neq i\} \\ &\quad \times P\{m_{t-\tau+2} \neq i \mid m_{t-\tau+1} \neq i\} \times \cdots \\ &\quad \times P\{m_t \neq i \mid m_{t-1} \neq i\} \\ &\quad \times P\{m_{t+1} = i \mid m_t \neq i\} \end{aligned} \tag{1}$$

と表わすことができる．この各因数は p_i, p_{ij} を用いて次のように表現される．

$$P\{m_{t-\tau+1} \neq i\} = 1 - p_i \tag{2}$$

$$P\{m_{t+1}\neq i \mid m_t \neq i\}=1-\frac{p_i(1-p_{ii})}{1-p_i} \quad (3)$$

(この値を q_i と書く.)

$$P\{m_{t+1}=i \mid m_t \neq i\}=\frac{p_i(1-p_{ii})}{1-p_i} \quad (4)$$

以上により,

$$r_i(\tau)=q_i^{\tau-1}\cdot p_i(1-p_{ii}) \quad (5)$$

が求められる. したがって一般の CWS の発生確率 $r(\tau)$ は,

$$r(\tau)=\sum_{i=1}^{n} r_i(\tau)=\sum_{i=1}^{n} q_i^{\tau-1}\cdot p_i(1-p_{ii}) \quad (6)$$

である. このように命令参照を定常マルコフ連鎖と仮定すると, $r(\tau)$ は指数関数の線形結合として表わされる.

次に, $m_{t+1}=i$ でありかつ j を含むような CWS の出現確率 $r_{ij}(\tau)$ を求める. そのため, まず j を含まないものの出現確率から求める. これは,

$$\begin{aligned}&P\{i, j\notin W(t,\tau),\ m_{t+1}=i\}\\&=P\{m_{t-\tau+1}\neq i, j\}\\&\quad\times P\{m_{t-\tau+2}\neq i, j \mid m_{t-\tau+1}\neq i, j\}\times\cdots\\&\quad\times P\{m_t\neq i, j \mid m_{t-1}\neq i, j\}\\&\quad\times P\{m_{t+1}=i \mid m_t\neq i, j\}\end{aligned} \quad (7)$$

と表わされるが, このうち,

$$P\{m_{t-\tau+1}\neq i, j\}=1-p_i-p_j \quad (8)$$

$$\begin{aligned}&P\{m_{t+1}\neq i, j \mid m_t\neq i, j\}\\&=1-\frac{p_i-p_ip_{ii}-p_jp_{ji}}{1-p_i-p_j}-\frac{p_j-p_jp_{jj}-p_ip_{ij}}{1-p_i-p_j}\end{aligned}$$

(この値を q_{ij} と書く.) (9)

$$P\{m_{t+1}=i \mid m_t\neq i, j\}=\frac{p_i-p_ip_{ii}-p_jp_{ji}}{1-p_i-p_j} \quad (10)$$

であるから,

$$\begin{aligned}&P\{i, j\notin W(t,\tau),\ m_{t+1}=i\}\\&=q_{ij}^{\tau-1}\cdot\{p_i-(p_ip_{ii}+p_jp_{ji})\}\end{aligned} \quad (11)$$

ゆえに, 式(5)と(11)により

$$\begin{aligned}&r_{ij}(\tau)=P\{i\notin W(t,\tau),\ j\in W(t,\tau),\ m_{t+1}=i\}\\&=P\{i\notin W(t,\tau),\ m_{t+1}=i\}\\&\quad-P\{i, j\notin W(t,\tau),\ m_{t+1}=i\}\\&=q_i^{\tau-1}\cdot p_i(1-p_{ii})\\&\quad-q_{ij}^{\tau-1}\cdot\{p_i(1-p_{ii})-p_jp_{ji}\}\end{aligned} \quad (12)$$

となる. 一方, CWS 法によるコスト c_{ij} はモジュール i と j の一方を次参照モジュールとし他方を含むような CWS の発生回数を表わしているから, この $r_{ij}(\tau)$ を用いて,

$$c_{ij}=T\times\left\{\sum_{j=1}^{n} r_{ij}(\tau)+\sum_{i=1}^{n} r_{ji}(\tau)\right\} \quad (13)$$

となることが分かる. 式(12)の形からも明らかなように, $r_{ij}(\tau)$ は一般に τ について単調減少であるとは限らず, また単調減少でない場合はひとつの極大値をもつ曲線を描くことが証明できる. したがって, c_{ij} も同様の曲線を描いて変化すると考えられる.

以上のように, 命令参照列を定常マルコフ連鎖と仮定することにより, CWS 法における各種の統計量を p_i と p_{ij} から計算することが可能である.

2.3 最適パラメータの範囲

CWS 法では, モジュール・ワーキング・セットを規定するパラメータとして, ウィンドウ・サイズ τ を使用する. この τ の最適値は対象とするプログラムの動作に依存しており, すべてのプログラムに共通の値は存在しない. しかし, 個別のプログラムについては, より適切な τ の値の範囲がある.

τ を小さくした最も極端な場合は $\tau=1$ であるが, これは nearness matrix[2] に一致する. この nearness matrix と τ を適当に大きくした場合の CWS 法のコスト行列とでは, 後者を用いたプログラム再構成の方が優れており, これは Ferrari[3] が既に述べている通りである. τ の値が小さければ, ひとつのモジュール参照についてそれ以前に参照されたモジュールとの関係を観察する範囲は狭い. これを CWS 法の立場から見ると, 本来 CWS 法は, ページ参照におけるページ・フォールトをモジュール参照のレベルで模擬するものであるから, τ を小さく設定すると CWS が過剰に発生して, 現実のページ・フォールトの発生とのずれが大きくなる. すなわち τ を過小に設定することは, 現実のページ・リプレースメント・アルゴリズムとの関連で, プログラム再構成の効果を低下させる要因となる.

逆に τ の上限について考える. ウィンドウ・サイズを τ とした時の CWS の発生数を $N(\tau)$ と書くと, $N(\tau)$ は τ について単調減少である. つまり,

$$N(\tau)\searrow 0 \quad (\tau\nearrow\infty) \quad (14)$$

また, 一般にウィンドウ・サイズが増大するとページ・ワーキング・セットは飽和するが, モジュール・ワーキング・セットにもこれと同様の性質がある. そこで, この飽和点を τ_w と書くと,

$$\tau_w<\tau \Rightarrow W(t,\tau) \text{ は } \tau \text{ についてほぼ一定} \quad (15)$$

となる. これよりひとつのモジュール参照列において,

$\tau_w < \tau \Rightarrow N(\tau)$ は τ についてほぼ一定 (16)

(14)と(16)により

$\tau_w < \tau \Rightarrow N(\tau) \fallingdotseq 0$ (17)

となることが分かる．すなわち，$\tau_w < \tau$ の時は CWS はほとんど発生せず，したがってコスト行列の成分の大部分は0である．コストは2つのモジュールを同じページに割付ける望ましさの尺度であるから，これがほとんど0であれば有効なクラスタリングはできない．すなわち，$\tau_w < \tau$ では効果的なプログラム再構成を行うことはできない．

このように，τ は過小でも過大でも不適当であり，最適値はある一定範囲内に存在する．

3. 適用例と結果

3.1 対象プログラムと適用条件

プログラム再構成方式は，比較的規模の大きいプログラムで（数十ページ以上），かつ再配置可能なモジュールに分割されているものに適用するのが最も有効である．これには，コンパイラやオンライン・プログラムなどがある．特にオンライン・プログラムでは，ひとつのモジュール内における滞留ステップ数が短く，モジュール遷移が頻繁に起きるので，プログラム再構成の効果は高い．ここでは，オンライン実時間システムにおいて通信管理プログラムとユーザ・プログラムのインタフェースをとるオンライン制御プログラムにプログラム再構成を適用した．このプログラムは，全容量が約 174 KB (=86 ページ)，全モジュール数は 63 モジュールである．後に述べる2段階プログラム再構成方式を適用する際には，1ページ・サイズより大きい 11 モジュールをさらに細くセクションに分割し，1モジュール＝1セクションのものを含めて，全セクション数は 480 セクションとした．

セクション参照列はセクション数 7,628 個のものを用いたが，この中には対象プログラムの範囲外でもページングの対象となる部分の参照を含んでいる．しかし，OS の常駐部などのページングと無関係な部分の参照は除外した．

3.2 再構成の尺度

再構成効果の評価のために，OS のページ・リプレースメント・アルゴリズム（以下，PRA と略す）を Least Recently Used 法および Working Set 法（以下，LRU 法，WS 法と略す）と仮定した場合のページ・フォールト発生数を，命令トレース結果を入力とするシミュレーションにより求めた．このシミュレーションを再構成前と再構成後の状態について行い再構成の効果を比較評価した．評価指数としては，各シミュレーションについて次の値を用いた．

α: PRA を LRU 法と仮定した時のページ・フォールト発生数の再構成前に対する再構成後の比率

β: PRA を WS 法と仮定した時のページ・フォールト発生数の再構成前に対する再構成後の比率

この定義から，α と β は1より小さい程プログラム再構成の効果は上がったことになる．この2つの指数は，一方が良ければ他方も良い値となり互いに強い相関がある．また各シミュレーションは，仮定する主記憶容量もしくはウィンドウ・サイズを変えて行ったが，ひとつの再構成結果に対する α と β の値としては，それらの平均値を用いた．

3.3 確率モデルとの比較

まず，**2.2** 節の確率モデルが現実の状態をどの程度記述しているか比較する．**図1,2**は，対象としたプログラムの一部約 8 KB を 49 個のモジュールに分割し，CWS の発生数 $N(\tau)$ ならびにコストの総和 $\Gamma(\tau)=\sum_{i<j} c_{ij}$ について調べたものある．

$N(\tau)$ が τ に関して単調減少することは，測定値については $W(t,\tau)$ が単調増大であることから，計算値については式(6)において $q_i<1$ であることから明らかである．また $\Gamma(\tau)$ の測定値がひとつの極大値をもつ曲線を描くことは，$N(\tau)$ は単調減少し，モジュール・ワーキング・セット・サイズは単調増加するという相反する変化の相乗効果によるものである．

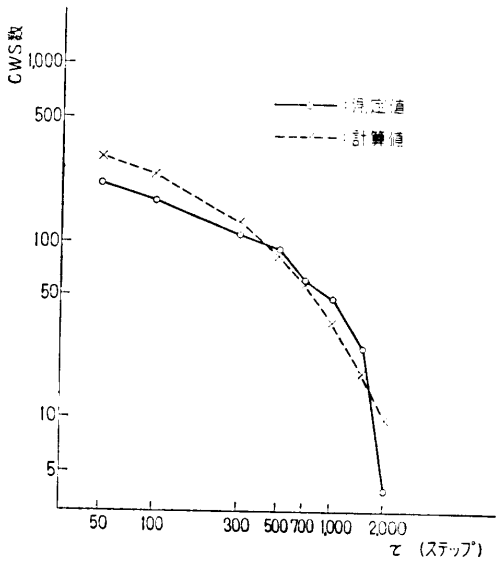


図 1 Critical Working Set の発生数
Fig. 1 Number of Critical Working Sets.

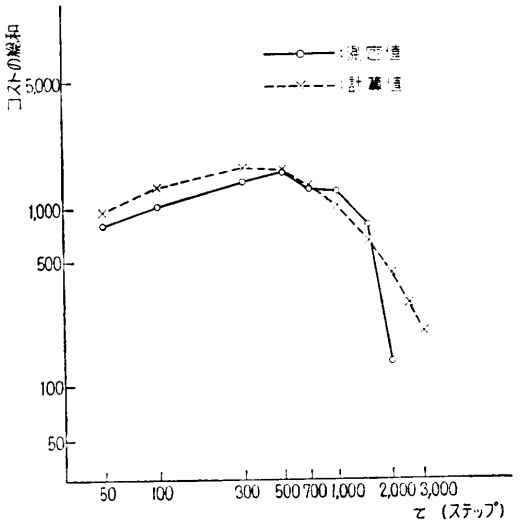


図 2 コストの総和
Fig. 2 Total cost.

確率モデルでは命令参照に定常マルコフ性を仮定したが，現実には局所性によりプログラムの実行過程はいくつかの phase に分かれている．そして，ひとつの phase ではプログラム中の特定の部分を頻繁に参照し，別の phase では異なる部分を参照する傾向が強い．特に τ の値が小さい場合には，モジュール・ワーキング・セットにはひとつの phase のモジュールのみが含まれている．そして，次参照モジュールもこれと同じ phase 内のものであることが多いから，CWS の発生数は定常マルコフ性を仮定した計算値よりも小さく測定される．この事実は，図1,2において裏付けられている．

3.4 コストの妥当性

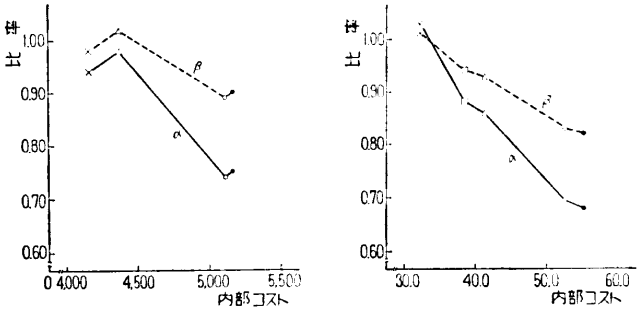
コストはプログラム再構成の次の段階であるクラスタリングにおいて，どのモジュールを同じページに割付けるかを決定する規準となるもので，その妥当性は大きな問題である．すなわち，クラスタリングではコストの値が2つのモジュールを同一ページに割付ける望ましさを正しく表わしているものと仮定して，同一ページに含まれたモジュール間のコストの総和（以下，内部コストと呼ぶ）がでる限り大きくなるような割付けを行う．したがって，コストは内部コストが大きいクラスタリング程再構成の効果が上がるという性質をもっていなければならない．

そこで，モジュールのページへの割付け方法として内部コストの異なる数種類の割付けを行い，それぞれの内部コストと再構成の効果の間の相関を調べた（**図 3**）．τ=5,000 の場合は内部コストの増加に伴ない α,β はほぼ線形に減少しており，コストとして必要な上記の性質を備えている．すなわち，τ の値を適切に設定すれば，CWS 法はコストの算出法として十分妥当なものになる．一方，τ=1 の場合は内部コストが増加しても α,β は減少するとは限らず，これはコストとして好ましいとは言えない．τ=1 の場合は nearness matrix になるから，結局 CWS 法はパラメータ τ を導入して nearness matrix を拡張・改良した方法であると言える．

3.5 最適パラメータの推定

パラメータ τ については **2.3** 節でとり得る値の範囲を調べたが，ここでは再構成以前に予め最適な τ を推定する一方法を考察する．

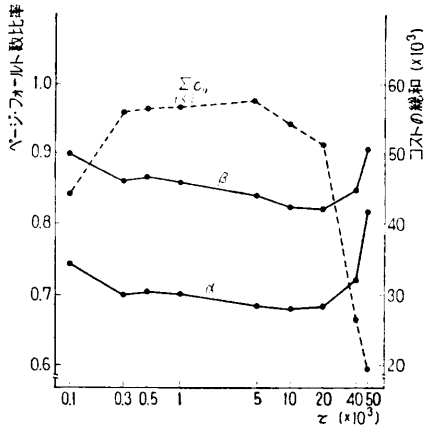
図 4 はコストの総和 $\Gamma(\tau)$ の τ による変化と再構成の効果 α,β を重ねて書いたものである．これによると，τ<300 では過小であり τ>40,000 では過大である．（このプログラムでは τ_w=55,000 である．）しかし，その間の 300≦τ≦40,000 では α と β はほぼ横ばいである．したがって，この比較的広い範囲にわたって再構成の効果に大きな差はなく，τ の値としてこの範囲内の適当な値を用いて良い．この範囲は $\Gamma(\tau)$ が 50,000 以上の大きな値を有する τ の範囲とほぼ同一である．この理由は，$\Gamma(\tau)$ が大きければコスト行列は2つのモジュールを同じページに割付ける望ましさをより細く分類しており，クラスタリングの基準値としてより敏感な行列となっているためである．このように，$\Gamma(\tau)$ により α,β の変化を推定することができる．したがって，τ の最適値としては，まず $\Gamma(\tau)$ を式(13)より求め，これが最大となる範囲から τ を選べ



(a) τ=1　　(b) τ=5,000

α: LRU 法による再構成前後のページ・フォールト数比率
β: WS 法による再構成前後のページ・フォールト数比率

図 3 内部コストと α,β の関係
Fig. 3 Relationship between internal-cost and effectiveness of restructuring.



α: LRU 法による再構成前後のページ・フォールト数の比率
β: WS 法による再構成前後のページ・フォールト数の比率

図 4 α, β ならびにコストの総和とウィンドウ・サイズの関係
Fig. 4 Effectiveness of restructuring and total cost vs. window-size τ.

ば良い．$\Gamma(\tau)$ の計算値と現実の測定値には多少のずれがあるが，最適な τ の許容範囲が相当広いので，最適値を推定する上にはほとんど差し支えない．

4. 2段階プログラム再構成方式

4.1 プログラム再構成適用時の問題点

プログラム再構成方式は，同一ページに割付けるモジュールを変更することにより，プログラム局所性を高めるものである．したがって，各モジュールの大きさは1ページ・サイズより小さいことが前提である．(1ページ・サイズの 1/3〜1/10 が望ましいとされている[2]．) しかし現実のプログラムの中には，1ページ・サイズより大きいモジュールを含むものが多い．このようなモジュールにはそのままでは再構成の効果を望むことはできない．ところが，この大きなモジュールはたとえ数は多くなくてもプログラム全体の相当部分を占め，参照頻度も高いので，何らかの対策を講ずる必要がある．

この問題に対処するために，既成のモジュールにとらわれずに，モジュールを1ページ・サイズ以下のさらに細い単位に分割する．便宜上これをセクションと呼ぶ．そして，このセクションを単位としたプログラム再構成を行う．したがって，セクション分割は同一のモジュール内で別の記号アドレスに無条件分岐する箇所で行う．また必要ならば無条件分岐命令を挿入して分割する．そして再構成後に再びコンパイル（アセンブル）し直す必要がある．

この場合，もし全セクションを一斉に再構成すると，異なるモジュールのセクションが混じり合ってモジュールは不連続なものになる．すると，再配置可能な単位であるというモジュール本来の性質はくずれる．そこで，セクションの再構成は同一モジュールの中でのみ行うこととし，その後モジュール全体を再構成する方法を提案した．これを2段階プログラム再構成方式と呼ぶ．以下，この方式について提案した根拠とその適用効果について論ずる．

4.2 2段階プログラム再構成の根拠

プログラム再構成では，再構成の最小単位内部の局所性には全く関知できない．益田と塩田[5]によると，一般にサイズの大きいモジュールには複数の機能が含まれており，プログラム実行時のひとつの phase ではその一部分が利用されていて，全体的にはメモリ使用効果率が低下するとの報告がある．本論文で対象としたプログラムにおいても，セクションに分割してみるとモジュール内部の参照状態にはかなりの偏りがあることが分かった（**図 5**）．本来このようなモジュールは，個々の機能ごとに分割して作成することが，局所性を保つ上には最も望ましい．にもかかわらず，これらをひとつのモジュールにまとめたことは，モジュール内部の局所性を弱める結果を招いている．ここに各モジュール内でセクションを再構成する可能性があり，2段階プログラム再構成方式は，これとモジュール全体の再構成を併用するものである．したがって，セクションはひとつのモジュールに含まれる複数の細い機能単位に分割すれば良いことがわかる．

図5によると，未参照のセクションがあるが，これらは実質的には再構成の対象とはならず，単に一箇所に集められるだけである．これを実効的に再構成する

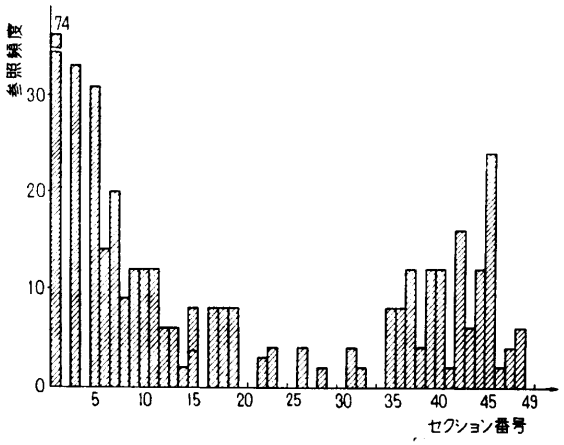


図 5 セクション参照の頻度
Fig. 5 Frequency of section refferences.

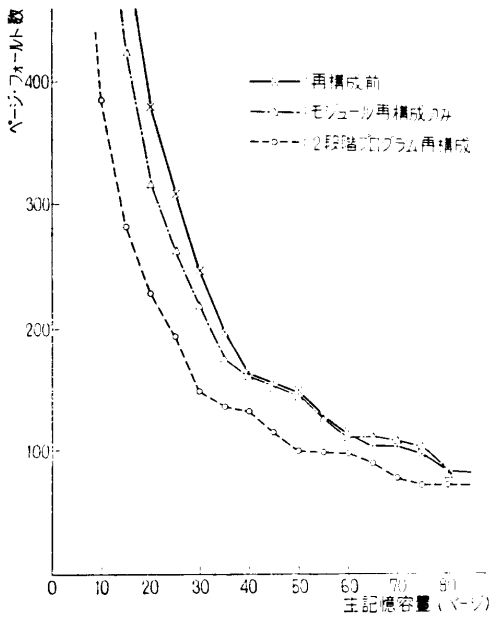


図 6 2段階プログラム再構成の効果
Fig. 6 Effectiveness of 2-step restructuring.

には，これらを参照する別のセクション参照列が必要である．

4.3 適用結果

先に述べたオンライン制御用プログラムを2段階プログラム再構成し，命令トレースを入力とするシミュレーションによりページ・フォールトの発生数を調べた結果を**図 6** に示す．これから分かる通り，2段階の再構成を行う効果がはっきり現れている．特に，主記憶容量が 40 ページ以上の場合にはモジュールの再構成だけでは効果がなかったが，2段階にすることによりこの範囲でもページ・フォールトの発生数を 32～12% 減少させることができた．

セクション再構成が有効となるのは，これを行うモジュールが頻繁に参照され，かつモジュール内部の局所性が低い場合である．対象としたプログラムでは，セクション再構成を行った 11 個のモジュールのサイズは合計で 63 KB（全体の 36%）であるにもかかわらず，これらのモジュールの参照回数は 6,327 回（全体の 83%）であった．すなわち，これらのモジュールは他の部分より単位メモリ量当たり8倍以上多く参照される部分である．しかも，同一セクション内の平均滞留ステップ数は 11.7 ステップと短く，ひとつのモジュール内で頻繁に分岐を繰り返している．したがって，セクション間の遷移が激しく，しかもこの遷移は従来の割付の方法では連続的な参照とはならないものが多かったことが分かる．また，11 個のモジュールを 428 個のセクションに分割し，セクション・サイズを平均 131 バイトと細く行ったため，セクション再構成の自由度が十分に高かったことも効果を上げるのに役立ったと考えられる．

5. むすび

プログラム再構成方式におけるコスト算出の一方法である CWS 法について，確率モデルを用いた解析とパラメータ τ による再構成効果の解析を行った．これにより，CWS 法の諸性質を明らかにすると共に，その有効性を確認し最適な τ の推定方法を考察した．確率モデルには命令参照を定常マルコフ連鎖と仮定し，モジュール参照確率とモジュール間遷移確率により CWS 法で使用する諸統計量の期待値を算出した．この結果は，τ の最適値の推定に応用可能である．

次に，プログラム再構成を適用する際の問題点である1ページ・サイズより大きなモジュールに対する解決策として，2段階プログラム再構成方式を提案した．この方法は，サイズの大きいモジュール特有の内部局所性の悪さを改善することが目的である．特徴は，再構成後のモジュールが不連続とならないことにある．対象とするプログラムの性質にも依るが，この2段階プログラム再構成方式により再構成の効果に明らかな向上が見られた．今後は，再構成後のプログラム修正に対する硬直性や，モジュールを仮想メモリ上で詰めて再構成することの影響などの．実用上の問題点について検討を続ける予定である．

最後に，本研究に対して貴重なご助言を頂いた筑波大学益田隆司助教授，ならびにデータ取得にご協力頂いた(株)日立製作所ソフトウェア工場の方々に心から感謝の意を表します．

参考文献

1) Denning, P. J.: The Working Set Model of Program Behavior, Comm. of ACM, Vol. 11, No. 5, pp. 323-333 (1968).
2) Hatfield, D. J. and Gerald, J.: Program Restructuring for Virtual Memory, IBM Systems J., Vol. 8, No. 2, pp. 168-192 (1971).
3) Ferrari, D.: Improving Locality by Critical Working Sets, Comm. ACM, Vol. 17, No. 10, pp. 614-620 (1974).
4) 益田隆司，塩田博行：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，情報処理，Vol. 15, No. 9, pp. 662-669 (1974).

5) 益田隆司，塩田博行：仮想システムにおけるプログラムの局所性とその最大化，情報処理，Vol. 16, No. 12, pp. 1055-1063 (1975).
6) Denning, P. J.: Multiprogrammed Memory Management, Proc. of the IEEE, Vol. 63, No. 6, pp. 924-939 (1975).
7) Doob, J. L.: Stochastic Process, New York: Wiley (1953).

(昭和54年10月5日受付)
(昭和54年12月20日採録)