B3 計算機システムのシミュレーションとその効率について

金沢正憲，萩原　宏　（京都大学工学部）
北川　一　（京都大学大型計算機センター）

1. まえがき

一般に，計算機システムは，多くの確率変数やパラメータ（X）によってその動作環境が与えられる複雑な待ち行列プロセスであり，その動作状況が種々のパフォーマンス測度（P）として定まるような$P=F(X)$の関係と考えることができる．システム・パフォーマンスの解析のためのシミュレーションは，モデル化された計算機システムの動作をシミュレートして，過渡的および定常的な状態における入力Xと出力Pの関係Fを求めることになる．ところで，計算機システムのシミュレーションの効率を考える場合，問題となるのは，シミュレーションに要する計算時間と得られる結果の精度である．

このような意味での効率は，まずモデルの作成と重要な関係にある．実システムをモデル化する段階において，システム・パフォーマンスの隘路となる部分がモデルに十分反映されていること，一方，シミュレーションを簡単化するための種々の仮定を，求めようとする結果にできるだけ影響の少ない範囲内で設定することが大切である．このためには，シミュレーションの予備的段階としての実システムなどによる解析，シミュレーションの結果の解析からモデル作成へのフィード・バックなどが必要となろう．

また，シミュレーションと実システムの差異は，システム・パラメータの一部となる入力データと実際のデータの違いからも生じる．シミュレーションに限らず，有効な解析方法が得られたとしても，実用的な解析を行なうためには，種々の確率変数について実測されたデータの利用を考えねばならない．

つぎに，処理技法上の問題として，まず1回のシミュレーション・ランにおける計算時間と得られる結果の精度であるが，これはシミュレーションの初期条件および収束判定による打切り条件の与え方により，時間と精度をバランスさせることにある．いたずらに長くシミュレートしても無駄な計算をしていることになり，また短か過ぎても定常解は得られないことになり，必要十分な試行回数をどのように決めるかが問題である．

一方，計算機システムのシミュレーションでは，パラメータの種々の組合せと結果の関係，または最適結果を与えるパラメータの組合せなどを知ろうとする場合が多い．このような時は，通常，多大の計算時間が必要となるので，シミュレーションの効率として，全シミュレーション・ランに要する時間の短縮を考えることも有効となる．そのための手法として，パラメータの変更と初期条件の設定に工夫を行なって1回のシミュレーション・ランを短縮する方法，

Direct Search法による最適パラメータの探索，さらには，シミュレータに学習機能を取り入れて，パラメータの同じような組合せに対する無駄な計算を避ける方法などが考えられる．また，学習機能は，結果の精度の上で若干問題はあるが，シミュレーションの利用によるシステム・アイデンティフィケーションへの一つのアプローチにもなると考えられる．

われわれは，効率良いシミュレーションのやり方について，上のような手法を考察し，現在行なっているマルチプログラミング・システムのシミュレーションに取入れて実験を行なったので，それについて以下に述べる．

2. シミュレーションの効率について

2．1 時間と精度

(i) 収束判定と打切り

定常的状態を求めるシミュレーションの場合，出力される結果がどれぐらいの精度，あるいは揺ぎを持っているかを知ることは，結果の信頼性を吟味する点において重要であるとともに，効率の点からも必要なことである．前もって，シミュレーションの試行回数を決定されうる場合は，その必要がないであろうが，定常的状態を知るためのシミュレーションの場合には，一義的に試行回数を決定できないのが普通である．あらかじめ試行回数を決定しておくことは，必要以上の試行を行なうか，または，十分な精度の結果を得られないまゝで，シミュレーションを打切ることになるであろう．

シミュレーション$\boldsymbol{P}=\boldsymbol{F}(\boldsymbol{X})$において，第$i$回目の試行による$p_j$の値を$p_j^i$とすると，

$$p_j^i = f(\boldsymbol{X}, \boldsymbol{y}_i) \quad (\text{但し，} \boldsymbol{y}_i \text{は入力乱数を示すパラメータ}) \qquad (1)$$

と表わせる．すなわち，p_j^iはパラメータの組合せ$\boldsymbol{X}$で試行した時の入力乱数$\boldsymbol{y}_i$によって生じる揺ぎを含んでいる．この揺ぎをいかに判定するかということが問題となる．それには，次の確率・統計の理論を利用することができる．[1)]

〔中心極限定理〕

$\{x_i\}$は共通な分布をもった互いに独立な確率変数の系列で，$\mu=E(x_i)$，$\sigma^2=V_{ar}(x_i)$が存在すると仮定し，$S_n=x_1+\cdots\cdots+x_n$とすると，$(S_n-n\mu)/(\sqrt{n}\sigma)$の分布は，$n\to\infty$のとき正規分布$N(0,1)$に収束する．

つぎに正規母集団の平均値の信頼区間について考えてみる．

正規母集団$N(\mu,\sigma^2)$からの標本$(x_1, x_2, \cdots, x_n)$の平均値μ，ならびに分散σ^2の最尤推定量X，S^2は

$$\left.\begin{aligned} X &= (1/n)\sum_{i=1}^{n} x_i \\ S^2 &= (1/n)\sum_{i=1}^{n}(x_i - X)^2 \end{aligned}\right\} \qquad (2)$$

となる．また，不偏分散Vは

$$V = (1/(n-1))\sum_{i=1}^{n}(x_i - X)^2 \qquad (3)$$

となる．Xの区間推定を行なうために，新しい統計量として

$$T=(X-\mu)/(S/\sqrt{n-1}) \qquad (4)$$

を作れば，Tは自由度$\phi=n-1$のt分布に従い，この分布はμとSに無関係になる．$t_\phi(\varepsilon/2)$を自由度ϕのt分布の上側$(\varepsilon/2)$点とすれば，

$$P_r\{-t_\phi(\varepsilon/2)<T=(x-\mu)/(S/\sqrt{n-1})<t_\phi(\varepsilon/2)\}=1-\varepsilon \qquad (5)$$

となり，信頼上限および下限をμ_U，μ_Lとすれば，

$$\left.\begin{aligned}\mu_U&=X+t_\phi(\varepsilon/2)S/\sqrt{n-1}\\ \mu_L&=X-t_\phi(\varepsilon/2)S/\sqrt{n-1}\end{aligned}\right\} \qquad (6)$$

となる．すなわち

$$\mu\in\{X-t_\phi(\varepsilon/2)S/\sqrt{n-1},\ X+t_\phi(\varepsilon/2)S/\sqrt{n-1}\} \qquad (7)$$

なることが，危険率εでいえる．

以上の理論を系列p^i_jに適用し，シミュレーションによって得られるp^i_jをいくつかまとめて，系列$\{\overline{p}_k\}$を作り，この標本$(\overline{p}_1, \overline{p}_2, \cdots\cdots, \overline{p}_n)$によって区間推定を行なうことができる．

このような区間推定法により，前もって信頼区間の幅を決定しておけば，余分な試行を行なうことなしに，必要な精度の結果を求めることができる．

(ji) 初期設定とパラメータ変更

定常状態を求めるシミュレーションでは，その初期の状態の設定によって，区間推定による収束の度合いが変化する．例えば，待ち行列の長さが0で，使用中のものが全くない状態（0状態（empty & idle）と呼ぶ）からシミュレーションを開始すれば，立上がりの状態による揺ぎが区間推定の値に影響を及ぼすであろう．このように通常，シミュレーションの試行の最初で現われる過渡的な値による揺ぎを少なくするために，次のような処理方法が考えられる．

(イ) 最初から，何個かの結果を無視することによって，過渡的な値を取除く．

(ロ) パラメータの値を変更する場合に，毎回0状態にするのではなくて，そのまゝの状態にしておき，しかも，パラメータの値が少し変化するようなパラメータの組を次にシミュレートする．これにより，過渡的な値の揺ぎを小さくする．（Slide法と呼ぶ）

以上の2つの方策が考えられる．つぎに，過渡的な状態の値（異常値）が入ってくることによって，収束がいかに変わるかについて，信頼区間$t_\phi(\varepsilon/2)S/\sqrt{n-1}$をもとに，少し考えてみる．

正規母集団$N(\mu, \sigma^2)$の標本を$(x_1, x_2, \cdots\cdots, x_n)$とし，最尤推定平均値，分散値を$X$，$S^2$とする．今$x_{n+1}=X+kS$なる異常値が入れば，

$$\left.\begin{aligned}\hat{X}&\equiv(1/(n+1))\sum_{i=1}^{n+1}x_i=X+kS/(n+1)\\ V&\equiv(1/n)\sum_{i=1}^{n+1}(x_i-\hat{X})^2=(n+1+k^2)S^2/(n+1)\end{aligned}\right\} \qquad (8)$$

となり，信頼区間は，

$$t_n(\varepsilon/2)S\sqrt{(n+1+h^2)/(n+1)^2} \qquad (9)$$

となる．

ここで，第1回目の試行の結果が異常値で，第2回目以後の試行の結果は，$N(\mu, \sigma^2)$に従っているとすれば，0状態から開始した場合および第1回目の結果を無視した場合の収束判定による打切り回数をそれぞれn，m回とすれば，同じ信頼区間を得るためには次式を満足せねばならない．

$$t_{m-1}(\varepsilon/2)S/\sqrt{m-1} = t_{n-1}(\varepsilon/2)S\sqrt{(n+k^2)/n^2}$$

すなわち

$$n^2(t_{n-1}(\varepsilon/2)/t_{m-1}(\varepsilon/2))^2-(m-1)(n+k^2)=0 \qquad (10)$$

となり，$(n-m-1)$回多く試行せねばならない．

2．2 Direct Search法

あらゆるパラメータの組合せに対するシミュレーションを試みる必要がなく，最適（最大または最小）の結果を出力するパラメータの値を見出せば良い場合がある．このような場合に考えられる一つの方法が，Direct Search法である．Direct Search法とは，試みた結果を順番に吟味していくことを基本にした方法で，単純な比較という作業手続きによって，それ以後の試みるパラメータの値を決めて行く．

函数$F(\boldsymbol{x})$が領域$\boldsymbol{D}$で唯一つの極大（または極小）しか持たない時，最大（最小）値は，Direct Searchによって求められる．ここでは，その一つの方法として，Hooke & Jeevesの方法[2]を使用した．なお，以下では最大値を求めることにする．

Hooke & Jeevesの方法には，次に設定すべきパラメータの値を決定するアルゴリズムに，Exploratory moveとPattern moveと呼ばれる2つのものがある．

Exploratory moveのアルゴリズム．

(i) $i \leftarrow 1$，$V \leftarrow F(\boldsymbol{X})$，但し，$\boldsymbol{X} \equiv (x_1, x_2, \cdots\cdots, x_d)$

(ii) $x_i \leftarrow x_i + \Delta x_i$

(iii) もし，$F(\boldsymbol{X}) > V$ならば，$V \leftarrow F(\boldsymbol{X})$，$i \leftarrow i+1$として，(ii)へ飛ぶ

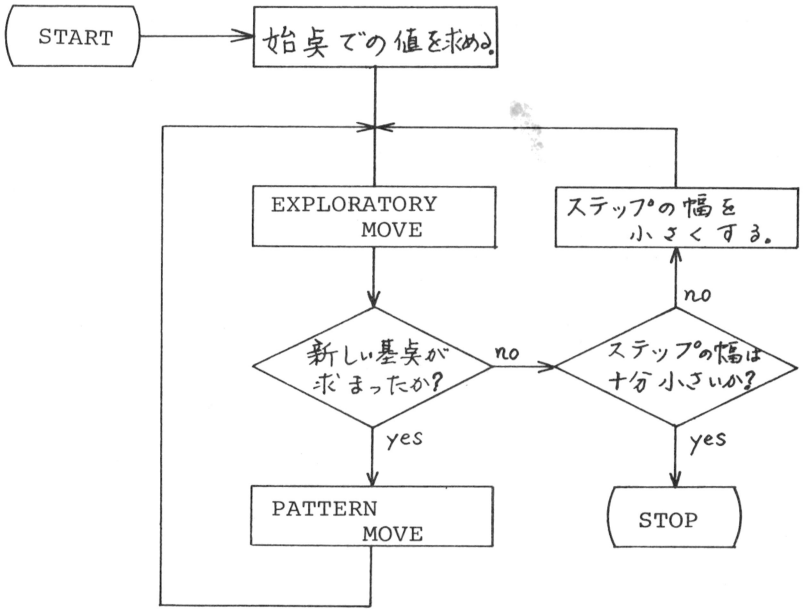
(iv) もし，$F(\boldsymbol{X}) \leqq V$ならば，$x_i \leftarrow x_i - 2\Delta x_i$とする．

(v) もし，$F(\boldsymbol{X}) > V$ならば，$V \leftarrow F(\boldsymbol{X})$，$i \leftarrow i+1$として，(ii)へ飛ぶ

(vi) もし，$F(\boldsymbol{X}) \leqq V$ならば，$x_i \leftarrow x_i + \Delta x_i$，$i \leftarrow i+1$として，(ii)へ飛ぶ

このアルゴリズムによって到達する点$\boldsymbol{x}^{\boldsymbol{B}}$は，基点と呼ばれる．また最初の点を始点と呼ぶ．

Pattern moveは，$\boldsymbol{x} = 2\cdot\boldsymbol{x}^{\boldsymbol{B}} - \bar{\boldsymbol{x}}^{\boldsymbol{B}}$によって点$\boldsymbol{x}$を求める．但し，$\boldsymbol{x}^{\boldsymbol{B}}$は現在の，$\bar{\boldsymbol{x}}^{\boldsymbol{B}}$は一つ前の基点である．そして，点$\boldsymbol{x}$が次のExploratory moveの始点になる．Hooke & Jeevesの方法のアルゴリズムをフロー・チャート形式で図－1に示す．



図－1　Direct Search Method

2．3　学習機能

シミュレーションの入力パラメータを$\boldsymbol{X}\equiv(x_1, x_2, \cdots\cdots, x_d)$，結果として求まるパフォーマンスの測度を$\boldsymbol{P}\equiv(p_1, p_2, \cdots\cdots, p_h)$とする時，このシミュレータによって表現されるシステムは，$\boldsymbol{P}=\boldsymbol{F}(\boldsymbol{X})$という関係でアイデンティファイされていることになる．$\boldsymbol{P}$で示されるシステム・パフォーマンスをR個のカテゴリーに分類し，パラメータ$\boldsymbol{X}$が与えられた時に，システムがどのカテゴリーに属しているかを判別できるような函数関係が求まれば，式によるシステム・アイデンティフィケーションが行なわれたことになる．そこで，シミュレータに学習機能を持たせ，種々のパラメータの組合せについてシミュレーションを行ないながら，システム・パフォーマンスのカテゴリーを識別できる函数を学習し，ある程度十分なシミュレーション・ランを繰返せば，その後は函数計算だけで属するカテゴリーを判別できるようなシステムを考えた．ここでは，次のような学習の方式を採用した．

カテゴリー1，2，……，Rについて，次のような識別函数$G_j(\boldsymbol{X})$，$j=1, 2, \cdots\cdots, R$を定める．

$$G_j(\boldsymbol{X})=w_{1j}f_1(\boldsymbol{X})+w_{2j}f_2(\boldsymbol{X})+\cdots\cdots+w_{Mj}f_M(\boldsymbol{X})+w_{M+1j} \qquad (11)$$

但し，$f_i(\boldsymbol{X})$，$i=1, 2, \cdots\cdots, M$は，重み$w_{1j}, w_{2j}, \cdots\cdots, w_{M+1j}$に対して独立な，パラメータ$\boldsymbol{X}$の函数であり，例えば，次のように与えられる．

(i)　線型函数

$$f_i(\boldsymbol{X})=x_k, \quad k=1, 2, \cdots\cdots, d \qquad M=d \qquad (12)$$

(ii) 二次函数

$$\left.\begin{aligned} &f_i(\boldsymbol{X}) = x_k^n x_l^m, \quad k, l = 1, 2, \cdots\cdots, d \ ; \ n, m = 0 \text{ および } 1 \\ &M = d(d+3)/2 \end{aligned}\right\} \qquad (13)$$

(iii) r 次多項函数

$$\begin{aligned} f_i(\boldsymbol{X}) &= x_{k_1}^{n_1} x_{k_2}^{n_2} \cdots\cdots x_{k_r}^{n_r}, \quad k_1, k_2, \cdots\cdots, k_r = 1, 2, \cdots\cdots, d \\ &\qquad n_1, n_2, \cdots\cdots, n_r = 0 \text{ および } 1 \end{aligned} \qquad (14)$$

$$M = \frac{(d+1)(d+2)\cdots\cdots(d+r)}{r!} - 1$$

今，一組のパラメータ$\boldsymbol{X} \equiv (x_1, x_2, \cdots\cdots, x_d)$が与えられた時，$G_j(\boldsymbol{X})$を$j = 1, 2, \cdots, R$について計算し，その中で最大値を取る$j$の値を，そのパラメータに対応するカテゴリーとして識別するとして，正しいカテゴリー識別が行なえるように重み$w_{1j}, w_{2j}, \cdots\cdots, w_{M+1j}$を訓練すればよい．

$$\left.\begin{aligned} &\boldsymbol{W}^{(j)} = (w_{1j}, w_{2j}, \cdots\cdots, w_{Mj}, w_{M+1j}) \\ &\boldsymbol{Y} = (f_1(\boldsymbol{X}), f_2(\boldsymbol{X}), \cdots\cdots, f_M(\boldsymbol{X}), 1) \end{aligned}\right\} \qquad (15)$$

とおくと，識別函数（11）式は，重みベクトル$\boldsymbol{W}^{(j)}$と付加パターン（パラメータ）ベクトル$\boldsymbol{Y}$の内積となっている．

$$G_j(\boldsymbol{X}) = \boldsymbol{W}^{(j)}\boldsymbol{Y} \qquad j = 1, 2, \cdots\cdots, R \qquad (16)$$

カテゴリーjの中にあるパラメータの組に対する応答が正しいというのはj番目の識別値が最大であるということである．そして，重みの初期値を任意に選び（例えば0），各パラメータのパターン毎に試行を繰返し，誤った応答を出すたびに次の訂正を行なっていく．すなわち，カテゴリーjに属するパターン$\boldsymbol{Y}$に対してi番目（$i \neq j$）の識別値が最大であったとすれば，i番目，j番目の重みベクトルを次のような絶対訂正法により修正する．

$$\left.\begin{aligned} &\boldsymbol{W}^{(j)} = \boldsymbol{W}^{(j)} + C\boldsymbol{Y} \\ &\boldsymbol{W}^{(i)} = \boldsymbol{W}^{(i)} - C\boldsymbol{Y} \end{aligned}\right\} \qquad (17)$$

但し，Cは$G_j(\boldsymbol{X})$が最大となるような最小の正整数である．

このような試行を繰返せば，必ず有限回の重みベクトル訂正により，必ず解重みベクトルへ至ることが保証されている．[3] しかし，d，R，r，学習に使用するパラメータ・パターンの種類などにより，有限回という表現に相当の偏差があろう．

学習の効率は，次のように示すことにする．J種類の与えられたパラメータ・パターンによりL回繰返し学習して，その結果の訓練された重みベクトルを用いてN種類（学習に使用したJ種類を含む）のパラメータを識別した時

$$\begin{aligned} &N_0 \quad : \text{正しいカテゴリーとして判定した個数} \\ &N_1 \quad : \text{正しいカテゴリーから} \pm 1 \text{誤って判定した個数} \\ &\vdots \\ &N_i \quad : \text{正しいカテゴリーから} \pm i \text{誤って判定した個数} \end{aligned} \qquad (18)$$

とすると，

$$q_0 = N_0 / N \qquad \text{：正解率}$$
$$q_i = N_i / N,\ i = 1, 2, \cdots\cdots \qquad \text{：}\pm i \text{ 誤ったカテゴリーに判別した率} \qquad (19)$$

となる．

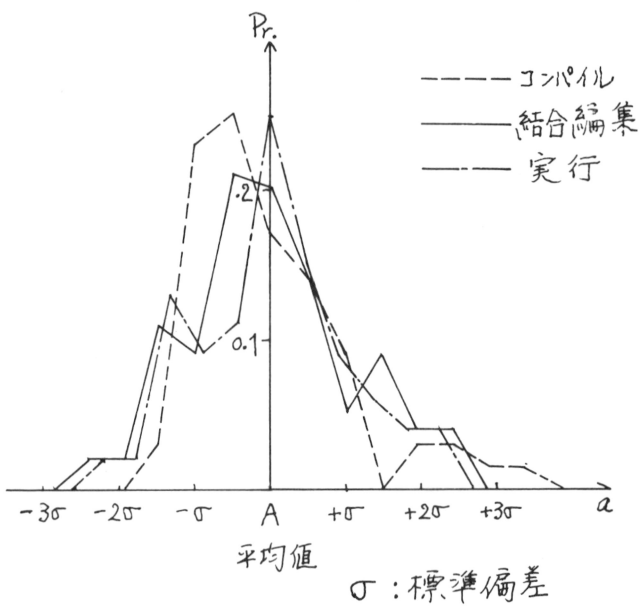
3. マルチプログラミング・システムのシミュレーション

3.1 モデルと方法

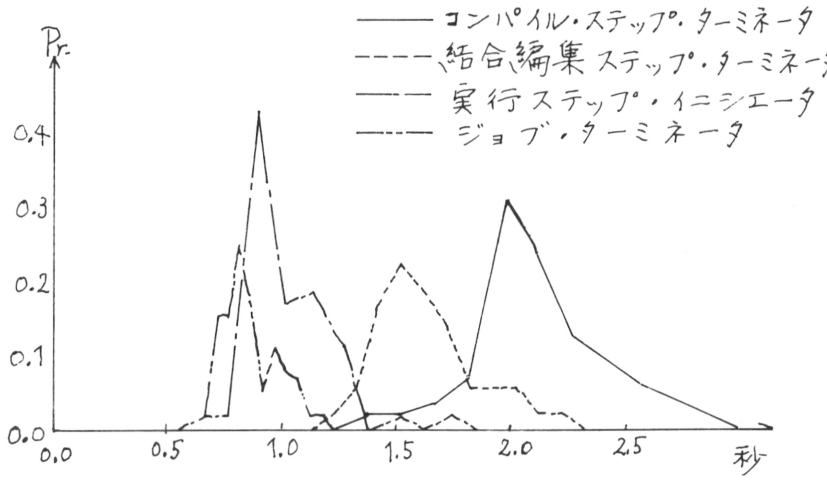
ここで行なうマルチプログラミング・システムのシミュレーションは，その対象として，京都大学大型計算機センターのFACOM 230－60システム（2CPU，コア・メモリ192K語，マルチプログラミングによるバッチ処理，ユーザのプログラミング言語は殆んどがFORTRAN）を取上げ，システムの動作状況を測定し，そのデータに基づいて，オペレーティング・システムをモデル化し，シミュレーションによってシステム・パフォーマンスの評価を行なった．

(i) システムの分析

システムの動作状況を把握するため，ジョブ処理中に記録されるアカウント情報を利用した．多重度1でジョブ処理を行ない，各ジョブ・ステップ（1ジョブは，コンパイル，結合編集，実行の3つのジョブ・ステップからなる）に関して，入出力時間（ファイル装置とコア・メモリ間の情報転送時間），スケジュラ時間（コンソール・タイプライタにスケジュラ開始時刻と終了時刻を打出して求める）の分布を調べた．入出力時間は，処理プログラムの入出力要求の回数，入出力量などからなる複雑さに依存する．この複雑さを示すものとして，コンパイル，結合編集ステップでは，CPU専有時間を，実行ステップではライン・プリンタ出力行数を取上げ，入出力時間との関係を一次式で近似した．図－2にその傾きの分布を図－3にスケジュラ時間の分布を示す．



図－2　入出力時間の一次近似による傾きの分布



図—3　スケジュラ時間の分布

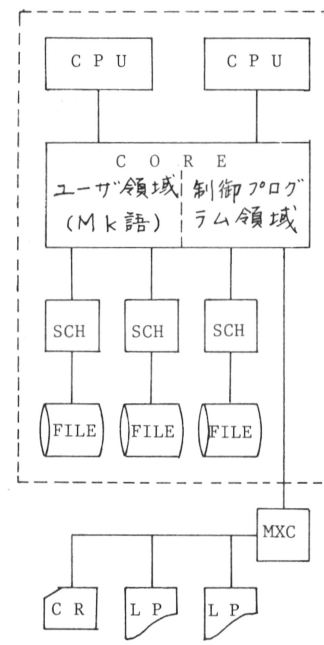
(ii)　入力データ

入力データのうち，ジョブの各ジョブ・ステップにおける入出力時間，スケジュラ時間は，システムの分析の結果に従って，近似式の傾き，時間の分布を正規分布で近似し，乱数を発生させて求めた．一方，ＣＰＵ専有時間，コア・メモリ専有語数は，アカウント情報のリアルなデータをそのまゝ用いて，シミュレーションの入力データとした．

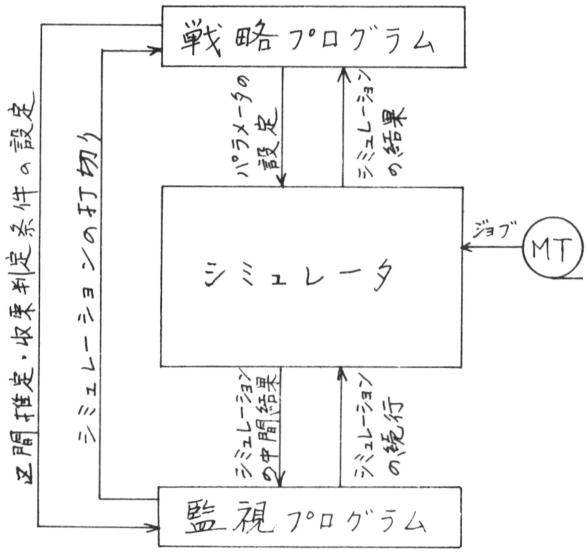
(iii)　シミュレーション・プログラムの構成

モデル化されたシステムの構成は図—4に示されている．

シミュレーション・プログラムは，図—5に示すように，3つの部分，戦略プログラム，シミュレータ，監視プログラムからなっている．戦略プログラムは，監視プログラムへ収束判定，区間推定などのための必要な値を与えるとともに，シミュレータのパラメータの値を決定するルーチンである．監視プログラムは，シミュレータから，中間結果により，区間推定，収束判定を行ない，シミュレーションの続行・打切りを決定するルーチンである．シミュレータはさらに，タスク管理，リソース管理，スケジュラ管理，多重度管理などのシステムをシミュレートしているルーチンと，パフォーマンス・ルーチンからなっている．パフォーマンス・ルーチンは，パフォーマンスの測度に関する情報の採集を行なうル



図—4　システム構成



図－5　シミュレーション・プログラム

ーチンである.

(iv)　システム・パフォーマンスの評価

システム・パフォーマンスを評価する上で重要と考えられるパラメータとして，ＣＰＵの速度比，コア・メモリのユーザ領域，設定多重度の３つを選び，一方，パフォーマンスの測度としては，各リソースの使用率，スケジュラの走行率，実行多重度などを取上げた.

システム全体を評価する式として

$$P_1 = T / \Sigma C_i$$
$$P_2 = T \Sigma (C_i U_i) / (\Sigma C_i)^2 \quad \} \qquad (20)$$

の２つを考えた. T は単位時間当りのジョブ・ステップ処理件数である. C_i はリソースの単位価格の比である. ＣＰＵの速度比 s による価格の割合は，Grosch の法則に従うものとした. 各リソースの台数を N_i とすると（ i ＝ＣＰＵ，ＣＨ，ＣＯＲＥとする），

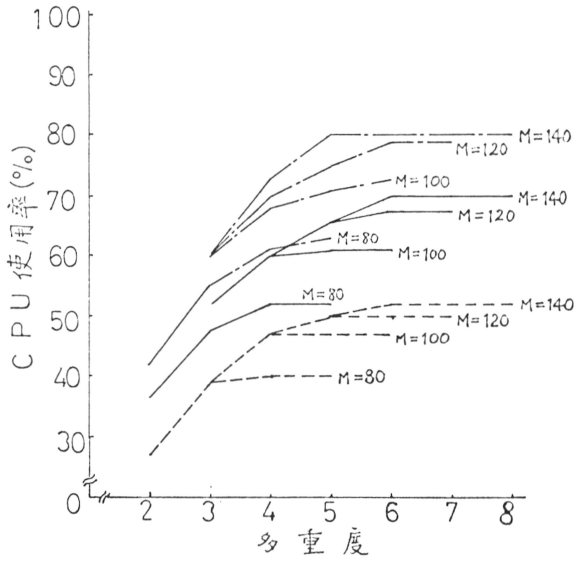
$$\Sigma C_i = \frac{1}{\sqrt{s}} N_{CPU} C_{CPU} + N_{CH} C_{CH} + N_{CORE} \cdot C_{CORE} \qquad (21)$$

となる. 一方，U_i はリソース i の利用率を示している.

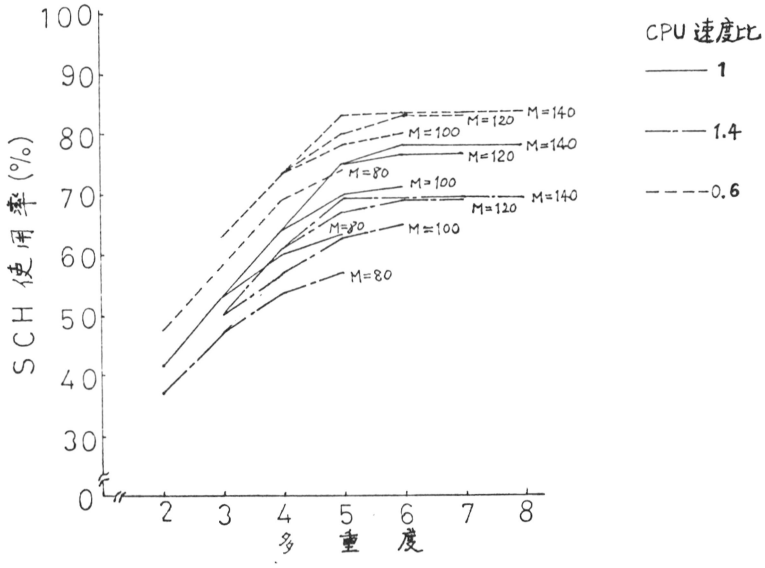
３．２　シミュレーションの結果

ここでは，2.1で述べたＳｌｉｄｅ法により，パラメータの値を少しずつ変化させて，シミュレーションを連続的に実行させた. その結果を図－６，７に示す.

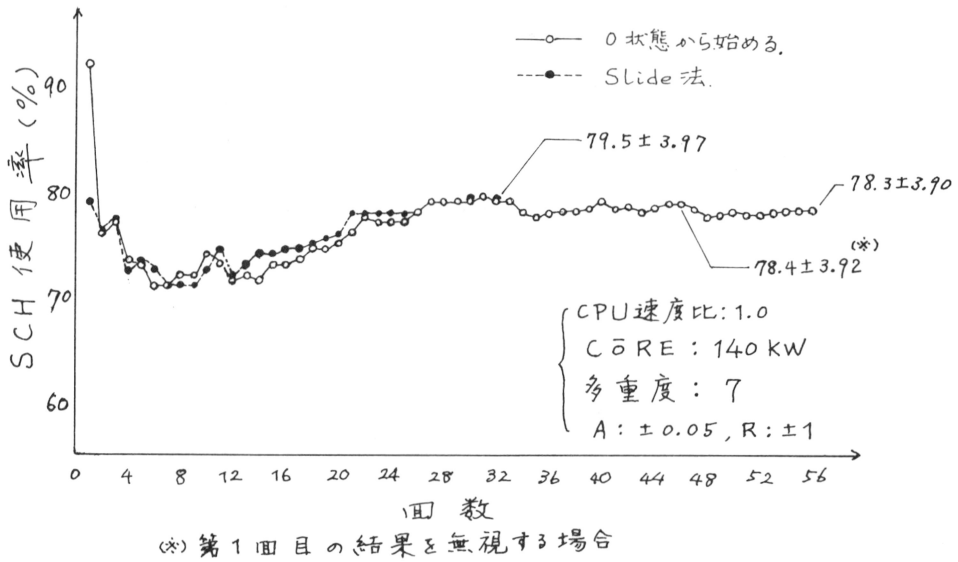
また，2.1で述べた収束を早める方法の有効性を示す一例として，０状態から始めた場合およびＳｌｉｄｅ 法による場合の平均値を図－８に示す. ０状態から始めた場合に，第１回目の結果を過渡状態による異常値とみなし，第２回目以後の結果によって，区間推定を行なえば，第



図－6　CPUの使用率



図－7　チャネルの使用率



図－8　チャネルの平均使用率

46回目で十分の精度が取れている．即ち（10）式に$k=3\cdot3$，$n=56$を代入すれば，$N\fallingdotseq47$（但し，$t_{46}(\frac{\varepsilon}{2})/t_{55}(\varepsilon/2)=1$とする）となる．このようにシミュレーションの効率が良くなっていることが，分るであろう．この場合，Slide 法が更に効率良くなっているのは，第2回目のデータにも過渡的状態の影響が現われているためと思われる．

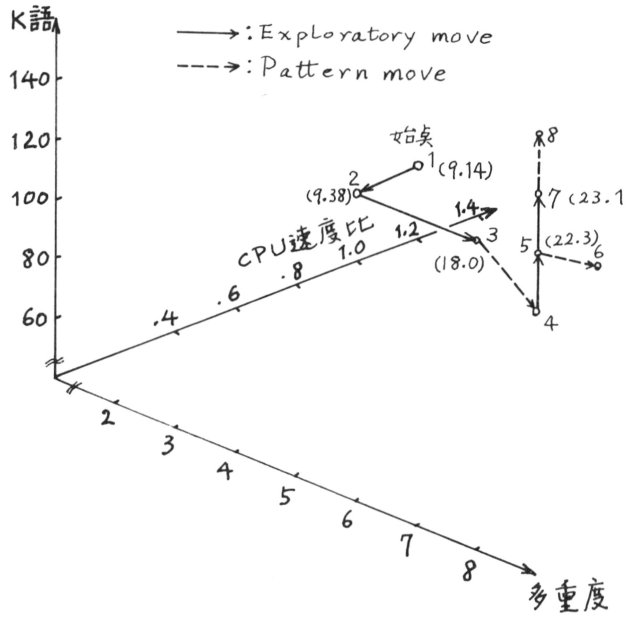
信頼区間による打切り条件は，結果の値と信頼区間の相対的割合(R)および信頼区間の絶対的な値(A)の2つの場合を考えた．今回のシミュレーションでは，$R=\pm0.05$，$A=\pm1$とした．但し，結果の値の単位はパーセントである．

3.3　Direct Searchの実験例

システム・パフォーマンスの測度として，（20）式のP_1とP_2を取上げて，その値が最大になるパラメータの値を求めるために，Direct Search を利用した．P_1, P_2 の値を決定する函数は，あらかじめ与えられた変域で，高々一つの極大しか持たないことは，保証されているものとした．そのP_2に関するDirect Search の動きを図－9に示す．但し，Exploratory move は成功したものだけしか示していない．

3.4　学習機能の実験例

マルチプログラミング・システムのシミュレーションにおいて，2.3に述べた学習機能を取入れて実験を行なった．付加パターンとして与えるパラメータは，CPU速度比（0.6～1.4），コア・メモリのユーザ領域（80～140k語），設定多重度の3種であり，またシステム・パフォーマンスの測度として，（20）式のP_2（上のようなパラメータの変域では，6～24の値をとる）を使用し，表－1のようなR個（$R=2,4,6,9$）のカテゴリーに分類した．そしてJ個



図－9　Direct Search　　P_2：システム・パフォーマンス

のシミュレーション結果を用いて，J種類のパラメータ・パターンに対しL回学習（識別函数の次数r：1, 2, 3,）を繰返した後，N種類のパラメータ・パターン（学習に使用したJ個を含む）に対する応答の精度（N_0, N_1, ……, q_0, q_1, …）を求めた．学習繰返し回数LはL_Mまで行ない，$L=1 \cdots\cdots L_M$でq_0が最大となる時のL，N_0，N_1，q_0，q_1を表－2に示した．

（表－2）　カテゴリーの分類

カテゴリー数 R	システム・パフォーマンス P_2	カテゴリー j
2	～15	1
	15～	2
4	～10	1
	10～15	2
	15～20	3
	20～	4
6	～ 9	1
	9～12	2
	12～15	3
	15～18	4
	18～21	5
	21～	6
9	～ 8	1
	8～10	2
	10～12	3
	12～14	4
	14～16	5
	16～18	6
	18～20	7
	20～22	8
	22～	9

（表－２）　学習効率の一例

q_0, q_1：（％），＊：$q_0+q_1=100\%$

	R	J	L_M	$r=1$			$r=2$			$r=3$		
				L	N_0 q_0	N_1 q_1	L	N_0 q_0	N_1 q_1	L	N_0 $1q_0$	N_1 q_1
$d=3$ $N=140$	2	140	87	14*	136 97.1	4 2.9	80*	138 98.5	2 1.5	54*	139 99.2	1 0.8
		70	55	6*	134 95.7	6 4.3	29*	136 97.1	4 2.9	48*	138 98.5	2 1.5
		46	59	14*	134 95.7	6 4.3	20*	135 96.4	5 3.6	36*	136 97.1	4 2.9
	4	140	96	42*	107 76.4	33 23.6	26*	111 79.2	29 20.8	49*	106 75.7	33 24.3
		70	81	76*	102 72.8	38 27.2	51*	114 81.4	26 18.6	73*	111 79.2	29 20.8
		46	53	35*	92 65.7	48 34.3	19*	92 65.7	48 34.3	23*	100 71.4	40 28.6
	6	140	184	129*	84 60.0	56 40.0	143*	93 66.4	47 33.6	134*	97 69.2	43 30.8
		70	50	45	75 53.5	56 40.0	45	83 59.2	51 36.4	28	83 59.2	50 35.7
		46	48	36	71 50.7	61 43.5	44	83 59.2	50 35.7	26	86 61.4	47 33.5
	9	140	11	3	39 27.8	59 42.1	1	41 29.2	46 32.8	1	40 28.6	68 48.5
		70	11	2	48 34.3	56 40.0	4	31 22.1	40 28.6	8	41 29.2	52 37.1
		46	26	2	45 32.1	53 37.8	22	52 37.1	52 37.6	20	70 50.0	48 34.2

4. あとがき

効率良いシミュレーションを行なうためのいくつかの技法について述べ，さらにマルチプログラミング・システムのシミュレーションにおいてそれらを実験したが，特にSlide法やDirect Search法は，シミュレーションに要する時間（計算時間だけでなく，パラメータの設定などに関して人間が考えたりする時間も含めて）の短縮に有効であり，また，計算機システムの解析などの分野では，これらの手法を利用して効果が上がる場合が比較的多いと考えられる．さらに，ここには述べなかったが，シミュレーションに対話型処理機能を取入れて効率を高める方法もある．パラメータの変更や収束判定などに人間の判断を加えることにより，さらに高等な戦術をとることもできるので，ここに述べた技法をＴＳＳの環境で行なってみて，対話型の有用性をも確かめてみたい．

学習機能は，シミュレーションを行なう上で直接にどの程度の効果が望めるかは少し問題であるが，シミュレータを利用してシステム・アイデンティフィケーションを行ない，その結果をモデルやシミュレータの作成上へフィード・バックできれば大変有効となるので，ここに取上げた．

最後に，計算機システムの解析を行なう場合，シミュレーションの他に，実システムの測定や数学的解析の手法もあるが，実用的な結果を効率良く得ようとすれば，これらのうちの一方法だけでは不十分と考えられ，それぞれの長所を生かすような工夫が必要であろう．

参　考　文　献

1）　Feller, W. : An Introduction to Probability Theory and Its Applications, Vol. 1, John Wiley & Sons, (1957)

2）　Kowalik, J. & M. R. Osborne : Methods for Unconstrained Optimization Problems, Elsevier, (1968)

3）　Nilsson, N. : Learning Machines, McGraw-Hill, (1965)

本 PDF ファイルは 1972 年発行の「第 13 回プログラミング―シンポジウム報告集」をスキャンし、項目ごとに整理して、情報処理学会電子図書館「情報学広場」に掲載するものです。

この出版物は情報処理学会への著作権譲渡がなされていませんが、情報処理学会公式 Web サイトの https://www.ipsj.or.jp/topics/Past_reports.html に下記「過去のプログラミング・シンポジウム報告集の利用許諾について」を掲載して、権利者の捜索をおこないました。そのうえで同意をいただいたもの、お申し出のなかったものを掲載しています。

過去のプログラミング・シンポジウム報告集の利用許諾について

情報処理学会発行の出版物著作権は平成 12 年から情報処理学会著作権規程に従い、学会に帰属することになっています。

プログラミング・シンポジウムの報告集は、情報処理学会と設立の事情が異なるため、この改訂がシンポジウム内部で徹底しておらず、情報処理学会の他の出版物が情報学広場 (=情報処理学会電子図書館) で公開されているにも拘らず、古い報告集には公開されていないものが少からずありました。

プログラミング・シンポジウムは昭和 59 年に情報処理学会の一部門になりましたが、それ以前の報告集も含め、この度学会の他の出版物と同様の扱いにしたいと考えます。過去のすべての報告集の論文について、著作権者（論文を執筆された故人の相続人）を探し出して利用許諾に関する同意を頂くことは困難ですので、一定期間の権利者捜索の努力をしたうえで、著作権者が見つからない場合も論文を情報学広場に掲載させていただきたいと思います。その後、著作権者が発見され、情報学広場への掲載の継続に同意が得られなかった場合には、当該論文については、掲載を停止致します。

この措置にご意見のある方は、プログラミング・シンポジウムの辻尚史運営委員長 (tsuji@math.s.chiba-u.ac.jp) までお申し出ください。

加えて、著作権者について情報をお持ちの方は事務局まで情報をお寄せくださいますようお願い申し上げます。

期間：2020 年 12 月 18 日 〜 2021 年 3 月 19 日
掲載日：2020 年 12 月 18 日

プログラミング・シンポジウム委員会

情報処理学会著作権規程
https://www.ipsj.or.jp/copyright/ronbun/copyright.html