JAERI-M
9501

計算機ジョブ処理最適多重度の決定

1981年6月

浅井　清・高橋　國夫*・藤井　実

日本原子力研究所
Japan Atomic Energy Research Institute

この報告書は，日本原子力研究所がJAERI-Mレポートとして，不定期に刊行している研究報告書です．入手，複製などのお問合わせは，日本原子力研究所技術情報部（茨城県那珂郡東海村）あて，お申しこしください．

JAERI-M reports, issued irregularly, describe the results of research works carried out in JAERI. Inquiries about the availability of reports and their reproduction should be addressed to Division of Technical Information, Japan Atomic Energy Research Institute, Tokai-mura, Naka-gun, Ibaraki-ken, Japan.

JAERI－M 9501

計算機ジョブ処理最適多重度の決定

日本原子力研究所東海研究所計算センタ
浅井　清・高橋國夫*・藤井　実

（1981年4月28日受理）

本報告で述べる研究の最終的目標は，ひと言でいえば，夜間自動運転において，いかにうまく計算機システムを停止させるかということである。単に計算機を停止させることであれば，現在計算機メーカから提供されている標準的な自動運転の機能によってすでに実現されている。毎夜異なった組合せのジョブが数百個入力されるが，それらを複数の計算機システムに配分し，計算機資源を最大限に使用しながら全部のシステムをほとんど同時に停止させること，すなわちうまく計算機システムをとめることは難しい。この問題を解決するには，入力されたジョブ群に対して演算装置，入出力装置がどのような振舞いをするかを正確に知る必要がある。

本報告において筆者らは，入力されたジョブを演算装置が処理するときの最適多重度とジョブ終了時間を求める計算機プログラムを待ち行列理論の手法によって構成し提案した。また，その手法の妥当性を検証するために現実の計算機オペレイティング・システムの動きを模擬するシミュレーション・プログラムを作成した。それが上記手法と同一の入力ジョブ群について，ほとんど同一の計算結果を与えることがわかった。このことから夜間ジョブのスケジューリング，および現実のオペレイティング・システムの複雑なジョブ・スケジューリングを制御する方法についてある程度明るい見通しが得られた。

＊富士通（株）

JAERI-M 9501

Determination of Optimal Degree of Multiprogramming and Prediction of Job Processing Time

Kiyoshi ASAI, Kunio TAKAHASHI* and Minoru FUJII

Computing Center, Tokai Establishment, JAERI

(Received April 28, 1981)

In the report the authors have presented a job scheduler for determination of optimal degrees of multiprogramming and prediction of processing time for a given set of jobs. It is assumed that the jobs' computational times and input/output frequencies are known beforehand. The jobs are sorted by the scheduler according to a LPT rule before scheduling.

A Poisson arrival and a constant service time are assumed with each job's input/output operation.

A modified round-robin algorithm is used to schedule a given set of jobs.

A computational result of the scheduler is validated by a simulator which resembles closely with round-robin type operating systems.

The scheduler is useful for automatic control of computer sytems and the simulator is useful for simulations of existing operating systems. Both the scheduler and the simulator are written in Fortran language.

Keywords: Computer, Multiprogramming, Scheduling, Simulation, Round-robin

---

* Fujitsu, Ltd.

目　　次

Contents

1. はじめに

日本原子力研究所における計算機システムの運用方法ではFig. 1.1にみられるごとく昼間においては3システムそれぞれが異なった用途に当てられている。

夜間における運用はこれとは全く異なり，3システムはバッチ・ジョブ専用となる。このときこれらの3システムの運用をおこなう上で望ましい条件は

(1) 3システムがほとんど同時にジョブ終了を完了し，同時に電源切断が可能なようにジョブのスケジューリングをおこなうこと，ジョブ処理が明朝9時までに完了しないときは，9時までに完了できる範囲でジョブ・スケジューリングをおこない，翌日の運用をさまたげないこと，

(2)夜間ジョブのスケジューリングは，演算装置，入出力装置などの計算機資源をできるだけ有効利用するようおこなうこと

などである。

昭和56年2月の夜間ジョブ数はリモート・ジョブ・ステージョンから投入されたものと，クローズ・ジョブ受付で受付けられたものを含めて4000件あり，1日平均では200件となる。これらのジョブは入出力装置（I／O）よりは演算装置（CPU）を多く使用する。しかし演算装置は高速化されてゆく傾向にあるにもかかわらず入出力装置の速さはさほど改善されていないので，現在CPUよりのジョブもごく近い将来はI／Oよりのジョブとなる。すなわち単位時間に発信される入出力命令の数が増加し，ジョブ実行多重度の影響が大きくなる。したがって200～300ジョブを3システムに適当に配分してこれらシステムを同時に停止させることは熟練した操作員にとってもできないことである。

そこで自動ジョブ・スケジューリングのための道具が必要になる。この道具は計算機プログラムであって，このプログラムを操作員が事前に5分間程度操作することによって夜間の全ジョブのスケジューリングをおこなえることがのぞましい。

計算機自動運転のためのハードウェアとソフトウェアおよびその関連図をFig. 1.2に示す。

無人運転制御装置は計算機メーカが用意する部分と計算機を設置している計算センタが用意する部分とに分れることがある。

ジョブのスケジューリングは本来計算センタがおこなうべき性質のものである。スケジューリングの目標は前述の2項目であるが，その目標を達成するためには，ひとつの計算機システムと与えられたジョブ実行系列に対して，

(i)CPU系のジョブ処理を予測する初等的理論と手法を持つこと，

(ii)上記の(i)を現実の計算機のジョブ・スケジューリング方法に合致するよう変形すること，

(iii)磁気ディスク，大容量記憶装置，磁気テープと階層づけられたファイルの入出力処理を予測する初等的理論と手法を持つこと，

(iv)与えられたジョブ実行系列中のジョブの特性，使用するファイルの属性に関するデータを計算機システム内に集積し利用可能とすること

が必要である。

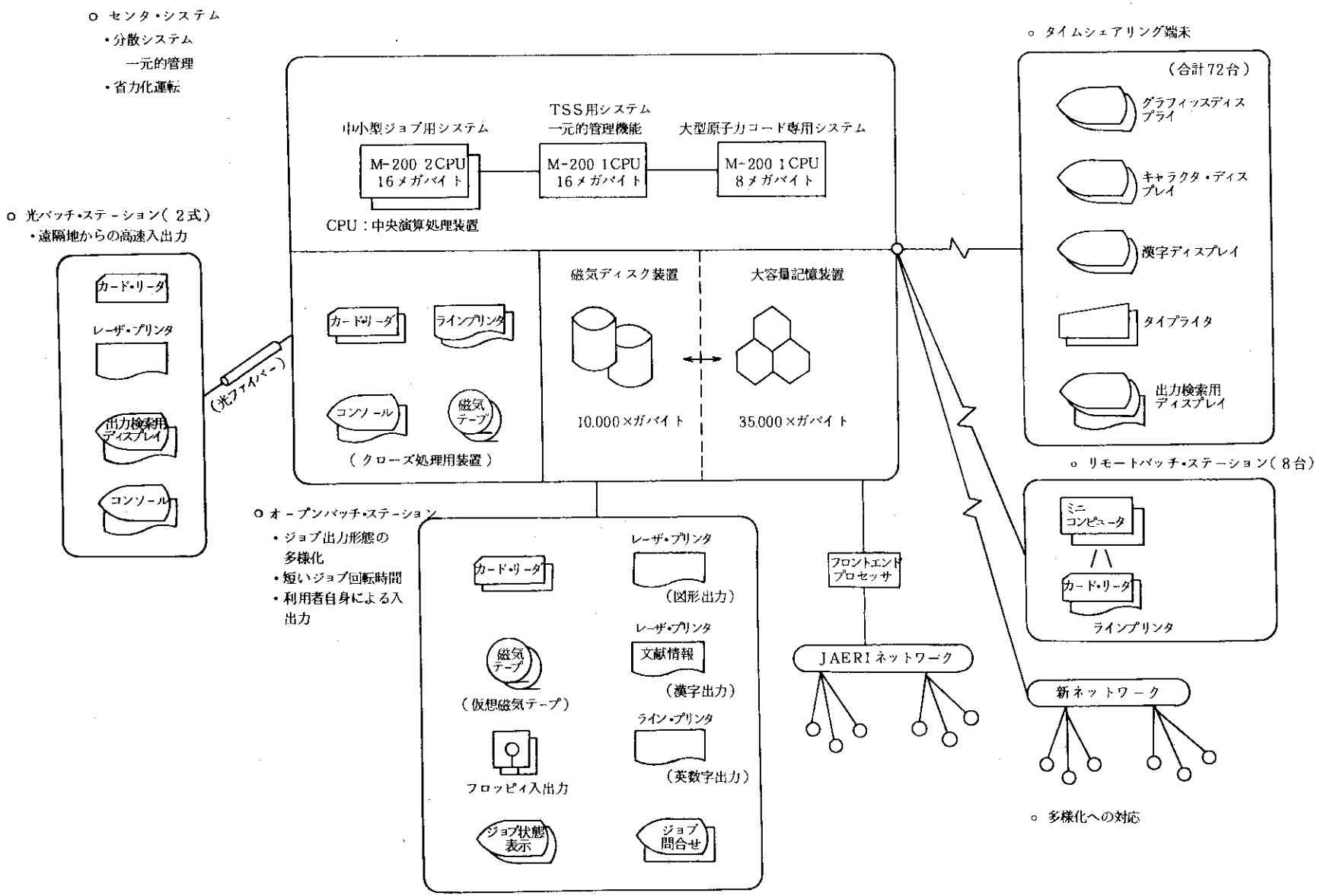


Fig. 1.1 Computing facilities at JAERI

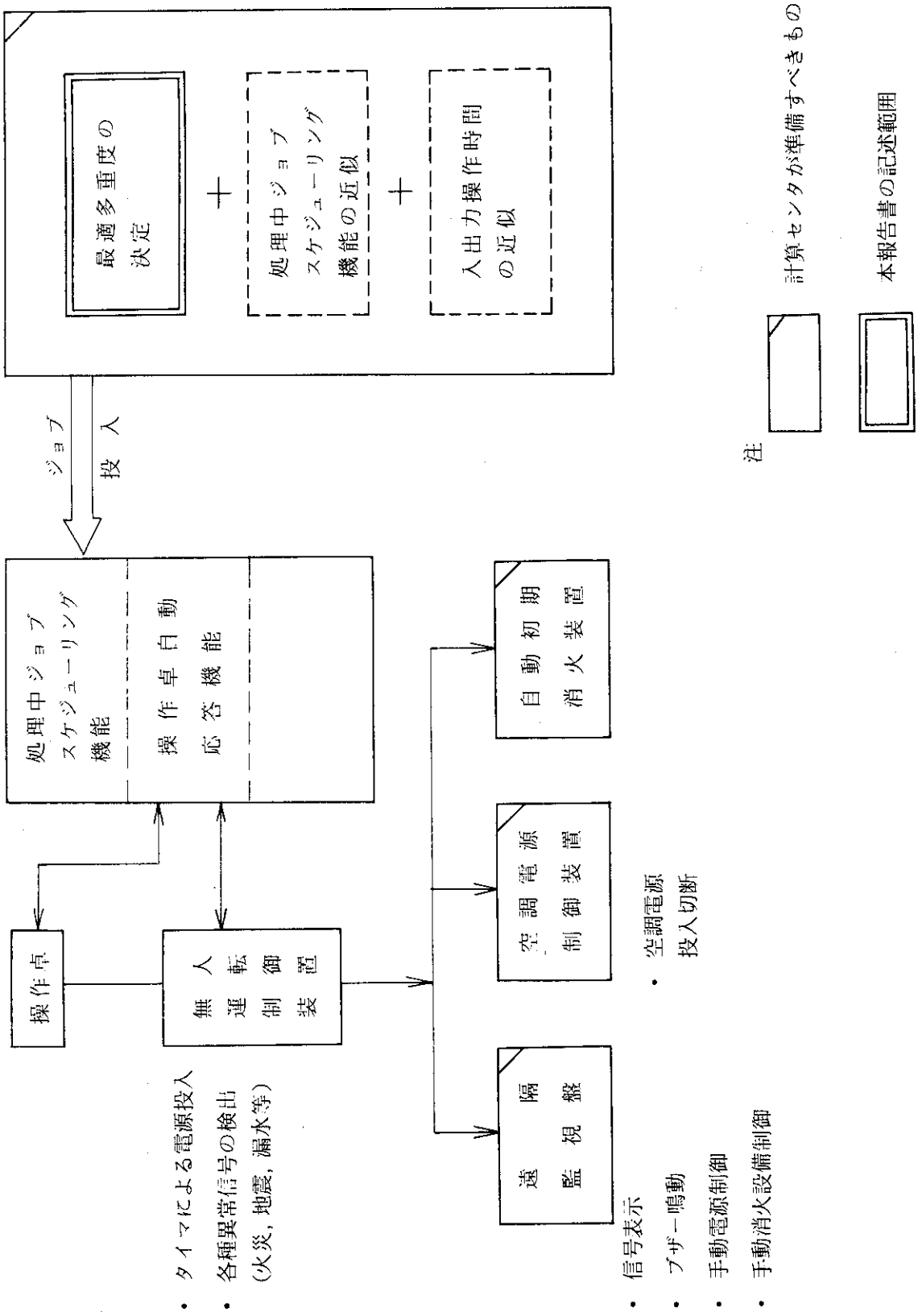


Fig. 1.2 Facilities and functions for automatic computer control

本報告は上記4項のうち第(i)項について実用的な方法を提案し，その予測結果の妥当性をシミュレーションで示したものである。

スケジューリングの方法を考えるにあたって次の5つの仮定を採用した。

(1)入力されるジョブの演算装置使用時間と入出力回数は既知である。
(2)起動されたジョブは主記憶から追い出されることはない。
(3)オペレイティング・システムのオーバーヘッドはない。
(4)使用可能な実記憶量を超えない範囲でジョブ多重度を決定する。
(5)入出力1回当りに要する時間は固定である。

上記(1)の仮定の根拠を第2章で示す。第3章では既存の各種ジョブ・スケジューリング手法を簡単に概観する。第4章では(1)の前提からラウンド・ロビン手法によるジョブ・スケジューリングにおけるサービス時間分布を計算することができ，個々のジョブの終了時刻を予測することができるスケジューラについて述べる。第5章では，現実のオペレイティング・システムの動きを模擬するシミュレータについて述べる。これでスケジューラの妥当性を検証する。第6章ではスケジューラ，シミュレータの問題点，長短について述べる。

2. ジョブ計算機資源使用量予測

2.1 ジョブの計算機資源使用量の予測

現在，ジョブは利用者が入力時に指定した計算機資源使用の上限値にもとづいてスケジューリングされ，処理されている。しかし，利用者の指定した上限値と実際にジョブが使用する資源量にはかなりの差がある。

ここでは，計算機が利用者に関する記憶をもつことにより，ジョブ処理前に実際にジョブが使用する資源量を精度よく推定できることを示す。Fig. 2.1に計算機資源使用量予測を行うときの概念図を示す。

この章は，原研計算センタのジョブ処理データを使った予測テストについて述べる。

2.2 予測テスト

予測は，演算装置使用時間（CPU時間），主記憶装置使用量（MEM量），ファイル入出力回数（I／O回数）について行った。ジョブ処理データは会計情報から収集した。

〔ジョブ入力情報〕

入力ジョブに対して利用者が指定する

(1) CPU使用時間の上限値（T値），

(2) MEM使用量の上限値（C値），

(3) I/O使用回数の上限値（I値），

(4) ライン・プリンタの出力量の上限値（W値）

と，ジョブ入力時に収集できる

(5) カード形式の入力データ枚数（input）

を使う。

実際には，T，C，I，W の設定値はTable 2.1のように各上限値と対応している。

〔利用者に関する記憶〕

この方式の持つ記憶には次の6種類がある。利用者別の平均値がない場合にはシステム平均値を用いて予測する。

(1)システム平均値：μ_k（$k=1$：CPU時間，$k=2$：MEM量，$k=3$：I／O回数）

資源利用の上限値（設定値）に対する資源の実際使用量の割合である。μ_1はCPU使用割合，μ_2はMEM使用割合，μ_3はI／O使用割合である。nをこの記憶が作動してからのジョブ処理件数とする。

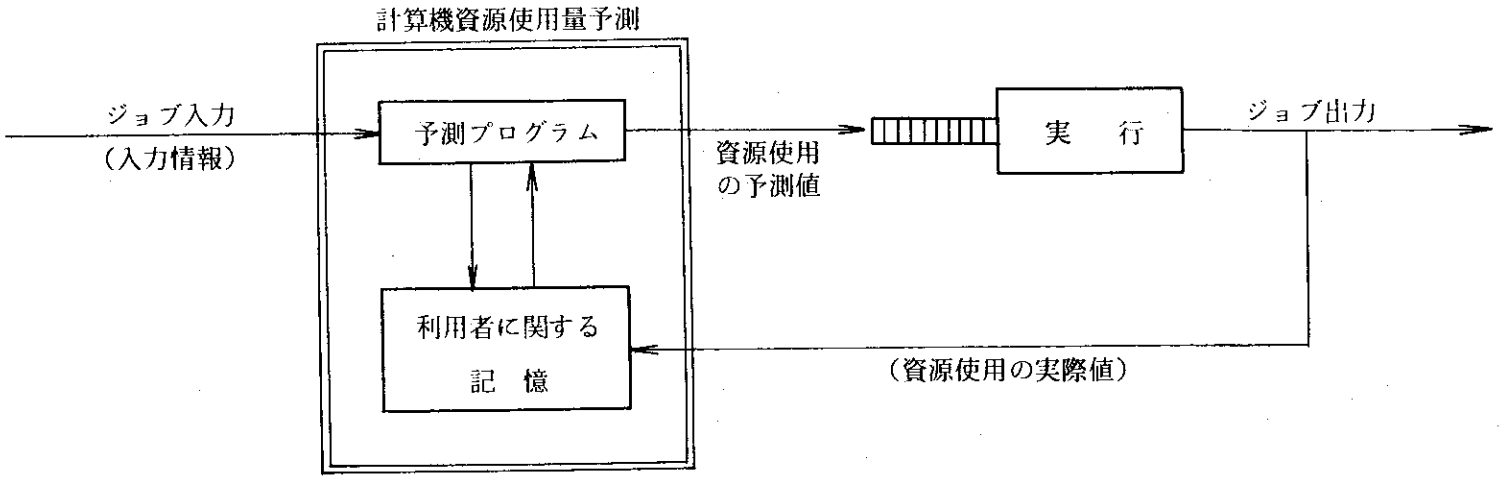


Fig. 2.1　Prediction of job characteristics

Table 2.1　Values for specification and corresponding computer resources

設定値 / 資源	0	1	2	3	4	5	6	7	8	9	10	11	12
CPU時間（T）	5 秒	10 秒	30 秒	1 分	2 分	5 分	10 分	20 分	30 分	45 分	60 分	90 分	120 分
MEM 量 （C）	256 KB	512 KB	768 KB	1,024 KB	1,536 KB	2,048 KB	2,560 KB	3,072 KB	4,096 KB	5,120 KB	6,144 KB	7,168 KB	8,192 KB
I/O 回 数（I）	500	1,000	2,000	4,000	10,000	20,000	30,000	40,000	50,000	500,000			
印刷行数（W）	2,000	3,000	4,000	6,000	8,000	12,000	16,000	20,000	28,000	36,000	44,000	52,000	60,000

$$\mu_k = \sum_{i=1}^{n} \left(\frac{\text{実際使用量}}{\text{設定上限値}}\right)_{ik} / n$$

(2)設定値別システム平均値：μ_{kl}（$l = 0 \sim 12$）

設定値別の資源使用割合である。n_{kl}を各資源の設定値別ジョブ処理件数とする（例えばT.9を設定したジョブ処理件数が10件あると$n_{1,9} = 10$である）。

$$\mu_{kl} = \sum_{i=1}^{n_{kl}} \left(\frac{\text{実際使用量}}{\text{設定上限値}}\right)_{kl} / n_{kl}$$

(3)利用者別平均値：μ_{uk}（$u = 0 \sim$ 利用者数）

利用者別の資源使用割合である。n_uを各利用者のジョブ処理件数とする。

$$\mu_{uk} = \sum_{i=1}^{n_u} \left(\frac{\text{実際使用量}}{\text{設定上限値}}\right)_{uk} / n_u$$

(4)利用者別設定値別平均値：μ_{ukl}

利用者別，設定値別の資源使用割合である。n_{ukl}を各資源の利用者別，設定値別のジョブ処理件数とする。

$$\mu_{ukl} = \sum_{i=1}^{n_{ukl}} \left(\frac{\text{実際使用量}}{\text{設定上限値}}\right)_{ukl} / n_{ukl}$$

(5)利用者別短期記憶：MS_u（m_1，11）

各利用者の最新m_1個のジョブ処理データの記憶を短期記憶と呼び，以下のデータ項目を保持する。

① T値
② C値
③ I値
④ W値
⑤ 入力データ枚数
⑥ CPU使用設定上限値
⑦ CPU使用実際値
⑧ MEM使用設定上限値
⑨ MEM使用実際値
⑩ I／O使用設定上限値
⑪ I／O使用実際値

この中で，⑥⑧⑩は，①②③の値をTable 2.1で変換すればよいので記憶しておく必要はないが，ここでは便宜上入れた。

(6)利用者別長期記憶：ML_u（k，l，11）

各利用者の資源別，設定値別の最新ジョブ処理データの記憶を長期記憶と呼び，(5)と同じ

項目をもつ。

以上に必要な記憶容量：MCは，利用者数N＝1000人の場合，

(i)短期記憶をm＝10まで保存するとMC＝約500k語，

(ii)m_1＝5，長期記憶は除去，不必要なデータを削除するとMC＝約40k語

となる。

〔予測アルゴリズム〕

記憶を用いた資源使用量の予測アルゴリズムをFig. 2.2に示す。予測方法を決めるのに用いた入力情報のチェック方法をTable 2.2に示す。表中の類似という用語は，

(i) CPU時間の予測の場合

T値は同じで，C値，I値，W値のいずれか1つが入力情報のそれと1だけ違う，

(ii) MEM量の予測の場合

C値，T値は同じで，I値，W値のいずれか1つが入力情報のそれと1だけ違う，

(iii) I／O回数の予測の場合

I値，T値は同じで，C値，W値のいずれか1つが入力情報のそれと1だけ違う

ことを意味する。

Table 2.2 Check pattern of input information

ESTM	入力情報のチェック
1	短期記憶の最新のものと同一入力情報である。
2	〃 2番目 〃
＼	＼
10	〃 10番目 〃
11	長期記憶のTの設定値別のものと同一入力情報である。
12	〃 C 〃
13	〃 I 〃
14	短期記憶の最新のものとinputだけが違う。
15	〃 2番目 〃
＼	＼
23	〃 10番目 〃
24	短期記憶の最新のものと類似である。
25	〃 2番目 〃
	＼
33	〃 10番目 〃
34	長期記憶のTの設定値別のものとinputだけ違う。
35	〃 C 〃
36	〃 I 〃
37	長期記憶のTの設定値別のものと類似である。
38	〃 C 〃
39	〃 I 〃
40	利用者別設定値別の平均値がある。
41	利用者別の平均値がある。
42	システムの設定値別の平均値がある。
43	システムの平均値がある。
44	最初の入力ジョブである。

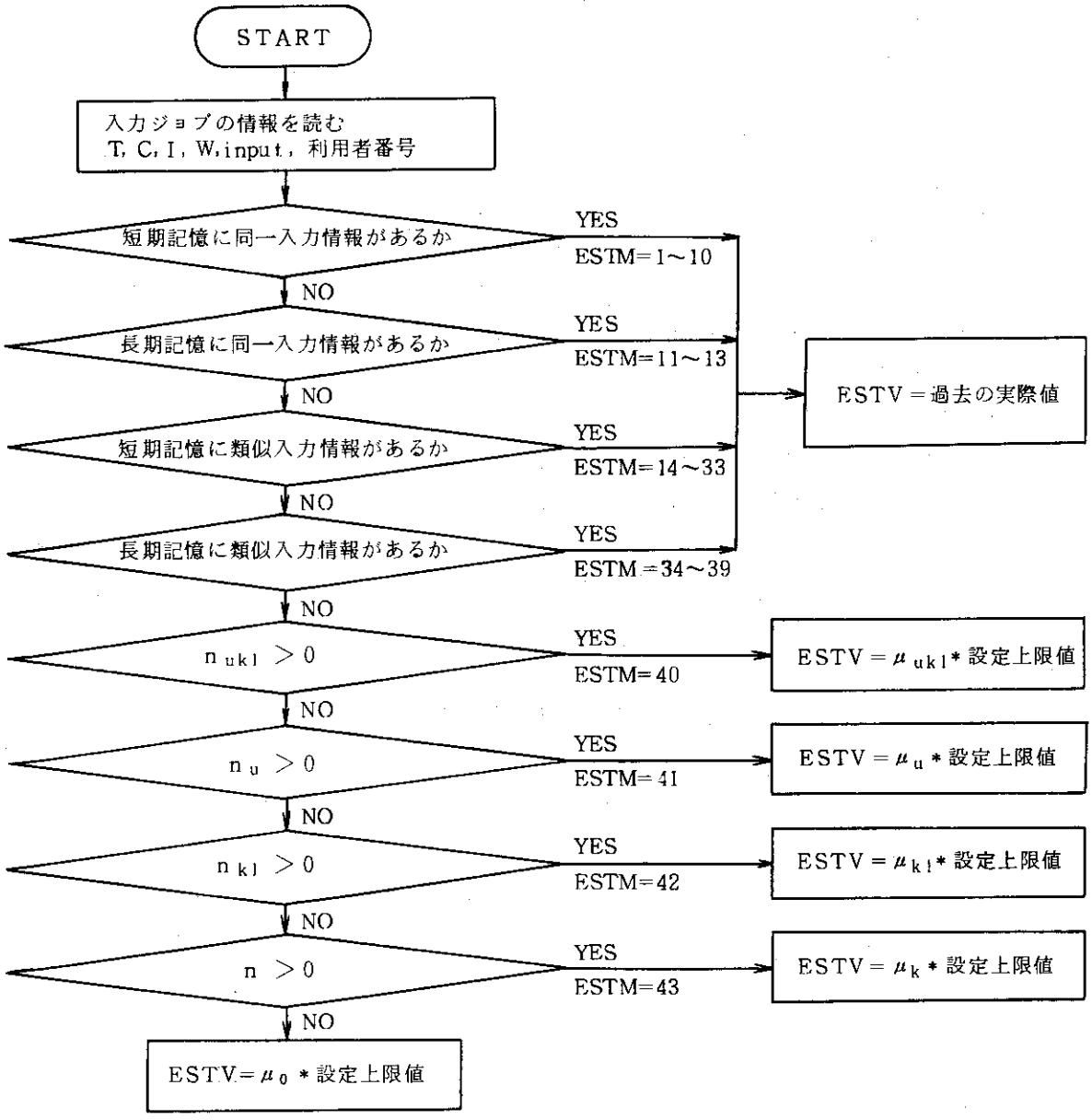


Fig. 2.2　Prediction flow

2.3 予測結果

予測は，昭和55年12月の会計情報より，原研計算センタのCPU使用時間の多い大口ユーザ30人に対して行った。

予測対象のジョブは以下の3グループである。

(1)全ジョブ

(2)大型ジョブ（CPU時間の設定上限値≧30分）

(3)夜間ジョブ（ジョブ投入が18時～翌朝9時）

予測の前調査で，投入ジョブの数%がエラーで比較的早い段階ですぐ終了してしまうので，これらのエラー・ジョブを除去するため，次の前処理をした場合を考えた。

① 前処理は行わない。

② CPUの設定上限値の0.1%を実行してみる（この0.1%以上のCPU時間以内でジョブが終了したら，そのジョブの予測は終了したとみなす）。

③ CPUの設定上限値の1%を実行してみる。

上の②，③は，2時間ジョブにおいてはCPU時間が7.2秒，72秒にあたる。これらは，翻訳エラー，結合編集エラー，入力データのエラーなどに対応する。

Table 2.3に予測結果を表わす。ここで予測精度は，予測したジョブ数をnとすると

$$\text{予測精度} = \left\{ 1 - \left(\sum_{i=1}^{n} \frac{|\text{予測値} - \text{実際使用値}|}{\text{設定上限値}} \right) / n \right\} \times 100\%$$

である。また予測寄与率は，予測を行うことによってどれだけ実際使用値との誤差が減少したかを示す割合である。

$$\text{予測寄与率} = \left\{ \frac{\text{予測を行ったときの精度} - \text{予測を行わなかったときの精度}}{100\% - \text{予測を行わなかったときの精度}} \right\} \times 100\%$$

Table 2.3 Accuracy of Prediction

ケースNo.	予測方法	対象ジョブ	前処理	予測精度			予測寄与率		
				CPU時間	MEM量	I/O回数	CPU時間	MEM量	I/O回数
1	予測しない場合（設定上限値利用）	全ジョブ	なし	42%	78%	25%	0%	0%	0%
2	〃	大型ジョブ	〃	68%	82%	28%	0%	0%	0%
3	〃	夜間ジョブ	〃	51%	80%	25%	0%	0%	0%
4	システム平均値のみ利用	全ジョブ	なし	69%	81%	78%	47%	14%	71%
5	〃	〃	0.1%	69%	81%	79%	47%	14%	72%
6	〃	〃	1.0%	71%	83%	79%	50%	23%	72%
7	〃	大型ジョブ	なし	70%	87%	82%	6%	28%	75%
8	〃	〃	0.1%	74%	89%	83%	19%	39%	76%
9	〃	〃	1.0%	78%	89%	84%	31%	39%	78%
10	〃	夜間ジョブ	なし	65%	84%	81%	29%	20%	75%
11	〃	〃	0.1%	67%	84%	82%	33%	20%	76%
12	〃	〃	1.0%	70%	84%	83%	39%	20%	77%
13	利用者平均値まで利用	全ジョブ	なし	71%	82%	79%	50%	18%	72%
14	〃	〃	0.1%	72%	82%	80%	52%	18%	73%
15	〃	〃	1.0%	74%	84%	80%	55%	27%	73%
16	〃	大型ジョブ	なし	71%	90%	87%	9%	44%	82%
17	〃	〃	0.1%	76%	91%	87%	25%	50%	82%
18	〃	〃	1.0%	80%	92%	88%	37%	56%	83%
19	〃	夜間ジョブ	なし	70%	87%	82%	39%	35%	76%
20	〃	〃	0.1%	74%	87%	82%	47%	35%	76%
21	〃	〃	1.0%	77%	88%	83%	53%	40%	77%

ケースNo.	予測方法	対象ジョブ	前処理	予測精度			予測寄与率		
				CPU時間	MEM量	I／O回数	CPU時間	MEM量	I／O回数
22	設定値別利用者平均値まで利用	全ジョブ	なし	76%	89%	89%	59%	50%	85%
23	〃	〃	0.1%	77%	90%	90%	60%	55%	87%
24	〃	〃	1.0%	79%	91%	90%	64%	59%	87%
25	〃	大型ジョブ	なし	72%	95%	91%	12%	72%	87%
26	〃	〃	0.1%	78%	96%	92%	31%	78%	89%
27	〃	〃	1.0%	83%	97%	92%	47%	83%	89%
28	〃	夜間ジョブ	なし	73%	87%	89%	45%	35%	85%
29	〃	〃	0.1%	76%	90%	89%	51%	50%	85%
30	〃	〃	1.0%	79%	92%	90%	57%	60%	87%
31	短期記憶を1つもった場合	全ジョブ	なし	79%	92%	91%	64%	64%	88%
32	〃	〃	0.1%	80%	82%	91%	65%	64%	88%
33	〃	〃	1.0%	81%	93%	92%	67%	68%	89%
34	〃	大型ジョブ	なし	73%	95%	92%	16%	72%	89%
35	〃	〃	0.1%	79%	97%	93%	34%	83%	90%
36	〃	〃	1.0%	84%	97%	93%	50%	83%	90%
37	〃	夜間ジョブ	なし	74%	90%	89%	47%	50%	85%
38	〃	〃	0.1%	78%	92%	90%	55%	60%	87%
39	〃	〃	1.0%	81%	93%	91%	61%	65%	88%
40	短期記憶を10もった場合	全ジョブ	なし	81%	94%	92%	67%	73%	89%
41	〃	〃	0.1%	82%	94%	93%	69%	73%	91%
42	〃	〃	1.0%	83%	95%	93%	71%	77%	91%
43	〃	大型ジョブ	なし	74%	95%	92%	19%	72%	89%
44	〃	〃	0.1%	80%	97%	93%	37%	83%	90%

ケースNo.	予測方法	対象ジョブ	前処理	予測精度			予測寄与率		
				CPU時間	MEM量	I/O回数	CPU時間	MEM量	I/O回数
45	〃	〃	1.0%	85%	97%	94%	53%	83%	92%
46	〃	夜間ジョブ	なし	76%	91%	91%	51%	55%	88%
47	〃	〃	0.1%	79%	93%	91%	57%	65%	88%
48	〃	〃	1.0%	82%	94%	92%	63%	70%	89%
49	長期記憶をもった場合	全ジョブ	なし	81%	94%	93%	67%	73%	91%
50	〃	〃	0.1%	82%	94%	93%	69%	73%	91%
51	〃	〃	1.0%	83%	95%	93%	71%	77%	91%
52	〃	大型ジョブ	なし	75%	95%	93%	22%	72%	90%
53	〃	〃	0.1%	81%	97%	94%	41%	83%	92%
54	〃	〃	1.0%	86%	97%	94%	56%	83%	92%
55	〃	夜間ジョブ	なし	77%	92%	91%	53%	60%	88%
56	〃	〃	0.1%	80%	93%	91%	59%	65%	88%
57	〃	〃	1.0%	83%	94%	92%	65%	70%	89%
58	予測期間を長くした場合	全ジョブ	なし	82%	94%	94%	69%	73%	92%
59	〃	〃	0.1%	83%	95%	94%	71%	77%	92%
60	〃	〃	1.0%	84%	95%	94%	72%	77%	92%
61	〃	大型ジョブ	なし	77%	96%	94%	28%	78%	92%
62	〃	〃	0.1%	82%	98%	95%	44%	89%	93%
63	〃	〃	1.0%	87%	99%	96%	59%	94%	94%
64	〃	夜間ジョブ	なし	79%	93%	92%	57%	65%	89%
65	〃	〃	0.1%	81%	94%	92%	61%	70%	89%
66	〃	〃	1.0%	84%	94%	92%	67%	70%	89%

〔予測を行わない場合〕

利用者が指定した設定上限値に対する資源の実際使用量は，全ジョブでCPU＝42％，MEM＝78％，I／O＝25％，大型ジョブでCPU＝68％，MEM＝82％，I／O＝28％である。CPUは，大型ジョブになるほど利用者の設定が正確になるが，まだ誤差が32％以上もある。ファイル入出力回数は，まだ利用者の設定がおおまかである。

〔システム平均値のみを用いた場合〕

I／Oの予測精度は，25％から80％へと急上昇する。CPU時間も全ジョブ，夜間ジョブに対してはかなり良くなるが，大型ジョブに関してはあまり良くならない。これは，利用者が大型ジョブに関してかなりCPU使用時間に注意を払っているためと見られる。

〔利用者平均値まで利用した場合〕

大型ジョブのI／O回数と夜間ジョブのCPU時間に寄与が見られる。

〔設定値別利用者平均値までを利用した場合〕

設定値別の平均値をもつことは，全ジョブでCPU＝5％，MEM＝7％，I／O＝10％も予測精度が上がる。これは予測寄与率では，CPU＝9％，MEM＝32％，I／O＝13％である。

〔短期記憶を1つ持った場合〕

これは，利用者ごとにその利用者が直前に実行させたジョブの処理データを1つ記憶していることである。これにより，全ジョブでCPU＝3％，MEM＝3％，I／O＝2％と更に予測精度が上昇する。

〔短期記憶を10持った場合〕

これは全ジョブでCPU＝2％，MEM＝2％，I／O＝1％と更に予測精度を上昇させる。

〔長期記憶をもった場合〕

これは，全ジョブでCPU＝0％，MEM＝0％，I／O＝1％の予測精度の上昇だけで，記憶量の割には，予測精度の上昇には寄与しない。

〔前処理の効果〕

これは，大型ジョブなどにおいて，利用者がCPU時間の設定上限値を2時間とし，利用者も2時間近くCPUを使用することを期待していたにもかかわらず，入力データのミスとか翻訳エラーなどで，数秒～十数秒のCPU時間でエラー終了となっているジョブに対して効果がある。

前処理の効果は，Table 2.3のケースNO＝58～66を見ると，CPU時間の設定上限値のそれぞれ0.1％，1.0％を処理することで，以下のようになっている。全ジョブに対しては，あまり効果はない。大型ジョブに対しては，前処理0.1％で，CPU＝5％，MEM＝2％，I／O＝1％，前処理1.0％ではCPU＝10％，MEM＝3％，I／O＝2％も予測精度が上昇している。

この前処理の実際の運用例はつぎのようになる。大型ジョブ（CPU時間≧30分）には，翻訳，入力データのチェックまでを行うジョブ・クラスを設け，MEM量が多くても優先的に処理を行うようにする。

〔予測の精度〕

ここに述べた方法による予測の精度は，最終的にCPUで84～87％，MEMで94～99％，I／Oで92～96％である。MEM，I／Oはかなりよく予測できるが，CPUの予測誤差は約15％とまだかなり高い。これらの原因には以下のものが考えられる。

① 大型計算センタで不特定多数の利用者を相手に多様なジョブ処理を行っている所では，個々のジョブに研究的色彩が強く，同じジョブを実行しても実行のたびにCPU使用時間が違う。CPU使用時間が設定上限値に対して30％一様に変動すると，この方式ではCPUの予測誤差は10％となる。

② 大型ジョブにおいては，本来CPUを5時間程度使用するジョブの場合，原研の運用においてCPU使用時間の上限を2時間にしているため，利用者が2時間，2時間，1時間とジョブをつないで実行しているものがある。このため，このジョブを同じ制御文で実行するとTable 2.4のように予測精度の平均が67％と低くなる。これには，ジョブ入力の周期性を考慮することも必要である。

Table 2.4 Prediction example of periodic computer use

ジョブNo.	CPU設定上限値	予測CPU時間	実際CPU使用値	予測精度
1	120 分	—	120 分	—
2	120 分	120 分	120 分	100 %
3	120 分	120 分	60 分	50 %
4	120 分	60 分	120 分	50 %
5	120 分	120 分	120 分	100 %
6	120 分	120 分	60 分	50 %
7	120 分	60 分	120 分	50 %
		⋮		

3. 各種ジョブ・スケジューリング手法

計算機にジョブを割当て，かつ実行を制御するジョブ・スケジューリングには大まかに分類して2つの方法がある。

ひとつは決定論的（deterministic）方法と呼ばれるものである。この方法では，実行されるジョブの性質，実行時のシステムの振舞い，ジョブ実行の拘束条件などのすべてが事前にわかっていることが前提条件となっている。

他の方法は確率論的（probabilistic）方法と呼ばれるもので，ジョブとシステムの振舞いはこまかくはわかっていないが，大勢としては確率論的分布に従うものと仮定されている。

これら2つの方法の検証の手段のひとつにシミュレーションの方法がある。これは実際のシステムを模擬するプログラムを作成し，定められたジョブ実行系列についてシステムの動きを追跡しようとするものである。

われわれの場合は上記3つの方法を次のようにして使用する。

(1)決定論的方法によってジョブ実行系列の第1近似を作る。

(2)確率論的方法によって最適なジョブ多重度を決定し全ジョブの処理完了時間を推定する。

(3)得られた推定値が正しいかどうかを検証するために実行ジョブ系列についてシミュレーションをおこなう。

3.1 ジョブ多重度の決定

計算機夜間自動運転を円滑かつ効率よくおこなうためには，与えられたジョブの実行系列について当該計算機がジョブ処理を完了する時刻を事前に正確に推定できることが必要である。計算機の演算装置（CPU）1台に複数の入出力制御装置（入出力チャンネルなど）が接続されているのが普通であり，これらの入出力制御装置は並行動作可能である。したがって入出力制御装置の並行動作によってジョブ処理時間の短縮を期待することができる。入出力制御装置を並行動作させるためには同時に実行されるジョブの数を多くすればよい。このときのジョブ個数はジョブ実行多重度（Degree of Multiprogramming）と呼ばれている。従来このジョブ実行多重度はジョブ処理の環境を考慮した適当な値が感覚的に選ばれている。しかしこの多重度はオペレイタ・コンソールからでも，あるいは近年どの計算機システムにおいても開発されつつあるオペレイタ・コンソール自動応答機能ソフトウェアによっても変更できるので，処理すべきジョブ・パターンに応じて多重度を変化させることが可能となった。

そこで次に問題となるのはどのようなジョブ・パターンのときに多重度をどのように変化させればよいかということである。これは古くて新しい問題である。多重ジョブ（プログラム）処理の概念は商用大型計算機の出現した頃に提案され，かつ実現されている。しかしその実現の方法は計算機によっても異なり，理論的な解析が十分におこなわれているとは言い難い。例えば米国の計算機学会（ACM：Association for Computing Machinery）が発行する著名な季刊論

文誌（JACM：Journal of the Association for Computing Machinery），および月刊実務技術論文誌（CACM：Communications of the Association for Computing Machinery）の1965年から1980年までの16年間ジョブ・スケジューリングに関する論文を見ればそのことがわかる。

この16年間にJACMにジョブ・スケジューリング，計算機性能評価に関係する論文は60数編掲載されている。このうち確率論的手法にもとづくものは約39編，決定論的手法にもとづくもの約15編，シミュレーションその他にもとづくもの約8編である。このうち陽にジョブ実行多重度に触れているものは約3編にすぎない。

そのひとつはA. Brandwajn[1]によるもので，頻繁にページ・フォールトを引起す過負荷状態の計算機システムにおいてCPU使用率を最大にするジョブ多重度を計算している。入力率（ページ・フォールト率）はBeladyのlife time functionを仮定し，CPUのサービス分布関数は平均が既知の指数分布としている。ユーザ・プログラムが発生する入出力についての考慮はない。

他のひとつはT. G. Price[2]によるもので，M／G／I／Nの先着順サービスの計算機システムにおけるCPU使用率の上限と下限の巾を求め，その巾とジョブ多重度との関係をグラフで表現している。

またI. Mitrani[3]は演算装置とチャネル装置がFig. 3.1のように接続された処理系において与えられたK個のジョブを固定ジョブ多重度Nで処理するときの全ジョブの終了時間を大雑把に推定する方法を提案した。ただし，これは個々のジョブの終了時間には使えない。

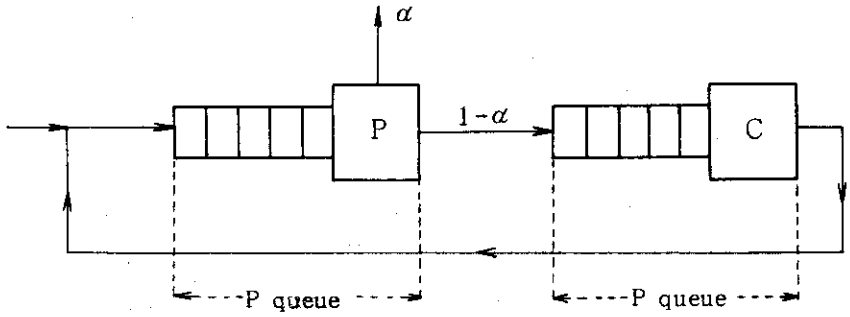


Fig. 3.1　Mitrani's CPU−channel network[3]

CACMについてみると過去16年間にジョブ・スケジューリングあるいは性能評価に関する論文は約30数編ある。そのうち確率論的手法によるものは約16編，決定論的手法によるものは約3編，シミュレーションその他の手法によるものは約14編である。

このうちCPU使用率とジョブ実行多重度について触れているのは2編である。この2編のうちV. I. Wallace[4]によるものは前述のBrandwagenと同様の状況において，CPU使用率とジョブ実行多重度との関係をグラフ化している。Wallaceはこの問題をマルコフ・チェインの平衡確率を計算するプログラムを使用し数値計算で求めている。

他のひとつはD. Towsley et al.[5]によるものでCentral Serverモデルによって CPUと

入出力装置（I／O），I／OとI／Oが並行動作するときのCPU使用率の増加を推定している。多重度は固定である。

以上の事実からみてもジョブ多重度決定法は古くて新しい問題であることがわかる。

これら以外では最近M. Ishiguro[6]がバッチ処理の固定多重度におけるジョブ処理の時間遅れについて論じている。

3.2 決定論的スケジューリング（Deterministic Processor Scheduling）

3.2.1 Flow－shopスケジューリング（Flow－shop Scheduling）

（Ai，Bi）のnケの対が与えられたときnジョブの完了時間を最小にする。ここでA_iは演算装置（CPU）時間，B_iは入出力（I／O）時間とみてよい。

ジョブJ_jをJ_{j+1}より先行させるのは

$\min(A_j, B_{j+1}) < \min(A_{j+1}, B_j)$

のときである。このときジョブJ_jはタスクA_jとタスクB_jとから成り，タスクA_jはタスクB_jに先行するものとする。

ButenとShen[7]はタスクA_j用にmケのプロセッサ，タスクB_j用にnケのプロセッサが用意されているときT_j ＜・T_jとなるのは

$\min(A_i/m, B_j/n) < \min(A_j/m, B_j/n)$

の条件をみたすときであること，およびこのときの上，下限を求めている。

この上，下限の式は巾が広すぎて実際の夜間ジョブ・スケジューリングには使えない。

(1) Johnson[8]の方法，Buten and Shenの方法を採用したときは適切な（A_1，B_1），（A_2，B_2），……，（A_n，B_n），あるいは（A_1/m，B_1/n），（A_2/m，B_2/n），……，のジョブ列を決めなければならない。ところが，私達の一夜の夜間ジョブの数は現在200～300あるので上記ジョブ列を決定することは容易ではない。5分程度の時間のうちに操作員が適切なジョブ系列を決定することは難しい。

(2)このようにひとつのジョブを2つのタスクA_i，B_iにまとめてしまうことは多重プログラム環境のよい近似とならない，という第2の難点もある。なぜならCPUからみた場合，ひとつのジョブはCPU時間の総和がA_j，I_i回の入出力割込みをおこなう小さなタスクの集合である。あるいは平均のCPU滞在時間がA_i/I_i，滞在回数I_iのジョブとみることもできる。ジョブJ_iがI_i回入出力操作をおこなっている間に他の複数のジョブがCPUサービスを受けることができるというのが多重プログラム処理の基本概念である。このとき最適なジョブの組合せ（ジョブの実行系列）とジョブ個数（ジョブ実行多重度）を求めることが我々の目標であるが，ひとつのジョブが要求する資源をA_i，B_iと大きく決めてしまうと多重プログラム処理の実態に合わなくなる。

3.2.2 周期的ジョブ・スケジューリング（Periodic Job Scheduling）

多重プログラム処理の考え方からみればA_i，B_iを分割，細分化したほうがよい。この方法で決定論的なジョブ・スケジュールをおこなう方法として周期的ジョブ・スケジュール法[9]がある。これはTable 3.1のごとき頻度で発生するタスクをFig. 3.2のようにならべて該当する装置の有

効利用をはかろうとするものである。この方法で考えるとわれわれの場合は頻度，実行時間の第1近似として，それぞれ $A_i / (A_i + B_i)$，A_i を採用することができる。

Table 3.1　Example of periodic Jobs

Jobs	Frequency	Period	Execution time
J_1	1/4	4	1
J_2	1/8	8	2
J_3	1/16	16	1 ½
J_4	1/32	32	5
J_5	1/64	64	3

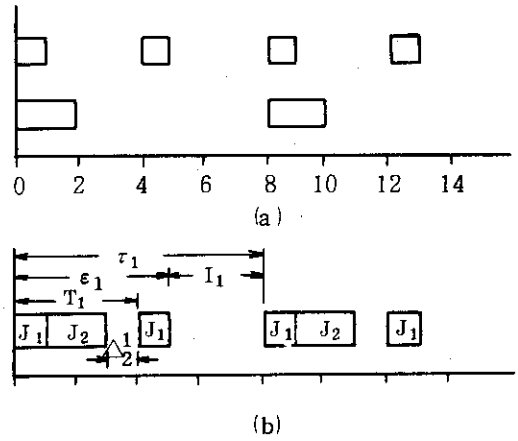


Fig. 3.2　Periodic job scheduling [9]

この方法の問題点は(1) Johnsonの方法と同じく，適切なジョブ実行系列を決定することが難しいこと，(2) Fig. 3.2のごとく強制的に発生する装置アイドル時間 $\triangle_2^1$ をどこまで短縮できるかということである。

周期的ジョブ・スケジューリングのこの難点をおぎなう方法にLiu and Laylond [10]のrate monotonic schedulingとdead line schedulingがある。上記(1)の問題についていえばrate monotonic schedulingは頻度の高いタスクに高い優先権を与えて実行する。このようなジョブ系列を作ることに時間はかからない。このとき同時に実行されているタスクの実行多重度をmとすると装置使用率の下限Uは

$$U = m\left(2^{\frac{1}{m}} - 1\right)$$

で与えられる。

ここでUはわれわれの場合は

$$U = \sum_{i=1}^{m} A_i / (A_i + B_i)$$

で定義される。

$m = 2$ のとき $U = 2\left(2^{1/2} - 1\right) \cong 0.83$,

$m = 3$ のとき $U = 3\left(3^{1/3} - 1\right) \cong 0.78$,

$m \to \infty$ のとき $U \to \ln 2$ となる。

mが大きくなると使用率の下限Uの値は小さくなるが，このスケジューリング法をジョブ系列決定の第1近似として用いることはできる。しかしこの後でのべるスケジューリング法によっても，われわれのジョブ処理環境では高い装置使用率を達成できることがわかっている。

上記(2)の難点を解消するスケジューリング法にdead line schedulingがある。

このスケジューリング法では，現在実行されているタスクのうちで，次回に要求がくる時間間隔がもっとも短いものから順に優先権が与えられる。この優先権決定法は最初から決定されたも

のではなく，ジョブ処理環境の変化に従う動的なものである。通常の計算機のオペレーティングシステムではこのようなスケジューリング法は採用されていない。

このスケジューリング法が適用可能なのは実行されているm個のタスクの間につぎの条件が成立しているときである。

$$A_1/(A_1+B_1)+\cdots+A_m/(A_m+B_m)\leq 1$$

不等号が成立しないときはその所定の周期内に実行が完了しないタスクがでてくるが，われわれの多重プログラム処理の環境では全体のジョブ処理がおくれるだけなのであまり問題はない。したがって条件式

$$A_1/(A_1+B_1)+\cdots+A_m/(A_m+B_n)\cong 1$$

をみたすタスクは実行すべきタスク，したがってジョブ系列の第一近似としてよいと考えられる。しかしここでも問題となるのはこの式をみたすジョブの組合せを求めるには時間がかかることである。

3.2.3 LPTジョブ・スケジューリング（Largest Processing Time (LPT) Job First Scheduling）

n個の処理装置が利用可能であるときm個のジョブをこれらのプロセッサに配分する方法のひとつにLargest Rocessing Time job first scheduling[11]がある。これはまた単にLPT schedulingと呼ばれることが多い。この方法の良い点は，どのようなジョブ実行系列をとっても，それがLPT schedulingの方法に従っている限りは，処理完了時間が最短（W_O）であるジョブ系列と処理完了時間が最長（W_L）であるジョブ系列の時間W_O，W_Lとプロセッサ数nについて

$$W_L/W_O \leq 4/3 - 1/3n$$

という不等式が成立することである。

ただしここで各ジョブはそれぞれが独立で，順序について制約がないと仮定している。

ひとつのジョブの実行時間をA_i+B_iとみなしたときは，このLPT法によるジョブの実行系列決定は非常に単純になり，そのための時間もあまりかからない。とくにどのジョブJも$A_i \gg B_j$の傾向を持っているときは，このLPT法で定めたジョブの実行系列は，われわれの期待するジョブ実行系列の第1近似となり得る。

また長時間ジョブをまず最初に片づけておくことは，夜間ジョブ処理において重要なことである。処理時間の短いジョブが未処理で残っても，それらは昼間の運用原則を変更することなく昼間ジョブとして処理できるが，長時間ジョブが未処理で残った場合はそうはいかない。

逆にジョブが$A_i \ll B_j$の傾向を持っているときは注意を要する。$A_j \ll B_j$であるジョブは演算装置（CPU）を使用する時間よりは入出力装置を使用する時間のほうが長いことを意味する。

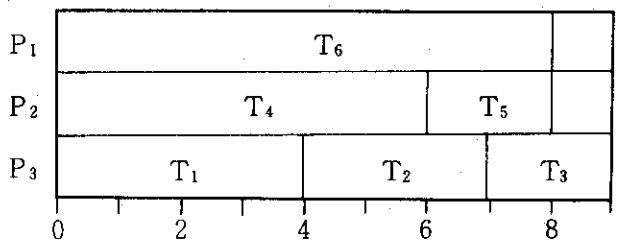
演算装置の最近の性能向上にともなって従来は$A_j \cong B_j$であったジョブは$A_j \ll B_j$の傾向を示すようになってきた。

ひとつの演算装置には通常複数の入出力制御装置が接続されているので，$A_j \ll B_j$のジョブが多いときは多重プログラム処理の効果が顕著に現われる。この効果を定量的に推定し夜間自動ジョブ処理に役立てることはこれまでおこなわれていなかった。

本報告において筆者らはそのための手法と得られた結果を示し，それによって夜間ジョブの自動スケジューリングのみならず，ジョブ処理の環境データを利用した計算機の知的な運転が可能であることを示唆しよう。

筆者らの計算センタにおける現在の夜間ジョブは$A_j \cong B_j$あるいは$A_j \gg B_j$となるものが多いので，LPT法が定めたジョブの系列は該当するプロセッサが対象とするジョブ実行系列の第1近似とみなすことができる。したがってジョブの実行系列決定の方法にLPT法を採用した。

LPT法を採用したときの例を Fig. 3.3 にかかげる。これはジョブを3個のプロセッサに配分したときの例である。



Pi : i th processor

Ti : i th job

Fig. 3.3 Example of LPT shedule[9)]

ひとつの演算装置（CPU）に複数の入出力制御装置（チャンネルなど）が接続されているのが普通である。そして計算機制御プログラムの資源配分の方法はA_1, B_1 を細分化し，ジョブ（A_1,B_1）が入出力動作をおこなっているときは，CPUをジョブ（A_2,B_2）に解放する。A_2, B_2 なども細分化されて実行される。

われわれの仮定では入出力動作は入出力制御装置の最大数までは並行動作可能であるとしている。

このときつぎの(1), (2)の問題に解決を与えることができれば夜間ジョブの自動処理に，少なくともCPUに関しては解決の糸口を見出したことになる。

(1)与えられたジョブ実行系列に対してCPU資源をもっとも有効に使用するジョブ実行多重度の決定。

(2)このときのジョブ処理完了時間の推定。

これらの問題を決定論的な手法で解くことは難しい。なぜなら第1には前述のButen and Shenの方法に見られるように，ジョブを細分化してもジョブ処理完了時間が必らずしも短縮できるとは限らないからである。この事実は決定論的方法におけるAnomaly of Multiprocessing Timing[11), 12)]として知られている。

第2には現実の計算機オペレイティング・システムのCPU資源配分の方法は決定論的手法で取扱うには複雑すぎるということによる。

そこでわれわれは上記(1), (2)の問題を解決するために2つの手法を考案した。

そのひとつは確率論的（待ち行列理論的）手法にもとづくジョブ・スケジューリング方法であり，他の方法は現実のオペレイティング・システムの動きを忠実に再現するシミュレーション手法によるものである。

両者の手法はそれぞれFortranプログラムで実現されている。以後前者をスケジューラ，後者をシミュレータと呼ぶことにする。

3.3 確率論的スケジューリング（Probabistic Scheduling）

計算機性能評価，性能予測ではFig. 3.4にみられるcyclic queueモデルやFig. 3.5のcentral serverモデルを待ち行列理論を使って解く方法が用いられてきた。これらのモデルは現実の計算機システムの記述に有用であることが多くの報告でのべられている。ただし，それらは入力率 λ_1，λ_2 …，装置使用率 μ_1，μ_2 …の確率的分布と1，2次のモーメントが既知の場合である。通常入力率の分布はポアソン分布が，装置利用率の分布は指数，またはアーラン分布が仮定される。

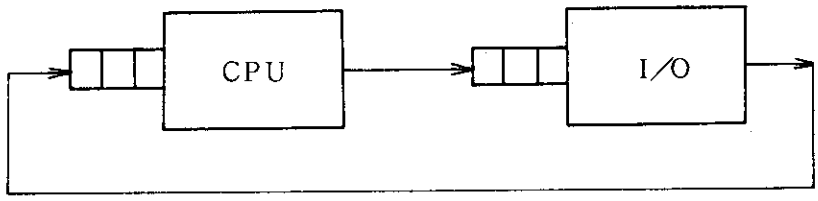


Fig. 3.4　Cyclic queue model

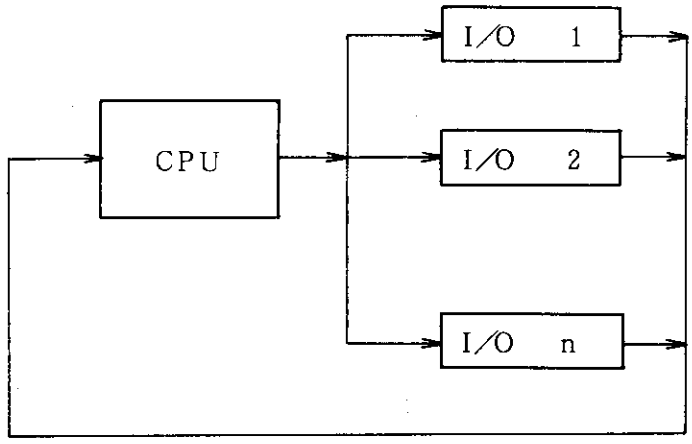


Fig. 3.5　Central server model

適当なジョブ群を計算機システムで処理し，λ_i，μ_i をまず測定すると，これらのパラメータは当該計算機のオペレイティング・システムのオーバーヘッドなども含んでいる。このパラメータを使って上記モデルを作成し，同じジョブ群を処理するという仮定のもとに，CPU台数，チャンネル台数の増減に関し，計算機システムの性能の内挿，外挿をおこなうと，現実の値とよい一致を示す[13),14)]。

このとき使用するジョブ群の特性は年間平均のジョブ・パターンのこともあり，ベンチマーク・テスト用のジョブ・パターンのこともある。

これは結果論である。ジョブを実行した結果得られた値を使って大局的あるいは平均的な性能評価をおこなうときはこの方法も有効である。

われわれの夜間ジョブ・スケジューリングにこの従来の方法は使えない。夜間に実行されるジョブの組合せは毎回変化するのでパラメータの値は常に変化する。パラメータの値がジョブ実行後にわかるというのでは意味がない。というわけで短期的，あるいは局所的なジョブ・スケジューリングを正確におこなうことは非常に難しい。ところがわれわれが最近見出したジョブについての特性を利用すれば，この問題は意外と簡単に解決できることがわかった。この特性については第2章で既に記述した。

原研計算センタで処理されているジョブをひとつひとつ眺めてみると，それらのCPU使用時間，入出力回数，記憶容量は前回の処理時とさほど違いがないことがわかった。どのジョブもこれら計算機資源を必要とする上限値を指定して入力されるが，前回の実際使用量を計算機システムで記憶しておけば，この指定値に関係なく必要量を推定できることがわかっている。つまりひとつのジョブ J_j を $J_j = (a_j, b_j * c)$ で特徴づけることができるのである。ここで a_j，b_j，c は，それぞれCPU使用時間，入出力回数，一回当りの平均入出力時間である。現在は十分な記憶容量を持っているので，記憶容量についてはとりあえず考慮の対象外とする。

$J_j = (a_j, b_i * c)$ を (A_j, B_j) と表現することもある。

4. スケジューラ

この章では夜間ジョブをスケジュールするためのスケジューラのひとつの方式を提案する。このスケジューラは，ひとつの計算機システムに入力されて処理を待っている個々のジョブの要求CPU時間と入出力回数が既知であるという前提条件のもとで，CPU利用率を最高にするジョブ実行多重度を決定する。

前章でも述べたように大型計算機システム群に入力される個々のジョブの特性（ここではCPU使用量a_j，入出力回数b_j）は1カ月程度の短期間では変らない。したがって前回使用のa_j，b_jをジョブ特性を示す値として使用することができる。しかし入力されるジョブの組合せは毎回変化するので計算機システムにとっては毎回異ったジョブ処理となる。このために毎夜のジョブ処理終了時間は同じではない。ここで述べるスケジューラはジョブ・スケジューリングの方法のうちではRound Robin（以下RRと略す）法として知られている古典的なものを少し修正したものである。RR法（Fig. 4.1）においてはジョブは入力率λ（単位時間当りのジョブ入力個数）でCPUサービスを要求する。サービスはひとつのまとまった装置時間Q（これをタイム・クワンタムという）単位でおこなわれ，時間Qでおわらないジョブは CPUサービスを待っている待ち行列の最後について再びサービスを待つ。ただしこのフィード・バック・ループは新しいジョブの到着とはみなさない。すなわち入力率λに寄与するとはみなさない。

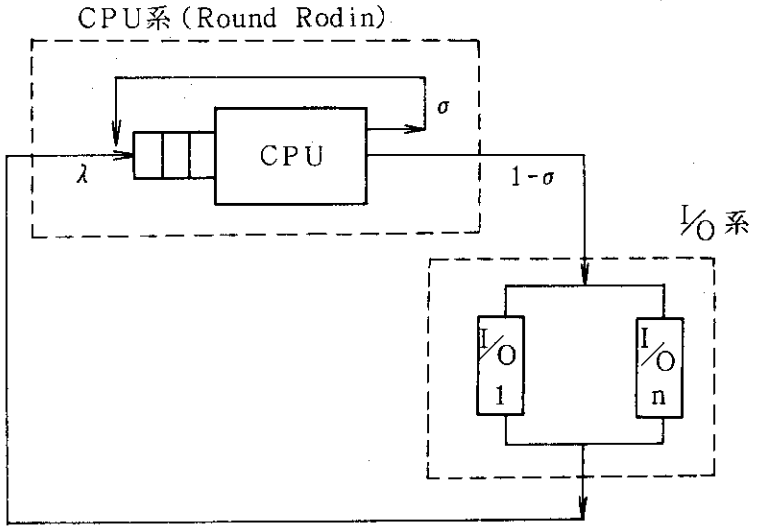


Fig. 4.1　Our model

(1) RR法の特長はCPUサービス時間分布を陽に計算できることである。Fig. 4.1において，CPUサービスを受けたジョブがフィードバックされる確率をσとすると，i回のタイム・クワンタを要求するジョブがサービスを受ける確率g_iは

$$(4.1)\quad g_i = \sigma^{i-1}(1-\sigma),\ i = 1, 2, \cdots$$

で与えられる。このg_iを幾何分布に従うという。

サービス時間分布の期待値$E(S)$は

$$(4.2)\quad E(S)=\sum_{i=1}^{\infty}(iQ)g_i=\frac{Q}{1-\sigma},$$

またCPU利用率ρは

$$\rho=\lambda E(S)$$

となる。

(2) このσをシステムに入力されているジョブの特性から割出すことができればシステムのジョブ処理性能を推定することができる。

いまジョブ

$$J_i=(a_i, b_i\cdot c)\quad i=1, \cdots, M$$

が1多重でFig. 4.1のシステムで実行されるとするとCPUへの入力率λ_iは近似的に

$$\lambda_i=(b_i+1)/(a_i+b_i\cdot c)$$

とかける。ここで各ジョブJ_iが入力率がそれぞれλ_iのポアソン分布に従ってCPU要求をおこなうと仮定しよう。説明の便宜上このひとつのCPU要求を擬ジョブJ_iと呼ぶことにする。

$$Q=\min(a_i/b_i),\ i=1, \cdots, M,$$

$$N_i\cong(a_i/b_i)/Q,\ i=1, \cdots, M,$$

ただしN_iは$(a_i/b_i)/Q$を四捨五入した整数値とし，上記M個のジョブがFig. 4.1のシステムにおいて滞在していると仮定するとCPUサービスを要求する全ジョブの到着率λの近似値は

$$\lambda=\sum_{i=1}^{M}\lambda_i$$

となる。また$\lambda'=\sum_{i=1}^{M}N_i\lambda_i$とすると入力率が$\lambda$であるときこの$\lambda'$はフィードバックまで勘定に入れたCPUサービス要求の率を示している。したがってFig. 4.1でジョブがCPU系から離れていく確率$1-\sigma$は

$$(4.3)\quad 1-\sigma=\lambda/\lambda'$$

で近似できる。

(3) kクワンタを要求するジョブのCPU系における滞在時間W_kの計算法はKleinrock[15]，Coffman & Denning[16]らによって与えられている。

kクワンタを要求する擬ジョブの第1回目の入力率$\lambda_k=\lambda_k^{(1)}$は，実際の入力率よりも大きく出ている筈である。なぜなら各擬ジョブは多重度1で処理されるとしてλ_kが計算しているから。そこでこのλ_kを使ってW_kを計算すると，W_kの値は実際のW_kよりも大きくなる。このW_kでλ_kを再計算する。この操作を繰返す。模式的に書くと

$$(\lambda^{(1)}{}_k\text{の増大})\rightarrow(W_k{}^{(1)}\text{の増大})\rightarrow(\lambda^{(2)}{}_k\text{の減少})\rightarrow$$
$$(W_k{}^{(2)}\text{の減少})\rightarrow(\lambda^{(3)}{}_k\text{の増大})\rightarrow\cdots\rightarrow\text{収束。}$$

W_kの計算はCoffman & Denning[16]の導出とほとんど同じであるが，ここでシステム内に滞在する擬ジョブ数が有限であるという点が一般のRR方式と異なる。各ジョブは並行タスクを発生しないものとする。したがって現在実行中のジョブが終了し，新しいジョブが起動されない限

りはジョブ実行多重度は変化しない。

(4) Mをジョブ実行多重度，k クワンタのサービスを要求する擬ジョブがCPU系に到着したときのシステムに滞在する擬ジョブ数がj 個のときの条件つき待ち時間を$W_k(j)$とすると，待ち時間W_kは

$$(4.4)\quad W_k = \sum_{j=0}^{M} P_j W_k(j)\ ,$$

ここで$\{P_j\}_{j=0}^{M}$はCPU系にj 個の擬ジョブが存在する定常確率を示す。これはCPU系にj 個のジョブが存在するときの確率と同じである。この確率の定義と計算法については後で述べる。以下Coffman & Denning に従ってW_kの計算法を述べる。

k クワンタのサービスを要求する擬ジョブはCPU系の待ち行列をK回通過することになる。1回の通過は擬ジョブが待ち行列の最後に加わった瞬間から次に待ち行列の最後に加わるまで，あるいはそれが最後のクワンタであればその擬ジョブがCPU系を離れるまでとする。

$U_i(j)$，j ＝ 1，2，…，K をCPU系にj 個のジョブが存在するときに擬ジョブがi 番目の通過に要する時間をあらわす確率変数とすると，

$$(4.5)\quad W_k(j) = \sum_{j=1}^{k} E[U_i(j)]\ .$$

$U_i(j) = X$，$i \geq 2$とすると，$(X/Q) - 1$はi 番目の通過をおこなう擬ジョブの前に存在しているジョブの数を示しており，このうち平均すると$\sigma[(X/Q) - 1]$が(j ＋ 1)番目の通過をおこなう。現在注目しているジョブがi 番目の通過をおこなう間にCPU系に到着する擬ジョブの平均個数はλXであるから

$$(4.6)\quad E[U_{i+1}(j) \mid U_i(j) = X] = \sigma Q[(X/Q) - 1] + \lambda X Q + Q$$

上式の両辺に確率密度dF (X)を乗じて積分すると，

$E[E[X \mid Y]] = E[X]$であるから，

$$(4.7)\quad E[U_{i+1}(j)] = (\lambda Q + \sigma) E[U_i(j)] + Q(1-\sigma),\ i = 2,\ 3,\ \cdots\cdots$$

またi ＝ 1のときは

また現在注目している擬ジョブがCPU系に到着したとき，すなわちi ＝ 1のときはシステム内に，当該擬ジョブまで含めてj 個の擬ジョブが存在するから式(4.6)においてX／Q－1 ＝ jである。したがって，

$$(4.8)\quad E[U_2(j)] = \lambda Q E[U_1(j)] + Q(\sigma j + 1)。$$

式(4.7)から

$$(4.9)\quad E[U_i(j)] = (\lambda Q + \sigma)^{i-2} E[U_2(j)] + Q(1-\sigma)[1 - (\lambda Q + \sigma)^{i-2}] / (1 - \lambda Q - \sigma)$$

(4.9)を(4.5)に代入し，

$\lambda Q+\sigma=\alpha$，$\lambda Q/(1-\sigma)=\rho$ とおくと

$$(4.10)\quad W_k(j)=E[U_1(j)]+(k-1)Q/(1-\rho)$$
$$+Q[\lambda E[U_1(j)+\sigma j-\rho/(1-\rho)](1-\alpha^{k-1})/(1-\alpha)。$$

式（4.4），（4.10）を組み合せ，また

$$W_1=\sum_{j=0}^{M} P_j E[U_1(j)]$$

であるから

$$(4.11)\quad W_k=W_1+\frac{(k-1)}{1-\rho}$$
$$+Q\left[\lambda W_1+\sigma\bar{n}-\frac{\rho}{1-\rho}\right]\frac{1-\alpha^{k-1}}{1-\alpha},\quad k\geq 1,$$

ここで $\bar{n}=\sum_{j=0}^{M} P_j \cdot j$ は CPU系における平均待ち行列個数である。この $\bar{n}$ は以下に述べる有限待ち行列の平均待ち行列個数で近似する。

有限待ち行列系の最大待ち行列個数をM，状態確率を P_n とすると，定常状態における行列個数はn個からn－1個へ，またはn－1個へと変化するから，λ を系への入力率，μ をサービス率とすると，

$$(4.12)\quad \lambda P_{n-1}+\mu P_{n+1}=(\lambda+\mu)P_n,\quad 1\leq n\leq M-1,$$

$$(4.13)\quad k P_1=\lambda P_0,\quad n=0,$$

$$(4.14)\quad \lambda P_{n-1}=\mu P_M,\quad n=M,$$

$$(4.15)\quad \sum_{0}^{N} P_n=1$$

の関係が成立する。[17),18)] これから $\frac{\lambda}{\mu}=\rho \gtrless 1$ のときは

$$P_n=\rho^n P_0\quad(\rho \gtrless 1)$$

$$P_0=\frac{1}{\sum_{0}^{M}\rho^n}=\frac{1-\rho}{1-\rho^{M+1}}$$

から

$$(4.16)\quad P_n=\frac{1-\rho}{1-\rho^{M+1}}\rho^n,$$

となり，平均待ち行列個数 $\bar{n}$ は

$$(4.17)\quad \bar{n}=\frac{1-\rho}{1-\rho^{M+1}}\sum_{0}^{M} n\rho^n=\frac{\rho\{1-(M+1)\rho^M+M\rho^{M+1}\}}{(1-\rho)(1-\rho^{M+1})}。$$

$\rho=1$ のときは

$$P_n=P_0,\quad \sum_{0}^{M} P_n=(M+1)P_0=1$$

から　$P_0 = \dfrac{1}{M+1}$，$\bar{n} = \sum_{0}^{M} n_M / (M+1) = M/2$。

ある擬ジョブが待ち行列のうしろに到着したときのCPUでのサービス残り時間の期待値を E（Qr）とすると，

（4.18）$W_1 = E(Q_r) + \bar{n}_q Q + Q$。

ここでは $\bar{n}_q$ は平均待ち行列個数（サービス中の擬ジョブは含まない）で，この $\bar{n}_q$ は $\bar{n}_q = \bar{n} - \rho$ なる関係をみたす。ここで ρ はCPUの使用率であって $\rho = \lambda E(S)$ で定義される。E（Q_r）はジョブが行列に到着したときに，処理進行中のジョブがそのクワンタム内で費す残り時間である。このE（Q_r）は $\rho Q/2$ [16]となる。

λ は $\lambda = \lambda_1 + \lambda_2 + \cdots\cdots + \lambda_M$ である。

(5) このようにしてkクワンタを要求する擬ジョブでのCPU系での滞在時間 W_k が計算される。しかしこのままでは（4.1）式で与えられる g_i の計算法に問題があって正しい答は得られない。それを例で示そう。

システム内にTable 4.1のジョブが滞在しているとする。（4.1）式の定義によって σ の値は $1 - \sigma = 4/(8+4+11+1) = 4/24 = 0.167$ となり，g_i の値はFig. 4.2の斜線の部分のようになる。

Table 4.1　Example of jobs

ジョブ番号	要求クワンタ数	入力率	g_i 値
1	8	1	0.046
2	1	1	0.167
3	4	1	0.097
4	11	1	0.027

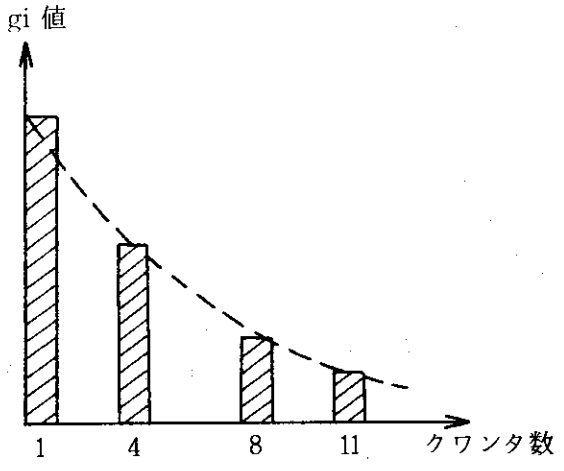


Fig. 4.2　g_i values and required quanta

Fig. 4.2 から明らかなように我々の W_k の計算法にはこのままではつぎの2つの欠点があることがわかる。

(a) RR 法では CPU 系に到着する擬ジョブが無限個の要求クワンタ数をもつ場合も勘定に入れているが現実にはそのようなことは起り得ない。

(b) ジョブ数が有限個であるから，Fig. 4.1 の斜線以外の g_i 値をもつ擬ジョブが存在しない。

この2つの難点を解決するためにつぎのように擬ジョブ群を細分割し，g_i 曲線の近似を試みる。

$$\ell = \sum_{i=1}^{M} N_i \ , \quad M\text{はジョブ多重度},$$

$$S = \sum_{i=1}^{L} i \cdot g_i \ , \quad L = \max \{ N_i \mid i = 1, \cdots, M \},$$

$$\beta = S / \ell \ ,$$

$$m_i = g_i / \beta$$

とすると，β はひとつのワンタムのジョブが g_i 曲線のなかで占めるべき面積を示している。このときの g_i 曲線は Fig. 4.3 のごとく横軸 L の点で打切られていて，L＋1 以後の g_i は無視する。m_i は g_i 値の部分に1つワンタムの擬ジョブなら $i \cdot m_i$ 個，i ワンタムのジョブなら m_i 個収容できることを示している。

$$S = 3.71, \quad \beta = 3.71/24 = 0.155$$

であるから，収容可能な擬ジョブの個数は Table 4.2 のようになる。S は $1/(1-\sigma) = 24/4 = 6$ に近いことが望ましいから，この例では近似による誤差は非常に大きいことになる。

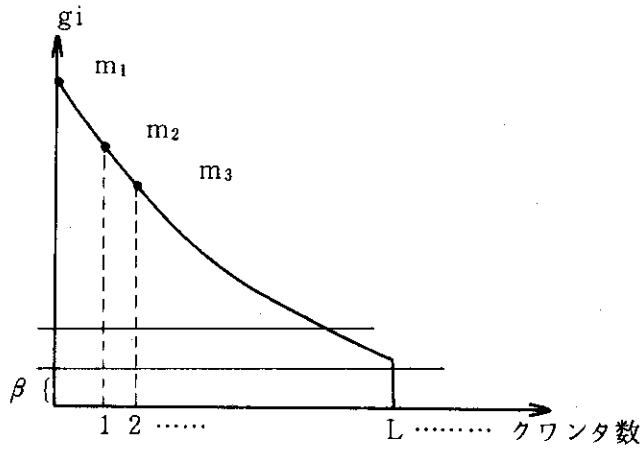


Fig. 4.3 Refined g_i

4つのジョブから発生する擬ジョブの要求するクワンタを振分ける方法は数多くあり，任意性があるために，ここで各ジョブの処理優先順位を導入することも可能であるが，われわれの場合は先行するジョブをより小さな i 値の部分へ寄せてゆく，すなわち早く処理されるようにしている。

Table 4.2 Example of allowable pseudo job combinations

クワンタ数	g_i 値	$i \cdot g_i$ 値	収容可能個数
1	0.167	0.167	1.1
2	0.139	0.278	0.90～1.8
3	0.116	0.348	0.75～2.2
4	0.097	0.388	0.63～2.5
5	0.080	0.400	0.52～2.6
6	0.067	0.402	0.43～2.6
7	0.056	0.392	0.36～2.5
8	0.046	0.368	0.30～2.4
9	0.039	0.351	0.25～2.3
10	0.032	0.320	0.21～2.1
11	0.027	0.297	0.17～1.9
合　計	0.866	3.71	0.62～22.9

いずれにしてもkクワンタを要求する擬ジョブのCPU系における滞在時間 W_k は，γ_i を前頁で述べたごとく

$$(4.18)\quad W_k = \sum_i \gamma_i W_i ,$$

$$k = \sum_i \gamma_i \cdot i$$

として近似される。

(6) 以上のようにして第1近似 λ_k を使って待ち時間 W_k^j の第1近似が得られた。
この W_k^j を使ってジョブ J_j の滞在時間 E_k^j は

$$E_k^j = (\text{I／O 回数})_j \times (W_k^j + 1\text{回のI／O 時間})$$

$$= b_j \cdot (W_k^j + C)。$$

新しい $\lambda_k^{(2)}$ は

$$\lambda_j^{(2)} = \frac{(\text{I／O回数})_j}{E_j^{(1)}} = \frac{b_j}{b_j \cdot (W_k^j + C)} = \frac{1}{W_k^j + C}$$

として求める。

$$\lambda^{(i+1)} = \lambda_1^{(i+1)} + \cdots + \lambda_M^{(i+1)}$$

として $\dfrac{|\lambda^{(i+1)} - \lambda^{(i)}|}{\lambda^{(i)}} < \varepsilon_\lambda$ ，ε_λ ：given.

となるまで計算を繰返す。（λ－iteration）。
上の式をみたす $\lambda^{(i)}$ を λ とするとCPU利用率 ρ_M は

$$\rho_M = \lambda E(S)$$

Mは系内でのジョブ滞在数。

(7) ジョブ多重度MをM＝1，2，…と動かしてρ_Mが最大となるM あるいは

$$\frac{|\rho_{M+1}-\rho_M|}{\rho_M} < \varepsilon_M,\ \varepsilon_M : \text{given}$$

となるMを求めるべき多重度とする。

(8) 多重度Mによってジョブ処理がおこなわれたとする。最初に終了するジョブJ_jの経過時間をE_jとすると，ポアッソン入力の仮定から，この間に各ジョブJ_kが発信した入出力回数b'_kは

$$b'_k = \sum_{i=0}^{\infty} i\frac{(\lambda_k E_j)^i}{i!}e^{-\lambda_k E_j} = \lambda_k E_j \sum_{1}^{\infty}\frac{(\lambda_k E_j)^{i-1}}{(i-1)!}e^{-\lambda_k E_j} = \lambda_k E_j .$$

b_jはジョブJ_jの現実の入出力回数b_jと一到することがのぞましいが実際はジョブ・パターンによって差がでてくる。

(9) $b_k - b'_k > 0$を新しいb_kとする。$b_k - b'_k \leq 0$のときは計算を打切る。$a_k - a'_k$を新しいa_kとし，$E_k = a_k + b_k \cdot c$によって新しいE_kを求め，b_k / E_kを新しいλ_kとする。前回の同様操作で新しいMを求める。

(10) 上記操作を1システムに割当てられたNジョブについておこない該当システムの終了時間を求める。

他システムの終了時間と比較し，最短時間で終了するシステムLPTルールでジョブの配分を試みる。

(11) 以上の操作を流れ図で表示したものがFig. 4.4（概略図），Fig. 4.5（a～f）（詳細図）である。計算結果の検討は第6章においておこなう。

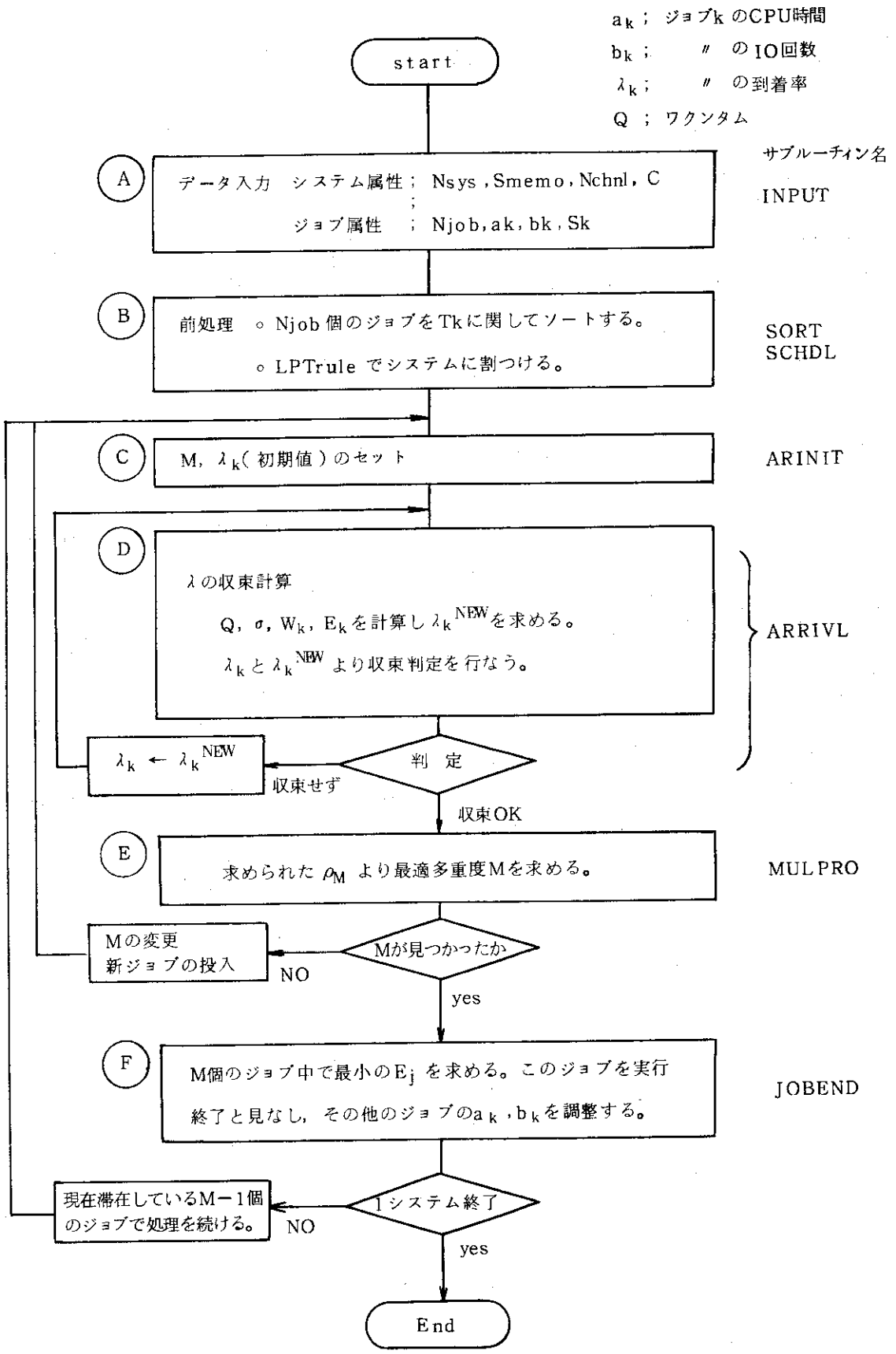


Fig. 4.4　General flow of scheduler

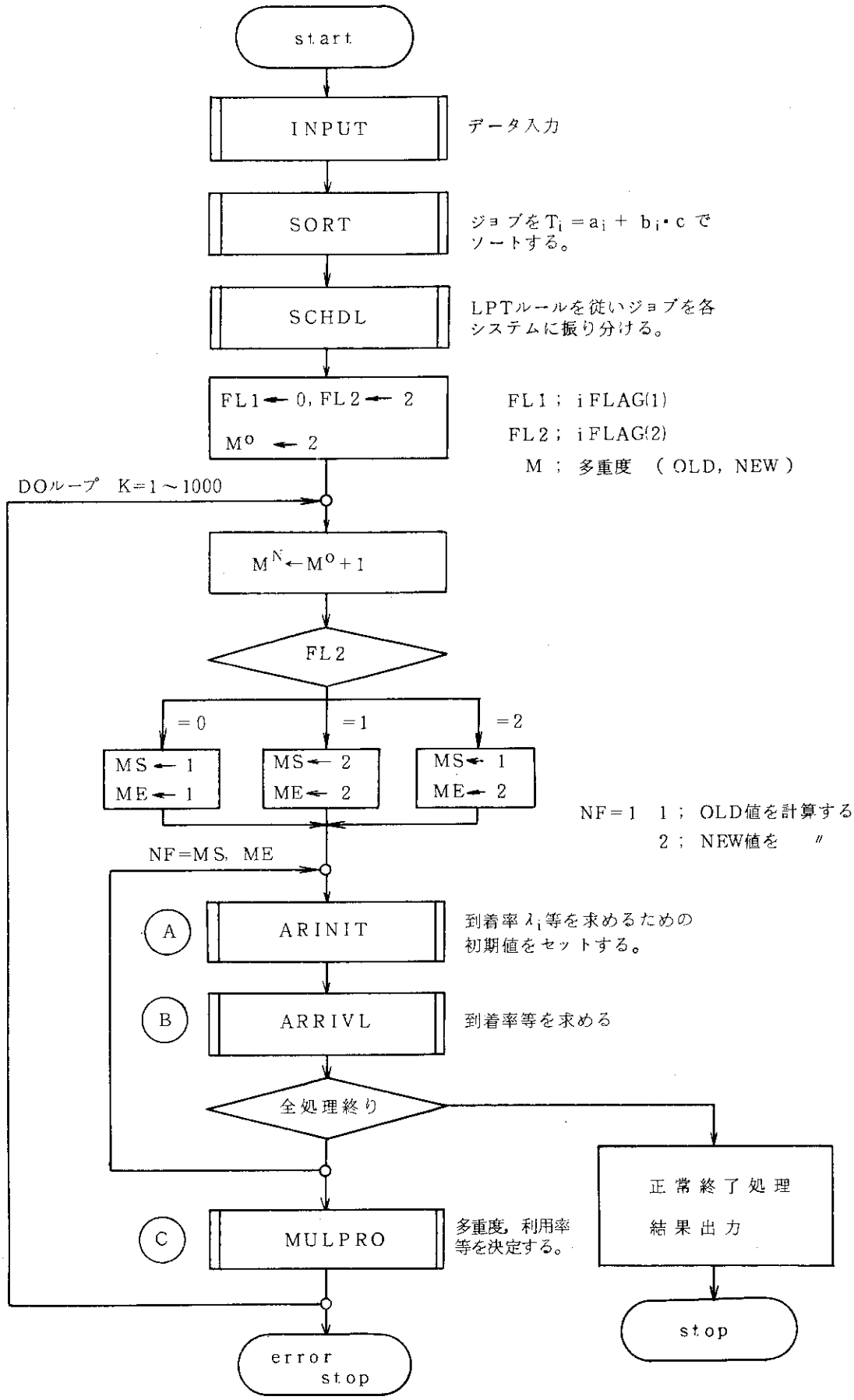
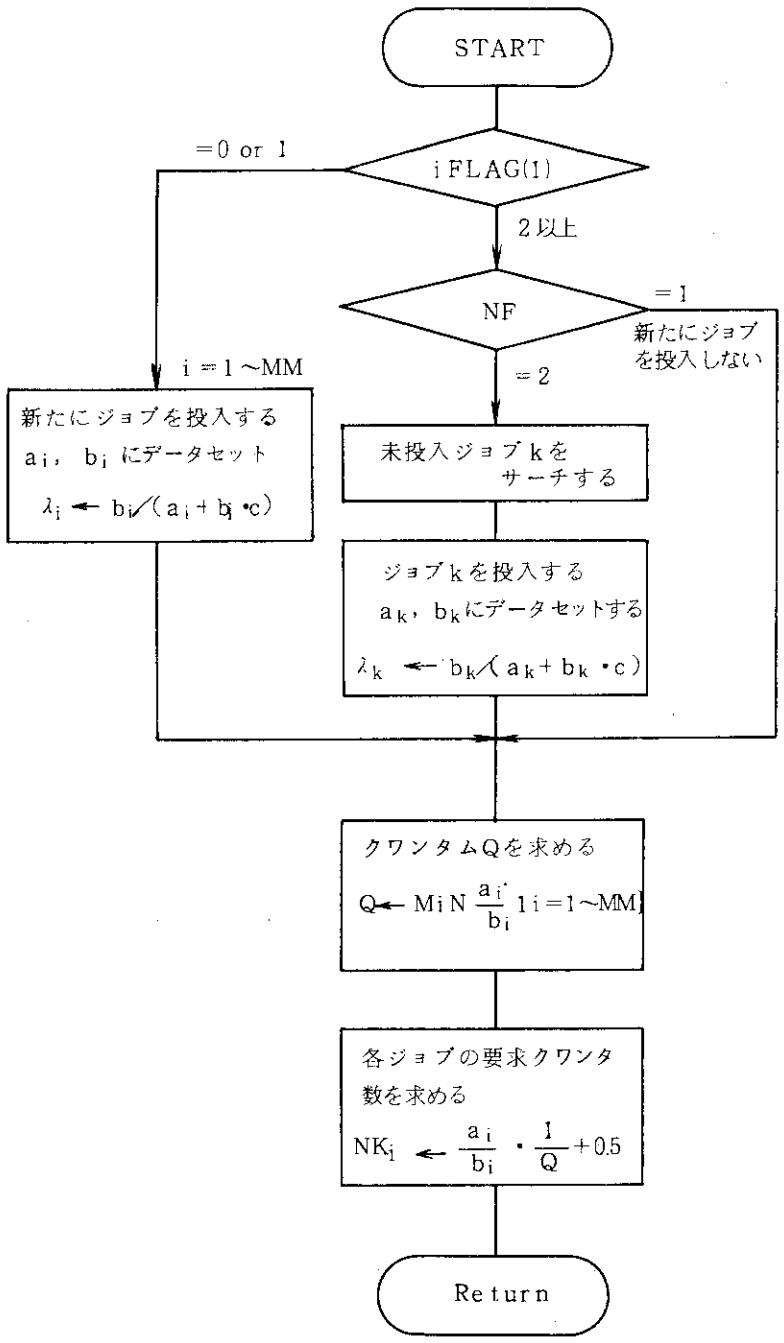


Fig. 4.5(a) Detail flow of scheduler (MAIN)

Ⓐ　サブルーチンARINIT（NF, NS, QQ, LQ, KM, IRET）
到着率を求めるための初期データをセットする。



NF＝　0；Old data
　　1；New data

NS；システム番号

iFLAG；コントロールフラッグ

MM　；多重度

a_i　；CPU使用時間

b_i　；IO回数

c　；1回あたりのIO時間

Q；クワンタム

NK_i；要求クワンタ数

Fig. 4.5(b)　Detail flow of scheduler (ARINIT)

Ⓑ　サブルーチン ARRIVL（NF, NS, CC, QQ, LQ, KM, IRET）

到着率を収集計算により求める。

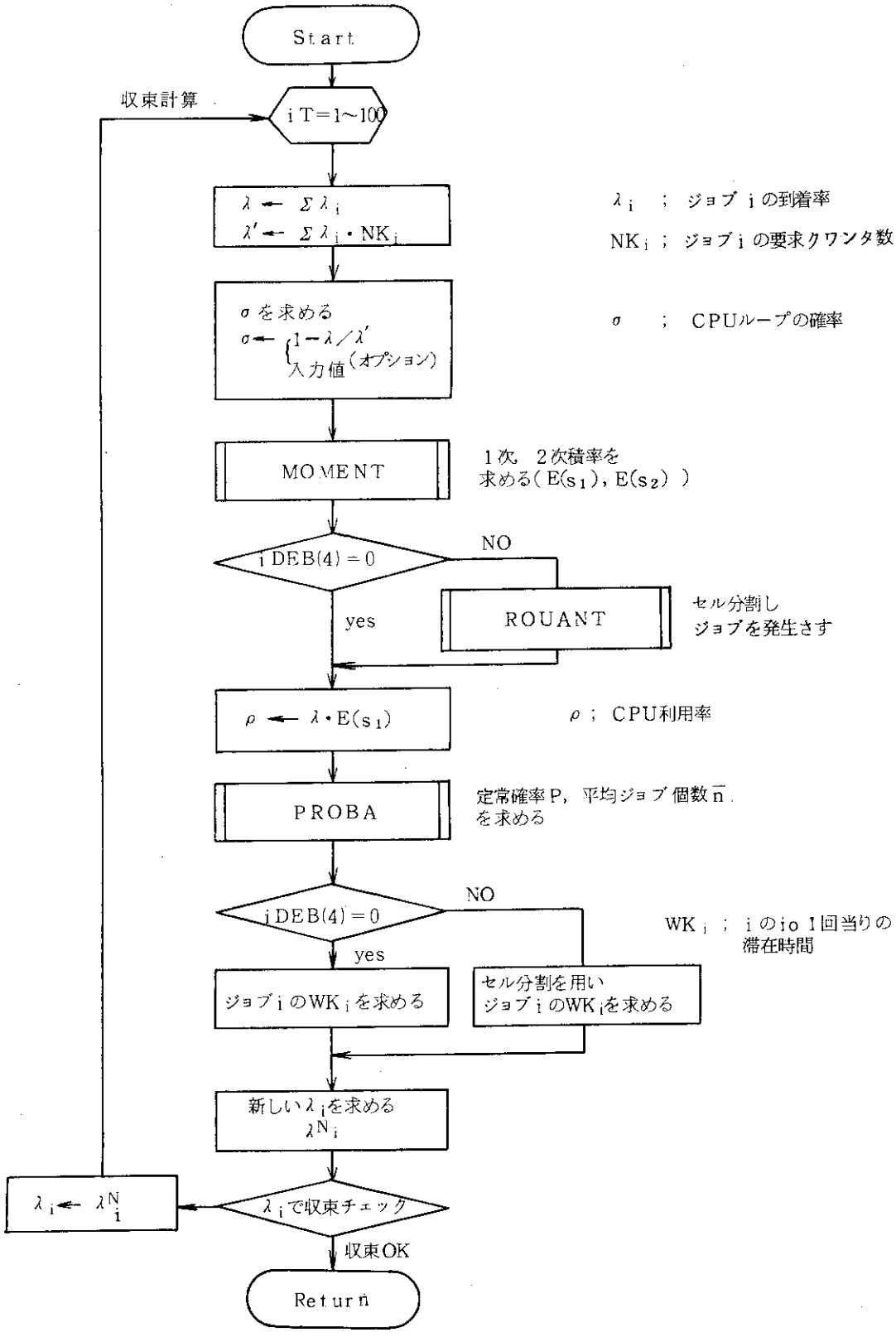


Fig. 4.5 (c)　Detail flow of scheduler (ARRIVL)

© サブルーチンMULPRO（NS, IRET）
多重度を決定するコントロールルーチン

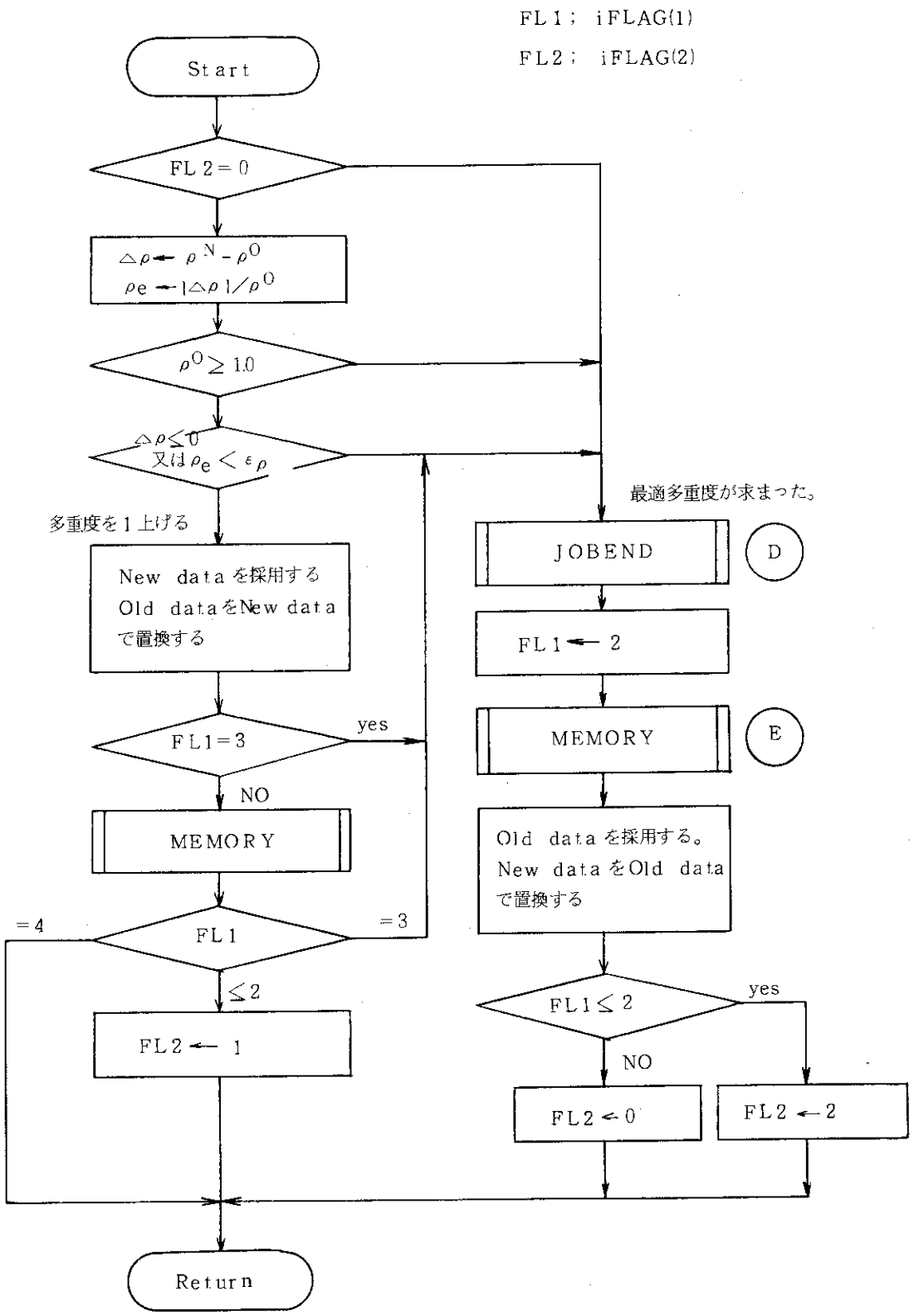


Fig. 4.5(d) Detail flow of scheduler (MULPRO)

⑪ サブルーチン JOBEND (NF, NS, IRET)

終了ジョブを処理する。

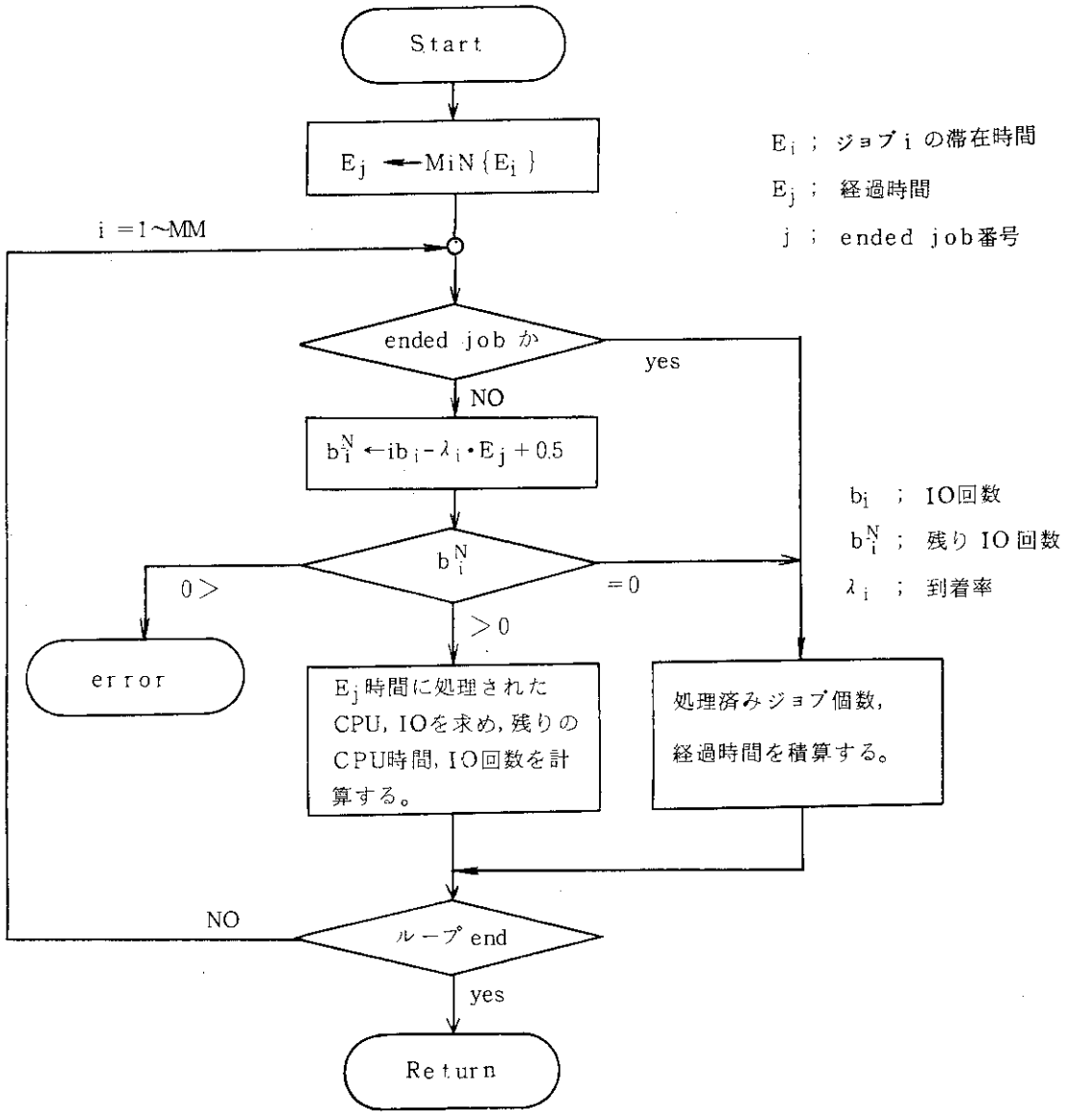


Fig. 4.5(e) Detail flow of scheduler (JOBEND)

Ⓔ　5 ブルーチンMEMORY (NS, MM, ICPU, IFLAG, IRET)
メモリーチェックを行なう。

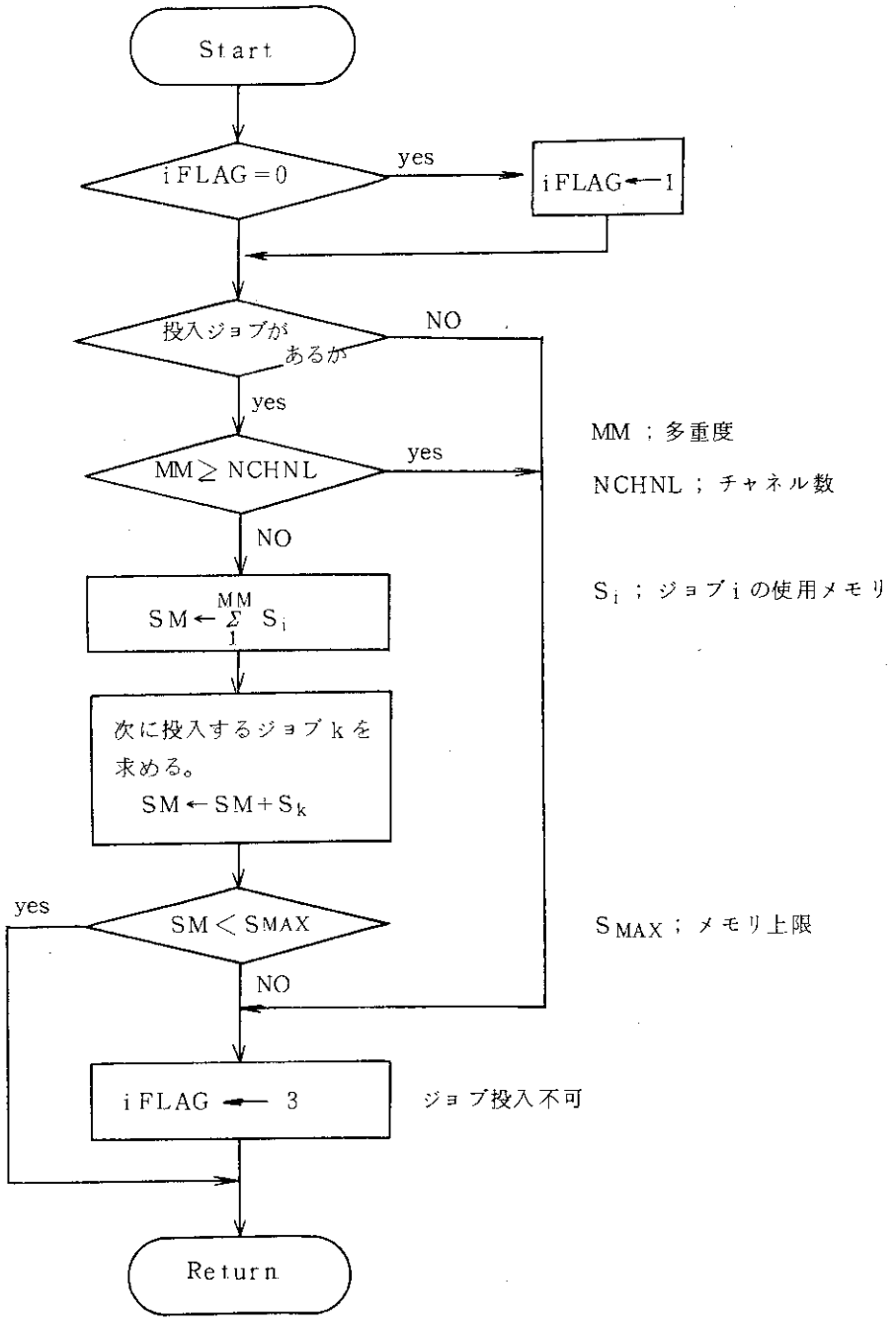


Fig. 4.5(f)　Detail flow of scheduler (MEMORY)

5. シミュレータ

前章で述べたスケジューラーを検証するためにシミュレータを作成した。スケジューラは到着率はポアソン分布，サービス時間は幾何分布にもとづくと仮定して計算している。現実の計算機の動きを模擬するシミュレータを作り，スケジューラが予測の対象とした同じ実行ジョブ系列をこのシミュレータで処理するとどうなるであろうか。

この章ではこのシミュレータの構成法について述べる。結論をいえばシミュレータとスケジューラの計算結果はかなり良い一致を示す。

5.1 シミュレータの概略

シミュレータへの入力：シミュレータへの入力は前章で述べたスケジューラへの入力と同じであるが，これにジョブ到着時間が追加されている。すなわち，シミュレータにおいては，ひとつのジョブJ_iは$J_i=(a_i, b_i, T_i, M_i)$であらわされる。ここでT_iはシミュレータ・システムにジョブJ_iが到着した時間である。いまの場合にはシミュレータにはスケジューラのジョブを開始した時間が与えられる。そしてスケジューラの指定する多重度M_iにしたがって直ちにジョブの実行が開始され待ち時間はない。したがってT_iはジョブJ_iのシミュレータ・システムにおけるジョブ開始時間にほとんど等しい。

シミュレータの振舞い：シミュレータは前章Fig. 4.1のRRシステムを模擬する。このRRシステムがFig. 4.1のシステムと異なる点はおもに次のとおりである。

(i) CPUのタイム・クワンタムQはジョブ実行開始時には常に$Q=\min(a_i/b_i)$として設定される。ここまではスケジューラと同じであるが，CPUあるいはI／O処理の途中でQが再分割されたときは，それら再分割されたうちの最小のタイム・クワンタムが新しいQとして再分割された時点から使用される。

(ii) CPUでQだけ時間が経過する間に，I／OでもQの時間が経過するとしている。またQを再分割する契機はCPU側でもI／O側でも発生する。すべてのジョブの1回のI／O当りの平均CPU占有時間$\triangle t_i$は，ある時点のQの整数倍ではないので，$\triangle t_i = nQ + \triangle t_i'$となる。この$\triangle t_i'$は新しいQとなることがある。1回当りのI／O時間Cはわれわれの場合は定数と仮定しているが，$C = nQ + \Delta C$となることが多い。このΔCも新しいQとなることがある。このようにしてCPUとI／Oの割込み発生のタイミングを検知する。これをQの再分割と呼ぶ。

(iii) ひとつのI／O要求は，それがI／O装置の処理の対象となったときは，処理が終了するまでそのI／O装置にとどまる。すなわち処理はRRではない。

(iv) 実時間の計測：シミュレータでは時間の経過はQによって計り，実時間はジョブ入力時刻によって計る。シミュレータはリアル・タイマLRTR，ロング・インタバル・タイマLITR，ショート・インタバル・タイマSITRを持ち，それらの間の関係はFig. 5.1のようになっている。

スケジューラとシミュレータのデータの授受はFig. 5.2のとおりである。
シミュレータの処理の概略はFig. 5.3のとおりである。

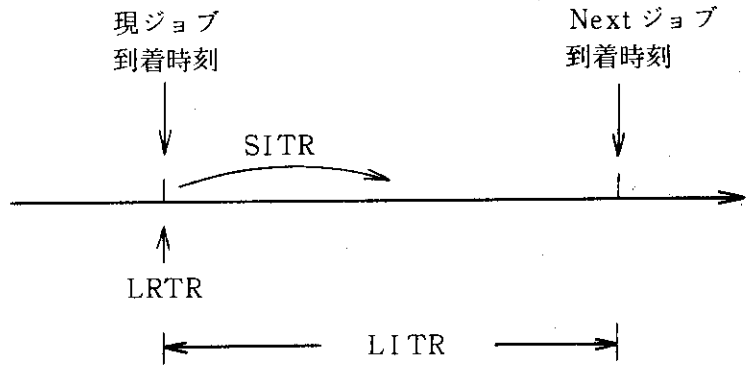


Fig. 5.1　Relations between timers

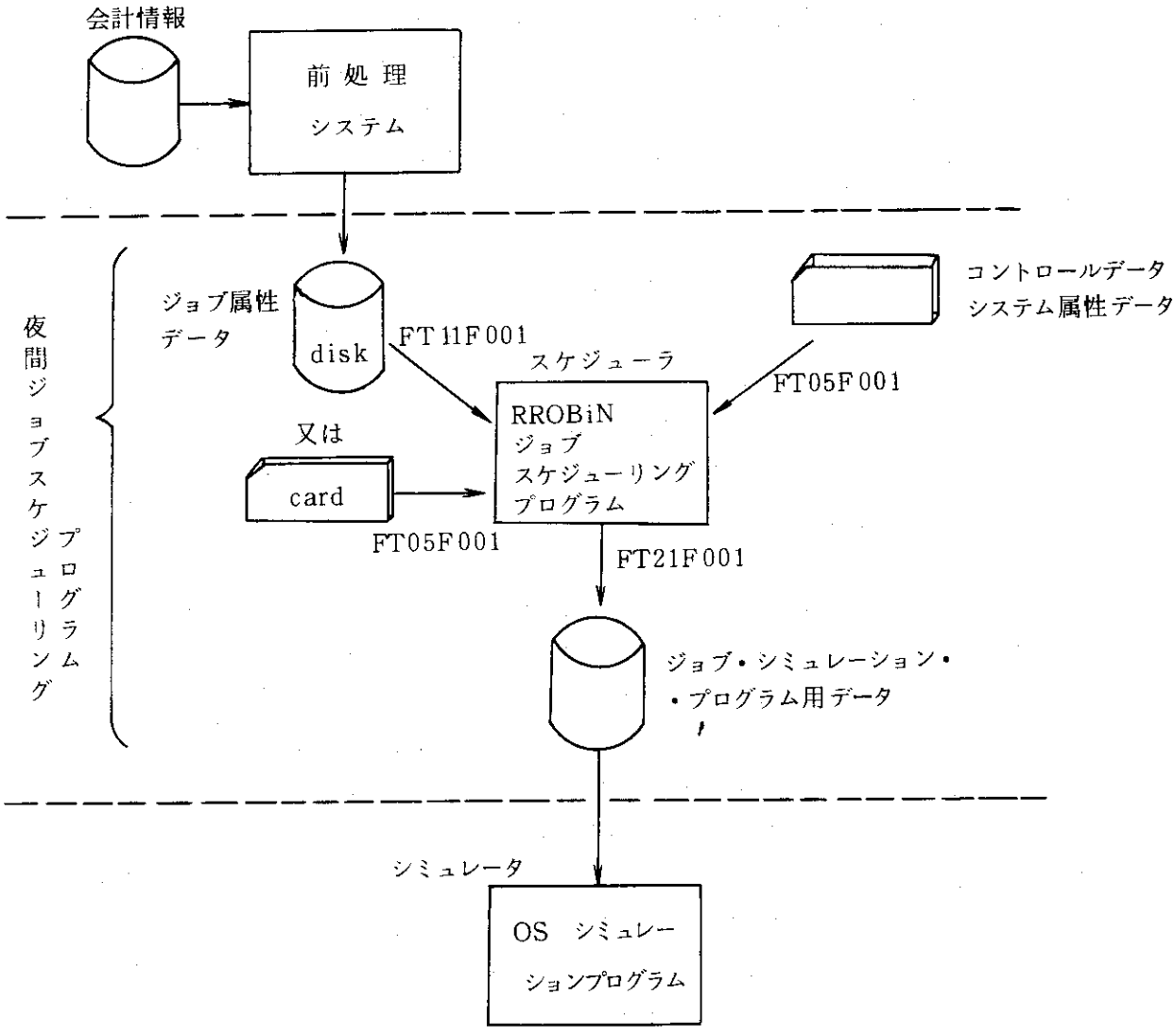


Fig. 5.2　Data flow of scheduler and simulator

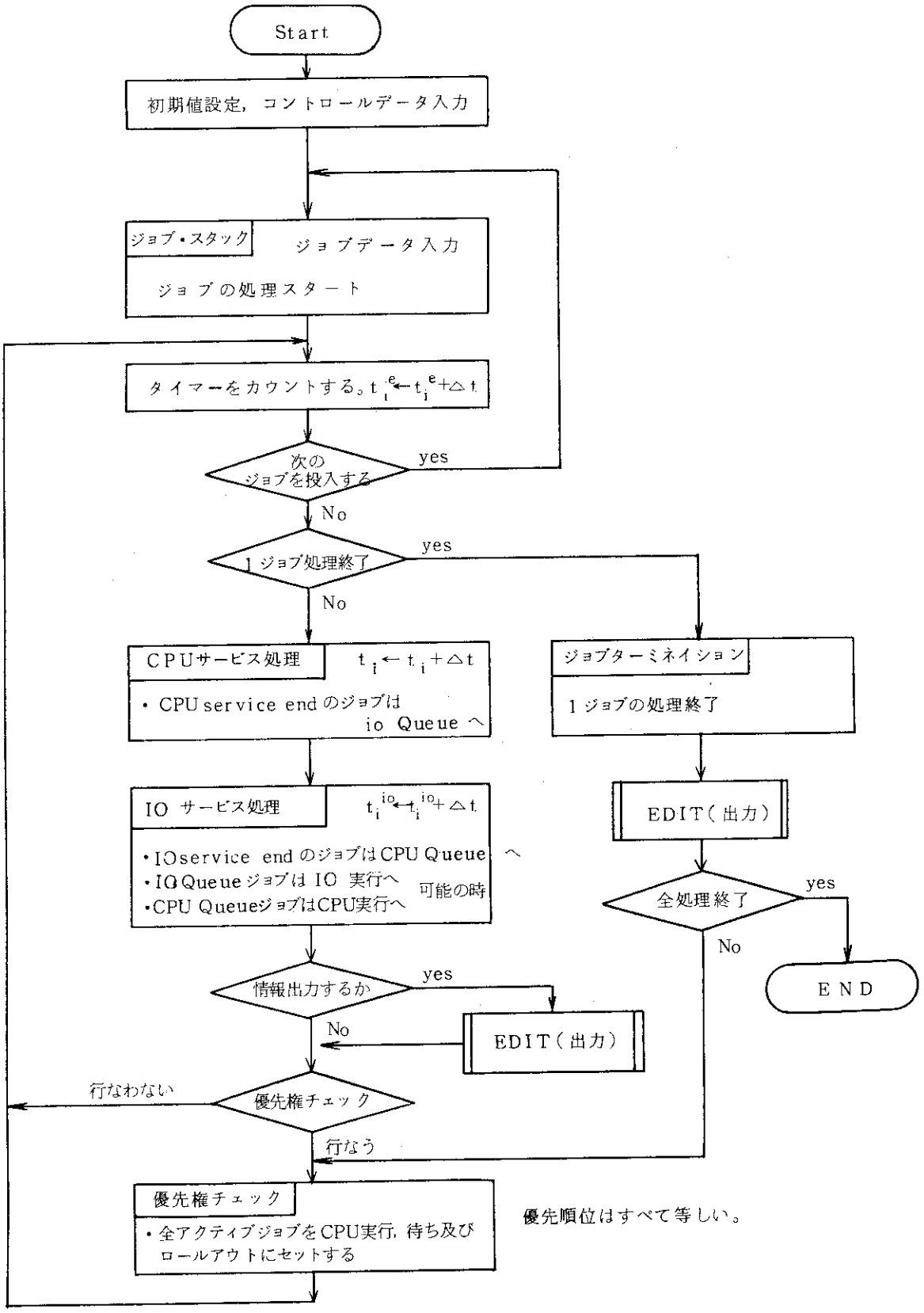


Fig. 5.3 General flow of simulator

5.2 シミュレータの詳細

シミュレータの処理の詳細図をFig. 5.4（ a〜e ）に示す。

Ⓐ ジョブスタック＆スタート

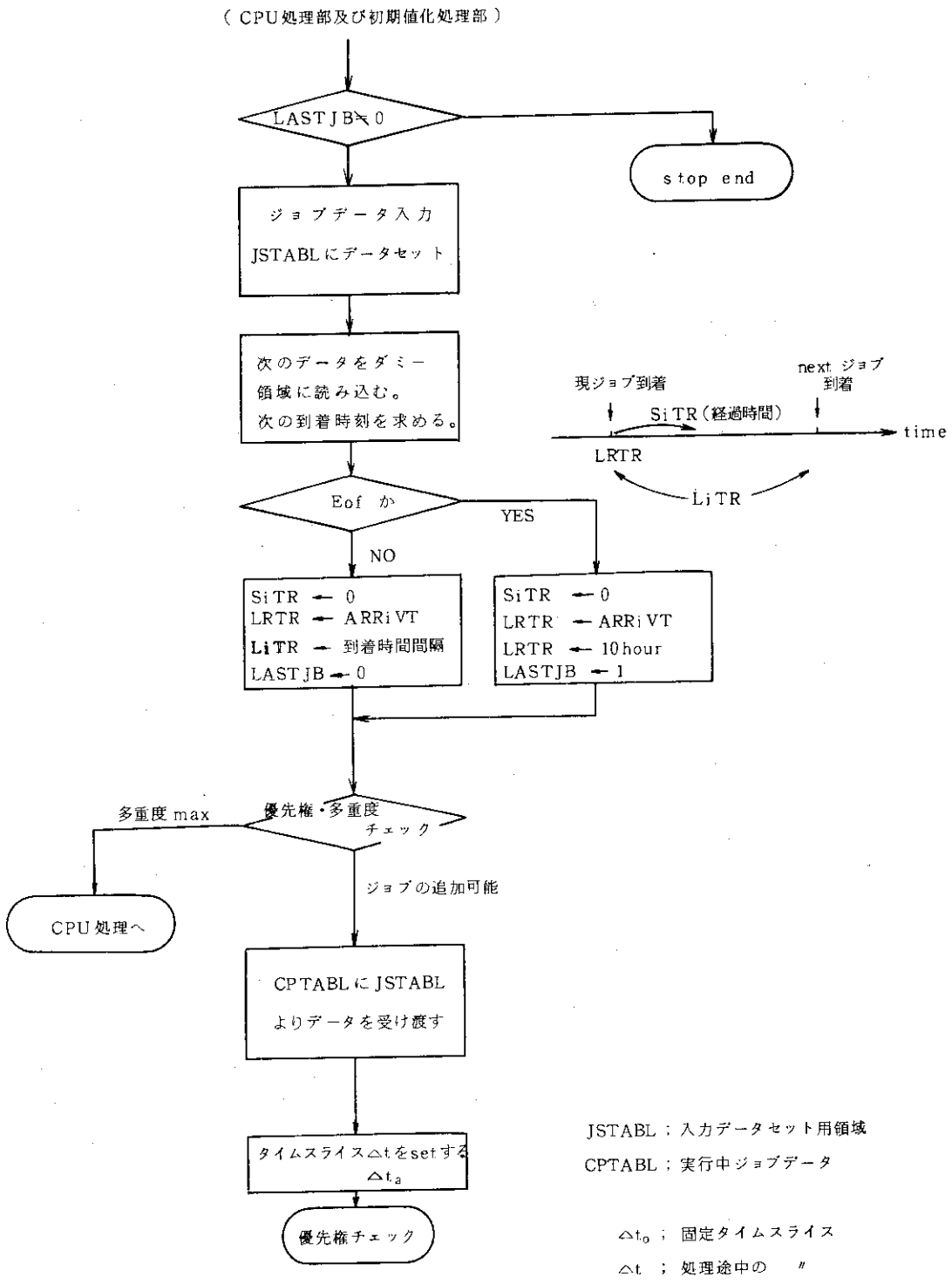


Fig. 5.4 (a) Detail flow of simulator

Ⓑ CPUサービス処理部

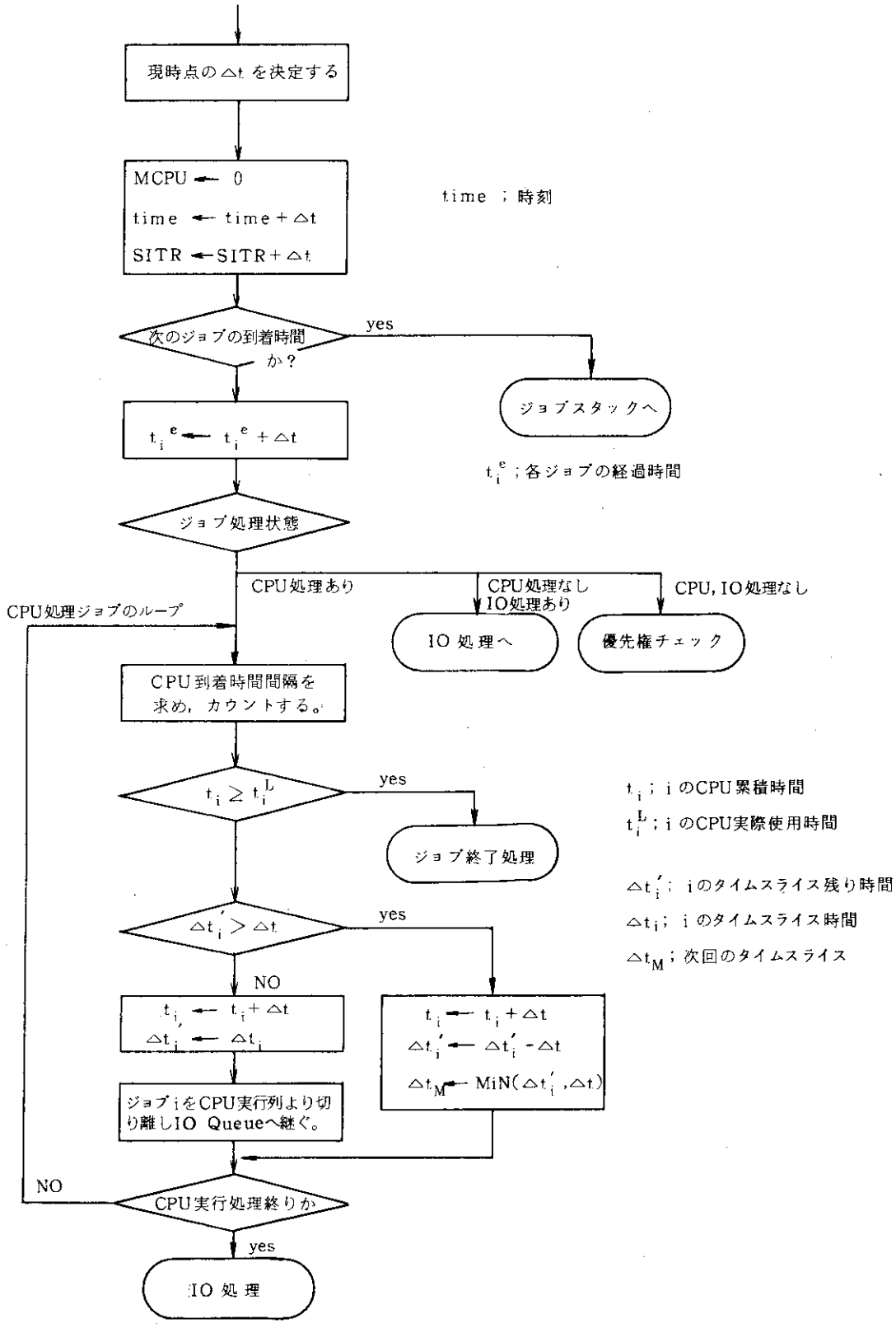


Fig. 5.4 (b)　Detail flow of simulator

© IOサービス処理

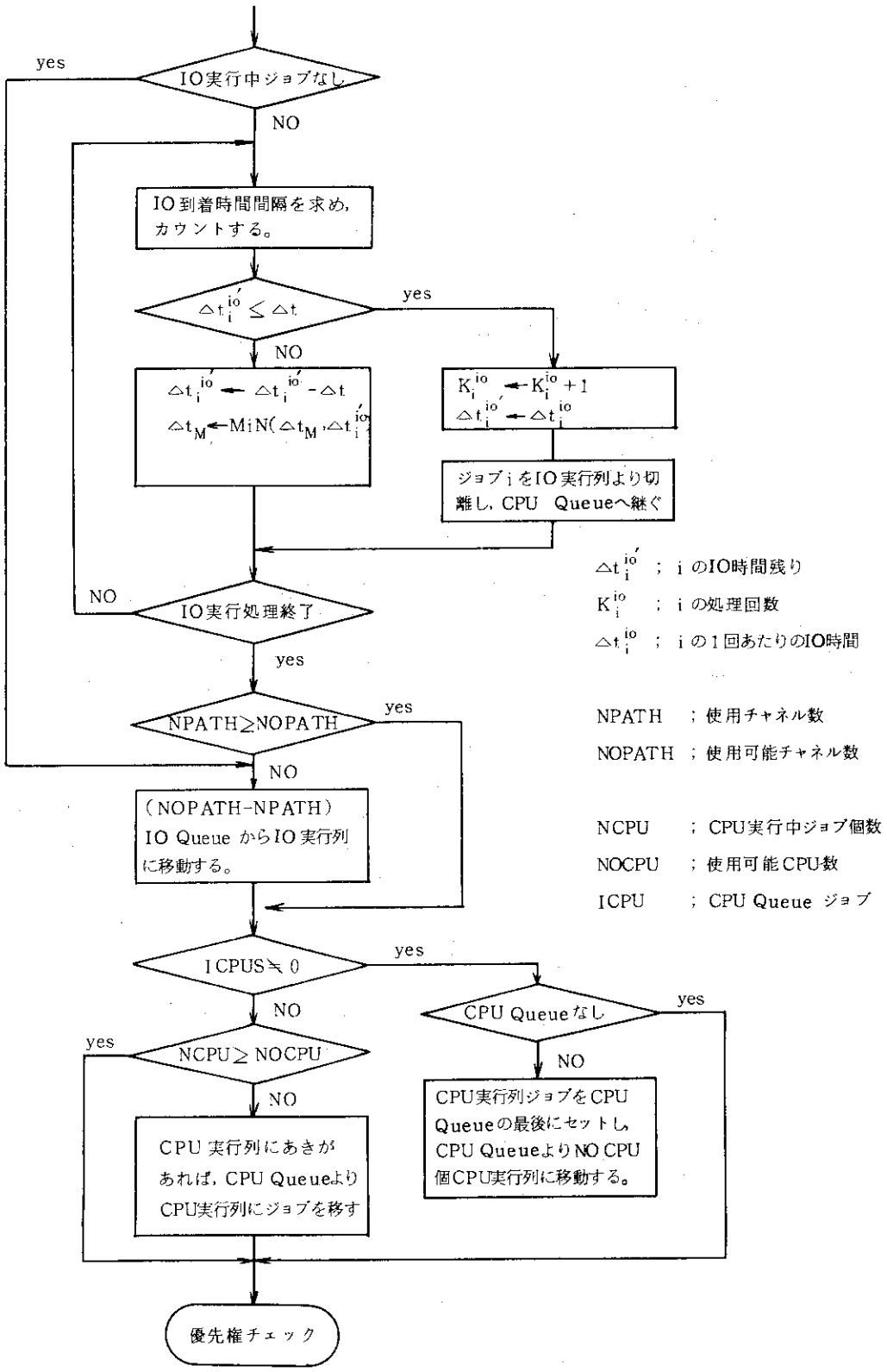


Fig. 5.4 (c) Detail flow of simulator

⑪ ジョブ終了

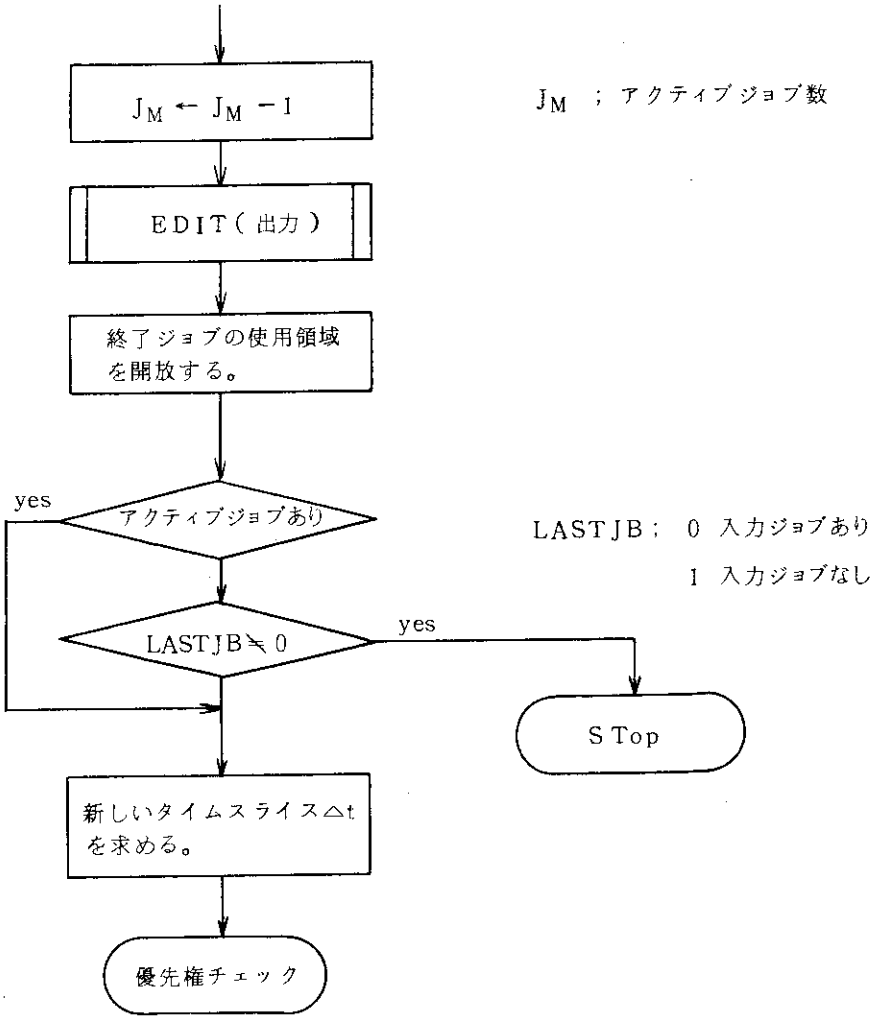


Fig. 5.4 (d) Detail flow of simulator

Ⓔ　優先権チェック

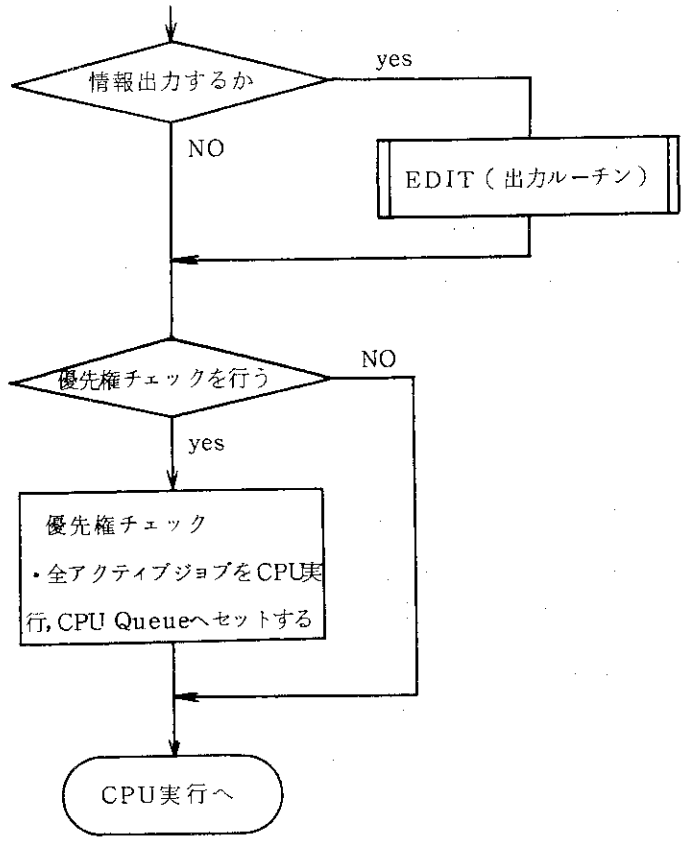


Fig. 5.4 (e)　Detail flow of simulator

シミュレータで使用する変数とその意味は，以下のとおり。

シミュレータで用いるデータはリスト構造になっている。そのデータ用変数をTable 5.1 に掲げる。

Table 5.1　Variables for simulator

配　　列	タイプ	意　　味
CPBASE(10)	i	6種処理状態の先頭番地（NF2(j)を示す）
NF2(1000)	i	リスト処理用変数
CPTABL(390)	R	処理中ジョブ・データ用変数（実数）↖同一番地
CPUTBL(360)	i	〃（整数）↙同一番地
JSTABL(3000)	i	入力用ジョブ・データ用変数（整数）↖同一番地
JSTBFL(3000)	R	〃（実数）↙同一番地
JSTEMP(30)	i	入力用ダミー変数
MHiST(30)	i	CPU到着間隔回数カウント用変数
MHiSTi(30)	i	I／O 〃　　〃

以下ａ)～ｅ)で各変数の内容について説明する。

ａ）処理状態別先頭地　CPBASE(10)

NF2の番地を示すポインタ

CPBASE(1)；アクティブジョブの先頭番地　システム内の全ジョブ

(2)；CPU処理中ジョブの　〃

(3)；CPU待ちジョブの　〃

(4)；I／O処理中のジョブの〃

(5)；I／O待ちジョブの　〃

(6)；ロールアウト中ジョブの〃

ｂ）リスト処理用変数　NF2(1000)

NF2(2i－1),CPTABL(or　CPUTBL)上の番地をジョブ単位に示す。

NF2(2i)；同一処理状態に関してNF2上の次の番地を示す。

例をFig. 5.5 に示す。

（例） CPU待ち状態

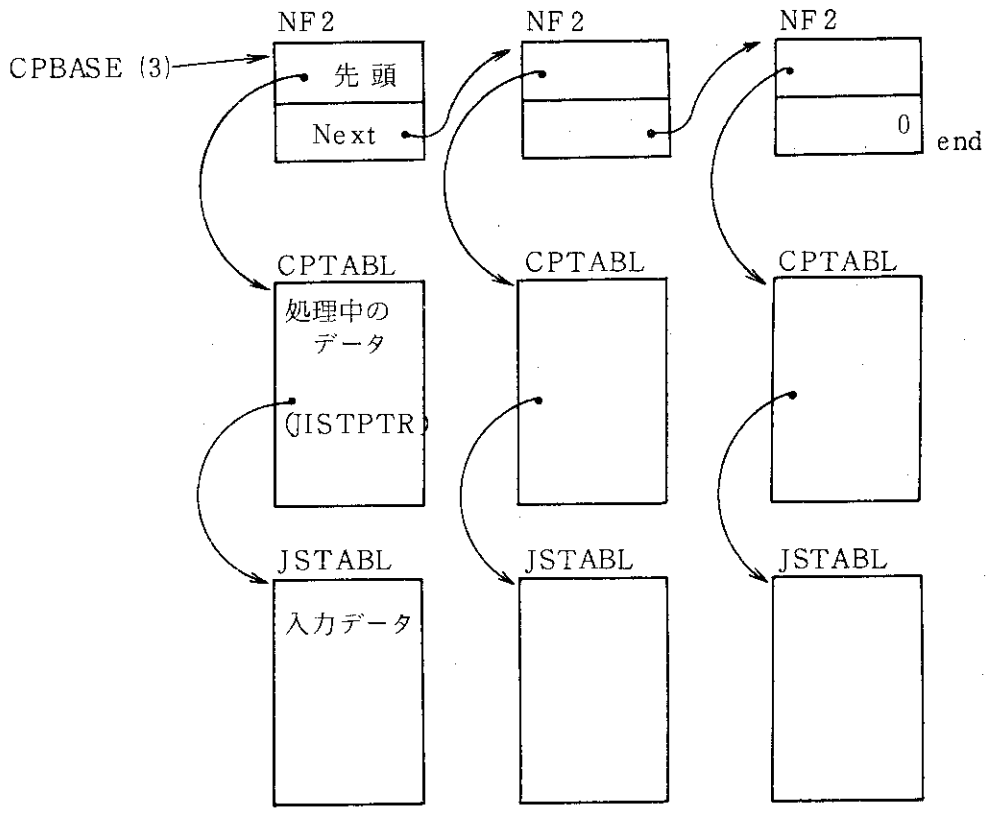


Fig. 5.5 Example of simulator's list processing

ジョブの処理状態の移行に伴ってNF2の連結，切り離しを行い，
CPTABL，JSTABLの検索を可能にしている。

c）処理中ジョブのデータ用変数

- CPTABL (390) （CPUTBL (360)。その内容はTable 5.2のとおり。
- 各ジョブの先頭番地 M＝13(i－1)＋1

Table 5.2 Contents of CPTABL

相対番地	相対番地名	タイプ	意味
0			投入ジョブ番号
1	STATUS	i	ジョブの状態（注１）
2	PRIOTY	R	優先権
3	SVRATE	R	サービス率
4	DTI	R	Δt_i ； ジョブのタイムスライス
5	DTPI	R	Δt_i ； タイムスライス残り時間
6	ACCUMT	R	t_i ； 累積CPU時間
7	CPTLIM	R	t_i^L ； 実際のCPU使用時間
8	ELAPSE	R	t_i^e ； 経過時間
9	IOTIME	R	t_i^{io} ； １回辺りのi／o時間
10	DTIOPI	R	t_i^{io} ； 〃 〃 の残り
11	JSTPTR	i	JSTABLの対応先頭番地
12	NXTPTR	i	next ポインタ
6	CPUT	R	

（注１） <STATUS> = 0 初期状態
1 CPU実行中
2 〃 待ち
3 I／O実行中
4 〃 待ち
5 ロールアウト中

d）ジョブの入力データ用変数

JSTABL (3000)，JSTBFL (3000)。その内容は Table 5.3 のとおり。

各ジョブの先頭番地，M＝30（i－1）＋1

番地 1～17 は入力データをセットする。

Table 5.3 Contents of JSTABL

相対番地	相対番地名	タイプ	意味
0	JSNO	i	投入ジョブの順番号
1	JSARIV	i	ジョブスタート時間 （ms）
2	JSRQCP		未使用
3	JSRQME		未使用
4	JSUSCP JSCPUP	i	CPU時間使用量 （ms）
5	JSUSME		未使用
6	JSIOS	i	I／O 回数使用量
7	JSELAP		未使用
8	JSCHAN	i	チャネル時間使用量（ms）I／O回数×I／O時間
9	JSROLL		未使用
10	JSCARD		未使用
11	JSLINE		未使用
12	SVCLAS	i	パフォーマンスグループ 1
13	JSVRAT		サービス率（未使用）
14	JSPRTY	i	優先権 999
15	JSSCNO		未使用
16			
17	JSCPIO		未使用
18			
19			
20			
21	INSTAK	i	処理状態フラック
22	JSLCPU	i	最新のCPU処理開始時間
23	JSLIO	i	〃 I／O 〃 〃
24	JSIOS 2	i	処理済みI／O回数
⋮			
30			

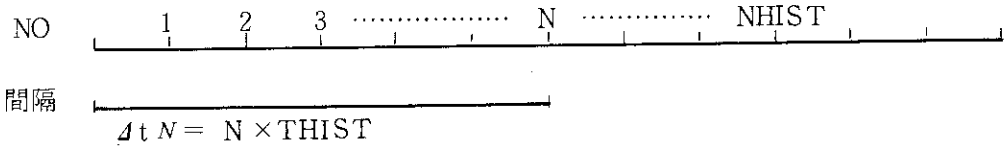
e) 到着時間間隔調査用変数

MHiST (30) ； CPU 用テーブル

MHiSTi (30) ； I／O 用 〃

入力データ { THIST； 時間間隔設定用データ（ms）
NHIST ； 間隔最大個数

{ MHIST (N)
MHISTI (N)



あるジョブが $\Delta t_N > \Delta t > \Delta t_{N-1}$ である様な時間間隔 Δt で CPU 又はI／Oへ到着した時，カウンター MHIST (N) 又はMHISTI (N)が 1 加算される。カウントアップされた最終結果をヒストグラム等に作図し，到着率の調査に用いる。

6. 予測計算結果の検討

6.1 スケジューラへの入力

スケジューラへの入力は原研計算センタの会計情報ファイルから一夜の夜間ジョブを抽出して作成した。

113個のジョブをLPTルールで3システムに分配し，そのうちのひとつのジョブ系列（Table 6.1）をスケジューラへの入力とした。

Table 6.1においてI/O timeとあるのは各ジョブの入出力回数に30ミリ秒の入出力時間を乗じて得られたものである。1回の入出力時間を30ミリ秒と仮定する限りは，このジョブ系列は演算装置寄り（CPUバウンド）である。

Table 6.1 Input jobs

** LPT SCHEDULING **

SYSTEM NO. = 1
NUM OF JOB = 37

LPT	JOB	I/O	CPU TIME	I/O TIME	TOTAL TIME	QUANTUM	MEMORY
1	99	1853	7.198E+03	5.559E+01	7.254E+03	3.885E+00	1.544E+03
2	8	50000	4.599E+02	1.500E+03	1.960E+03	9.199E-03	1.288E+03
3	73	36163	8.470E+02	1.085E+03	1.932E+03	2.342E-02	8.720E+02
4	104	2921	1.755E+03	8.763E+01	1.843E+03	6.009E-01	3.468E+03
5	97	912	1.640E+03	2.736E+01	1.668E+03	1.799E+00	1.048E+03
6	42	254	1.113E+03	7.620E+00	1.120E+03	4.381E+00	1.184E+03
7	1	4869	8.548E+02	1.461E+02	1.001E+03	1.756E-01	1.416E+03
8	41	20191	4.765E+01	6.057E+02	6.534E+02	2.360E-03	5.600E+02
9	103	1063	6.074E+02	3.189E+01	6.393E+02	5.714E-01	1.108E+03
10	16	13178	2.130E+02	3.953E+02	6.083E+02	1.616E-02	5.400E+02
11	29	56	4.635E+02	1.680E+00	4.652E+02	8.277E+00	1.144E+03
12	57	7935	1.632E+02	2.381E+02	4.012E+02	2.057E-02	5.400E+02
13	77	5682	1.587E+02	1.705E+02	3.292E+02	2.794E-02	1.732E+03
14	50	5733	1.268E+02	1.720E+02	2.988E+02	2.212E-02	5.400E+02
15	7	171	2.715E+02	5.130E+00	2.767E+02	1.588E+00	1.600E+03
16	96	7924	2.137E+01	2.377E+02	2.591E+02	2.697E-03	5.760E+02
17	19	2760	1.201E+02	8.280E+01	2.029E+02	4.350E-02	1.032E+03
18	112	5032	3.924E+01	1.510E+02	1.902E+02	7.798E-03	1.024E+03
19	48	3561	4.400E+01	1.068E+02	1.508E+02	1.236E-02	8.320E+02
20	66	2387	7.028E+01	7.161E+01	1.419E+02	2.944E-02	5.400E+02
21	63	1139	6.896E+01	3.417E+01	1.031E+02	6.054E-02	1.540E+03
22	110	2818	8.340E+00	8.454E+01	9.288E+01	2.960E-03	5.080E+02
23	109	2576	7.790E+00	7.728E+01	8.507E+01	3.024E-03	5.080E+02
24	24	1917	2.413E+01	5.751E+01	8.164E+01	1.259E-02	8.320E+02
25	11	1442	2.790E+01	4.326E+01	7.116E+01	1.935E-02	5.760E+02
26	27	919	8.600E+00	2.757E+01	3.617E+01	9.358E-03	1.372E+03
27	13	919	8.070E+00	2.757E+01	3.564E+01	8.781E-03	1.372E+03
28	21	629	4.960E+00	1.887E+01	2.383E+01	7.886E-03	8.320E+02
29	93	471	6.090E+00	1.413E+01	2.022E+01	1.293E-02	9.200E+02
30	52	191	9.860E+00	5.730E+00	1.559E+01	5.162E-02	5.760E+02
31	34	357	4.230E+00	1.071E+01	1.494E+01	1.185E-02	1.800E+02
32	91	223	6.400E-01	6.690E+00	7.330E+00	2.870E-03	8.320E+02
33	62	93	7.000E-01	2.790E+00	3.490E+00	7.527E-03	1.200E+02
34	61	93	6.900E-01	2.790E+00	3.480E+00	7.419E-03	1.200E+02
35	17	37	2.050E+00	1.110E+00	3.160E+00	5.541E-02	7.680E+02
36	100	4	1.000E-01	1.200E-01	2.200E-01	2.500E-02	1.920E+02
37	72	1	8.000E-02	3.000E-02	1.100E-01	8.000E-02	3.200E+02
TOTAL		186474	1.640E+04	5.594E+03	2.199E+04		

6.2 計算にあたっての制限条件

多重度を抑制する要素はCPU利用率，主記憶容量，入出力チャンネル数である。抑制の方法は次のようにした。

(i) CPU利用率：多重度の計算においてCPU利用率が1を起える場合は，最初に1を超えたときの多重度を求めるべき多重度とした。CPUは1台である。

(ii)主記憶容量：10 MB（メガバイト）までとし，多重度はこの容量を超えない範囲とした。

(iii)チャネル台数：多重度はこの台数を超えない範囲とした。この章で述べる計算では7台としている。これらチャネルは，この台数までは任意に多重化でき，その1回の入出力時間は30ミリ秒としている。

したがってTable 6.1のジョブ処理の経過時間はおよそ16400＋5600／7＝17200秒になる筈である。

6.3 計算結果の検討

(1) Table 6.1のジョブ系列をスケージューラでスケジュールし，その経過時間を求めた。またスケジュールした結果を出力しそれをシミュレータへの入力とした。その結果をTable 6.2に示す。

(2) 最初に終了する10個のジョブについてジョブの処理終了となった時刻毎に，それまでの最適多重度，次善の多重度，それぞれに対応するCPU利用率を図で表わしたものがFig. 6.1である。

Table 6.2 Elapsed time

推　定　値	スケジューラ	シミュレータ
17200 秒	17000 (16220)	17000

(3) 上記のジョブについて同時刻にスケジューラ，シミュレータで動いている同一ジョブのCPU系への到着率 λ_i を比較するとTable 6.3のようになる。

(4) Table 6.2の結果をみるとスケジューラは良い結果を与えているように見えるが，その予測計算の過程を追ってみるとさまざまな問題をはらんでいることがわかる。スケジューラは統計的仮定といくつかの近似手法から成り立っている。このうち予測精度に影響をもつと思われるおもな項目を挙げる。

(i)式（4.18）で表わされた1クワンタムのジョブの待ち時間 $W_1 = E(Q_r) + n_q Q + Q$ において，CPUサービス時間の期待値 $E(Q_r)$ は $E(Q_r) = Q/2$ としているが，われわれの場合はこうならないことが頻繁に起る。たとえば1クワンタ＝3秒のジョブが2個実行されているときは，1回の入出力時間が30ミリ秒と小さいために，CPU系で1個のジョブが処理されはじめたときに次のジョブが，またCPU系に到着する。このときは $E(Q_r) = Q$，$n_q = 0$，であるから $W_1 \cong Q + Q = 2Q$ となり，$W_1 \cong Q/2 + Q = 1.5Q$ ではない。それにもかかわらず $W_1 = 1.5Q$ としているために λ の値が大きくなり，$\rho > 1$ となる。Fig. 6.1の多重度＝2のときにこれが発生している。

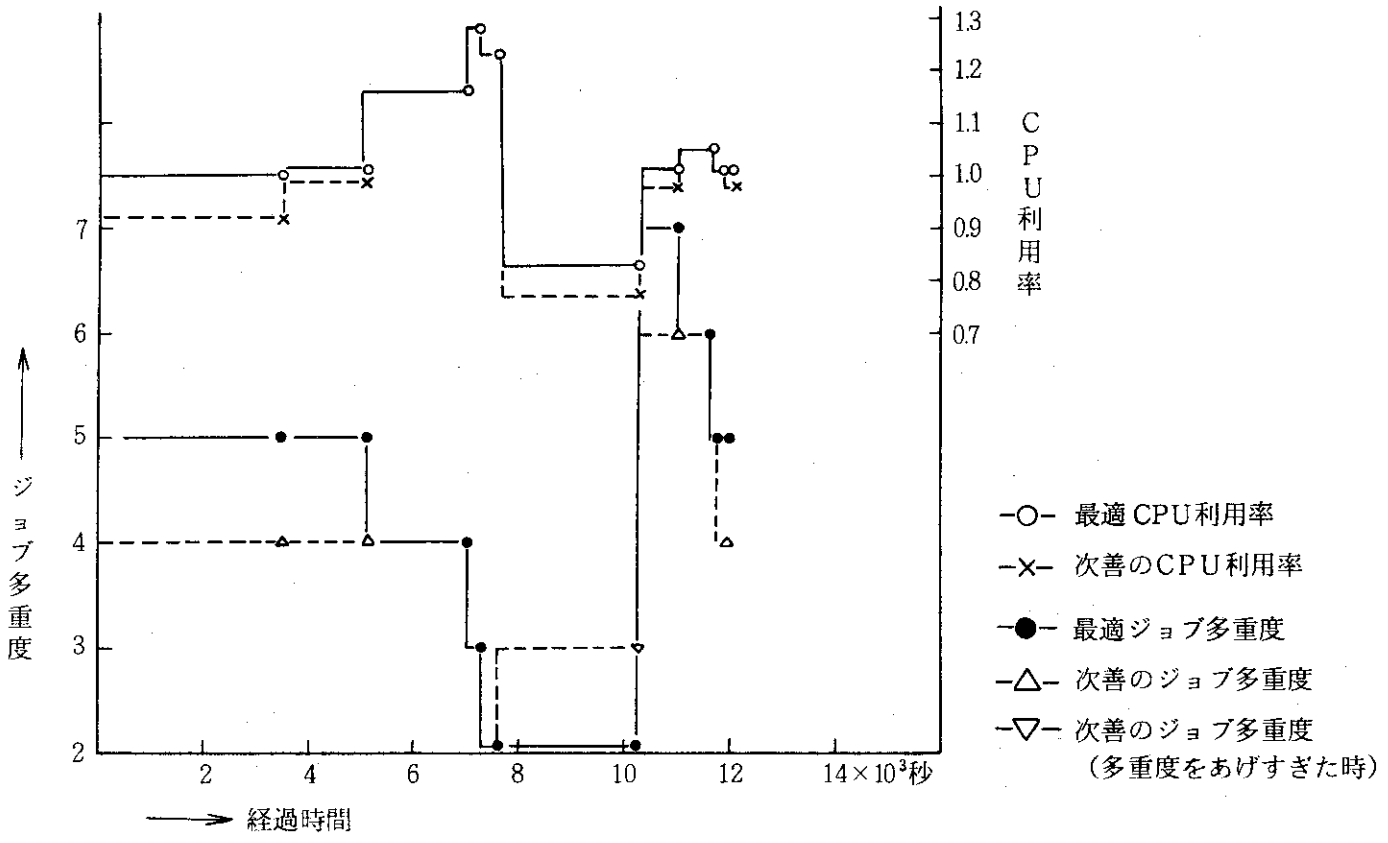


Fig. 6.1　Optimal degree of multiprogramming and CPU utilization

Table 6.3　Comparison of input rate λ_i

時刻	システム内のジョブ番号	λ_i（スケジューラ）	λ_i（シミュレータ）
1	1, 2, 3, 4, 5	0.069, 15., 6.8, 0.38, 0.11	0.057, 15, 7.6, 0.37, 0.12
2	1, 3, 4, 5, 6	0.067, 7.8, 0.38, 0.12, 0.041	0.053, 8.1, 0.34, 0.11, 0.046
3	1, 4, 5, 6	0.094, 0.51, 0.16, 0.055	0.065, 0.40, 0.14, 0.057
4	1, 7	0.13, 1.8	0.074, 1.7
5	1, 8, 9, 10, 11, 12, 13	0.056, 24., 0.36, 9.0, 0.022, 4.9, 3.5	0.046, 23., 0.31, 7.9, 0.021, 6.6, 5.2
6	1, 9, 10, 11, 12, 13	0.065, 0.41, 10., 0.024, 6.2, 3.1	0.045, 0.30, 8.8, 0.022, 5.5, 5.5
7	1, 9, 14, 15, 16	0.070, 0.43, 0.026, 6.6, 0.10	0.053, 0.35, 0.024, 8.4, 0.13

(ii)夜間ジョブはCPU寄りのものが多いために，CPU系における負荷は待ち行列理論で計算されているものより大きい場合がある。したがって処理系の平均待ち行列個数$\bar{n}$は式（ 4.17 ）の理論値よりも大きいことがしばしばある。

(iii)各ジョブの要求クワンタ数N_iは$N_i = Q_i / Q_{min} + 0.5$として求める。したがって$Q_2 = 1.3\,Q_1$と$Q_1$の2個のジョブの要求クワンタ数はそれぞれ1として計算される。2個のジョブの演算装置時間が長いときはその誤差は大きくなる。Fig. 6.1の多重度＝2のときにこの事態が発生している。

(iv) Kクワンタを要求する擬ジョブの待ち時間W_kは$W_k = \Sigma \gamma_i W_i$として計算される。kが大きいときは式（ 4.11 ）において（ k－1 ）Q／（ 1－ρ ）の値が相対的に大きくなるために，上の近似式は精度が良くなる。またQ／（ 1－σ ）の値を固定してQの値を0に近づけても同じことが言える。

(v)サービス分布密度g_iを最大要求クワンタ数までで打切っている。これもいくらか誤差を生む原因になっている。

上記のうち(iii)～(v)はクワンタ値Qを小さくすることで解決できよう。しかし(i)～(ii)は簡単に処理できない。

たとえば(i)の例をあげる。Table 6.1のジョブをスケジュールすると，CPU時間7198秒を要するジョブが最後まで残る。これはシミュレータでも同じである。しかしスケジューラでは，このとき，次の不適切な計算をおこなう結果となる。$W_i = E(Q_r) + n_q + Q$においてE（Q_r）＝Q／2としているが，これはE（Q_r）＝0が正しい。この部分は該当するジョブが1多重で処理されていて，その残りCPU時間は2400秒，残り入出力回数は622回，これまでの経過時間は13800秒と予測されている。したがってE（Q_r）＝0のときは16220秒で全ジョブが終了する筈である。

さらに(i)と(ii)の混合した例をあげる。Fig. 6.1の多重度4，3，2のとき$\rho > 1$となっているのは次のような理由による。

(a)実際の残りサービス時間はQ／2ではなくQに近い。

(b)実際の平均待ち行列個数$\bar{n}$は理論値よりも大きい。

(c) Qの値が3.9秒と大きい。

(d)上記(a)，(b)を理論値で計算しているために待ち時間W_kの値が小さくなり，したがってλ_kの値は大きくなって$\rho > 1$となる。

6.4 スケジューラとシミュレータの長短

いずれの方法においてもタスク・スケッチや入出力操作時のオペレイティング・システムのオーバヘッドを計算できる点は長所である。さらに言えば次のようになる。

6.4.1　スケジューラの長短

長所

(1)短時間で終るスケジューリング

短時間でジョブのスケジューリングが可能である。本報告書に挙げた夜間ジョブのスケジューリングは1システムについて，FACOM M—200 計算機のCPU時間で3.6秒であった。このことから現状の夜間ジョブを3システムに振り分けてスケジュールし，その結果生じる多少の処理時間の凹凸を調整したとしても15秒あればよい。

(2)最適CPU利用率の推定

ジョブが入力されるたびに，そのジョブの特性まで含めたジョブ処理環境における最適なCPU利用率を推定することができる。

(3)計算機性能の内外挿

スケジュールに入力するジョブの特性，すなわちCPU時間，I／O回数などを変更することによって各種ジョブ・パターンに対する各種計算機の性能を予測することができる。

短所

(1)実際処理時間との誤差

入力されるデータは，到着がポアソン分布，サービス時間が幾何分布，擬ジョブ1回の処理時間は正確にタイム・クワンタム量Qと等しいと仮定されている。ジョブの組み合せによっては，この仮定は必ずしも正しくないので誤差を生じる。また近似計算式によっても誤差を生じる。

6.4.2　シミュレータの長短

長所

(1)現実模擬の容易さ

シミュレータは実際のオペレイティング・システム（ＯＳ）のジョブ処理方法を模擬することができるので，その予測結果はスケジューラのそれに比較すると，より実際の処理時間に近いといえる。また現実の複雑なOSの動きを近似することも容易である。シミュレータはCPU，チャネル台数などを任意に変化させることができる。

短所

(1)計算時間の長さ

シミュレータは，スケジューラと同一のジョブ・パターンの入力について46秒の模擬時間を要する。これはスケジューラの約13倍である。また，スケジューラの計算時間は処理対象としているジョブの特性にさほど環境されないが，シミュレータの計算時間は大きな影響を受ける。

(2)最適多重度決定の難しさ

シミュレータで最適多重度を決定する場合は，いくつかの多重度で並行して計算しなければならないので時間がかかり難しい。

6.5 残された問題

残された問題を列挙しておく。

(1)ジョブ特性をより精確に反映する平均待ち行列個数$\bar{n}$，およびサービス残り時間$E(Q_r)$を求めること。

(2)IBMのSystem Resource Manager(SRM)，あるいはこれに類似のスケジューリング機能を模擬できるスケジューラを作ること。ここでわれわれが提案したスケジューラを修正，拡張すればこれは可能であるように見える。

(3)大容量記憶装置，磁気ディスク装置などの階層化された2次記憶装置の入出力動作を精密に模擬するスケジューラへと拡張すること，

などである。

謝　　辞

本報告書で述べたシミュレータは，もともとはFACOM 230－75計算機のジョブ・スケジューリング機能であるHOPEの振舞いを模擬するために作成したものである。HOPEはIBMのSRMに類似の機能を持っていた。原研計算センタの一カ月分のぼう大な会計情報を整理し，すべてのジョブを到着時刻順に分類してシミュレータへの入力を作成していただいた富士通(株)システムズ・エンジニア加藤菊和氏，またこの作業の手配をしてくださった富士通原子力システム部第一システム課長(当時)清水和五氏，FACOM 230－75のオペレイティング・システムについて御教示いただいた富士通OS部の田淵治樹氏に感謝します。また常日頃から御協力いただいている原研計算センタの皆様，および御指導，助言いただいている茨城大学工学部磯田和男先生に感謝します。

6.5 残された問題

残された問題を列挙しておく。

(1)ジョブ特性をより精確に反映する平均待ち行列個数$\bar{n}$，およびサービス残り時間$E(Q_r)$を求めること。

(2)IBMのSystem Resource Manager(SRM)，あるいはこれに類似のスケジューリング機能を模擬できるスケジューラを作ること。ここでわれわれが提案したスケジューラを修正，拡張すればこれは可能であるように見える。

(3)大容量記憶装置，磁気ディスク装置などの階層化された2次記憶装置の入出力動作を精密に模擬するスケジューラへと拡張すること，

などである。

謝　　辞

本報告書で述べたシミュレータは，もともとはFACOM 230－75計算機のジョブ・スケジューリング機能であるHOPEの振舞いを模擬するために作成したものである。HOPEはIBMのSRMに類似の機能を持っていた。原研計算センタの一カ月分のぼう大な会計情報を整理し，すべてのジョブを到着時刻順に分類してシミュレータへの入力を作成していただいた富士通(株)システムズ・エンジニア加藤菊和氏，またこの作業の手配をしてくださった富士通原子力システム部第一システム課長(当時)清水和五氏，FACOM 230－75のオペレイティング・システムについて御教示いただいた富士通OS部の田淵治樹氏に感謝します。また常日頃から御協力いただいている原研計算センタの皆様，および御指導，助言いただいている茨城大学工学部磯田和男先生に感謝します。

参 考 文 献

1) A. Brandwajn : A Queuing Model of Multiprogrammed Computer Systems Under Full Load Conditions, J. ACM, Vol.24, No.2, pp.222-240 (1977)

2) T.G. Price : A Note on the Effect of the Central Processor Service Time Distribution on Processor Utilization in Multiprogrammed Computer Systems, J. ACM, Vol.23, No.2, pp.342-346 (1976)

3) I. Mitrani : Nonpriority Multiprogramming Systems Under Heavy Demand Conditions - Customer's Viewpoint, J. ACM, Vol.19, No.3, pp.445-452 (1972)

4) V.L. Wallace : Degree of Multiprogramming in Page-on-Demand systems, C. ACM, Vol.12, No.6, pp.305-308 (1969)

5) D. Towsley, et al. : Models for Parallel Processing Within Programs : Application to CPU: I/O and I/O: I/O Overlap, C. ACM, Vol.21, No.10, pp.821-830 (1978)

6) M. Ishiguro : Running Time Delays in Processor-Sharing Systems, JIP, Vol.3, No.1, pp.38-44 (1980)

7) R.E. Buten & V.Y. Shen : A Scheduling Model for Computer Systems with Two Classes of Processors, Proc. 1973 Sagamore Computer Conf. on Parallel Processing, pp.130-138 (1973)

8) R.W. Conway 他著，関根智明監訳：スケジューリングの理論，PP. 118－131，日刊工業新聞社，昭和46年10月

9) M.J. Gonzalez, Jr. : Deterministic Processor Scheduling, ACM Computing Surveys, Vol.9, No.3, pp.173-204 (1977)

10) C.L. Liu & J.W. Layland : Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment, J. ACM, Vol.20, No.1, pp.46-61 (1973)

11) R.L. Graham : Bounds on Multiprocessing Timing Anomalies, SIAM J. Appl. Math., Vol.17, No.2, pp.416-429 (1969)

12) R.L. Graham : Bound for Cert in Multiprocessing Anomalies, Bell System Tech. J., Vol.45, pp.1563-1581 (1966)

13) W. Chiu, et al. : Performance Analysis of a Multiprogrammed Computer System, IBM Jour. of Res. & Develop., Vol.19, No.3, pp.263-271 (1975)

14) 浅井，藤井他：FACOM 230－75計算機の性能評価(1)，JAERI－M 8714，日本原子力研究所，1980年3月

15) L. Kleinrock : Analysis of a Time-Shared Processor, Nav. Res. Log. Quart., Vol.11, No.10, pp.59-73 (1964)

16) E.G. Coffman, Jr. & P.J. Denning : Operating Systems Theory, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1973

17) 西田俊夫：待ち行列の理論と応用，朝倉書房，昭和46年11月

18) A.O. Allen : Probability, Statislics, and Queueing Theory, Academic Press, New York, 1978

JAERI-M レポート 9501 正誤表

ページ	行	正	誤
7	上5	$(\quad)_{ikl}$	$(\quad)_{kl}$
	上8	$(\quad)_{iuk}$	$(\quad)_{uk}$
	上12	$(\quad)_{iukl}$	$(\quad)_{ukl}$
19	上3	$jobs$	$Jobs$
23	下1	(a_j, b_j*c)	(a_j, b_i*c)
26	上12	K	K
28	上1	$n \cdot M$	n_M
	上4	$\bar{n}_g$	$\bar{n}q_g$
30	table 4.2 最下段	5.62～22.9	0.62～22.9
	上2	で述べたごとくに定め	で述べたごとく
32	上4	クワンタム	ワンタム
34	下4	$MiN\{\frac{a_i}{b_i} \mid i=1\sim MM\}$	$MiN\{\frac{a_i}{b_i} 1 i=1\sim MM\}$
37	上6	b_i	ib_i
22	上6	Probabilistic	Protalistic