EDPS 化日程計算の一方法*

粟　野　敏　雄**

Abstract

It is very important to estimate when system design of the objective is completed, owing to use of an electric computer, not only for current computer business completion, but also for newly establishing a computer system or making additions to a computer system. In this paper, a method of estimation and calculation of a term of system design is described in 3 parts.

1. By PERT network system design, it is easy to calculate theoretically the shortest number of days for system design.
2. Next estimation is the number of programming days. As usual, the estimation is (estimated program steps)/(ability of programmer per one day). In this paper, they are calculated by introducing the "risk" concept.
3. As element of activities in network is very complicated and its effect exercises a great influence all the network, they cause difficulties in calculating the number of days. For instance, when they check out a newly designed system, there is always feedback in the process. The feedback times have to be minimized, also, the number of amendements and re-trials must be minimized.

1. まえがき

電子計算機の利用において，対象業務のシステム設計がいつごろ完成するかを予測することは，稼動計画をたてる上に非常に重要なことである．本論文においては，この日程計算を性格上から，1. 情報システムの EDPS 化，2. プログラム作成，3. 修正と再試行とに区分し，それぞれ，1. PERT，2. 基本的因子危険率の概念，3. OR 手法を導入して，従来より，より信頼性のあるものにしようとするものである．

一般に情報システムの EDPS 化をはかる場合，要素の決定ができ，その手順が定型化されれば，システム設計の PERT ネットワークを作成して，日程計算を理論的に導き出しうるのであるが，そのネットワーク中のアクティビティに含まれる要素が非常に複雑になり，全体に及ぼす影響が大であるという特徴がある．したがって，1つの手段だけでは正確な日程算出を与えることは不可能であるが，大体の算出は可能である．

次に，プログラミングの日程算出は，従来は，はじめに JOB 全体としてのプログラム・ステップ数の見当をつけ(過去の経験による)，次に1人で日に何ステップ組めるかということから全体の期間を推定し，これを各プログラム担当要員に配分して，プログラムごとに作業計画をたてるのである．この方法は一見容易なようであるが，実際にははなはだ不確実性が多いものである．第1に推定されたステップ数のバラツキは個々のプログラマ，個々の与えられたプログラムの性質により非常に大きいので，ステップ数そのものが正確につかめない．第2にメンテナンスしやすい整理されたプログラムを作成するためには，時間をかけてステップ数の減少をはかる場合はしばしばあることで，ステップ数の増減，即，日数の増減とはならない．ステップ数の見積り上の不確実性から，1人当りの稼動量の設定が不確実となり，日程計算に大きな誤りを生ずることとなる．本論文においては，従来のステップ数という概念を用いず，基本的因子危険率という概念をもとにして日程計算をした表により，直接プログラミングの日程算出を行なってこの問題の解決をはかっている．ここにいう危険率とは，ある JOB のプログラム完了予定日が決められたものと仮定したとき，実行の結果，その予定日をこえるかも知れない確率のことである．危険率には，総合危険率と個別危険率とがあり，総合危険率は全プログラム日程について予定日をこえるかも知れない確率であり，総合危険率を全プ

* Calculation and estimation of EDPS system design time in days, by Toshio Awano (Nippon Telegraph and Telephone Public Corporation)

** 日本電信電話公社

ログラムを構成する各プログラム，およびディバッグに一律に分配したとき，分配された危険率を個別危険率という．総合危険率から個別危険率を求めるには，表4を使用すればよい．

最後に，**図1**は電電公社における EDPS 化の処理体系である．現実に情報システムを EDPS 化する場合，立案されたシステム設計書が一度でパスするということはまず考えられない．特に破線で示したごとく，現場で試行する段階において，試行結果による措置をとったりするときなど，常にフィードバックして修正するということが生ずるが，この段階においては，新旧両システムが併行実施されているので費用もかかり，実際にはかなり長い期間を要している．したがって，この段階では，フィードバックされてきた修正ユニットの修正を行ない，再び試行をはじめ，それが終了するまでの全時間については検討を行ない，その最適日程を求める必要がある．すなわち，修正ユニット個々についてみれば，修正時間・試行時間は長いものもあり，短いものもあり，区々である．これらの個々のユニットの順序づけを行なって，すべてのユニットを処理するまでの経過時間を最少にするような計画をたてる必要がある．

以上区分された3部分について日程計算を説明し，その根拠と実施例について述べてある．

2. 情報システム EDPS 化の日程計算

電子計算機利用のためのシステム設計の処理過程は図1のごとくであるが，それを詳細に解明した1つのシステム設計法[1]がある．**図2**はこのシステム設計法を構成図により示したものである．いま，**図3**のごとくこの定型化されたシステム設計法の PERT ネットワークを作成し，このネットワークに作業要素ごとの推定時間を書き入れるならば，EDPS 化するに要する日程はそのクリティカル・パスを求めることにより，容易に求めることができる[2]．図中の数字は**表1**の例題について求めたものである．この所要日数は機械計算については 128 日，手計算については 72 日となる．

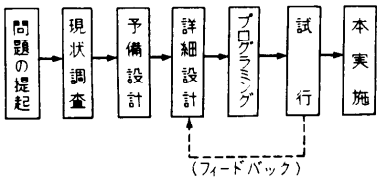
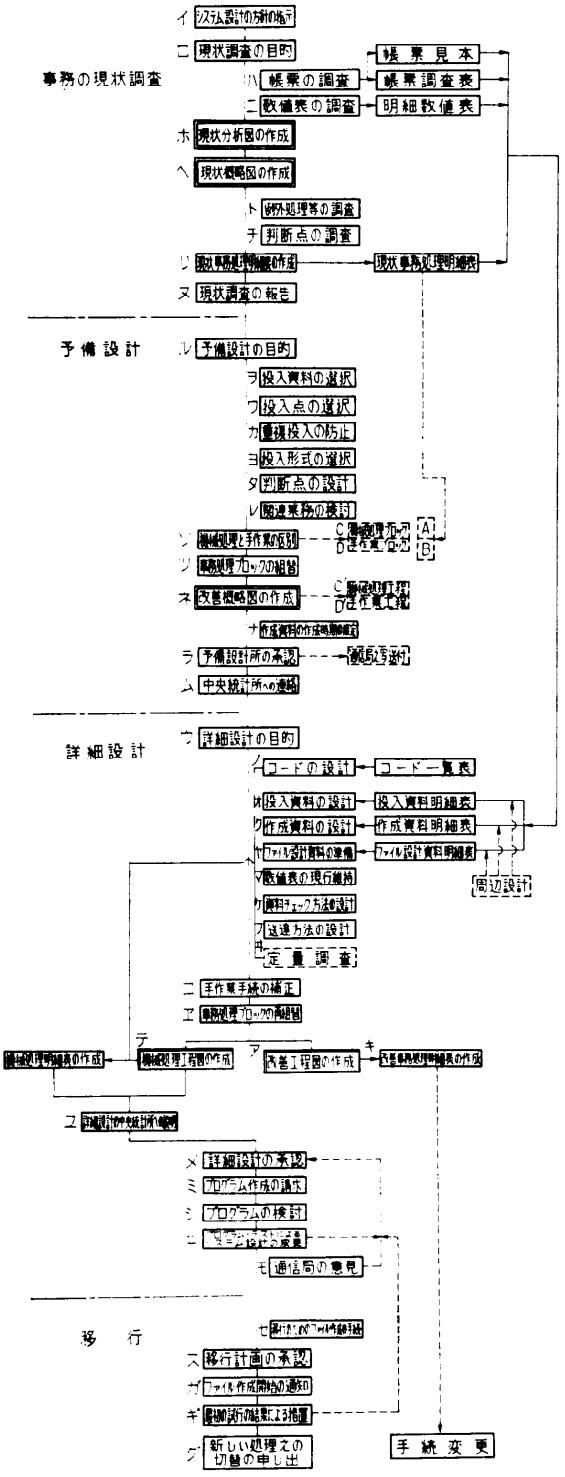


図 1 EDPS 化の過程
Fig. 1 EDPS progress.



注：カタカナ文字は作業記号を示す．

図 2 EDPS 化のためのシステム設計法の構成図
Fig. 2 Organization of EDPS system design.

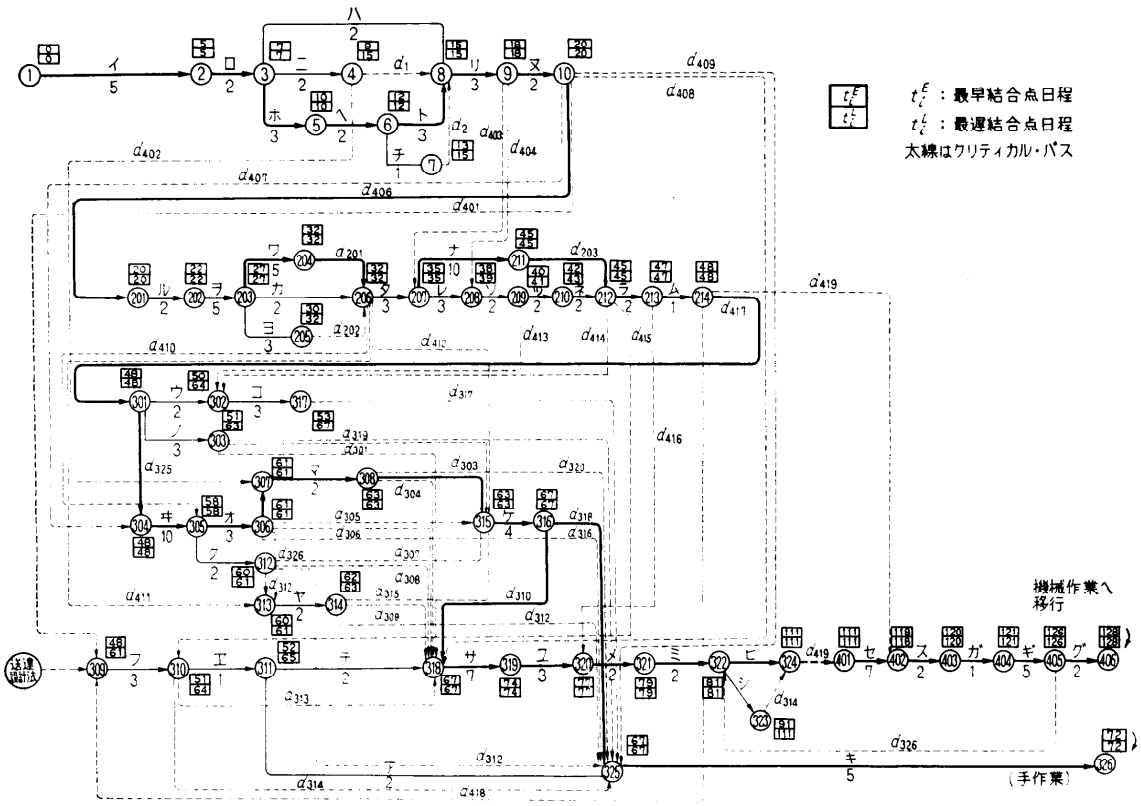


図 3　EDPS 化のためのシステム設計の PERT ネットワーク
Fig. 3　PERT network system design of EDPS.

表 1　作業リスト
Table 1　Work list.

結合点番号 (i, j)	作業内容	作業記号	先行作業	所要日数
(1, 2)	システム設計の方針の指示など	イ	なし	5
(2, 3)	現状調査の目的	ロ	イ	2
(3, 8)	帳票の調査	ハ	ロ	2
(3, 4)	数値票の調査	ニ	ロ	2
(3, 5)	現状分析図の作成	ホ	ロ	3
(4, 8)	ダミー	d_1		0
(5, 6)	現状概略図の作成	ヘ	ホ	2
(6, 8)	例外処理等の調査	ト	ヘ	3
(6, 7)	判断点の調査	チ	ヘ	1
(7, 8)	ダミー	d_2		0
(8, 9)	現状事務処理明細表の作成	リ	ハ, ニ, ト, チ	3

以下省略する

3.　プログラムの作成日程計算

プログラム作成の日程算出は索表によることとし，プログラム作成関係とデバッグ関係に分かれる．この要点は次のとおりである．

1)　本プログラム作成日程表は COBOL 言語によるプログラムに限る．

2)　作成の根拠に危険率の概念を導入した．

3)　因子と水準の関係を設定して表により直接日程算出ができるようにした．入力要素は構造，I/O 数，項目数，難易性の 4 因子とし，入力要素の情報が若干入手できないときでも，残りの情報だけで最適日程が定められるように，どの因子にも 0 水準を設けた．ある因子の情報が不足する場合は，その因子の 0 水準をとればよい．因子と水準の関係を示すと**表 2** のとおりである．

4)　プログラミング技術が進歩した場合，たとえば，既成サブルーチンの活用などにより，プログラミングが徐々に容易になっていった場合にも，この方法は適用できるように，因子として“難易性”を設定してある．プログラムが容易であると判断される場合には，その水準を低くする．

過去の実測データより算出した各水準におけるプログラム作成期間およびデバッグ期間は**表 3** のようにな

表 2 因子と水準
Table 2 Factors and levels.

因子	因子の説明	水準の判定		水準
構造 A	①チェック，②ファイル突合，③計算処理一般，④リポート編集という4つの機能をプログラムの機能要素と呼び，プログラムには少なくとも1つの機能要素を対応させる。因子 A の水準の決定は，いま対象としているプログラムに何個の機能要素をもたせたいかによって決まる。エラー・リストの編集は機能要素「リポート編集」を対応させる。	そのプログラムにもたせたい機能要素の数が	1個の場合	1
			2個の場合	2
			3個の場合	3
			4個の場合	4
I/O 数 B	プログラムが必要とするすべての入出力ファイル数によって因子 B の水準が決まる。ただし，同一プログラム内において内部記憶の不足を補う意味で使用するワーク・ファイルはこの対象から除く。	プログラムの入出力ファイルの数が	2個の場合	2
			3個の場合	3
			4個の場合	4
			5個の場合	5
			6個以上の場合	6
項目数 C	上記入出力ファイルに含まれるすべての項目の数によって因子 C の水準が決まる。ただし，単に後続プログラムに中継させるためだけに，入出力する項目は対象から除く。		1～100個の場合	1
			101～300個の場合	2
			301個以上の場合	3
難易性 D	前記3因子（因子 A, B, C）の補正因子として，以下の事項を総合的に補正する。 ①プログラムそのもの，難易性 ②担当プログラマの熟練性 ③サブルーチンの整備状態 ④そのほか各種の不均衡	プログラムの難易性の点で	かなり容易と考えられる場合	1
			やや容易と考えられる場合	2
			普通と考えられる場合	3
			やや困難と考えられる場合	4
			かなり困難と考えられる場合	5

上記各因子について水準0を設け，その因子の水準が不確定状態の場合に適用する

表 3 プログラム作成期間（単位週）
Table 3 Programming term (week).

水準	プログラム作成期間				デバッグ期間		
	因子 A	因子 B	因子 C	因子 D	因子 A	因子 B	因子 E
1	3,409		4,907	2,887	1,648		3,131
2	5,246	4,581	5,058	3,489	4,345	4,053	3,519
3	8,920	4,927	5,968	4,694	7,719	4,123	4,004
4	12,594	5,619		5,899	11,093	4,120	4,489
5		6,311		7,104		4,296	4,975
6		7,003				4,383	
0	5,018	5,056	5,012	3,938	4,140	4,134	4,134

る．表3によると因子 A は5水準，因子 B は6水準，因子 C は4水準，因子 D は6水準であり $A_iB_jC_kD_l$（i, j, k, l は因子 A, B, C, D のそれぞれの水準）の期間については

プログラム作成の場合 $5\times6\times4\times6=720$

デバッグの場合 $5\times6\times6=180$

であるから

計 $720+180=900$

の表を作成することとなる．

ここで，それぞれ特性ごとに，ある危険率を指定する．プログラム作成期間やデバッグ期間の分布法則は Weibull 分布と考えてよい．その確率密度関数は

$$f(x)=\frac{\beta}{\alpha}(x-\gamma)^{\beta-1}\cdot e^{-\frac{(x-\gamma)}{\alpha},\beta} \quad (x\geqq\gamma) \quad (1)$$

$$=0 \quad (x<\gamma)$$

ここに

α：尺度を決めるパラメータ

β：分布の形を決めるパラメータ

γ：分布の位置を決めるパラメータ

α, β, γ を定めるため x に実測データ $x_1, x_2, \cdots\cdots, x_n$ を代入し

$$\frac{\partial}{\partial\alpha}\log\{f(x_1:\alpha,\beta,\gamma)\times f(x_2:\alpha,\beta,\gamma)\times\cdots\cdots \times f(x_n:\alpha,\beta,\gamma)\}=0$$

$$\frac{\partial}{\partial\beta}\log\{f(x_1:\alpha,\beta,\gamma)\times f(x_2:\alpha,\beta,\gamma)\times\cdots\cdots \times f(x_n:\alpha,\beta,\gamma)\}=0$$

$$\frac{\partial}{\partial\gamma}\log\{f(x_1:\alpha,\beta,\gamma)\times f(x_2:\alpha,\beta,\gamma)\times\cdots\cdots \times f(x_n:\alpha,\beta,\gamma)\}=0 \quad (2)$$

を解くことにより最適の α, β, γ を求めることができる．

危険率を

$$g(x)=\int_t^\infty f(x)dx$$

で定義すると(1)より

$$g(t)=e^{-\frac{(t-\gamma)^\beta}{\alpha}} \quad (3)$$

を得る．

いま，COBOL のプログラム作成に要した 18 個の

JOB についての実測データを(2)に代入してみると，$\alpha=1,546$, $\beta=1,247$, $\gamma=0.867$ を得る．したがって，(1)は指数分布をもってきわめてよく近似することがわかる．

(1)において $\alpha=\mu_\alpha$, $\beta=1$, $\gamma=0$ とすると

$$f_1(x)=\frac{1}{\mu_\alpha}\cdot e^{-\frac{x}{\mu_\alpha}} \quad (x\geqq 0)$$
$$=0 \quad (x<0) \qquad (4)$$

危険率 $g_1(t)$ は

表 4 総合危険率個別危険率対応表*
Table 4 Correspondence of general "risk" and individual "risk".

総合危険率(%) ＼ アクティビティの数	クリティカルパス上のアクティビティの波 表中の数値は単一アクティビティ危険率(%)																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ジョブ完了予定日についての危険率の指定(%) 5	5	10	15	20	20	20	25	25	25	25	30	30	30	30	30	30	30	30	30	30
10	10	15	20	25	25	25	30	30	30	30	30	30	35	35	35	35	35	35	35	35
15	15	20	25	25	30	30	30	35	35	35	35	35	35	35	35	35	35	35	35	40
20	20	25	30	30	35	35	35	35	35	35	35	35	40	40	40	40	40	40	40	40
25	25	30	30	35	35	35	35	40	40	40	40	40	40	40	40	40	40	40	40	40
30	30	35	35	35	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40
35	35	35	40	40	40	40	40	40	40	40	45	45	45	45	45	45	45	45	45	45
40	40	40	40	40	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45
45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45
50	50	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45	45
55	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
60	60	55	55	55	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
65	65	60	55	55	55	55	55	55	55	55	50	50	50	50	50	50	50	50	50	50
70	70	60	60	60	55	55	55	55	55	55	55	55	55	55	55	55	55	55	55	55

* プログラミング日程計画法：中央統計所標準課より

表 5 プログラム単一アクティビティ日程表*
Table 5 Calculation of individual program.

構造 A	I/O数 B	項目数 C	難易性 D	プログラム単一アクティビティ危険率(%) 表中の期間単位は週 5	10	15	20	25	30	35	40	45	50	55	60	65	70
2	3	1	1	8.5	6.5	5.5	4.5	4	3.5	3	2.5	2.5	2	1.5	1.5	1.5	1
			2	10.5	8	6.5	5.5	5	4	3.5	3	3	2.5	2	2	1.5	1.5
			3	14	11	9	7.5	6.5	5.5	5	4.5	4	3.5	3	2.5	2	1.5
			4	18	13.5	11	9.5	8	7	6	5.5	4.5	4	3.5	3	2.5	2
			5	21.5	16.5	12.5	11.5	10	8.5	7.5	6.5	5.5	5	4.5	3.5	3	2.5
			0	15	11.5	9.5	8	7	6	5	4.5	4	3.5	3	2.5	2	2
		2	1	9	7	6	5	4.5	3.5	3	3	2.5	2	2	1.5	1.5	1
			2	11	8.5	7	6	5	4.5	4	3.5	3	2.5	2	2	1.5	1.5
			3	14.5	11	9.5	8	7	6	5	4.5	4	3.5	3	2.5	2	1.5
			4	18	14	11.5	10	8.5	7.5	6.5	5.5	5	4	3.5	3	2.5	2
			5	22	17	14	11.5	10	9	7.5	6.5	6	5	4.5	3.5	3	2.5
			0	15.5	12	9.5	8	7	6	5.5	4.5	4	3.5	3	2.5	2	2
		3	1	12	9	7.5	6.5	5.5	5	4	3.5	3	3	2.5	2	1.5	1.5
			2	13.5	10.5	8.5	7.5	6.5	5.5	5	4	3.5	3	2.5	2.5	2	1.5
			3	17.5	13.5	11	9.5	8	7	6	5.5	4.5	4	3.5	3	2.5	2
			4	21	16	13.5	11.5	9.5	8.5	7.5	6.5	5.5	5	4	3.5	3	2.5
			5	24.5	19	15.5	13	11.5	10	8.5	7.5	6.5	5.5	5	4	3.5	3
			0	18	14	11.5	9.5	8.5	7.5	6.5	5.5	5	4	3.5	3	2.5	2
		0	1	9	7	5.5	5	4	3.5	3	3	2.5	2	2	1.5	1.5	1
			2	11	8.5	7	6	5	4.5	4	3.5	3	2.5	2	2	1.5	1.5
			3	14.5	11	9	8	6.5	6	5	4.5	4	3.5	3	2.5	2	1.5
			4	18	14	11.5	9.5	8.5	7.5	6.5	5.5	5	4	3.5	3	2.5	2
			5	21.5	16.5	13.5	11.5	10	8.5	7.5	6.5	6	5	4.5	3.5	3	2.5
2	3	0	0	15	11.5	9.5	8	7	6	5.5	4.5	4	3.5	3	2.5	2	2

* プログラミング日程計画法：中央統計所標準課より

$$g_1(t)=\int_t^\infty f_1(x)\,dx=e^{-\frac{t}{\mu_a}} \qquad (5)$$

となる.

本日程計算においては(5)に $\mu_a=(A_iB_jC_kD_l$ の作成日程の平均値) の値を入れ, 1) 総合危険率, 個別危険率対応表, 2) プログラム単一アクティビティ日程表, 3) デバッグ単一アクティビティ日程表が作られている. **表 4, 表 5, 表 6** にその一部を示す.

いま, ある JOB のプログラム作成日程を定める場合, まず, 総合危険率を定める. 総合危険率が 20%, アクティビティの数が 7 であれば, 表 4 において個別危険率は 35% となる. 次に個別のプログラムの構造 A, I/O 数 B, 項目数 C, 難易性 D において, それぞれ水準を選定する. もし, $A_2B_3C_1D_0$ を得たとすると表 5 より 5 週を得る. デバッグの日程についても同様に行なう.

4. 修正と再試行とに要する最適日程計算

試行の段階において修正ユニットが発見され, 詳細設計の段階にフィードバックされ, 修正されプログラミングを組みなおし, 試行の段階に送られ, ここで再試行される場合である. この場合修正に際しては, 通常システム担当とプログラム担当とは一体であるので, 以後修正担当という. フィードバックの方法については, いろいろの方法があるが, 本論文では修正ユニットが r 個になったら試行側より修正側へ送付する r-固定方式によるものとする. 修正ユニットが発生してから修正を受けるまでの平均待ち時間 W は

$$W=\frac{1}{\mu}\left\{\sum_{j=1}^{r}\frac{\gamma_j}{1-\gamma_j}\right\}+\frac{r-1}{2}\left(\frac{1}{\mu}+\frac{1}{\lambda}\right)$$

で示される[3].

ここに μ は修正に要する時間, λ は修正ユニットの発生率, $\gamma_j(j=1, 2, \cdots\cdots, r)$ は修正ユニットが十分存在するときに到着間隔の間に処理が完了する修正ユニットの数の確率母関数を $K(Z)$ としたとき, Z に関する方程式 $Z^r=K(Z)$ の単位円内 ($|Z|<1$) の r 個の根である.

W と $\rho(=\lambda/\mu)$ との関係を示すと**図 4** のようになる.

いま, 修正と再試行を行なうに際し, 修正を行なう側を A, 試行を行なう側を B とし, n 個の修正ユニットが発生したとする. このとき修正個数 n に $1, 2, \cdots\cdots, n$ のように番号を付し, それぞれの修正日数を $A_1, A_2, \cdots A_i, \cdots, A_n$, 再試行に要する日数を $B_1, B_2, \cdots, B_i, \cdots, B_n$ とする. すると**図 5** のガントチャートにより, 修正ユニット 1 は A_1 の処理が行なわれ, その間 B は遊休となり, A_1 日後にはじめて B が処理を行なうこととなる. 再試行は必ず修正が行なわれてからなされるのであるから, A には遊休日はなく連続的に仕事を続けてゆくことができる. したがって, 再試

表 6 デバッグ単一アクティビティ日程表
Table 6 Calculation of individual debug.

構造 A	I/O数 B	難易性 D	デバッグ単一アクティビティ危険率(%) 表中の期間単位は週 5	10	15	20	25	30	35	40	45	50	55	60	65	70
2	2	1	10	7.5	6	5	4.5	4	3.5	3	2.5	2.5	2	1.5	1.5	1
		2	11	8.5	7	6	5	4.5	4	3.5	3	2.5	2	2	1.5	1.5
		3	12.5	9.5	8	6.5	5.5	5	4.5	4	3.5	3	2.5	2	2	1.5
		4	14	10.5	9	7.5	6 5	6.5	5	4	3.5	3	3	2.5	2	1.5
		5	15.5	12	9.5	8	7	6	5.5	4.5	4	3.5	8	2.5	2	2
		0	13	10	8	7	6	5	4.5	4	3.5	3	2.5	2	2	1.5
	3	1	10	7.5	6.5	5.5	4.5	4	3.5	3	2.5	2.5	2	1.5	1.5	1
		2	11	8.5	7	6	5	4.5	4	3.5	3	2.5	2	2	1.5	1.5
		3	12.5	9.5	8	7	6	5	4.5	4	3.5	3	2.5	2	2	1.5
		4	14	11	9	7.5	6 5	5.5	5	4.5	3.5	3.5	3	2.5	2	1.5
		5	15.5	12	10	8.5	7	6	5.5	4.5	4	3.5	3	2.5	2	2
		0	13	10	8	7	6	5	4.5	4	3.5	3	2.5	2	2	1.5
	4	1	10	8	6.5	5.5	4.5	4	3.5	3	2.5	2.5	2	1.5	1.5	1
		2	11.5	9	7	6	5.5	4.5	4	3.5	3	2.5	2.5	2	1,5	1.5
		3	13	10	8	7	6	5	4.5	4	3.5	3	2.5	2	2	1.5
		4	14.5	11	9	7.5	6.5	6	5	4.5	4	3.5	3	2.5	2	1.5
		5	16	12	10	8.5	7.5	6.5	5.5	5	4	3.5	3	2.5	2.5	2
2	4	0	13	10	8.5	7	6	5.5	4.5	4	3.5	3	2.5	2.5	2	1.5

* プログラミング日程計画法: 中央統計所標準課より

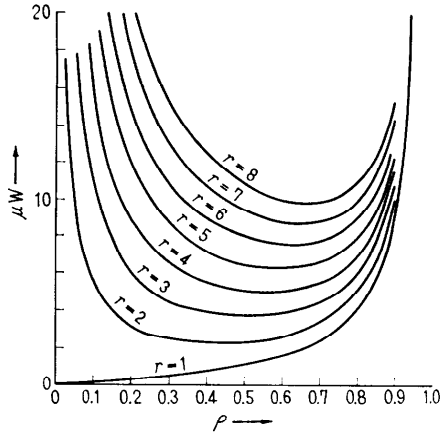


図 4 フィードバックまでの所要時間 (r-固定方式)
Fig. 4 Feedback indispensable times (r fixed system).

行がすべて完了するまでの総処理時間は，A_1 の最初から B_n の終わりまでの経過時間として算出できる．B の i 番目の試行直前の遊休日数を x_i で表わすと，B の総試行日数は $\sum_{i=1}^{n} B_i$，Bの総遊休日数は$\sum_{i=1}^{n} x_i$ となるから，B の総経過日数 T は

$$T=\sum_{i=1}^{n-1}(B_i+x_i)$$

いま，$X(n)=\sum_{i=1}^{n} x_i$ とおくと，一般に

$$X(n)=\max\left[\left(\sum_{i=1}^{n}A_i-\sum_{i=1}^{n=1}B_i\right),\ \left(\sum_{i=1}^{n-1}A_i-\sum_{i=1}^{n-2}B_i\right),\ \cdots\cdots,\ \left(\sum_{i=1}^{2}A_i-\sum_{i=1}^{1}B_i\right),\ \sum_{i=1}^{1}A_i\right]$$

$$=\max_{1\leqq r\leqq n}\left(\sum_{i=1}^{r}A_i-\sum_{i=1}^{r-1}B_i\right) \qquad (6)$$

ただし，r は正の整数，$X(n)$ は有効さ関数である．問題は $X(n)$ を最小にするような処理順序を求めればよいことになる．これについては Johnson[4] と Bellman[5] が解析的に最適順序を得るための方法を発表している．

1つの仕事処理順序 $L(1, 2, \cdots\cdots, m-1, m, m+1, \cdots\cdots, n)$ とその m 番目と $(m+1)$ 番目を入れかえた順序 $L'(1, 2, \cdots, m-1, m+1, m, m+2, \cdots, n)$ とを考えて

$$F(r)=\sum_{i=1}^{r}A_i-\sum_{i=1}^{r-1}B_i \qquad (7)$$

とおくと，(6)は

$$X(n)=\max_{1\leqq r\leqq n}[F(r)]$$

$r=1, 2, \cdots, m-1, m+2, \cdots, n$ の場合には L にたいする $F(r)$ と L' にたいする $F(r)$——これを $F'(r)$ とする——は等しい．すなわち，$F(r)=F'(r)$ である．しかし，$r=m, m+1$ の場合には(7)は

$$F(m)=\sum_{i=1}^{m}A_i-\sum_{i=1}^{m-1}B_i,\quad F(m+1)=\sum_{i=1}^{m+1}A_i-\sum_{i=1}^{m}B_i$$

$$F'(m)=\sum_{i=1}^{m-1}A_i+A_{m+1}-\sum_{i=1}^{m-1}B_i,\quad F'_{(m+1)}=\sum_{i=1}^{m+1}A_i-\sum_{i=1}^{m-1}B_i-B_{m+1}$$

よって

$$\max\left[F(m), F(m+1)\right]=\max\left[\left(\sum_{i=1}^{m}A_i-\sum_{i=1}^{m-1}B_i\right),\ \left(\sum_{i-1}^{m+1}A_i-\sum_{i=1}^{m}B_i\right)\right]$$

$$\max\left[F'(m),\ F'(m+1)\right]=\max\left[\left(\sum_{i=1}^{m-1}A_i+A_{m+1}-\sum_{i=1}^{m-1}B_i\right),\ \left(\sum_{i=1}^{m+1}A_i-\sum_{i=1}^{m-1}B_i-B_{m+1}\right)\right]$$

したがって

$$\min(A_{m+1}, B_m)\leqq\min(A_m, B_{m+1}) \qquad (8)$$

ならば

$$\max[F(m), F(m+1)]\geqq\max[F'(m), F'(m+1)] \qquad (9)$$

(8)と(9)とにおいて，m 番目の仕事と $(m+1)$ 番目の仕事の処理時間を比較して，最小値が A_{m+1} または B_m であるときには，経過時間の最大値は $F(m)$ または $F(m+1)$ であるから，m 番目の仕事と $(m+1)$ 番目の仕事とを入れかえたほうが，経過時間が短くなるという可能性がある．この操作を繰り返してゆけば，いま，最小値が A_s の場合は s 番目の仕事は他のすべての仕事に優先し，最小値が B_t の場合は t 番目の仕事は，すべての仕事のあとにまわされる．

(8)が等号の場合には(9)も等号となるから入れかえても同じである．

したがって，最適仕事処理順序を得るための手順を次のように定めることができる．

手順 1 すべての A_p, B_p について，その最小値

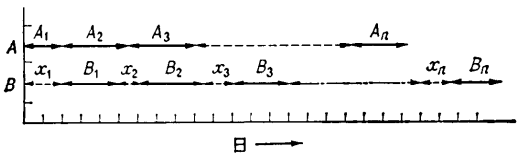


図 5 A, B のガントチャート
Fig. 5 Gantt chart for A and B work.

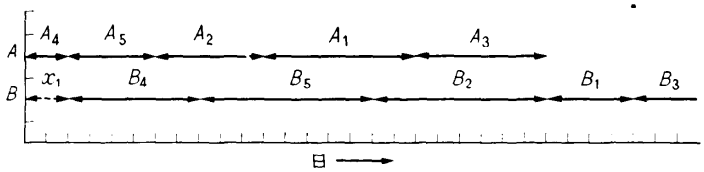


図 6 図5に対する最適処理順序のガントチャート
Fig. 6 Most suitable procedure series for Fig. 5.

$\min_p (A_p, B_p)$ をもとめる.

手順 2 それが A_t であれば, S 番目の修正ユニットを最初に配置し, B_t であれば t 番目の修正ユニットを最後に配置する.

手順 3 つぎに s 番目のあるいは t 番目の修正ユニットを除いた残りの修正ユニットについて, 手順1, 手順2を行ない, 修正ユニットの全部の順序が決定するまでこの作業を繰り返す.

いま, r-固定方式($r=5$)で送付されてきた修正ユニットを検討した結果**表7**を得た. このときの最適処理順序を求める. 最小値は $A_4=2$ であるから修正ユニットを最初に行なう. 次にこの修正ユニットを除いた残りの中で最小値は $B_3=3$ であるから, これを最終の仕事とする. 修正ユニット3および4を除いた残りの中では A_5 か B_1 が最小であるから, 修正ユニット5を2番目に修正ユニット1を最終の修正ユニット3の前に配置する. かくして, 最適の処理順序 4→5→2→1→3 が得られる. したがって, **図6**のようなガントチャートが得られ, これから全経過日数は 31日, 試行側の最少遊休日2日が得られる.

表 7 修正ユニット ($r=5$)
Table 7 Data ($r=5$) to be corrected.

区別 / 修正ユニット	A_i	B_i
1	7	4
2	5	8
3	6	3
4	2	6
5	4	8

6. むすび

情報システム EDPS 化の日程計算は, 前にも述べたとおり非常に複雑であるが, 電子計算機の稼動計画をたてるうえにきわめて重要なことである. しかし, 従来より系統だった計算方法は示されていなかったが今回1つの計算方法が示されたわけである. 日程計算はその計算が平易であって, しかも, 求められた結果が正確であることを要する. 本論文において, EDPS 化システムの日程計算は PERT ネットワークに要素の推定値を書き入れることにより, プログラミングは数値表により, フィードバックは図表により, 修正と再試行はガントチャートを描くことにより, 従来よりも確実な根拠にたって求められる. 今後なお, 種々な方法を考案し, より厳密な数量化の研究を行なう必要がある.

最後に本論文作成に関し, 東北大学大泉教授, 木村教授にご示唆をいただいた. また, 東北学院大学工学部長永井教授よりもご示唆をいただいた. ここに厚く感謝する. なお, ご指導およびご協力をいただいた当公社理事電気通信研究所緒方所長および中央統計所の方々に対し謝意を表わす.

参考文献

1) 電電公社システム研究会: システム設計の手引: 日本事務能率協会, 昭 39年6月15日初版発行,
2) 粟野敏雄: 直線型経営情報システム設計法の実際とその利点: 経営科学, 第13巻, 第2号, 1970.
3) 粟野敏雄: 経営情報システム設計の所要時間算出の問題: 経営科学, 第14巻, 第2号. 1970.
4) S. M. Johnson: Optimal two- and three-stage production schedules with set up times included, Nav. Res. Log. Quart., 1—1954.
5) R. Bellman: Mathematical aspects of scheduling theory, RAND Report, 1955, p.651.

(昭和45年12月8日受付)