日程管理に関する一考察

近畿大学　工学部　　正員　小川康彦

まえがき　従来、日程管理に用いられたバーチャートに対して、10数年前開発されたPERTの優位性は、これまでの使用経験によって実証されてきた。PERTはグラフ理論のひとつの適用例であり、グラフの枝（アーク）と点（ノード）にいろいろな特性を与えて、その上にプロジェクトの日程的な面を投影させたものである。PERTの最大の利点はクリティカル・パスという概念の発生と、可視性がありコミュニケーションに便利なことである。

PERT計算のアルゴリズムはノードを中心としてノードからでているアークについて調べる方法と、すべてのアークについての探索を基本としたものがある。ここで前者をノード型計算法、後者をアーク型計算法とよぶことにする。ノード型計算法はネットワーク図形をみながら手計算をする場合の手順を用いたものであり、電子計算機では全ネットワークの結合状態を調べる必要がある。アーク型計算法は結合状態を調べる手間をはぶくために、作業リスト順にアークを中心にして計算を行い、これを繰り返して収束させようとするものである。[1) 本報は、この二つの計算法を比較検討し、モデル化したネットワークについて、計算時間を比較した。

ノード型計算法のアルゴリズム　ノード番号がtopological orderingされた場合の最早時刻の計算手順を考えてみる。

n＝ノード数，m＝アーク数，t_{Ei}＝ノードiの最早時刻，D_{ij}＝アーク(i,j)の所要時間とすると、各ノードの最早時刻は次式によって求められる。

$$t_{E0}=0,\quad t_{Ei}=\max_{k}(t_{Ek}+D_{ki}),\ (i=1,\cdots,n-1)\quad\cdots(1)$$

図-1の流れ図より(1)式の計算の手間は$m\times n$の程度である。

最遅時刻の計算の手間も最早時刻と同様である。

アーク型計算法のアルゴリズム　「アークについての計算」を基本計算と考えて、最早時刻の計算手順を考えてみる。

初期値として $t_{Ei}^{(0)}=0,\ (i=0,1,\cdots,n-1)$

アーク(i,j)についての改良を $t_{Ej}^{(l+1)}=\max(t_{Ei}^{(l)}+D_{ij},\ t_{Ej}^{(l)})$ $\}\cdots(2)$

収束回数をLとすれば、計算の手間は$m\times L$の程度である。

最遅時刻の計算の手間も最早時刻と同様である。

(2)式からもわかるように、この計算法ではtopological orderingの必要はない。

また アークの順の与え方によって 収束回数Lが変ると想定される。

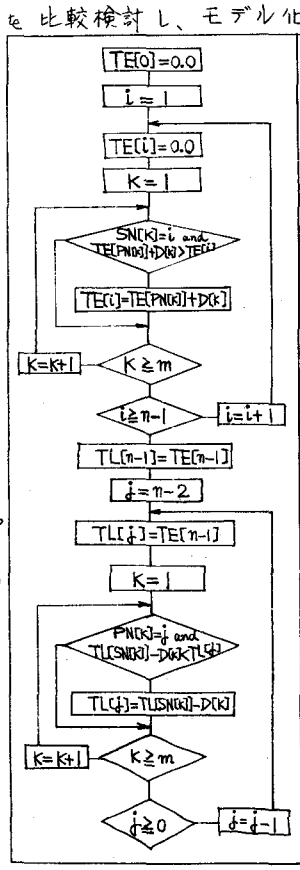

図-1 ノード型計算法の流れ図

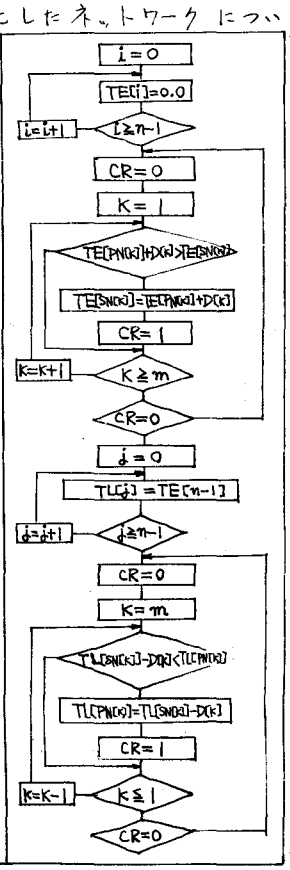

図-2 アーク型計算法の流れ図

アーク型計算の収束性　以下の計算ではネットワークを単純化して、すべての作業時間を $D_{ij}=1.0$ とする。最も簡単なネットワークとして、図-3の場合を考えると、作業順のつくり方は6通りある。その各々について (2)式を適用すると表-1の結果を得た。($\Rightarrow$ はクリティカル・パス)。表-1より作業順によって収束回数が異なることがわかる。最小が2回，最大が4回である。　同様のことを図-4, 図-5, 図-6で調べると、図-4では最小が2回、最大が5回，図-5では最小が2回、最大が4回，図-6では最小が2回，最大が5回である。

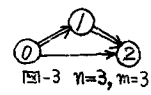

図-3 $n=3, m=3$

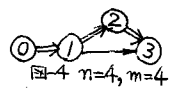

図-4 $n=4, m=4$

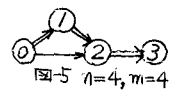

図-5 $n=4, m=4$

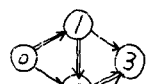

図-6 $n=4, m=5$

	作業順	$t_{ij}^{(0)}$	$t_{ij}^{(1)}$	$t_{ij}^{(2)}$	$t_{ij}^{(3)}$	$t_{ij}^{(4)}$	収束回数
ⓐ	(0,1)	0	1	1	1		3
	(0,2)	0	1	2	2		
	(1,2)	0	2	2	2		
ⓑ	(0,1)	0	1	1			2
	(1,2)	0	2	2			
	(0,2)	0	2	2			
ⓒ	(0,2)	0	1	2	2		3
	(2,1)	0	1	1	1		
	(1,2)	0	2	2	2		
ⓓ	(0,2)	0	1	1	2	2	4
	(1,2)	0	1	2	2	2	
	(2,1)	0	1	1	1	1	
ⓔ	(1,2)	0	1	2	2		3
	(2,1)	0	1	1	1		
	(0,2)	0	2	2	2		
ⓕ	(1,2)	0	1	2	2		3
	(0,2)	0	1	2	2		
	(0,1)	0	1	1	1		

表-1　図-3のアーク型計算の収束

これらの簡単なネットワークの考察によれば、作業順がクリティカル・パス順のとき収束回数が小さく，逆順の場合大きくなる。したがってアーク型計算法によれば、作業順の与え方次第で $m \times l < m \times n$ となり，ノード型計算法より計算時間をはやめることが可能である。

つぎに、ノード数を多くして図-7のネットワークA，図-8のネットワークBについて近畿大学 電子計算機 FACOM 231 を使って数値実験した結果を表-2に示す。

A，Bともに *topological ordering* されている。作業順 A_1 は (0,1)(0,3)(0,5)(1,2)(2,4)(2,6)(2,8)(3,4)(4,7)(5,6)(6,9)(7,10)(8,10)(8,11)
作業順 A_2 は (0,1)(1,2)(2,8)(8,10)(0,3)(3,4)(4,7)(7,10)(0,5)(5,6)(6,9)(9,10)(2,4)(2,6) である。作業順 A_3 は A_1 の逆順とする。
作業順 B_1 は (0,1)(0,2)(0,4)(1,3)(2,5)(3,6)(4,7)(4,8)(4,10)(5,9)(6,12)(7,11)(8,13)(9,14)(10,11)(10,13)(10,20)(11,12)(11,16)(12,15)(13,14)
(13,17)(14,18)(15,19)(16,20),(17,20)(18,21)(19,22)(20,24)(21,23)(22,24)(23,24) である。作業順 B_2 は B_1 の逆順とする。

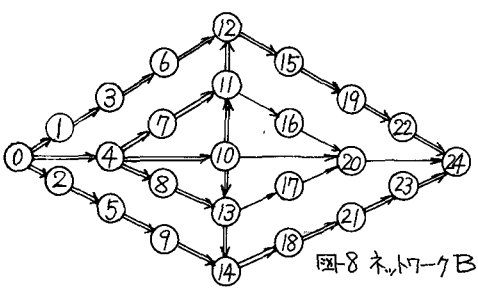

図-8 ネットワークB, $n=25, m=32$

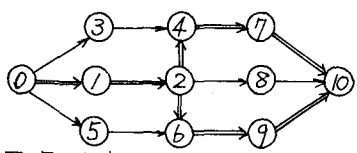

図-7　ネットワークA，$n=11, m=14$,

作業順	A_1	A_2	A_3	B_1	B_2
ノード型計算法	12	12	12	41	41
アーク型計算法	10	12	19	23	61

表-2　FACOM 231 によるPERT計算時間(秒)

まとめ　ノード型計算法とアーク型計算法の比較を要約すると次の通りである。

1) ノード型計算法は各ノードについて全ネットワークの結合状態を調べる必要があるが，アーク型計算法ではこの手間をはぶきアークを基本とした収束計算法を用いている。

2) ノード型計算法では前処理として *topological ordering* を必要とするが，アーク型計算法ではその必要はない。

3) アーク型計算法は作業順の与え方によっては収束が遅くなることもあるが，クリティカル・パスの順に作業リストをつくれば，はやく収束し、ノード型計算法より計算時間をはやめることができる。

実際問題としては、ほぼクリティカル・パスの順に作業が並んでいると考えられるので，電子計算機を用いるときは、アーク型計算法による場合が計算時間をはやめると想定される。

今後，さらに大きなネットワークについて数値実験を行い検討を加える予定である。

参考文献 1)「Topological Ordering なしの PERT・CPM 計算」　須永照雄　経営科学 第11巻 第2号