Ⅳ-111　PERT計算に関する一考察

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てき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。また、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方法の2つがある。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、電子計算機では各ノードで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2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されたものとして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。各ノードの最早時刻t_{Ei}は次式によって求められる。

$$t_{E0}=0,\quad t_{Ei}=\max_{k}(t_{Ek}+D_{ki}),\ (i=1,2,\cdots,n-1) \quad \cdots(1)$$

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は

$$\left.\begin{array}{l}\text{初期値として } t_{Ei}^{(0)}=0,\ (i=0,1,\cdots,n-1)\\ \text{アーク}(i,j)\text{についての改良を } t_{Ej}^{(k+1)}=\max(t_{Ei}^{(k)}+D_{ij},\ t_{Ej}^{(k)})\end{array}\right\}\quad\cdots(2)$$

である。すべてのjについて$t_{Ej}^{(k+1)}=t_{Ej}^{(k)}$で収束とする。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。また作業順の与え方によって収束回数Lがかわると想定される。

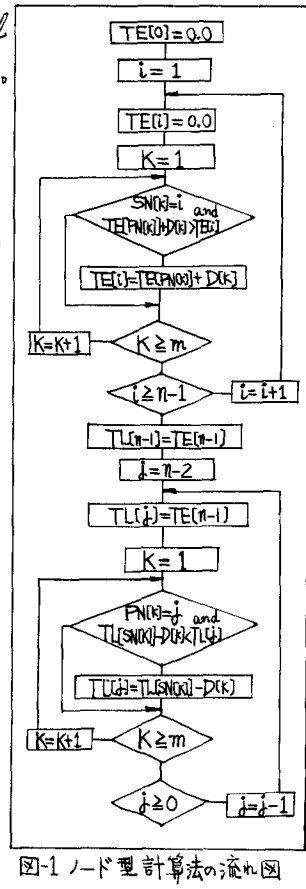

図-1 ノード型計算法の流れ図

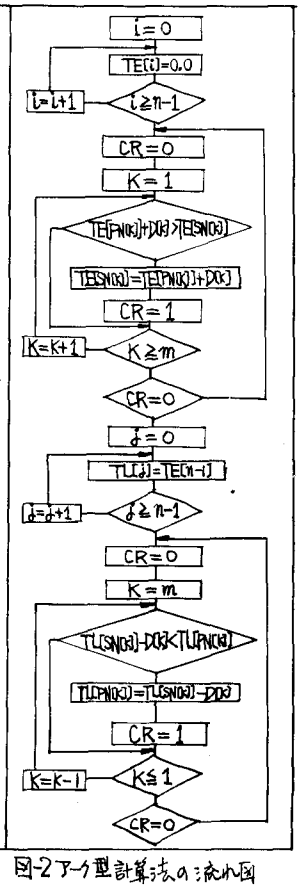

図-2 アーク型計算法の流れ図

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。(⇒は critical path)。その各々について(2)式を適用すると表-1の結果を得た。これより，作業順を critical path の順に与えた場合が収束回数が小さく，その逆順の場合が大きくなることがわかる。

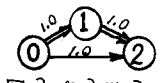

図-3　n=3, m=3

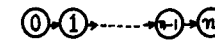

図-4 直列型PERT

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

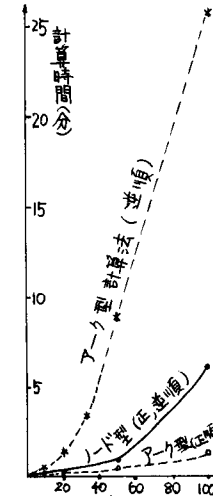

図-5, 直列PERT計算時間

つぎに図-4の直列型のPERTを，作業順を正順 (0,1)(1,2)……(n-1, n) とした場合と逆順 (n-1,n)(n-2,n-1)……(1,2)(0,1) とした場合について 近畿大学 電子計算機 FACOM 231 を用いて計算し，その所要時間を図-5に示す。この結果より，アーク型計算法を用いれば，作業順の与え方如何でノード型計算法より計算時間をはぶくことができる。一般の場合について検討するために，図-6のネットワーク[2]について作業順を種々かえて，数値実験した結果を図-7に示す。これより，一般にアーク型計算法がノード型計算法より，計算時間が短いと想定される。

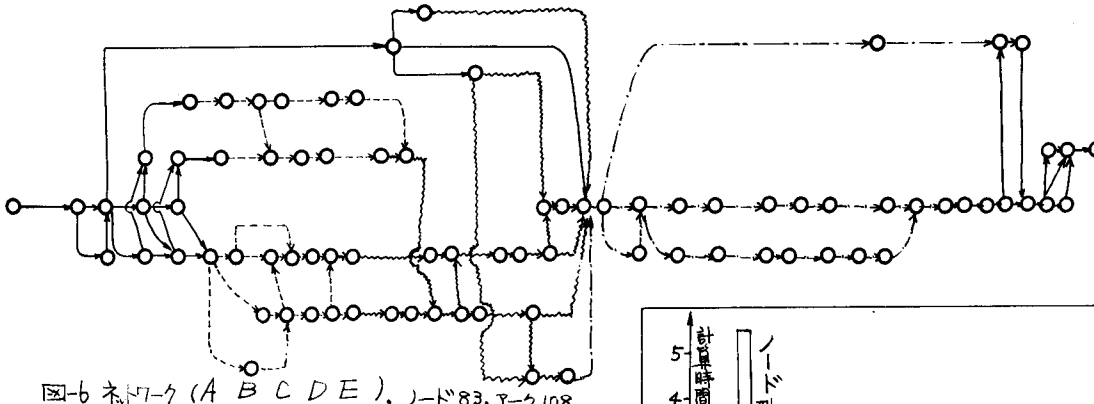
図-6 ネットワーク (A B C D E)，ノード83，アーク108

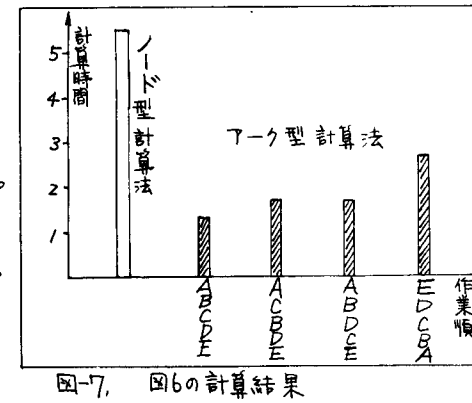

図-7,　図6の計算結果

まとめ　ノード型計算法とアーク型計算法の比較は次の通りである。

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，アーク型では，この手間をはぶき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) アーク型計算法において，作業順をほぼクリティカル・パスの順に与えればノード型計算法より計算時間を短縮することが可能である。

以上の検討より，使用目的によってはアーク型計算法が有利であると考えられる。

参考文献

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

2) 建設コンサルタンツ協会　「パート(PERT)による工程管理（その2）」