経営科学 第11巻 第2号 (1968年1月)

Topological Ordering なしの PERT・CPM 計算†

須永照雄*

1. まえがき

PERT・CPM のアルゴリズムはすっきりしたものであるが[1][2][3], そのプログラムとなるとそれ程簡単ではない. 特に現在, 前処理として topological ordering が必要とされ, PERT プログラムを作るにしても簡単ではない. 一般に思ったよりプログラムが複雑になる原因の一つは, 人手でやる計算手順と同じやり方を, 人間と機能の違う計算機にやらせることにあると思う. 例えば一つのノードからでているアークについてすべて調べるといった scanning は, 人間が図を見てやるのは局所的で簡単であるが, 計算機でやるときは全ネットワークの結合状態を調べる必要がある. 以下計算機の機能の点からみた PERT・CPM プログラムについての一案を述べ, また実際作ったものについて二, 三考察する.

2. 計算方法

ネットワークのノード番号が topological ordering された場合の最早時刻の計算手順を考えてみよう. [4][5]

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

$$t_{E0}=0$$

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

により計算される. ここで k はノード i のすべての先行結合点番号をとるので, 結局すべてのアークについて, その後続点番号が i となるものを探すことになる. このようなことをすべてのノード i について行なうので, 手間は $m\times n$ の程度になると考えられる. この計算法を吟味してみると,「すべてのアークについての探索」が基本になっていることが分る. そこで初めから「アークについての計算」を基本計算と考えてもっと手間をへらすことができないであろうか.

† 1967年2月16日受理 1966年度秋季研究発表会講演

* 九州大学工学部

さてまず，初期値として

$$t_{Ei}^{(0)}=0$$

とおく．アーク (i,j) についての改良を

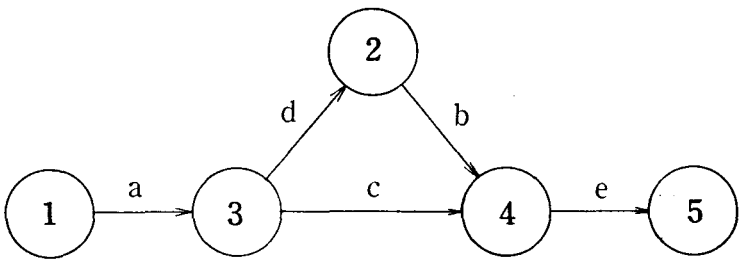
$$t_{Ej}^{(l+1)}=\max(t_{Ei}^{(l)}+D_{ij},\ t_{Ej}^{(l)})$$

とする．この改良をアーク順を追って行ない，改良の余地がなくなったとき計算は終了する．このときの手間は前述の $m\times n$ に比し大分少ないものとなる．具体的な例については次節に述べる．

ＣＰＭ計算も同じ方法で計算することができる．例えば最大フローを求めるのにラベリングを行なうが，これもノード順を追って計算するより，アーク順に従って計算した方が手間が少ない．これについては，後に簡単な例を用いて説明することにする．

3. PERT 計算の例

第１図，第１表で与えられたネットワークを例に説明する．ノード番号，アーク番号は一連番号であるがその番号付は任意でよい．ここではアークにアルファベットが付けられている．以下最早時刻を求める手順を示す．



第 1 図

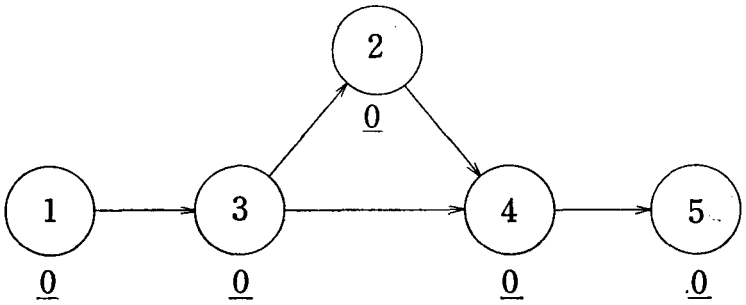
第 1 表

工　程	*i*	*j*	*Dij*
a	1	3	5
b	2	4	8
c	3	4	15
d	3	2	10
e	4	5	5

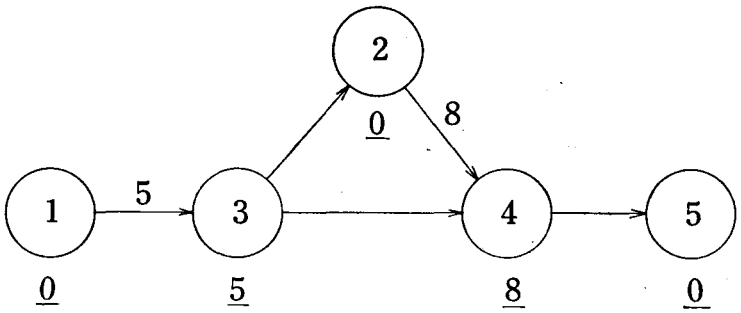
まず最初ノード時刻を全て零とおく（第２図）．これらを初期値として（*a*)工程より逐次的に計算を行なう．

(*a*)工程は５日かかるから③のノード時刻は少なくとも５でないといけない．

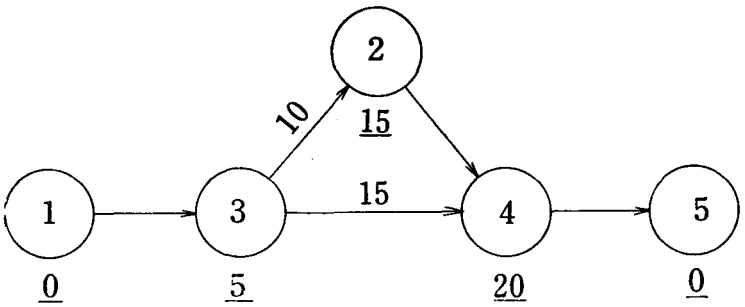
(*b*)工程は８日かかるから④のノード時刻は少なくとも８である（第３図）．



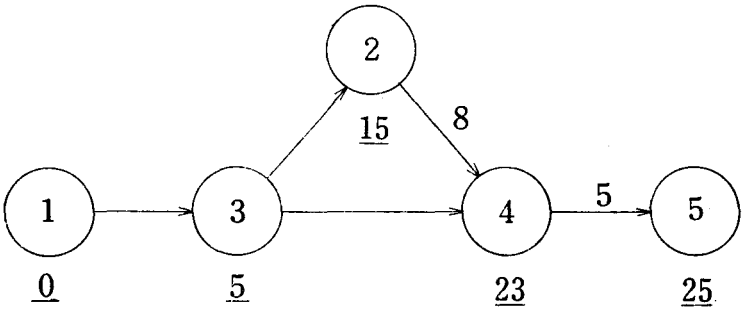
第 2 図



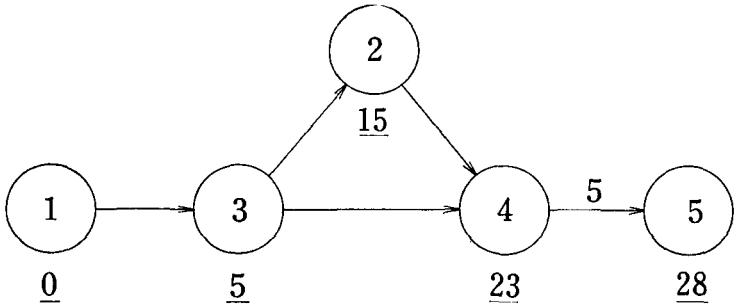
第 3 図



第 4 図



第 5 図



第 6 図

(*c*)工程は15日かかるから④のノード時刻は20となる.

(*d*)工程は10日かかるから②のノード時刻は15となる（第4図）.

(*e*)工程は5日かかるから⑤のノード時刻は25となる.

次にまたはじめの（*a*)工程にもどり計算を繰返す.

(*a*)工程について③のノード時刻は変化しない.

(*b*)工程については8日で$\underline{15}+8\geq\underline{20}$より④のノード時刻は23となる（第5図）.

(*c*), (*d*)工程についてはノード時刻は変化しない.

(*e*)工程については5日で$\underline{23}+5>\underline{25}$より⑤のノード時刻は28となる（第6図）.

これ以後の計算はノード時刻を変化させないので第6図の結果が最早時刻となる.

4. PERT のプログラム

前述の計算を ARGOL でプログラムしてみる. 最早時刻の計算の後に最遅時刻の計算を追加して示す.

まず次のように記号を定める.

P, Q＝アーク番号

I, J ＝ノード番号

M ＝アーク数

N ＝ノード数

TE〔I〕＝ノード *i* の最早時刻

TL〔J〕＝ノード *j* の最遅時刻

IP〔P〕＝アーク P の *i* 結合点

JP〔Q〕＝アーク Q の *j* 結合点

D〔P〕＝アーク P の所要時間

CR ＝計算終了のための判定

プログラムの主要部は次のようになる.

```
  for I: =1 step 1 until N do TE[I]=0;
L1: CR=0;
  for P: =1 step 1 until M do
  begin I: =IP[P]; J: =JP[P];
      if TE[I]+D[P]>TE[J] then
      begin TE[J]: =TE[I]+D[P];
            CR: =1
      end
  end;
  if CR=1 then go to L1;
  for J: =1 step 1 nntil N do TL[J]: =TE[N]1);
L2: CR: =0;
  for P: =1 step 1 until M do
  begin Q: =M+1-P;
        I: =IP[Q]; J: =JP[Q];
      if TL[J]-D[Q]<TL[I] then
      begin TL[I]: =TL[J]-D[Q];
            CR: =1
      end
  end;
  if CR=1 then go to L2;
```

最遅時刻の計算は最早時刻の手順とほとんど同じである．ただこの場合，アーク順は逆にして計算した方がよい．これは実際の場合，アーク順は topological ordering された状態に近いものと思われるからである．

以上のようにＰＥＲＴ計算のプログラムは簡単である．ノード番号が一連番号でないときは単に番号の付け換えのみでことはすむ．

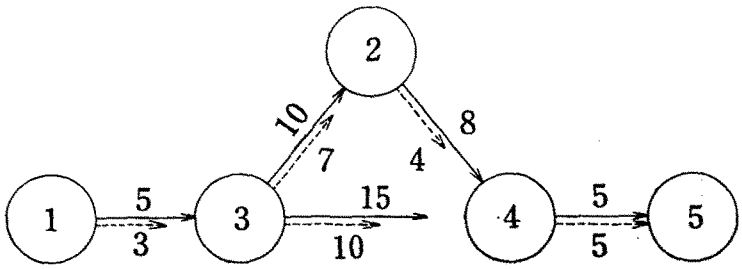
5. CPM 計算の例

ＣＰＭの計算にもＰＥＲＴの計算方法を使える．前述と同じ第２表，第７図に示すネットワークを例にして説明する．

最初のＰＥＲＴ計算における最早時刻の計算は前述した．かくて第８図に示すようなクリティカルパスを得る．短縮すべき工程をみつけるには費用勾配を capacity とする最大フローの問題を解けばよい．そのためアーク順を追ってラベリングをする．

---

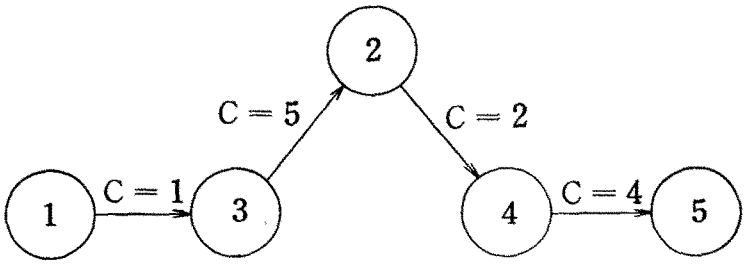
1) 簡単のため終点のノード番号をＮとしている。



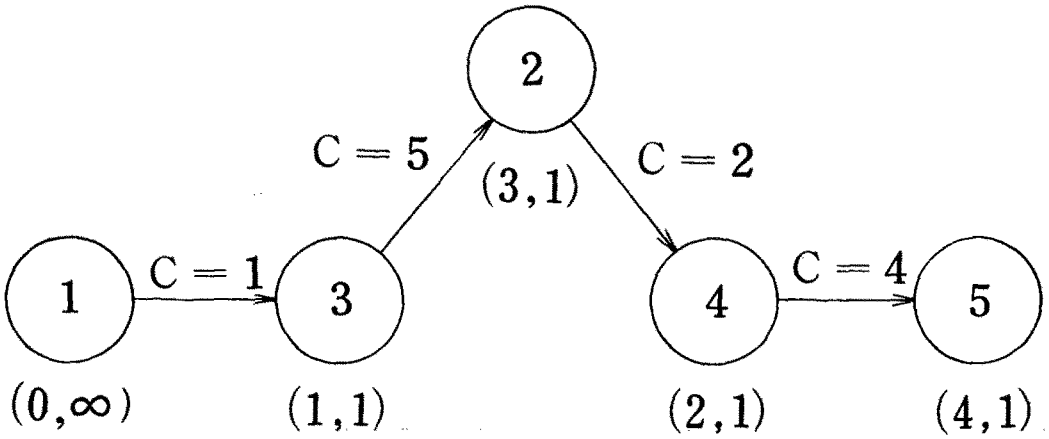
第 7 図

第 2 表

工 程	*i*	*j*	*Dij*	*dij*	*Cij*
a	1	3	5日	3日	1万/日
b	2	4	8	4	2
c	3	4	15	10	3
d	3	2	10	7	5
e	4	5	5	5	—



第 8 図



第 9 図

(a) については③のノードへ①よりフロー1が流せるゆえ③にラベル (1, 1) がつく[2)].

(b), (c) にはラベリングなし.

(d) については②のノードへ③より Max (1, 5) を流せるゆえ②にラベル (3, 1) をつける.

(e), (a) にはラベリングなし.

(b) については④のノードへ②より Max (1, 2) を流せるゆえ④へラベル (2, 1) がつく.

(c), (d) はラベリングなし.

(e) については⑤にラベル (4, 1) がつき，結局フローは1だけ流せることになる.

このようにCPM計算もすべてアーク順を追っての計算にすることができる.

6. 計算時間についての考察

ここで述べた計算法を用いるときの計算時間は，最初アークがどのような順序で計算機の中に並んでいるかによってくる．すなわちノード番号が topological ordering されとたきのアーク先行点番号の小さい方が先になるようにアークが並んでいるときが一番計算時間は少ないわけである．このようなアークの並び方をアークの topological ordering と呼べば，問題は計算のはじめにアークの topological ordering をするか，あるいはこれをしないでいきなり本計算を始めるかのどちらが有利かということである．実際問題としてはアークの並び方は大体 topological ordering された状態に近いものと思われるので計算時間は心配することはなく，プログラムが簡単になった分だけ有利ではないかと思われる．しかし問題によってはPERTの繰返し計算が非常に多いいものもあるわけで，そのときは計算時間が問題となろう．いずれにしても実際の計算例で検討すべき問題であるが，以下一つの実験例を示す.

アーク数200，ステップ数15程度のCPM計算について，アークが topological ordering された状態より計算したところ1分で終了した．次にデータカードの順序をかえてアーク順がランダムになるようにして計算したところ3分かかった．実際問題としてはこれ程悪いアークの並び方はしていないであろうから，この例については計算時間を心配する必要はないであろう.

最初に topological ordering する場合も，ここで述べた計算法を用いると1回の「すべてのアークについての計算」ですむので有効である.

7. まとめ

PERT CPM 計算について，プログラムが簡単でしかも計算時間の長くならない一つの手法を提案した．しかしこれに似た考えは已に文献にも見られ[6]，已存の理論を易しく解説したというような面もあるかも知れないが，ここでは自分で PERT CPM 計算のプログラムを作製して，そのとき考えたことを報告した次第である.

---

2) (a, 1) なるラベルがより便利である。

最後に電子計算機使用に当ってお世話になった九州大学計数施設や九州電力株式会社の関係者の方々にお礼申しあげます．また資料を頂きました電力中央研究所の中川友康氏に感謝の意を表します．

参 考 文 献

1. 近藤一夫（編），*RAAG Memoirs*, Vol 3, 学術文献普及会，1962.
2. Ford, L. R. Jr., and D, R, Fulkerson, *Flows in Networks*, Princeton University Press, 1962.
3. Kelley, Jr., J. E., Critical-Path Planning and Scheduling: Mathematical Basis, *Oper. Res.*, **9** (1961), 296—320.
4. 関根智明，PERT•CPM 入門，日科技連出版，1965.
5. 五百井清右衛門，ネットワークプランニング，日刊工業新聞社，1964.
6. Lass, S. E. PERT Time Calculation without Topological Ordering, *Comm. ACM*, **8** (1965), 172—174.