論　文

コード発生方式を用いた最短径路計算プログラム*

後藤　敏**　大附辰夫**　工藤安夫**

Abstract

An efficient shortest path calculation method suitable for real-time processing is presented. The algorithm used here is the one which has an algebraic analogy with the Crout elimination method for solving linear algebraic equations. By symbolic processing a computer program generates another program which represents the optimal reduced Crout algorithm in the sense that only non-trivial operations required for a given particular network structure are executed. Furthermore the generated program inculudes no DO-loop and requires no address calculation. The method proposed here, which may be called a shortest path version of the "code generation technique", is particularly powerful when a network of fixed sparseness structure must be solved repeatedly with different numerical values. In example networks, the execution of the generated program was observed to be five-to eight-fold as fast as that of existing shortest path programs.

1. まえがき

最短径路問題は通信網の設計問題，輸送問題，LSIの配線問題，自動車の径路誘導問題など各種のネットワーク問題の部分問題として重要な役割を演じている．これらの問題における大きな特徴として，ネットワーク構造を固定して，枝の重みだけを変化させて，何度も最短径路問題を解くことがあげられる．特に，自動車の径路誘導問題ではオンラインで最短径路問題を解く必要性から，(i)前処理に時間がかかっても，繰り返し行われる計算の実行時間は最大限に短縮すること，(ii)枝の重みに無関係に計算実行時間が定まることなどが要求される．

最短径路問題に対してのアルゴリズムとして，ポテンシャル法[1]，ダイキストラ法[2]などが良く知られており，最近は計算時間を短縮すべく，アルゴリズムの部分的改良やデータ構造，プログラミングの工夫などに関連した研究が広く行われている[3)~6)]．しかしこれらの方法は1回だけ最短径路計算を解く場合の計算時間を評価の基準にしており，ネットワーク構造を固定して，同じ計算を何回も繰り返す場合に適したものではない．また，計算実行時間が枝の重みに依存するという欠点を持っており，オンラインのシステムに応用するのは不適当である．

ところで，微分方程式の解析や統計解析において一定構造の線形方程式のパラメータを変えて，何回も繰り返し解く必要がある場合に，有利な方法として提案されたものに，コード発生方式がある[7]．コード発生方式はプログラムを実行する前に線形方程式の構造によって決まる演算順序に従って，計算に必要なコードを発生させておいて，繰り返し計算の実行時間を最大限に短縮させることを狙いとしている．すなわち，判定文や配列のアドレス計算（配列の要素の格納場所を探すに要する計算）を含まず，与えられたネットワーク構造において必要な演算だけが実行されるようなプログラムを前処理で発生しておくという方式である．

筆者らは線形方程式で提案されたコード発生方式を最短径路問題に適用し，ネットワーク構造が固定されている下で，枝の重みだけを変化させて，最短径路問題を繰り返し解くためのプログラムを作成したところ，計算実行時間が大幅に短縮できたので，その結果を報告する次第である．

---

* A Shortest Path Calculation Program based on Code Generation Technique by Satoshi GOTO, Tatsuo OHTSUKI and Yasuo KUDO (Central Research Laboratories, NiPPon Electric Company, Limited.)

** 日本電気(株)中央研究所

2. 最短径路問題

2.1 定　義

本論文においては，有向グラフ G をそれぞれ**節点**，**枝**とよばれる有限集合 V, E の順序対と考え，これを $G=(V, E)$ で表現する．ここで E の各々の元は2つの節点 $u, v \in V$ の順序対 $e=(u, v)$ であり，u, v をそれぞれ e の**始点**，**終点**とよぶ．ここで対象とするグラフ G はセルフループおよび同じ向きの並列枝を含まないものとする．グラフ G の節点の数を $n=|V|$，枝の数を $m=|E|$ とする．

最短径路問題においては有向グラフ $G=(V, E)$ の各々の枝 $e \in E$ に対して，非負の値*をもつ重み $w(e)$（枝 e の**長さ**とよぶ）が与えられたネットワークを対象とする．節点の順序集合 $P=\{v_1, v_2, \cdots, v_l\}$ において，

$$(v_i, v_{i+1}) \in E\,;\ i=1, 2, \cdots, l-1$$

が成立するとき，P を v_1 から v_l に至る**径路**とよぶ．また

$$w(P)=\sum_{i=1}^{l-1} w(e_i)\,;\ e_i=(v_i, v_{i+1}) \qquad (1)$$

を P の**長さ**とよぶ．節点の順序対 (s, t) に対して s から t に至る径路の集合を $P(s, t)$ で表わす．ある径路 $P \in P(s, t)$ において，

$$w(P) \leqq w(P') \qquad (2)$$

がすべての $P' \in P(s, t)$ に対して成立するとき，P を s から t に至る**最短径路**，$w(P)$ を s から t に至る**距離**とよぶ．

有向グラフ $G=(V, E)$ において節点の集合を $V=\{v_1, v_2, \cdots, v_n\}$ とすれば，各々の枝の長さは

$$a_{ij}=\begin{cases} w(e_{ij})\,;\ i \neq j \text{ かつ } e_{ij}=(v_i, v_j) \in E \\ \infty \quad\ ;\ i=j \text{ あるいは } (v_i, v_j) \notin E \end{cases} \qquad (3)$$

で定義される正方行列 $A=\{a_{ij}\}$ で表現できる．以後これを**隣接行列**とよぶこととする．ここで∞という記号は特定の節点対の間に枝が存在しないことを表現する意味で導入されたものである．しかし本論文で提案する最短径路プログラムは A のスパース性**を利用したものであり，∞要素は計算機メモリーの中に記憶しないことに注意されたい．一方，隣接行列 A で与えられるネットワークに対して，$n \times n$ の行列 $X=\{x_{ij}\}$ の各要素 x_{ij} が節点 v_i から節点 v_j に至る距離を与えるとき，X をこのネットワークの**距離行列**という．ここで X の対角項は0であることと考える．

更に，各々の節点対 $(v_i, v_j)\,;\ i \neq j$ に対して，v_i から v_j に至る最短径路上で v_j の直前の節点を $v_k\,;\ k \neq j$ とすれば

$$x_{ij}=x_{ik}+x_{kj}=x_{ik}+a_{kj} \qquad (4)$$

という関係が成立する．この関係を $r_{ij}=k$ で表わすものとすれば，各々の節点対間を結ぶ最短径路は行列 $R=\{r_{ij}\}$ によって表現することができる．ここで $r_{ij}=0$ とする．以後 R を**最短径路行列**とよぶこととする．例えば $r_{12}=7$, $r_{17}=3$, $r_{13}=1$ ならば，節点 v_1 から v_2 に至る最短径路は $P_{12}=\{v_1, v_3, v_7, v_2\}$ であることを示す．

以後，**最短径路問題**とは A を与えて，X と R の指定されたいくつかの行または列を求める問題***と解釈する．

2.2 アルゴリズムの評価

最短径路を求めるアルゴリズムとしては，ポテンシャル法（**付録1**参照）とダイキストラ法（**付録2**参照）が良く知られているが，これらはラベリング法というクラスに属すると考えられる．一方，連立一次方程式解析や逆行列計算のための手法として，ガウス法，クラウト法，ガウスジョルダン法など掃き出し法という名前で総称される手法が広く用いられているが，**3.**で述べるように演算の基本操作を適当に変更すればこれらに対応した最短径路アルゴリズムが得られる[8)~10)]．特にガウスジョルダン法に対応した最短径路アルゴリズムはフロイド法[11)]という名前で広く知られている．これらのアルゴリズムの能率を演算回数の上限で比較した結果が **Table 1** である．

Table 1 Number of operations in shortest path algorithm

アルゴリズム＼演算		比　較	加　算
ラベリング法	ポテンシャル法	n^3q	同　左
	ダイキストラ法	$3n^2q$	$\frac{1}{2}n^2q$
掃き出し法	ガ　ウ　ス　法	$\frac{1}{3}n^3+\frac{1}{2}n^2q+\frac{1}{6}q^3$	同　左
	ク　ラ　ウ　ト　法	〃	〃
	ガウスジョルダン法	$\frac{1}{2}n^3+\frac{1}{2}nq^2$	同　左

n: 節点の数 $n \gg 1$ と仮定する．
q: 始点の数

---

* 負の長さをもつ枝があっても，有向閉路に沿っての枝の長さの代数和が非負であれば，本論文で使用するアルゴリズムは適用できるが，議論を簡単にする意味で，枝の長さは非負とする．

** 連立一次方程式問題においては，0要素の多い行列をスパース行列とよぶが[12)]，最短径路問題では∞要素の多い行列をスパース行列とよぶ．

*** 例えば，X の i 行（j 列）はすべての節点 v_k, $k=1, 2, \cdots, n$ に対して $v_i(v_k)$ から $v_k(v_j)$ に至る距離を与える．

Table 1 の結果によれば，$q \ll n$ の場合はダイキストラ法が最良であり，$q=n$ の場合は掃き出し法が最良であることがわかる．一方ポテンシャル法は各々の枝の長さにばらつきが余りないときに有利なものであり，計算時間の平均値の意味で見積れば能率良いことが確かめられている[6]．

ところで，ポテンシャル法は付録1の操作 1) で行う枝 (v_i, v_j) の探索結果により，次に行う演算が決まる．また，ダイキストラ法は付録2の操作 2) で行う最小値選択の結果により，次に行う演算が決まる．すなわち，この2つの方法はネットワーク構造が同じでも，枝の長さにより演算順序が異なり，ネットワーク構造を固定して，枝の長さを変えて何回も解く問題には向かない．一方，掃き出し法は **3.** で述べるように，演算順序がネットワーク構造だけにより決まり，コード発生方式に適しているということができる．

3. 最短径路問題におけるコード発生方式

最短径路アルゴリズムにおける基本演算は，2つの非負の数 x, y に対する比較と加算であるが，便宜上これらをそれぞれ

$$x \oplus y = \min\{x, y\} \tag{5}$$

$$x \otimes y = x + y \tag{6}$$

で定義される演算記号で表わすものとする．ここで x, y の一方または両方が実数値ではなく ∞ という記号であることがあるが，その場合は

$$x \oplus \infty = x \tag{7}$$

$$x \otimes \infty = \infty \tag{8}$$

などと自然な解釈をするものとする．

一方，逆行列計算の掃き出しを基本とした解法における加算を⊕に，乗算を⊗に対応させれば，そのまま最短径路アルゴリズムとなることが知られている[8),9)]．ただし，最短径路問題では四則演算と異なり，⊕，⊗の逆演算は定義できないことに注意されたい．この対応関係を念頭におけば，式(7)，(8)及び

$$x \otimes 0 = x \tag{9}$$

が成立するという意味で，最短径路計算における∞，0はそれぞれ，逆行列計算における 0，1 に相当するものと考えられる．

さて，節点 v_i から v_j に至る距離 x_{ij} は

$$x_{ij} = \begin{cases} \min\limits_{k=1,2,\cdots,n} \{x_{ik} + a_{kj}\} & ; i \neq j \\ 0 & ; i = j \end{cases} \tag{10}$$

という関係を満たす．これを上記の演算記号を用いた行列表現をすれば，

$$X = A \otimes X \oplus I_n \tag{11}$$

という関係が導かれる[8)]．ここで I_n は対角項が 0，その他の要素が∞である n 次正方行列で，最短径路演算における単位行列と考えられる．

Table 1 において $q=n$ の場合，すなわち全節点間最短径路問題に対しては，掃き出し法に属する3つの解法の能率は全く同じであることがわかる．ところが $q<n$ の場合には，ガウスジョルダン法は演算回数の面でも，スパース性の保存の面でも他の2つに比べて劣っている．一方ガウス法とクラウト法は理論的な演算回数，スパース性の保存に関しては等価であるが，クラウト法は内積演算表現ができるため，計算を実際に行うときには，レジスターの有効な活用が可能であり，ガウス法に比べて計算時間が短縮できる可能性がある．よって，ここではクラウト法に対応した最短径路アルゴリズムを採用することとした．

逆行列計算との対応関係を念頭におけば，(11)式をクラウト法で逆行列を求める解法に対応した最短径路問題の解法を導くことができ，(12)式が得られる[12)]．

$$X = \prod_{i=1}^{n-1} U^{(i)} \otimes \prod_{i=1}^{n-1} L^{(n-i+1)} \otimes I_n \tag{12}$$

ここで $U^{(i)}, L^{(n-i+1)}$ は

$$U^{(i)} = \begin{bmatrix} 0 & & & & & \\ & 0 & & & \infty & \\ & & 0 & u_{i,i+1} & u_{i,i+2} \cdots & u_{i,n} \\ & & & \ddots & & \\ & & & & \ddots & \infty \\ & & \infty & & \ddots & \\ & & & & & 0 \end{bmatrix} \tag{13}$$

$$L^{(n-i+1)} = \begin{bmatrix} 0 & & & & & \\ & 0 & & & & \\ & & \ddots & & \infty & \\ & & & \ddots & & \\ & \infty & & & \ddots & \\ & & & & & \\ l_{n-i+1,1} & l_{n-i+1,2} \cdots & l_{n-i+1,n-i} & \cdot & & \\ & & \infty & & & 0 \end{bmatrix} \tag{14}$$

という構造をもっている．

(12)式は逆行列計算における Elimination Form of Inverse[13)] に対応している．逆行列計算との対応から，ここでは $U^{(i)}, L^{(n-i+1)}$ の各要素を求める計算を **3** 角分解計算とよび，$\prod_{i=1}^{n-1} L^{(n-i+1)} \otimes I_n$ を求める計算を前

進代入計算とよび，前進代入計算の結果を用い，(12)式よりXを求める計算を後退代入計算とよぶ．Xの指定された行及び列に無関係に，3角分解計算を行う必要があるが，前進代入計算及び後退代入計算はXの指定された行及び列に関してだけ行えばよい．

以下クラウト消去法に基づいて$X=\{x_{ij}\}$を求める手順を記述する．ただし，$R=\{r_{ij}\}$を求める操作については省略する．これを求めるには，各々のr_{ij}についてv_iからv_jに至るより短い径路が得られるたびに，v_jの直前の節点v_kの番号を$r_{ij}=k$と更新する操作を加えておけば良い．

(i) 3角分解計算

① 初期値設定

$$\left.\begin{aligned} l_{ij}&=a_{ij}\\ u_{ji}&=a_{ji}\end{aligned}\right\};\ j=1,2,\cdots,n-1;\ i=j+1,j+2,\cdots,n \tag{15}$$

② 第kステップ ($k=2,3,\cdots,n-1$)

$$\left.\begin{aligned} l_{ik}&=l_{ik}\oplus\sum_{j=1}^{k-1}l_{ij}\otimes u_{jk}\\ u_{ki}&=u_{ki}\oplus\sum_{j=1}^{k-1}l_{kj}\otimes u_{ji}\end{aligned}\right\};\ i=k+1,k+2,\cdots,n \tag{16}$$

次に，Xのh列だけを求めることが指定されたとする*．Xのh列を表わすベクトルを$\boldsymbol{x}^{(h)}$とし，h番目の要素が0で他は∞であるn次元ベクトルを$\boldsymbol{b}^{(h)}$とすれば，前進代入計算は

$$\boldsymbol{y}^{(h)}=\prod_{i=1}^{n-1}L^{(n-i+1)}\otimes\boldsymbol{b}^{(h)} \tag{17}$$

を求めることとなり，後退代入計算は

$$\boldsymbol{x}^{(h)}=\prod_{i=1}^{n-1}U^{(i)}\otimes\boldsymbol{y}^{(h)} \tag{18}$$

を求めることとなる．ここで前進代入計算をするとき，$\boldsymbol{b}^{(h)}$の性質より，$l_{ij}(j<h)$に対しては(17)式の計算を行う必要がないことに注意されたい．

$\boldsymbol{x}^{(h)}$, $\boldsymbol{y}^{(h)}$, $\boldsymbol{b}^{(h)}$のi番目の要素を$x_i^{(h)}$, $y_i^{(h)}$, $b_i^{(h)}$とすれば，次の計算を行うこととなる．

(ii) 前進代入計算

① 初期値設定

$$y_i^{(h)}=b_i^{(h)};\ i=1,2,\cdots,n \tag{19}$$

② 第kステップ ($k=h,h+1,\cdots,n$)

$$y_k^{(h)}=y_k^{(h)}\oplus\sum_{j=h}^{k-1}l_{kj}\otimes y_j^{(h)} \tag{20}$$

(iii) 後退代入計算

① 初期値設定

$$x_i^{(h)}=y_i^{(h)};\ i=1,2,\cdots,n \tag{21}$$

② 第kステップ ($k=1,2,\cdots,n-1$)

$$x_{n-k}^{(h)}=x_{n-k}^{(h)}\oplus\sum_{j=n-k+1}^{n}u_{n-kj}\otimes x_j^{(h)} \tag{22}$$

Xの指定されたq個の列を求めるに必要な演算回数の上限は，3角分解計算に$1/3n^3$，後退代入計算に$1/2n^2q$であり，前進代入計算は指定されたq個の節点を行列Aのできるだけ番号の大きな行及び列に対応させれば演算回数は最小となり，$1/6q^3$が得られる．

本論文で提案するコード発生方式では，与えられたグラフ構造（行列Aにおける非∞要素の位置）に基づいて，上記の計算手順のシミュレーションを行い，無駄な計算（$x\oplus\infty=x$, $x\otimes\infty=\infty$ など）を全て省き，かつ実行時における配列のアドレス計算を含まないようなプログラムを予め発生しておく．同じグラフ構造について，枝の長さを変化させて何回も最短径路問題を解く場合には，上記の前処理段階で発生されたプログラムを実行することによって，演算時間の大幅な短縮が期待できる．

さて前記のクラウト消去法は，隣接行列Aを行（列）番号の順に処理して行くものであり，グラフ$\boldsymbol{G}$の節点と行列Aの行（列）番号との対応のさせ方，すなわち**ピボット順**によってその手順が異なる．行列Aがスパースである（∞要素を多く含む）場合には，このピボット順は3角分解の結果表われる非∞要素の数に影響する．従って演算回数，発生されるコードの長さもピボット順に依存する．そこでコード発生方式においては，前処理の段階で上記の意味で最適ピボット順を求めることも重要な問題となる．

4. コード発生方式におけるプログラム構成

プログラムは大きく分けて，コードを発生する処理とコードを実行する処理の2つからなる．コード発生処理では，まずグラフ構造と最短径路を求める始点を与えて，発生するコードの長さ，すなわち計算実行時間をできるだけ短くするようなピボット順を決定する．次に，グラフ構造に従い，3角分解の計算に必要なコードを発生し，更に最短径路を求める始点に対して，前進代入計算と後退代入計算に必要なコードを発生させる．ここでは発生するコードの言語としては，FORTRAN を用いた．コード実行処理では与えられた枝の長さに従って，発生されたコードを実行し，最短径路を求める．なおここで使用した計算機は，NEAC 2200/M500 である．

* 行を求めることが指定された場合は，以後の議論で行列 A のかわりに A の転置行列を考えれば，全く同じことがいえる．

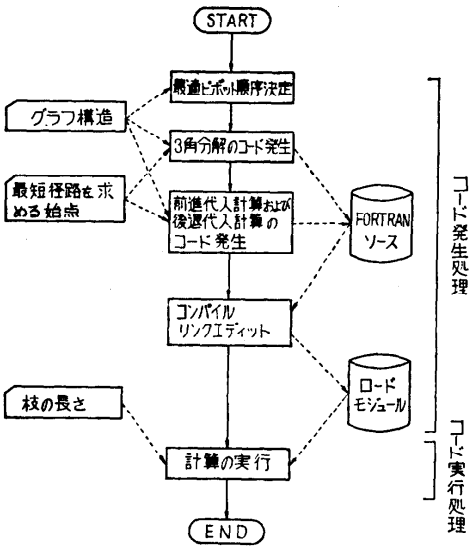


Fig. 1 Code generation method

4.1 コード発生処理

（i） 最適ピボット順序の決定

発生するコードを短くするためには，グラフ G の節点に1から n までの番号をふる際に，

1) 前進代入計算を省略するには，最短径路を求める始点にはできるだけ大きな番号をふる．

2) 3角分解後の非∞要素数をできるだけ少なくするようにする．

ことを考えねばならない．1) と 2) とを同時に考え，最適ピボット順序を決定することは困難なため，ここでは 1) に基づいて始点となる節点にはできるだけ大きな番号をふり，その後で 2) の処理を行うこととした．

2)は連立一次方程式における最適対角ピボット順序問題と同じ問題となる．最適ピボット順を求めるためのアルゴリズムは種々提案されているが[14)~16)]，厳密な最適解を求めるための能率良い算法は知られていない．ここでは，A が非対称（Gが有向グラフ）の場合に近似最適解を比較的簡単な操作で求められるマルコヴィッツの方法[17)]を用いた．

（ii） 3角分解計算のコード発生

(15)，(16)式によって3角分解計算に必要なコードを発生させる．この処理は最短径路を求める始点の数によらず，1度だけ行う．

（iii） 前進代入計算及び後退代入計算のコード発生

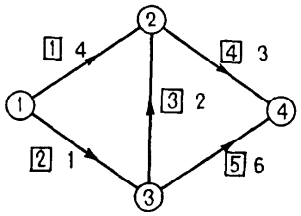
(19)～(22)式によって代入計算に必要なコードを発生させる．この処理は最短径路を求める始点ごとに行う．

4.2 コード実行処理

発生されたコード (FORTRAN 文) では，配列の添字は定数で表現されており，コンパイル時においてデータの格納場所が決定するために，計算の実行時においては配列のアドレス計算はしない．また，発生されたコード内には DO ループもないため，計算の実行時間は極めて速くなる．グラフ構造を固定して，枝の長さを変えて何回も解くときには，コード実行処理だけを繰り返し解けばよい．

4.3 コード発生の例題

Fig. 2 において節点①から他のすべての節点への最短径路を求める．Fig. 2 の枝の向きをすべて逆にし



(i)：節点番号
[i]：枝番号
i：枝の長さ

Fig. 2 An example network

たグラフで，すべての節点から節点①への最短径路を求めることとなり，このとき行列 A は

$$A=\begin{bmatrix} \infty & \infty & \infty & \infty \\ 4 & \infty & 2 & \infty \\ 1 & \infty & \infty & \infty \\ \infty & 3 & 6 & \infty \end{bmatrix} \tag{23}$$

となる．A を3角分解した結果，$L=\{l_{ij}\}$, $U=\{u_{ij}\}$, $R=\{r_{ij}\}$ の行列はそれぞれ

$$L=\begin{bmatrix} 0 & & & \\ 4 & 0 & \infty & \\ 1 & \infty & 0 & \\ \infty & 3 & 5 & 0 \end{bmatrix} \tag{24}$$

$$U=\begin{bmatrix} 0 & \infty & \infty & \infty \\ & 0 & 2 & \infty \\ & & 0 & \infty \\ \infty & & & 0 \end{bmatrix} \tag{25}$$

$$R=\begin{bmatrix} 0 & & & \\ 1 & 0 & 3 & \\ 1 & & 0 & \\ & 2 & 2 & 0 \end{bmatrix} \tag{26}$$

となる．よって前進代入計算により，行列Rの第1列$\boldsymbol{r}^{(1)}$及び$\boldsymbol{y}^{(1)}$は，

$$\boldsymbol{y}^{(1)}=\begin{bmatrix}0 & & & \infty\\ \infty & 0 & & \\ \infty & \infty & 0 & \\ \infty & 3 & 5 & 0\end{bmatrix}\otimes\begin{bmatrix}0 & & & \infty\\ \infty & 0 & & \\ 1 & \infty & 0 & \\ \infty & \infty & \infty & 0\end{bmatrix}\otimes\begin{bmatrix}0 & & & \infty\\ 4 & 0 & & \\ \infty & \infty & 0 & \\ \infty & \infty & \infty & 0\end{bmatrix}\otimes\begin{bmatrix}0\\ \infty\\ \infty\\ \infty\end{bmatrix}$$

$$=[0\ \ 4\ \ 1\ \ 6]^T \tag{27}$$

$$\boldsymbol{r}^{(1)}=[0\ \ 1\ \ 1\ \ 2]^T \tag{28}$$

となる．後退代入計算により，$\boldsymbol{x}^{(1)},\boldsymbol{r}^{(1)}$は

$$\boldsymbol{x}^{(1)}=\begin{bmatrix}0 & \infty & \infty & \infty\\ & 0 & 2 & \infty\\ & & 0 & \infty\\ \infty & & & 0\end{bmatrix}\otimes\begin{bmatrix}0\\ 4\\ 1\\ 6\end{bmatrix}$$

$$=[0\ \ 3\ \ 1\ \ 6]^T \tag{29}$$

$$\boldsymbol{r}^{(1)}=[0\ \ 3\ \ 1\ \ 2]^T \tag{30}$$

となり，節点①からの最短径路が求められた．

コード発生処理によって発生されるコードは**Fig. 3**となる．Fig. 3において各データは以下で示される1次元配列によって表現されている．

$A(i)$: 枝iの長さ（行列Aに対応）

$R(i)$: 行列Rに対応

$X(i)$: 節点①から節点iへ至る距離（ベクトル$\boldsymbol{x}^{(1)}$に対応）

$S(i)$: 節点①からの最短径路を表わす．($\boldsymbol{r}^{(1)}$に対応)

```
C     LU-DECOMPOSITION
      VLU = A(3) + A(4)
      IF(VLU.GE.A(5)) GO TO 10
      A(5) = VLU
      R(5) =R(3)
C     INITIAL SET FOR SUBSTITUTIONS
   10 S(1) = 0
      X(1) = 0
      X(2) = 30000
      X(3) = 30000
      X(4) = 30000
C     FORWARD SUBSTITUTIONS
      X(2) = A(1)
      X(3) = A(2)
      S(2) = R(1)
      S(3)=R(2)
      VLU = A(3) + X(2)
      IF(VLU.GE.X(4)) GO TO 20
      X(4) = VLU
      S(4) = R(3)
   20 VLU = A(5) + X(3)
      IF(VLU.GE.X(4)) GO TO 30
      X(4) = VLU
      S(4) = R(5)
C     BACKWARD SUBSTITUTIONS
   30 VLU = A(4) + X(3)
      IF(VLU.GE.X(2)) GO TO 40
      X(2) = VLU
      S(2) = R(4)
   40 CONTINUE
      RETURN
      END
```

Fig. 3 An example of generated codes

入力データとして，$A(1)=4$, $A(2)=1$, $A(3)=2$, $A(4)=3$, $A(5)=6$, 及び$R(1)=1$, $R(2)=1$, $R(3)=2$, $R(4)=3$, $R(5)=3$が与えられている．なお入出力文，宣言文については省略してある．

5. コード発生方式の評価

コード発生方式によって，どの程度計算実行時間が短縮されたかを計算機実験によって評価した結果を以下に示す．比較したプログラムは次のものである．

$P1$: 本文で提案したコード発生方式のプログラム

$P2$: $P1$と同じアルゴリズムを用い，コード発生を行わずに実行時にスパース処理を行ったもの．すなわち，判定文，DOループを含み，実行時に配列のアドレス計算を行う[12]．

$P3$: ダイキストラ法において（付録2），集合Tに属する節点の値が小さい順に取り出せるようなリスト構造を作成したプログラム[3)-6)]

$P4$: ポテンシャル法において（付録1），すべての節点v_j ($j=1,2,\cdots,n$)に対して，$x_i^{(h)}+a_{ij}\geq x_j^{(h)}$となる節点$v_i$を集合$V$から除いた節点だけが取り出せるようなリスト構造を作成したプログラム[6]

ここでコード発生方式における計算時間は，コード発生処理に要した時間は含まず，コード実行処理に要

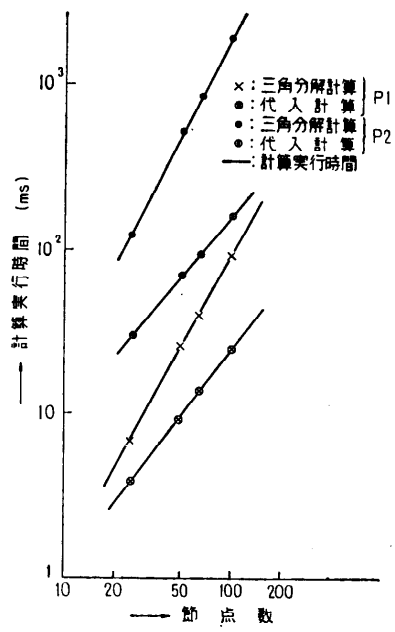


Fig. 4 Computation time to find shortest paths from a specified vertex to all vertices

した時間だけである．なお対象としたネットワークは碁盤型構造をもち，1から10までのランダムな整数値が枝の長さとして付加されている．

（i） 1点から全点への最短径路計算を実行するに必要な計算時間を $P1$ と $P2$ とで比較した結果が **Fig. 4**（前頁参照）である．$P1$ は $P2$ と比較し，三角分解計算で約 1/20，代入計算で約 1/7 で計算実行可能である．

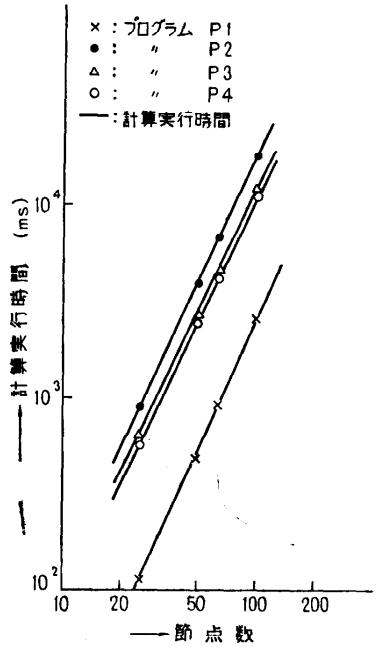


Fig. 5 Computation time to find all shortest paths

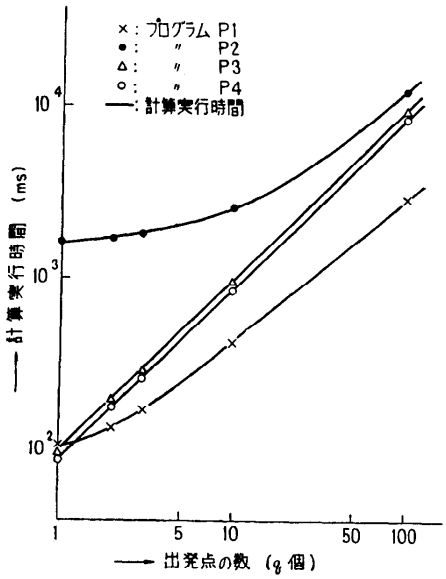


Fig. 6 Computation time to find shortest paths from q specified vertices to all vertices

（ii） 全点から全点への最短径路計算を実行するに必要な計算時間を $P1$〜$P4$ に関して比較した結果が **Fig. 5** である．$P1$ は $P2$ と比較し，1/7〜1/8 の時間で計算実行でき，また，$P3$ 及び $P4$ と比較して実行時間が 1/5〜1/6 となっている．

（iii） 100個の節点からなる碁盤型ネットワークにおいて，q 個の節点から全点への最短径路計算を実行するに必要な時間を $P1$〜$P4$ に関して比較した結果が **Fig. 6** である．$q=1$ のときのみ，$P1$ は $P2$ 及び $P3$ に対して劣るが，$q\geq 2$ に対してはコード発生方式が最も短時間に計算できることがわかる．

上記の計算実験結果の示すように，コード発生方式の導入によって，最短路計算の実行に要する C. P. U. 時間を大幅に短縮できる．しかし，$P1$ は $P2$〜$P4$ に比べて，発生されるコードが大きくなる（従ってメモリ使用量が大となる）という欠点を持つ．$P1$ における使用メモリー量は発生されたコードの長さ（従ってコードの実行時間）に比例し，$P2$〜$P4$ に比べてはるかに大きなメモリが必要となる*．一般にコード発生方式の導入が有利となるのは，「一定構造のネットワークにおいて，枝の長さを変更しながら何回も最短径路計算を行う」という前述の仮定に加えて，下記のいずれかの前提条件が満たされている場合である．

a）十分大きな内部メモリを持つ計算機の使用が可能であり，最短径路計算の実行時間には苛酷な要求が課せられている場合．

b）コードを十分長いセグメント単位に分割したときセグメント当たりのデータ転送時間が実行時間よりも長くない場合．この場合，コードの実行を終了するまでの経過時間は内部メモリが十分大きい場合の C. P. U. 時間にほぼ等しくなる**．

c）多重処理の能力を持つ大型計算機システムに加

---

* $P3$, $P4$ では与えられたグラフ G（行列 A の非∞要素の数）に比例する程度のメモリで十分である．$P2$ においては，新しく発生した非∞要素の記憶のためのエリアの分だけ $P3$, $P4$ よりも大きなメモリを必要とするが，$P1$ に比べればはるかに小さなメモリで十分である．

** 例えば，内部メモリを二つのエリア A, B に分割しておいて，エリア A に貯えられている k 番目のセグメントを実行している間にエリア B に $(k+1)$ 番目のセグメントを読み込み，k 番目のセグメントの実行が終了したときにコントロールをエリア B に移す，という操作を組み込んで，エリア A, B を交互に利用すれば良い．

わる一つのジョブとして最短径路計算が行われる場合.

この場合，このジョブが I/O (コードの読み込み) を行っている間は他のジョブが cpu を利用していると仮定すれば，プログラム $P1$ の実行コストはある定数 C_1, C_2 によって

C_1(C. P. U. 時間)+C_2(チャネル使用回数)

によって評価できると考えられる．このような使用状況においては，コードが長くなっても，これを適当にセグメントに分割してオーバーレイしておけば，チャネル使用によるコストは C. P. U. 時間によるコストに比べてはるかに小さい．

最短径路計算のコードを Fig. 1 に示すように FORTRAN などの高級言語として発生させると，かなりのコンパイル時間を要するという問題点もあるが，これは直接機械語を発生させることによって解決できる．

6. あ と が き

ネットワーク構造を固定した下で，枝の長さを変化させて，最短径路を何回も求める問題において，コード発生方式によって，計算実行時間を大幅に短縮できることが判明した．また，本文で提案した方法は枝の長さに無関係に計算実行時間が定まるので，自動車の径路誘導問題などオンライン・システムに利用するのに適した方法といえる．

コード発生方式を用いると C. P. U. 時間を大幅に短縮できるかわりに，発生されるコードが大きくなるという問題点が生じる．従ってこの方式が有効にできるためには，「内部メモリが十分大きい」，あるいは「外部メモリから内部メモリへのコードの転送時間を他の目的に利用できる」という使用形態が前提となる．応用によっては，プログラム $P1$ と $P2$ の中間的なものとしてプログラムのコードの代りに命令のアドレス表を発生させるという方式[18]の方が使用メモリ量が少ないという点で有利なこともある．いずれにしろ，最適なプログラミング方式は最短経路アルゴリズムを利用する目的に依存するということである．

謝辞．本研究は通産省自動車総合管制システムの関連基礎研究の一部として行なわれたものである．

参 考 文 献

1) H. Frank and I. T. Frish: Communication, transmission and transportation networks, pp. 192～198, Addison-Wesley, (1971).
2) E. W. Dijkstra: A note on problems in connection with graphs, Numerishe Mathematik, 1, p. 269 (1959).
3) 伊理(編): ネットワーク構造を有するオペレーションズ・リサーチ問題の電算機処理に関する基礎研究，報文シリーズ T-73-1，日本 OR 学会 (昭 48-03).
4) 富沢: 最短径路問題における Dijkstra 法の改良について，日本 OR 学会 1973 年度秋季研究発表会アブストラクト集，2-3-9.
5) E. L. Johnson: On shortest paths and sorting, Proc. ACM, pp. 510～517 (1972-8).
6) 後藤: スパース処理技法を用いた最短径路問題の解法，電子通信学会，回路とシステム理論研究会，CT 73-22 (1973-07).
7) F. G. Gustavson, W. Linger & R. Willoghby: Symbolic generation of an optimal Crout algorithm for sparse systems of linear equuations, J. ACM, Vol. 17, No. 1, pp. 87～109 (1970).
8) B. A. Carre: An algebra for network routing problems, J. Inst. Maths. Applics, Vol. 7, pp. 273～294 (1971).
9) M. Iri and M. Nakamori; Path-sets, operator semigroups and shortest-path algorithms on a network, RAAG Reseatch Notes, Third Series, No. 185, (Oct. 1972).
10) 片山，渡部: 通信網における最短路問題，昭 43 電気連合大会，No. 47.
11) R. W. Floyd: Algorithm 97-Shortest Path, C. ACM, 15, p. 345 (1962).
12) 後藤，大附: 全節点間最短径路問題のスペース処理プログラム，昭和 48 年度情報処理学会第 14 回大会.
13) R. P. Tewarson: Sparse matrices, Academic Press, New York (1973).
14) E. C. Ogbuobiri, W. F. Tinney and J. K. Walker: Sparsity directed decomposition for Gaussian elimination on matrices, IEEE Trans., PAS-89, 1, p. 141 (Jan. 1970).
15) T. Ohtsuki: A fast algorithm for finding an optimal ordering for vertex elimination of a graph, SIAM J. Comput., Vol. 5, No. 1 (March 1976).
16) 坂本，白川，尾崎: スパース連立方程式におけるピボットの順序づけ，情報処理，Vol. 13, p. 154 (昭 47).
17) H. M. Markowitzi: The elimination form of the inverse and its applicaton to linear programming, Manage. Sci. (1957-3).
18) 吉村，後藤，大附: アドレス表生成方式による最短路計算，昭和 50 年情報処理学会全国大会.

◇

付録 1 ポテンシャル法による最短径路問題の解法[1)]

節点 v_h より他のすべての節点への距離ベクトル $\boldsymbol{x}^{(h)}=(x_1^{(h)}, x_2^{(h)}, \cdots, x_n^{(h)})$ は

0) $x_h^{(h)}=0, x_i^{(h)}=\infty$ $(i\neq h)$ とする.

1) $\omega_j=x_i^{(h)}+a_{ij}$ として, $\omega_j<x_j^{(h)}$ を満たす枝 (v_i, v_j) を探す. $(i, j=1, \cdots, n)$. もしかような枝が見つかれば 2) へ行き, 見つからなければ操作は終る.

2) $x_j^{(h)}=\omega_j$ として 1)へ戻る.

により, 求められる. 演算回数の上限は比較と加算と同じで n^3 となる.

付録 2 ダイキストラ法による最短径路問題の解法[2)]

付録 1 の $\boldsymbol{x}^{(h)}$ は次の操作で求められる.

0) $V=T\cup P(T\cap P=\phi)$ なる集合 T と P とを定義し, $T=V, P=\phi$ とする. また, $x_h^{(h)}=0$, $x_i^{(h)}=\infty(i\neq h)$ とする.

1) $\min_{v_i\in T}\{x_i^{(h)}\}$ を満たす節点を v_{i0} とし, このときの値を x_{i0} とする. T 及び P を更新する.

$$P=P\cup\{v_{i0}\} \tag{31}$$

$$T=T-\{v_{i0}\} \tag{32}$$

2) T に属するすべての節点に対して,

$$x_j^{(h)}=\min\{x_j^{(h)}, x_{i0}+a_{i0j}\} \tag{33}$$

とする.

3) $T=\phi$ ならば操作終り, さもなければ 1) へ戻る.

以上の操作で $3n^2$ の比較と n^2 の加算が必要となる.

付録 3 ガウスジョルダン法による最短径路問題の解法[8)]

(11)式 $X=A\otimes X\oplus I_n$ を逆行列計算で知られるガウスジョルダン法で解けば,

$$X=\prod_{k=1}^{n} J_{n-k+1}\otimes I_n \tag{34}$$

となる. ここで $J_l=\{a_{ij}^{(l)}\}$ $(i, j=1, \cdots, n)$ とすれば, $a_{ij}^{(l)}$ は以下で表わせる.

$$a_{ij}^{(0)}=a_{ij} \tag{35}$$

$$a_{ij}^{(l)}=\begin{cases}(a_{il}^{(l-1)}\otimes a_{lj}^{(l-1)})\oplus a_{ij}^{(l-1)}; & j=l, \\ & \text{かつ } i\neq j \\ \infty; & j\neq l \text{ または } i=j\end{cases} \tag{36}$$

$$\begin{pmatrix} i, l=1, 2, \cdots, n \\ j=l+1, l+2, \cdots, n\end{pmatrix}$$

$J_l(l=1, 2, \cdots, n)$ を求める演算回数は $1/2n^3$ であり, 始点が q 個あるときの最短径路を求める積計算の演算回数は, q 個の始点を行列 A のできるだけ大きな行及び列番号に対応させることにより $1/2nq^2$ となる.

(昭和 50 年 6 月 21 日受付)
(昭和 51 年 4 月 2 日再受付)