Vol. 26 No. 2　　情報処理学会論文誌　　Mar. 1985

有向ネットワークにおいて閉路を含まない k 個の最短径路を求めるための手法†

杉本克行††　加藤誠巳†††

有向ネットワーク上で，閉路を含まない k 個の最短径路を求める新手法を発表する．本手法は，1個の木によって，終点までの閉路を含まない複数の径路を体系的に表現する．ところで，始点を固定して，ネットワーク内の各リンク長にある規則の変更を加えると，変更前と後で径路の距離の順位は変わらないとされている．ここでは，この原理を利用して効率よく木を育てることによって，目的径路を求める．本論文では，新手法のアルゴリズム，モデルネットワークによる具体例，格子状ネットワークにおける計算時間実測値，従来の手法との比較，を述べる．結論として，本手法は非常に効率がよく，実用上有力な手法であることがわかった．

1. まえがき

1番目だけでなく，2番目以降複数の最短径路を求める第 k 最短径路問題は，最短径路問題と同様に重要であり，なかでも，閉路を含めない第 k 最短径路問題は実用的な価値があり，いくつかの手法が提案されている．しかしながら，従来の手法は，計算時間がノード数 n の数乗と k の積に比例して増大し，実用上あまり効率がよいといえない．

そこで，今回遅小樹育法と名づけた手法を開発した．この手法は，リンクの遅延時間という概念と複数の径路を体系的に表すための木を導入することにより，アルゴリズムの効率化を行い計算時間の飛躍的向上を図ったものである．適用できるネットワークはリンク長が非負の無向および有向ネットワークであり，考え方はリンク長が負のネットワークにも応用できる．

2. 用語の定義

まず用語を定義しておく．

ノード：ネットワーク内の点のこと．ノード数が n のとき，各ノードを $1,2,\cdots,n$ と番号づける．

リンク：ノード間を結ぶ線のこと．リンクは方向と距離をもっている．リンク総数が m のとき各リンクを $1,2,\cdots,m$ と番号づけする．リンク x の始点を $O(x)$，終点を $S(x)$，距離を $d(x)(\geqq 0)$ と記す．

径路：径路は式(1)を満足するリンクの順列 $\boldsymbol{P}=\{x_1,x_2,\cdots,x_l\}(l\geqq 1)$ と考える．

$$S(x_i)=O(x_{i+1}) \quad (i=1,2,\cdots,l-1) \qquad (1)$$

$O(\boldsymbol{P})=O(x_1)$ を径路の始点，$S(\boldsymbol{P})=S(x_1)$ を径路の終点，$d(\boldsymbol{P})=\sum_{i=1}^{l} d(x_i)$ を径路の距離と定義する．

単純路：閉路を含まない径路を単純路と呼ぶ．

ノードの距離：ノード O_0 の距離は0とする．ノード $v(\neq O_0)$ の距離は，O_0 から v までの最短径路の距離のこと．これを $\pi(v)$ と記す．

リンクの時間：リンク x の時間 $\tau(x)$ は

$$\tau(x)=d(x)+\pi(O(x)) \qquad (2)$$

と定義される．

リンクの遅延時間：リンク x の遅延時間 $\delta(x)$ は

$$\delta(x)=\tau(x)-\pi(S(x)) \qquad (3)$$

と定義される．つまり，リンク x を通って $S(x)$ に着く径路の，最短の径路に対する遅れを示す．なお，この概念は従来から考えられている[5),10),18)]☆．

径路の遅延時間：径路 $\boldsymbol{P}$ の遅延時間とは $\sum_{i=1}^{l}\delta(x_i)$ のことを示す．これを $\delta(\boldsymbol{P})$ と記す．

第 k 単純路：O_0 から S_0 までの第 k 単純路 $(k\geqq 1)$ とは $O(\boldsymbol{P})=O_0, S(\boldsymbol{P})=S_0$ とするあらゆる単純路 $\boldsymbol{P}$ のうち，距離が k 番目に短いものを示す．ただし，同じ距離の単純路は任意に順位づけできるものとする．

単純路の木：ネットワーク上の S_0 までの単純路を体系的に表したものである．根以外の各節☆☆はネットワーク上のリンクに対応し，このうち根の息子は S_0 を終点とするリンクに対応する．節 i の対応リンクを λ_i と記す．節番号は根を1とし以後は発生した順序

† An Algorithm for Finding k Shortest Loopless Paths in a Directed Network by KATSUYUKI SUGIMOTO (Institute for Future Technology) and MASAMI KATOH (Department of Electrical-Electronic Engineering, Sophia University).

†† 未来工学研究所
††† 上智大学理工学部電気電子工学科

☆ たとえば，文献5)では Relative Cost，文献18)では Reduced Cost と名づけられている。

☆☆ ネットワーク上のノードと区別するため，木の点は節と呼ぶ。

とする．任意の節 i から根までさか上った径路は，ネットワーク上の一つの単純路を示す．この単純路を $\boldsymbol{P}_i$ と記す．つまり，i から1までの節を $i, p_2, p_3, \cdots, p_l, 1$ とすると，$\boldsymbol{P}_i$ は次のように定義される．

$$\boldsymbol{P}_i=\{\lambda_i, \lambda_{p_2}, \lambda_{p_3}, \cdots, \lambda_{p_l}\} \quad (4)$$

節 i の親子関係に対応して，$\boldsymbol{P}_i$ の親子関係も考える．なお，$\boldsymbol{P}_1$ を仮想単純路と考え，$\delta(\boldsymbol{P}_1)=0$ とする．

発芽リンク：$\boldsymbol{P}_i$ に対して，遅延時間が最小の息子を発生させることができるリンクを示し，β_i と記す．息子を発生できないときは0とする．

子孫遅延時間：次に発生できる，$\boldsymbol{P}_i$ の子孫の遅延時間の最小値を示し，$\delta_d(i)$ と記す．$\boldsymbol{P}_i$ の子孫を発生できないときは∞とする．

傍系遅延時間：次に発生できる，$\boldsymbol{P}_i$ の子孫以外の遅延時間の最小値を示し，$\delta_c(i)$ と記す．$\boldsymbol{P}_i$ の子孫以外を発生できないときは∞とする．

$f(v)$：単純路の木の探索中，各ノード v の通過状態を示すフラグである．現在の節から根までの間にノード v が出現していれば1，していなければ0である．

3. 遅小樹育法の原理

遅小樹育法は，まず全リンクの遅延時間を算出しておき，次はこの遅延時間を用いて単純路の木を育てる方法である．このためには以下の定理が基になっている．

まず，ネットワークは，従来から定理1の性質をもつとされている[5),10)]．

[定理1] $\boldsymbol{P}$ を O_0 から S_0 までの径路とすると，式(5)が成り立つ．

$$d(\boldsymbol{P})=\pi(S_0)+\delta(\boldsymbol{P}) \quad (5)$$

(証明略)

定理1は，遅延時間が小さい径路ほど距離が短い径路であることを示している．

よって，遅延時間が小さい順に節を発生しながら単純路の木を育てることにする．このとき，既成の木を辿りながら枝伸ばし地点を見つけ，枝を伸ばしたところからまた次の地点を探す，という操作を繰り返していく．

このとき，探索の効率化を図るために δ_d, δ_c および f を用いる．このためには以下の定理が基になっている．

[定理2] $\delta_c(i)=\delta_d(i)=\infty$ でなければ，通過する各節 i において，$\delta_c(i)$ と $\delta_d(i)$ を比較することによって，遅延時間最小の息子を発生できる節のいずれかに，到達できる．

(証明) $\delta_c(i)$, $\delta_d(i)$, $\delta(\boldsymbol{P}_i)+\delta(\beta_i)$ のうち $\delta(\boldsymbol{P}_i)+\delta(\beta_i)$ が最小の場合は，β_i を使って，遅延時間が最小の息子を発生できる．ところで，$\delta_c(i)<\delta_d(i)$ の間は上り続け，一度 $\delta_d(i)\leqq\delta_c(i)$ になったら下り続ければ，同じ節を2度通ることなく上記条件を満足する節 i に到達する．よって定理が成り立つ．(証明終)

以上の定理により本手法を実現するためには，木を探索中，通過する節の δ_c, δ_d を正しく設定する必要がある．以下の定理は設定できることを示している．

[補題1] まず，$f(S_0)=1$，その他のすべてのノード v については $f(v)=0$ とし，以降は節 i に下ってきたら $f(O(\lambda_i))=1$，節 i から上ったら $f(O(\lambda_i))=0$ とすれば，すべての f は正しく設定されている．

[補題2] ノード $O(\lambda_i)$ を終点とするリンク x の始点を v とするとき，$f(v)=0$ ならば，x を使って節 i の息子を発生できる．

[補題3] 節1を発生させたとき，$\delta(\boldsymbol{P}_1)=0, \beta_1=\Delta(S_0)$ の先頭リンク，$\delta_c(1)=\infty$，$\delta_d(1)=\delta(\beta_1)$ である．

補題1，2，3は単純路の木の定義からして明らかである．

[補題4] 現在の節を i とするとき，i の息子 j を発生させた場合，β_i は正しく再設定できる．

(証明) 補題1，2により設定できる．つまり，β_i は，まだ使用されていずかつ補題2の条件を満足するリンクのうち，遅延時間が最小のものとすればよい．(証明終)

[補題5] 節 i から新しく発生した息子 j に移動したとき，$\beta_j, \delta_d(j)$ は正しく設定できる．

(証明) β_j は補題1，2により設定できる．つまり，β_j は補題2の条件を満足するリンクのうち，遅延時間が最小のものとすればよい．また，j にはまだ息子がないから $\delta_d(j)=\delta(\boldsymbol{P}_i)+\delta(\lambda_j)+\delta(\beta_j)$ とすればよい．(証明終)

[補題6] 現在の節を i としたとき，$\delta(\boldsymbol{P}_i)$, i と i の祖先すべての δ_c, i の祖先以外すべての δ_d, すべての β がわかっているならば，i の父または息子 j に進んでもやはり $\delta(\boldsymbol{P}_j)$, j と j の祖先すべての δ_c, j の祖先以外すべての δ_d はわかる．

(証明) まず，父に進んだ場合，$\delta(\boldsymbol{P}_j)$ と $\delta_d(j)$ は式(6)，(7)により算出できる．

$$\delta(\boldsymbol{P}_j)=\delta(\boldsymbol{P}_i)-\delta(\lambda_i) \quad (6)$$

$$\delta_d(j)=\min\{\delta_d(S), \delta(\boldsymbol{P}_j)+\delta(\beta_j)\} \quad (7)$$

次に，子に進んだ場合，$\delta(\boldsymbol{P}_j)$ と $\delta_c(j)$ は式(8)，(9)

により算出できる.

$$\delta(\boldsymbol{P}_j)=\delta(\boldsymbol{P}_i)+\delta(\lambda_j) \quad (8)$$

$$\delta_c(j)=\min\{\delta_c(i), \delta_d(\mathcal{B}), \delta(\boldsymbol{P}_i)+\delta(\beta_i)\} \quad (9)$$

ここで S は j の息子のうちで δ_d が最小のもの，$\mathcal{B}$ は j の兄弟のうちで δ_d が最小のものを示す.

(証明終)

[定理3] 節1から出発して単純路の木の探索・発育を行っている間中，通過中の各節 i の $\delta(\boldsymbol{P}_i)$, $\delta_c(i)$, $\delta_d(i)$, β_i は正しく設定できる.

(証明) 節1については，補題3に従って設定できる．以降の節についても，補題4・5・6に従って設定できる. (証明終)

4. アルゴリズム

遅小樹育法のアルゴリズムはフェーズⅠとフェーズⅡの2段階に分かれる.

まずフェーズⅠにおいては，ラベル確定法という最短径路手法により，全リンクの遅延時間を求めていく．ラベル確定法[15]についてはヒープ法[9]，バケット法[12]，拡散法[16]などいくつかの手法が提案されているが，ここではヒープを用いた方法について説明する.

次にフェーズⅡにおいては，単純路の木を構築して行く.

以下にフェーズⅠおよびⅡのアルゴリズムを示すが，その前に2で定義した他に次の変数とリストを定義しておく.

$\underline{\boldsymbol{H}}$: リンクを登録するためのデータ構造で，先頭にはいつも時間が最小のリンクが来る．リストを用いる方法や2進木を用いる方法が考えられる.

$\underline{\Delta(v)}$: v を終点とするリンクを，遅延時間が小さい順に並べておくためのリスト.

$\underline{H(v)}$: $\Delta(v)$ の先頭リンク.

$\underline{N(x)}$: $\Delta(S(x))$ において x の後に接続されているリンク.

$\underline{p}$: 新しく発生する節の番号．節が一つ発生するごとにカウントアップされる.

$\underline{\mathcal{F}_i, \mathcal{B}_i, \mathcal{S}_i}$: これらは i の親族を表す変数で，この三つによりツリー構造が表現される．$\mathcal{F}_i, \mathcal{B}_i, \mathcal{S}_i$ は順に節 i の父，弟，長男を示す．たとえば**図1**(a)のように節 a, b, c があったとすると，それらの関係は図1(b)のようになる．このとき兄のほうが弟よりも，子孫遅延時間が小さいかまたは等しいようにする.

$\underline{\delta_0}$: 現在の節 i の遅延時間を示す．つまり，この値は $\delta(\boldsymbol{P}_i)$ に等しい.

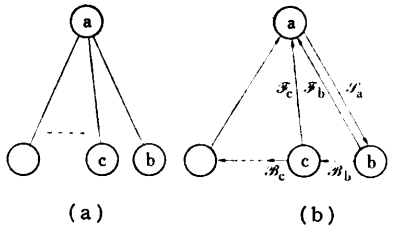


図1 ツリーのポインタ
Fig. 1 Pointers for the path tree.

$\underline{\boldsymbol{R}}$: 発見した第1～第 k 単純路を登録しておくためのリストである.

なお，上記変数のうち $H(v), N(x), \mathcal{F}_i, \mathcal{B}_i, \mathcal{S}_i$ は該当するものが存在しなければ0とする.

4.1 フェーズⅠのアルゴリズム

S1. $\boldsymbol{H}\leftarrow\Phi$, $\Delta(1)\sim\Delta(n)\leftarrow\Phi$, $\pi(O_0)\leftarrow 0$, $v\leftarrow O_0$, $\Delta(O_0)\leftarrow\{m+1\}$ とする．ここで Φ は空ヒープや空リストを示す．なお，リンク $m+1$ は処理の都合上設けた仮想リンクである.

S2. ノード v から発するすべてのリンクを $\boldsymbol{H}$ に登録する．このとき各リンク x について $\tau(x)\leftarrow d(x)+\pi(v)$ とする.

S3. $\boldsymbol{H}=\Phi$ の場合は **S6.** に進む．$\boldsymbol{H}\neq\Phi$ の場合は，$x\leftarrow\boldsymbol{H}$ のなかで τ が最小の任意のリンク，$\boldsymbol{H}\leftarrow\boldsymbol{H}-\{x\}$, $v\leftarrow S(x)$ とする.

S4. $\Delta(v)=\Phi$ の場合は $\Delta(v)\leftarrow\{x\}$, $\pi(v)\leftarrow\tau(x)$, $\delta(x)\leftarrow 0$ とする．$\Delta(v)\neq\Phi$ の場合は，$\Delta(v)\leftarrow\Delta(v)\cup\{x\}$, $\delta(x)\leftarrow\tau(x)-\pi(v)$ とする.

S5. $v\neq S_0$ の場合は **S2.** に，$v=S_0$ の場合は **S3.** に戻る.

S6. $\Delta(S_0)=\Phi$ の場合は，O_0 から S_0 への径路は存在しないので，終了する．そうでない場合は，フェーズⅡに進む.

4.2 フェーズⅡのアルゴリズム

フェーズⅡのアルゴリズムの概要を**図2**に示す．この図では定理2に従って $\delta_c(i), \delta_d(i), \delta(\boldsymbol{P}_i)+\delta(\beta_i)$ を比較することにより上るか，下るか，新しい息子を発生させるかを決めている.

図では省いたが，$\beta_i, \mathcal{F}_i, \mathcal{B}_i, \mathcal{S}_i$ も適時設定する必要がある．また定理4に従って β_i を設定するために変数 f を用いており，これも正しく設定する必要がある．詳細なアルゴリズムを以下に記す.

S1. $p\leftarrow 1$, $f(0)\sim f(n)\leftarrow 0$, $\boldsymbol{R}\leftarrow\Phi$, $\delta(0)\leftarrow\infty$, $\delta_c(0)\leftarrow\infty$, $\delta_d(0)\leftarrow\infty$, $\mathcal{F}_1\leftarrow 0$, $\mathcal{B}_1\leftarrow 0$, $\mathcal{S}_1\leftarrow 0$, $\delta_0\leftarrow 0$, $\beta_1\leftarrow H(S_0)$, $\delta_c(1)\leftarrow\infty$, $\delta_d(1)\leftarrow\delta(\beta_1)$, $O(0)\leftarrow 0$, $i\leftarrow 1$ と

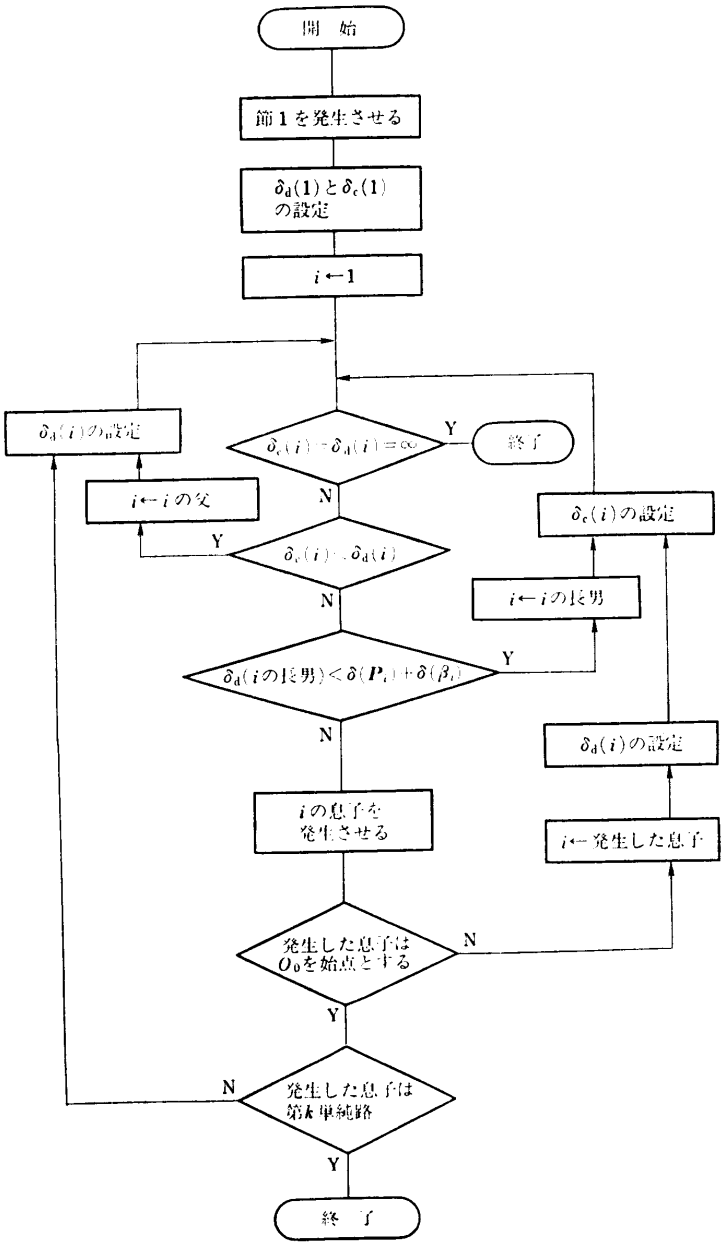


図 2 アルゴリズムの概要
Fig. 2 Flowchart of the algorithm.

する．ここで，ノード0，リンク0は処理の都合上設けた．

S2. $\delta_c(i)=\delta_d(i)=\infty$ のときは，これ以上単純路を発生させることができないので，フェーズⅡを終了する．$\delta_c(i)<\delta_d(i)$ ならば **S6.** に進む．$\delta_d(S_i)<\delta_0+\delta(\beta_i)$ ならば **S8.** に進む．

S3. $p\leftarrow p+1$, $\lambda_p\leftarrow\beta_i$, $\mathcal{F}_p\leftarrow i$, $\mathcal{B}_p\leftarrow 0$, $\mathcal{S}_p\leftarrow 0$, とする．

S4. $\beta_i\leftarrow N(\beta_i)$ とした後，$f(O(\beta_i))=1$ の場合は **S4.** に戻る．

S5. $O(\lambda_p)\neq O_0$ の場合は **S10.** に進む．$O(\lambda_p)=O_0$ の場合は O_0 を始点とする単純路が求まったので $\boldsymbol{R}\leftarrow\boldsymbol{R}\cup\{p\}$ とする．この結果 $\boldsymbol{R}$ の要素の数が k ならばフェーズⅡを終了する．でなければ **S7.** に進む．

S6. $f(O(\lambda_i))\leftarrow 0$, $\delta_0\leftarrow\delta_0-\delta(\lambda_i)$, とする．$i$ を $\mathcal{F}_i$ の息子として接続する．このとき $\mathcal{F}_i$ の各息子の δ_d が小さい順に並ぶような兄弟関係とする．$i\leftarrow\mathcal{F}_i$ とする．

S7. $\delta_d(i)\leftarrow\min\{\delta_d(\mathcal{S}_i),\delta_0+\delta(\beta_i)\}$, として **S2.** に戻る．

S8. $i\leftarrow\mathcal{S}_i$, $\mathcal{S}_{\mathcal{F}_i}\leftarrow\mathcal{B}_i$, とする．

S9. $f(O(\lambda_i))\leftarrow 1$, $\delta_c(i)\leftarrow\min\{\delta_c(\mathcal{F}_i),\delta_d(\mathcal{S}_{\mathcal{F}_i}),\ \delta_0+\delta(\beta_{\mathcal{F}_i})\}$, $\delta_0\leftarrow\delta_0+\delta(\lambda_i)$, として **S2.** に戻る．

S10. $i\leftarrow p$, $\beta_i\leftarrow H(O(\lambda_i))$, とする．

S11. $f(O(\beta_i))\neq 1$ の場合は **S12.** に進む．$f(O(\beta_i))=1$ の場合は $\beta_i\leftarrow N(\beta_i)$ として **S11.** に戻る．

S12. $\delta_d(i)\leftarrow\delta_0+\delta(\lambda_i)+\delta(\beta_i)$ として **S9.** に進む．

5. モデルによる説明

ここでは，具体的な処理を，**図 3** のネットワークで説明する．ここで，①～⑨はノード，矢印はリンク，□はリンク番号，間の数字はリンク長をそれぞれ示す．

いま，①から⑨までの第1～第3単純路を求めることにする．

5.1 フェーズⅠの説明

フェーズⅠについての具体的説明は紙面の都合上省略するが，ラベル確定法を用いて1～24までの全リンクの遅延時間を求める．結果は**表 1** のようになる．

5.2 フェーズⅡの説明

フェーズⅡは単純路の木を育てていく．図中では δ_c を△，δ_d を▽で表すことにする．○は節を示し，数字は上が節番号，下がリンク番号である．そのほか，---○ は発芽リンク，( ) はリンク遅延時間，⟶ は通った跡を示している．

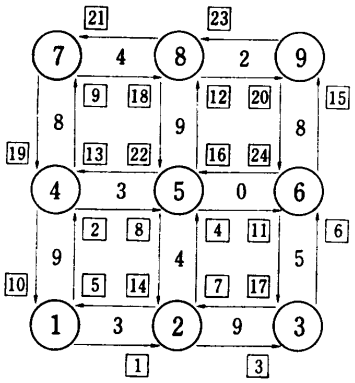


図 3 説明のためのネットワーク
Fig. 3 Model network.

表 1 フェーズIの結果
Table 1 Computation result of phase I.

ノード	π	Δ
①	0	25 (0), 5 (6), 10 (18)
②	3	1 (0), 14 (8), 7 (18)
③	12	17 (0), 3 (0)
④	9	2 (0), 13 (1), 19 (16)
⑤	7	4 (0), 16 (0), 8 (5) 22 (18)
⑥	7	11 (0), 6 (10)
⑦	17	9 (0), 21 (3)
⑧	16	12 (0), 18 (5)
⑨	15	15 (0), 20 (3)

( ) 内は遅延時間を示す.

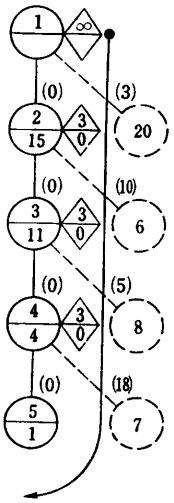


図 4 第1単純路の発見
Fig. 4 Finding the shortest-loopless path.

モデル・ネットワークで説明すると，まず1が作られ Δ が∞，∇ が0だから下に進む．次に2が作られ同様に下に進んでいき，5まで来たとき第1単純路が求まる（図4）．第1単純路は $P_5=\{1,4,11,15\}$ である．径路の登録はリスト R に径路番号5（＝節番号）を接続することにより行われ，$R=\{5\}$ となる．

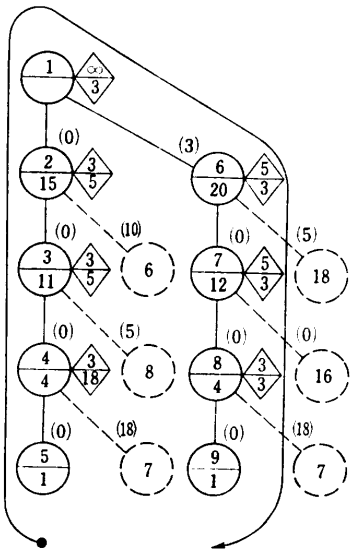


図 5 第2単純路の発見
Fig. 5 Finding the 2nd shortest-loopless path.

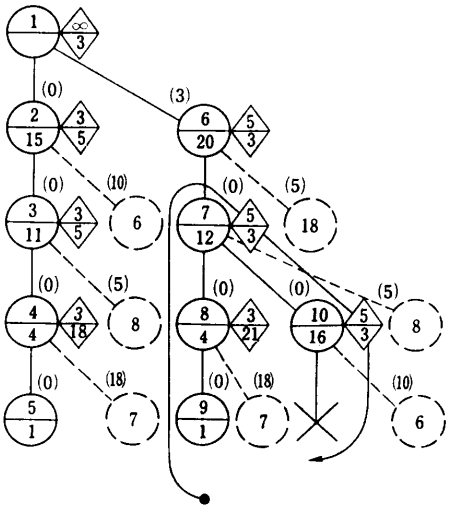


図 6 閉路の出現
Fig. 6 Encountering a loop.

さらに処理を続けると，上に戻って来て1から枝伸ばしが行われ，第2単純路が求まる（図5）．$R=\{5,9\}$ となり，第2単純路は $P_9=\{1,4,12,20\}$ である．

同様に処理を続けると，上に戻って来て7から枝伸ばしがされるが，10からは遅延時間が最小の単純路を発生できない．これは $\delta(x)=0$ のリンクがあっても，

それを辿っていくと閉路になるためである（**図6**）．×は閉路に行き当たったことを示している．

さらに探索を続けると，$\boldsymbol{R}=\{5, 9, 12\}$ となり，第3単純路が求まる（**図7**）．第4単純路以降も，これを繰り返していけば求めることができる．

6. 計算時間の評価

6.1 計算時間の試算

本手法の平面ネットワークにおける計算時間を試算してみることにする．ここで平面ネットワークにおけるリンク数はノード数に比例するものと考える．

まずフェーズⅠの計算時間 t_1 は，どのラベル確定法を使うかによって異なり，$O(n)$～$O(n \log n)$ である．

次にフェーズⅡの計算時間 t_2 であるが，これは単純路の木の探索・発育をする時間に相当する．

ここで，図4，5，7のように，枝伸ばし地点を探して，そこから（ ）が0のリンクを辿っていくことにより次々と新しい節が発生され，最後にノード①に到達する場合と，図6のように（ ）が0のリンクを辿って行っても，閉路に行き当たってしまいバックトラックが生じる場合がある．

よって，閉路に行き当たる回数の平均値を $\alpha(k, n)$ とすると，各図とも，探索に要する時間は一つの径路に含まれるノード数つまり $n^{1/2}$ にほぼ比例するといえるから，t_2 の平均値は $(k+\alpha(k, n))n^{1/2}$ に比例しよう．

6.2 計算時間の実測値

本手法の双方向性格子状ネットワークにおける実測値を**図8，9**に示す．図8，9とも，リンク長を 0～99999 の一様乱数，始点を格子の左下隅，終点を格子の右上隅として測定したものである．

図8はノード数を変化させたものである．図中のグラフとも，50個のサンプル・ネットワークを

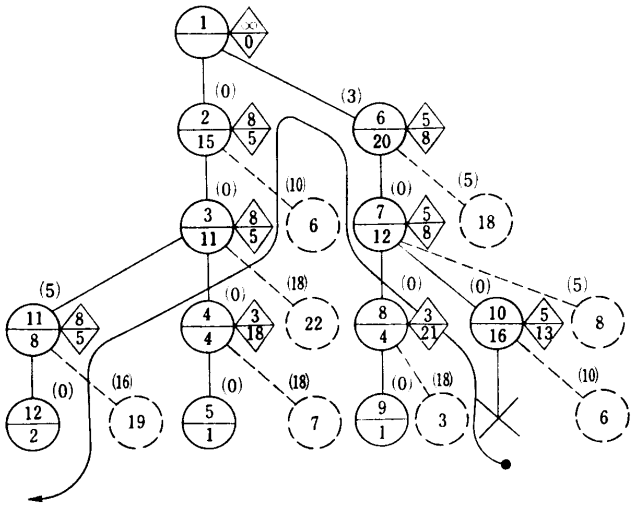
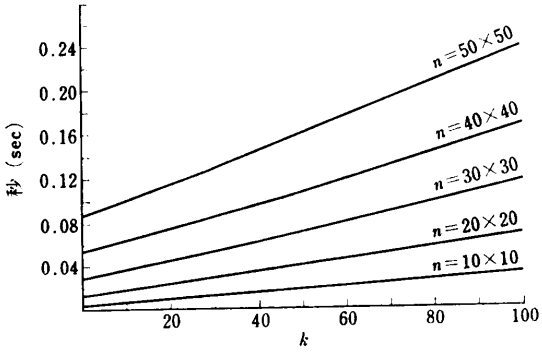
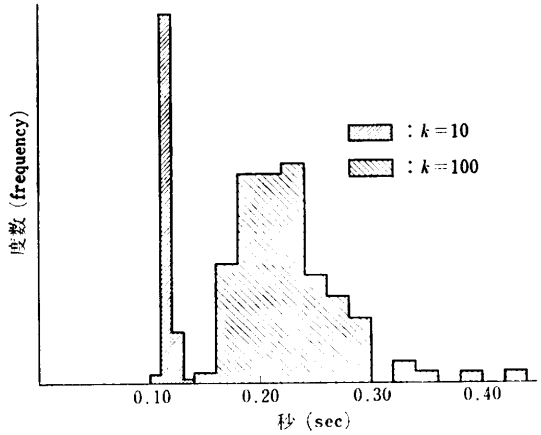


図 7　第3単純路の発見
Fig. 7　Finding the 3rd shortest-loopless path.



ネットワーク：双方向性格子
リ ン ク 長：0～99999（一様乱数）
始 点，終 点：1，n
サンプル数：50
機　　種：HITAC M 280 H
言　　語：FORTRAN

図 8　双方向性正方格子における平均計算時間
Fig. 8　Average time for finding k shortest-loopless paths on bidirectional grids.



ネットワーク：50×50 双方向性格子
リ ン ク 長：0～99999（一様乱数）
始 点，終 点：1，2500
サンプル数：100
機　　種：HITAC M 280 H
言　　語：FORTRAN

図 9　計算時間の分散度
Fig. 9　Histogram of k shortest-loopless paths computing time on 50×50 bidirectional grids.

発生させ各サンプルについて1回ずつ測定し，50回の平均をとったものである．この図について，t_1 ($k=0$ の部分)，t_2 と k,n の関係をべき乗回帰で求めた結果，

$$t_1=2.71\times10^{-5}\times n^{1.028} \quad (10)$$

$$t_2=1.896\times10^{-5}\times k^{1.066}\times n^{0.519} \quad (11)$$

が得られた．また，図9は，$n=50\times50$ として100個サンプルを発生させて $k=100$ まで測定し，$k=10$, 100 における計算時間のばらつきを見たものである．

式(11)からわかるように，フェーズⅡの平均時間は k と $n^{0.5}$ にほぼ比例している．このことから $(k+\alpha(k,n))n^{1/2}$ における α は k に比例し，全体として t_2 の平均値は $0(kn^{1/2})$ といってさしつかえないものと思われる．

また，図9を見る限り，100回サンプルをとって最悪の場合でも平均時間の2倍程度である．このばらつきはフェーズⅡによるものであり，フェーズⅠは一定だった．

参考までに，図9において，ツリーを作るために必要な領域の最大値は $k=10$ において p(節の数)$=829$, $k=100$ において $p=5724$ であった．

また，その他のネットワークにおける効率を評価するために，以下のようなネットワークに対していくつかサンプルを発生させて測定した．

① 各ノードから他のノードすべてに対してリンクを出した有向ネットワーク．リンク長はランダム．

② 正方形内部に50個の点をばらまき，各点から，最も近い4個の点に対してリンクを出した有向ネットワーク．リンク長はユークリッド距離．

①においては，$n=10, 20, 30, 40, 50$ のおのおのに対して，10個サンプルを発生させ $k=100$ までの平均値をとった．その結果，各 n とも t_2 は k に比例し，k に対する t_2 の勾配は n によって変化がなかった．②においては，サンプルを5個発生させ，$k=100$ として計21回測定を試みた．そのうち19回は，t_2 がほぼ k に比例した．残り2回は，非常に時間がかかったので，途中で計算を打ち切った．

このように，本手法は場合によっては非常に時間がかかる．たとえば，**図10**のネットワークがその一例である．この図において，ノード1から n までの単純路を求めたとき，部分ネットワーク G′ に多数のノードがあり，第2単純路 (1-3-n) の距離が非常に大きいとすると，第2単純路を求めるまでに G′ 内のノードを始点とする単純路を，無数発生させる必要がある．

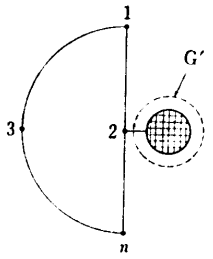


図10 効率の悪いネットワーク例
Fig. 10 A network on which the algorithm is unefficient.

なお，今回の測定においては，フェーズⅠではヒープでなくバケットを用いた．

7. 従来の手法との比較

これまで最も効率がよいとされている有向ネットワークに適用できる手法は，最短径路探索に $O(c(n,m))$ かかれば，第 k 単純路探索に $O(knc(n,m))$ かかるとされている[6),7)]．

また，無向ネットワークだけに適用される，$O(kc(n,m))$ の手法が加藤・茨木・三根より提案されている[11),14),17)]．

これらの手法は，いずれも最悪時間の効率化を図ったもので，平均時間がそれほどよくならない．たとえば $O(knc(n,m))$ の手法の平面ネットワークにおける平均時間を試算すると，$O(kn^{1/2}c(n))$ となる．本手法のフェーズⅠは $c(n)$ に相当するので，図8と比較して桁違いに時間がかかる．

また，閉路を許す第 k 最短径路の手法もいくつか提案されていて，最悪時間も閉路を含まない場合に比べると非常に効率がよい[1),3),4),8),13)]．よって，まず閉路を許す最短径路を k' 個求めておいて，そのなかから単純路を k 個選び出すのも一つの方法である．しかしながら，k' をいくつにしたら k 個まで求めることができるかは不明確である．とくに，長さが0または非常に小さい閉路が存在する場合，同じところをぐるぐる回り，無意味な径路しか発生しないようになる．

このためには，径路を体系的に表現し，一度閉路を含むとわかったらその系統は以後求めないようにする工夫が必要である．

また，平均時間の効率化を図るために，新しい径路は，既存の径路から分岐したものであるという考えも取り入れている．この考えは，従来の手法でもすでに採用していて，従来の手法は，最初最短径路の木を求めておき，どのノードに分岐したら最も近いかをその

つど計算している[1),2),18)]．しかし，いずれの手法も，径路を体系化していないので，単純路だけを求めるための効率のよい処理ができない．

8. まとめ

以上述べたように，本手法は平均的に見てきわめて高速である．また図9で示したように，ある程度のばらつきがあり，場合によっては時間がかかる．しかし，このような特殊なネットワークの生じる確率は小さく，k が小さいほどばらつきが小さいことが図9から明らかである．

また，リンク長が非負の場合についていままで述べてきたが，式(5)は $d(x_i)$ が負の場合にも成立する．よってフェーズⅡについては，長さが負のリンクを含むネットワークにも適用できる．一方，ここで述べているフェーズⅠはラベル確定法を用いており，リンク長は非負が前提となっている．何らかの手法がフェーズⅠに適用できるならば，本手法はリンク長が非負のネットワークについても適用可能である．

一般に第 k 最短径路問題は，最短径路問題が適用できる分野に適用され，最適な径路を求めるために使われよう．この場合，最適な径路を上位の複数の最短径路から選ぶ方法がとられる．距離が短いほど最適な径路に近いが，距離以外の要因があり，最短径路が必ずしも最適径路といえない場合に，これが必要である．

そのほかに，最小経費流問題にも適用できよう．短い径路から順々にフローを配分していけば，全フローを配分し終わったとき，最小経費流となる．本手法はこの問題にも適用できるものと思われる．

本手法の適用分野は交通網，通信網，回路網等のネットワークのほかに，ネットワークを利用して解ける各種の最適化問題があり，応用範囲が広い．

最後に本手法の名称であるが，本手法は，一口でいえば，遅延時間が最小になる地点を探して径路の木を育てる方法である．そこで正式には，遅延時間最小化式径路樹木発育法，略して遅小樹育法と呼ぶことにする．

謝辞 本手法の開発にあたり，未来工研の大井哲雄主任研究員および上智大学学生是永浩喜君に多大なご協力をいただきましたので，ここに謝意を表します．

参考文献

1) Hoffman, W. and Pavley, R.: A Method for the Solution of the Nth Best Path Problem, *J. ACM*, Vol. 6, No. 4, pp. 506-514 (1959).
2) Clarke, S., Krikorian, A. and Rausen, J.: Computing the N Best Loopless Paths in a Network, *Appl. Math.*, Vol. 11, No. 4, pp. 1096-1102 (1963).
3) Sakarovitch, M.: *The k Shortest Routes and the k Shortest Chains in a Graph*, p. 20, Operations Research Center, University of California, Berkeley, ORC 66-32 (1966).
4) Dreyfus, S. E.: An Appraisal of Some Shortest-Path Algorithms, *Oper. Res.*, Vol. 17, No. 3, pp. 395-412 (1969).
5) Tomizawa, N.: On Some Techniques Usefull for Solution of Transportation Network Problems, *Networks*, Vol. 1, pp. 173-194 (1971).
6) Yen, J. Y.: Finding the K Shortest Loopless Paths in a Network, *Manage. Sci.*, Vol. 17, No. 11, pp. 712-716 (1971).
7) Lawler, E. L.: A Procedure for Computing the K Best Solutions to Discrete Optimization Problems and Its Application to the Shortest Path Problem, *Manage. Sci.*, Vol. 18, No. 7, pp. 401-405 (1972).
8) Fox, B. L.: Calculating k th Shortest Paths, *INFOR*, Vol. 11, No. 1, pp. 66-70 (1973).
9) Jonson, D. B.: Efficient Algorthms for Shortest Paths in Sparse Networks, *J. ACM*, Vol. 24, No. 1, pp. 1-13 (1977).
10) Lawler, E. L.: Comment on Computing the k Shortest Paths in a Graph, *Comm. ACM*, Vol. 20, No. 8, pp. 603-604 (1977).
11) 加藤直樹，茨木俊秀，三根 久：無向グラフの第 k 最短単純路を求める $O(kn^2)$ のアルゴリズム，信学論(A)，Vol. J 61-A, No. 12, pp. 1199-1206 (1978).
12) Denard, E. V. and Fox, B. L.: Shortest-Route Methods: 1. Reaching, Pruning, and Buckets, *Oper. Res.*, Vol. 27, No. 1, pp. 161-186 (1979).
13) Shier, D. R.: On Algorithms for Finding the k Shortest Paths in a Network, *Networks*, Vol. 9, No. 3, pp. 195-214 (1979).
14) 山田洋一，茨木俊秀，長谷川利治：無向グラフにおける第 k 単純路の解法，日本オペレーションズ・リサーチ学会秋季研究発表会アブストラクト集，pp. 50-51 (1980).
15) 今井 浩：ネットワーク理論の現状，第3回数理計画シンポジウム論文集，pp. 31-44 (1982).
16) 加藤誠巳，倉部 淳：拡散法による最短径路探索の一手法，情報処理学会第25回全国大会論文

集, pp. 1415-1416 (1982).
17) Katoh, N., Ibaraki, T. and Mine, H.: An Efficient Algorithm for K Shortest Simple Paths, *Networks*, Vol. 12, No. 4, pp. 411-427 (1982).
18) De Queiros Vieira Martins, E.: An Algorithm for Ranking Paths in Acyclic Networks, *OR Spektrum*, Vol. 5, No. 2, pp. 87-90 (1983).

(昭和59年4月17日受付)
(昭和59年10月18日採録)