アルゴリズム２－３
（１９８８．７．２５）

# ラベリングースキャンニング法による最短経路問題について

佐藤　尚　，　野崎昭弘

国際基督教大学　理学研究科　国際基督教大学　理学科

辺の長さとして負のものもゆるすグラフの最短経路問題について、ＦＩＦＯ法の計算量の証明を利用して、ラベリングースキャンニング法による実行時間Ｏ（｜Ｖ｜｜Ｅ｜）とＯ（｜Ｖ｜｜Ｅ｜ｌｏｇ｜Ｅ｜）の新しい方法を考える。そして、これらの新しい方法とＦＩＦＯ法および両頭列法の比較を、いくつかのグラフを用いておこなう。これらの新しい方法は、複雑なグラフに対して、ある程度有効であることが分かった。

# On the shortest path problem wiht labeling-scanning method

Hisashi Sato and Akihiro Nozaki

International Christian University Division of Natural Science

10-2,Osawa 3-chome,Mitaka-shi,Tokyo 181 ,Japan

For the network optimization problem of finding shortest paths from a source to all the other vertices in a given directed graph whose edges have real-valued lengths , we devise several new methods in labeling and scanning based on a proof of running time of FIFO method. We compare these new methods with the existing FIFO method and a two-way sequence method.These new methods have been found efficient for complex graphs.

## 1　定義

G=(V,E)を、頂点の集合Vと辺の集合Eからなる有向グラフとする。各頂点ｖに対して、ｏｕｔ（ｖ）＝｛［ｖ,ｗ］∈Ｅ｝とする。Eに属する各辺には、ｌｅｎｇｔｈと言う実数（負でもよい）が割り振られており、[ｖ,ｗ]∈Eのｌｅｎｇｔｈをｌｅｎｇｔｈ(ｖ,ｗ)で表す。経路ｐのｌｅｎｇｔｈをp上の各辺のｌｅｎｇｔｈの和とし、ｌｅｎｇｔｈ(p)と書く。頂点ｓ∈Ｖから頂点ｔ∈Ｖへの最短経路は、ｓからｔへのｌｅｎｇｔｈ最小の経路のこととする。ここでは、１つの出発点ｓから、他のすべての頂点への最短経路を見つける問題を考える。

## 2　FIFO法

ここでは、FIFO法とその計算量の評価をＴａｒｊａｎに従って、述べることを目的とする［Ｔａｒｊａｎ］。

### ２．１　ラベルリング法

ここで、紹介する方法を、ラベリング法という。この方法が、停止したとき、出発点ｓから他のすべての点への、最短経路が得られる。

それぞれの頂点ｖにたいして、ｄｉｓｔ(ｖ)を出発点ｓからｖまでの、仮の距離とし、ｐ（ｖ）はｓからｖへの長さｄｉｓｔ(ｖ)の経路での、［ｐ(ｖ),ｖ]∈Eとなる頂点を表すこととする。初めに、ｄｉｓｔ(ｓ)=0,ｄｉｓｔ(ｖ)=∞(ｖ≠ｓ),ｐ（ｖ）=nullと初期化する。そこで、任意の[ｖ,ｗ]∈Ｅに対して、ｄｉｓｔ（ｖ）＋ｌｅｎｇｔｈ（ｖ,ｗ）≧ｄｉｓｔ（ｗ）となるまで、次の操作（Ｌａｂｅｌｉｎｇ　Ｓｔｅｐ）を繰り返す。

Ｌａｂｅｌｉｎｇ　Ｓｔｅｐ

ｄｉｓｔ（ｖ）＋ｌｅｎｇｔｈ（ｖ,ｗ）＜ｄｉｓｔ（ｗ）となる[ｖ,ｗ]∈Ｅを探し、

　　ｄｉｓｔ（ｗ）←ｄｉｓｔ（ｖ）＋ｌｅｎｇｔｈ（ｖ,ｗ）

　　ｐ（ｗ）←ｖ

と置き換える。

この時、ｄｉｓｔ（ｖ）に関して、次の２つの補題が成立する。

補題　２．１　ラベリング法において、ｄｉｓｔ（ｖ）が有限であれば、ｌｅｎｇｔｈがｄｉｓｔ（ｖ）のｓからｖへの経路が存在する。

（証明）Ｌａｂｅｌｉｎｇ　Ｓｔｅｐの回数に関する、帰納法による。

補題　２．２　ｐを、出発点ｓから頂点ｖへの任意の経路とする。ラベリング法が停止したとき、

　　　ｌｅｎｇｔｈ（ｐ）≧ｄｉｓｔ（ｖ）

が成立する。

（証明）ｐ上の辺の本数に関する、帰納法による。

この２つの補題により、次のことが分かる。

定理　２．３　ラベリング法が終わったとき、

(i)　ｄｉｓｔ（ｖ）が有限であれば、ｄｉｓｔ（ｖ）は出発点ｓからｖへの最短経路のｌｅｎｇｔｈを与える。

(ii)　ｄｉｓｔ（ｖ）が無限大であらば、ｓからｖへは到達不可能である。

(iii)　出発点ｓから到達できるところに、負の閉路があれば、ラベリング法は、決して終わらない。

ｐ（ｖ）に関しては、次の２つの補題が成り立つ。

補題　２．４　ラベリング法において、ｐ（ｖ）≠ｎｕｌｌであれば、
ｄｉｓｔ（ｐ（ｖ））＋ｌｅｎｇｔｈ（ｐ（ｖ），ｗ）≦ｄｉｓｔ（ｖ）
が成り立つ。ただし、等号はラベリング法が停止したときに成り立つ。
（証明）Ｌａｂｅｌｉｎｇ　Ｓｔｅｐの回数に関する、帰納法による。

補題　２．５　ラベリング法において、$p^k$（ｖ）＝ｖとなるｖ∈Ｖと自然数ｋが存在すれば、Ｇ＝（Ｖ,Ｅ）は負の閉路を持つ。
（証明）略

この２つの補題より、次のことが分かる。

定理　２．６　ラベリング法が停止したとき、ｐ（ｖ）≠ｎｕｌｌであれば経路ｐ＝（ｓ（＝$p^k$（ｖ）），$p^{k-1}$（ｖ），……，ｐ（ｖ），ｖ）が、出発点ｓからｖへの最短経路を与える。
（証明）次の２つ性質と、定理２．３、補題２．４、により明らか。
(i)　ｐ（ｖ）≠ｎｕｌｌ（ｖ≠ｓ）⇔ｄｉｓｔ（ｖ）＜∞
(ii)　ｄｉｓｔ（ｖ）＜∞　かつ　ｐ（ｖ）≠ｎｕｌｌ
⇒
ｄｉｓｔ（ｐ（ｖ））＜∞　　（Ｑ．Ｅ．Ｄ）

定理２．３および２．６により、ラベリング法が停止したとき最短経路が求まることが分かった。

## ２．２　ラベリングースキャンニング法とＦＩＦＯ法

前節のラベリング法は効率の良い方法ではないので、この節ではその改良であるラベリングースキャンニング法（ラベル修正法）と、そのｑｕｅｕｅによる実現であるＦＩＦＯ法について述べる。
ラベリングースキャンニング法では、各頂点を次の３つの状態

ｕｎｌａｂｅｌｅｄ，ｌａｂｅｌｅｄ，ｓｃａｎｎｅｄ

に分ける。初めに、出発点ｓに対してはｌａｂｅｌｅｄ、その他の頂点に対してはｕｎｌａｂｅｌｅｄとする。そこで、ｌａｂｅｌｅｄな頂点がなくなるまで、次の操作（Ｓｃａｎｎｉｎｇ　Ｓｔｅｐ）を繰り返す。

Ｓｃａｎｎｉｎｇ　Ｓｔｅｐ
ｌａｂｅｌｅｄな頂点ｖを選び、その状態をｓｃａｎｎｅｄにする。そして、ｄｉｓｔ（ｖ）＋ｌｅｎｇｔｈ（ｖ,ｗ）＜ｄｉｓｔ（ｗ）を満たす辺［ｖ,ｗ］に対して、Ｌａｂｅｌｉｎｇ　Ｓｔｅｐを適用し、ｗの状態をｌａｂｅｌｅｄに換える。

ｌａｂｅｌｅｄな頂点をｑｕｅｕｅで管理する方法を、ＦＩＦＯ法と呼ぶ。また、ｌａｂｅｌｅｄな頂点を、ｄｅｑｕｅｕｅで管理することにし、ｕｎｌａｂｅｌｅｄな頂点が、ｌａｂｅｌｅｄに変わるときにはｄｅｑｕｅｕｅの最後に挿入し、そうでない頂点が、ｌａｂｅｌｅｄに変わるときには、ｄｅｑｕｅｕｅの先頭に挿入する。そして、ｄｅｑｕｅｕｅの先頭からｌａｂｅｌｅｄな頂点を取り出す。これを、ＴＷＳＱ法（両頭列法）と呼ぶ［伊理　Ｉｍａｉ＆Ｉｒｉ］。

この節の最後に、ＦＩＦＯ法の時間量を調べる。そのためにＦＩＦＯ法の実行を、次のようなｐａｓｓという単位に分ける。

定義　２．７　ｐａｓｓ　０は、出発点ｓの最初のｓｃａｎｎｉｎｇだけからなる。ｊが正のとき、ｐａｓｓ　ｊはｐａｓｓ　ｊ－１の終わりまでに、ｑｕｅｕｅの中に蓄えらている、すべての頂点をｓｃａｎｎｉｎｇし終えるまでとする。

定理　２．８　ｓから到達可能な負の閉路が、存在しないとする。このとき、ＦＩＦＯ法はｐａｓｓ　｜Ｖ｜－１までで停止し、その実行時間はＯ（｜Ｖ｜｜Ｅ｜）となる。また、ｓから到達可能は負の閉路が存在すれば、ＦＩＦＯ法は決して止らない。

（証明）　各ｐａｓｓにおいて、頂点は高々１回しかｓｃａｎｎｉｎｇされないから、１回のｐａｓｓは、Ｏ（｜Ｅ｜）で実行できる。

ｓから頂点ｖへの最短経路がｋ個の辺を含むとする。このとき、ｐａｓｓ　ｋの初めまでにはｄｉｓｔ（ｖ）の値は、この最短経路のｌｅｎｇｔｈを与えたいることが、ｋに関する帰納法で証明できる。

したがって、補題２．１により、定理は正しい。　　　　　（Ｑ．Ｅ．Ｄ）

## 3　ラベリングースキャンニング法の別の実現法

ＦＩＦＯ法は、ｌａｂｅｌｅｄな頂点をｑｕｅｕｅを用いて管理することにより、ｐａｓｓの考えを前面に出すことなくラベリングースキャンニング法を実現することができた。そこで、ｌａｂｅｌｅｄな頂点を各ｐａｓｓごとに別々に管理することを考える。ここでは、各ｐａｓｓに対応するｌａｂｅｌｅｄな頂点を、異なる配列に順々に蓄えていく。そして、ｐａｓｓ　ｊ－１に対応する配列のすべての要素をｓｃａｎｎｉｎｇしおえたら、ｐａｓｓ　ｊに対応する配列の要素をある程度ｄｉｓｔの順に並べ換え（ｌａｚｙｓｏｒｔ）てからｓｃａｎｎｉｎｇをはじめることにする。すなわち、なるべくｄｉｓｔの値の小さいものからｓｃａｎｎｉｎｇできるように並べ換える。具体的アルゴリズムには次のようになる。

```
procedure shortestpath(vertices:set;s:vertex);
  var vertex v;array a1,a2;
      integer i1,i2;
  begin
    for v ∈ vertices begin dist(v)=∞;p(v):=nullend;
    i1:=0; i2:=1;
    a2[i2]:=s;dist(s):=0;
    while not empty(a2) do begin
      lazysort(i2,a2);
      swap(a1,a2);swap(i1,i2);
      while not empty(a1) do begin
        v:=a1[i1]; i1:=i1-1;
        for [v,w]∈out(v) do begin
          if dist(v)+length(v,w) < dist(w) then begin
            dist(w) := dist(v)+lenght(v,w);
            p(w) := v;
            if not w∈a2 then begin
              i2:=i2+1;a2[i2]:=w
             end
          end
        end
      end
    end
  end;
```

ｐａｒｔｉｔｉｏｎ（i,j,a）を、ａ［ｉ］、ａ［ｊ］を分割して、中央値より小さいキーが右に、大きいキーをもつののが左に来るようにし、右のブロックの先頭の位置を返すものとする。 ただし、キーはｄｉｓｔの値とする。このとき、ｌａｚｙｓｏｒｔとして次の４つのものを考える。

１． 2ブロック法

```
procedure lazysort(i:integer;a:array);
   var dummy:integer;
   begin
     if i>size then
       dummy := partition(1,i,a)
   end;
```

２． クイック法

```
procedure lazysort(i:integer;a:array);
  procedure lazyquick(m,n:integer);
    var k:integer;
    begin
      if (n-m) > size then
        k:=partition(m,n,a);
        lazyquick(m,k);
        layzquick(k+1,n)
      end;
  begin
    lazyquick(1,i)
  end;
```

３． 前方分割法

```
procedure lazysort(i:integer;a:array);
  var k:integer;
  begin
   if i > size then begin
     k:=i+1;
     while k>size do
        k:=partition(1,k-1)
   end
end;
```

４． 後方分割法

```
procedure lazysort(i:integer;a:array);
  var k:integer;
   begin
    if i > size then begin
     k:=1;
     while((i-k)>size do
       k:=partition(k,i)
    end
  end;
```

この４つのｌａｚｙｓｏｒｔに対応する、配列による実現方法をそれぞれ、２ブロック法、クイック法、前方分割法、後方分割法と呼ぶことにする。定理２．８より次のことが分かる

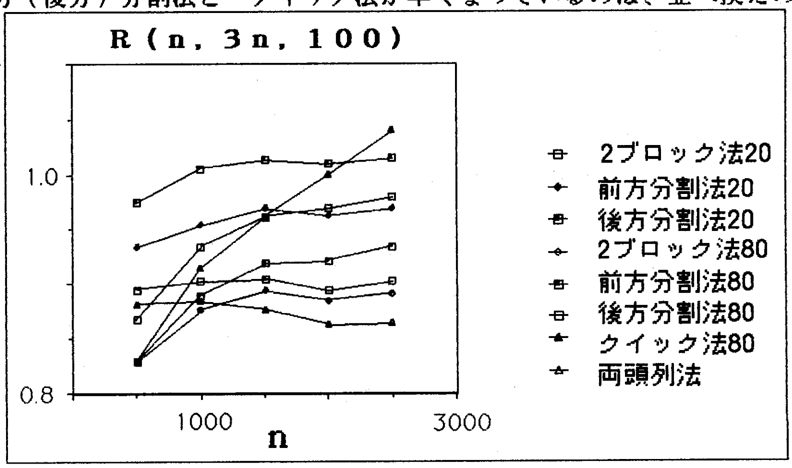
定理３．１　到達可能な負の閉路がなければ、これら４つの方法は必ず停止する。２ブロック法、前方分割法および後方分割法の実行時間は、Ｏ（｜Ｖ｜｜Ｅ｜）であり、クイック法はＯ（｜Ｖ｜｜Ｅ｜ｌｏｇ｜Ｅ｜）である。

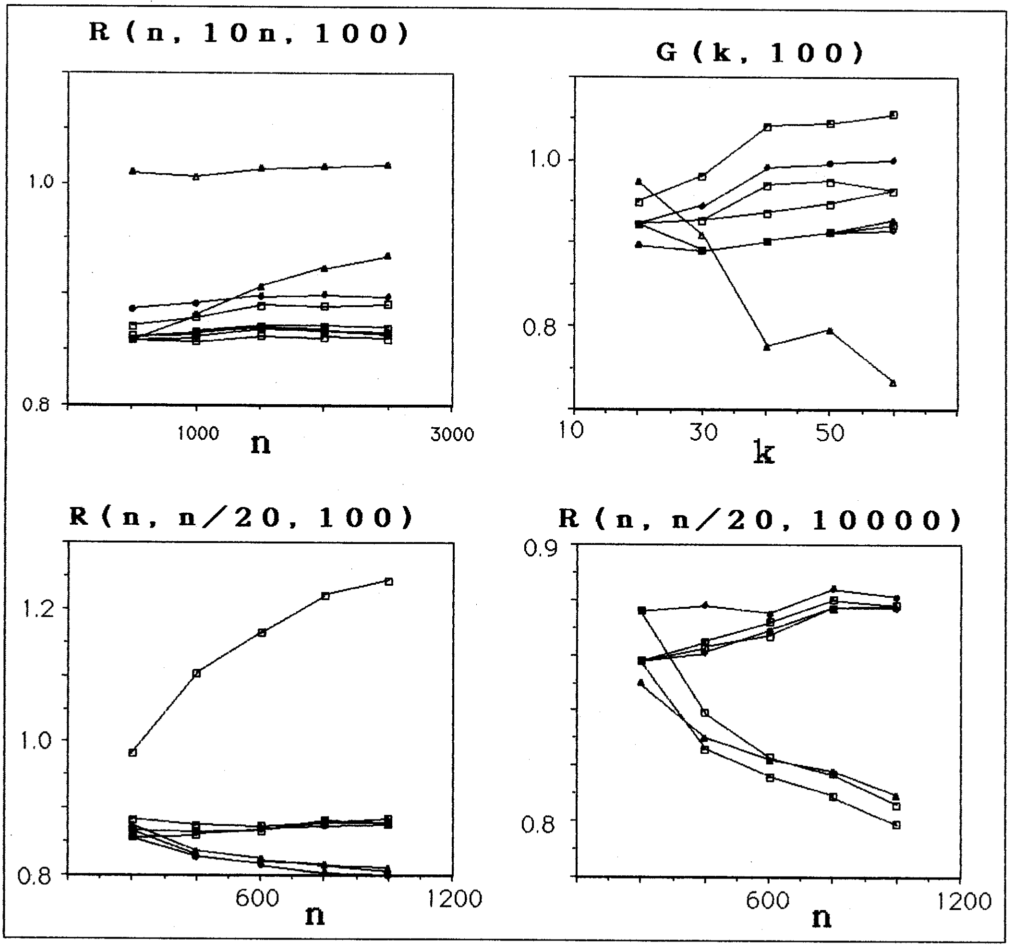
## ４　計算機実験

この章では、前章の４つの方法とＦＩＦＯ法および両頭列法を計算機実験により比較することにする。グラフを表わすデータ構造は［伊理］で示されている標準的データ構造を用いる。実験する方法は、２ブロック法、前方分割法、後方分割法ではｓｉｚｅを、２０と８０にしたもの、クイック法ではｓｉｚｅを８０にしたもの、ＦＩＦＯ法そして両頭列法の合計９つです。プログラムはすべてＣ言語で書き、プログラムの主要部分ではｐａｒｔｉｔｉｏｎとｌａｚｙｑｕｉｃｋを除き、関数（サブルーチン）を用いないようにした。また、計算機はＭｉｃｒｏ　Ｖａｘ（Ｕｌｔｒｉｘ）を使用した。データとして使用するグラフとして次の２つのタイプのものを考える。

（１）ランダムグラフＲ（ｎ，ｍ，ｌ）
ｎ個の頂点の集合にたいして、ｎ（ｎ－１）個の頂点のすべての組み合わせの中から、ｍ個をランダムに選び辺としたもの。各辺の長さは、１からｌまでのランダムな整数と|する
（２）格子状グラフＧ（ｋ，ｌ）
縦横それぞれｋ個の格子点からなる頂点数ｋ２、辺数４ｋ（ｋ－１）のグラフで、各辺の長さは１からｌまでのランダムな整数とする。

そこで、グラフとして、
Ｒ（ｎ，３ｎ，１００），Ｒ（ｎ，１０ｎ，１００）
（ｎ＝　５００,１０００,２０００,２５００）
Ｒ（ｎ,$n^2$／２０，１００），Ｒ（ｎ，$n^2$／２０，１００００）
（ｎ＝　２００,４００,６００,８００,１０００）
Ｇ（ｋ，１００）
（ｋ＝２０,３０,４０,５０，６０）
をもちいる。１つのグラフについて、出発点をランダムに１０個選びそれぞれの出発点に対して最短経路問題を解き、計算時間を調べる。これを、各グラフのそれぞれのｎに対して２５回繰り返し、平均の計算時間を計算する。これらの結果を、ＦＩＦＯ法の計算時間を１とし表わしたものが、次の一連のグラフです。
配列を用いた４つの方法の中で、大きいブロックと小さいブロックの２にしか並べ換えない２ブロック法がいがいに速いのがおもしろい。Ｇ（ｎ，$n^2$／２０，１００または１０００）に対しては、前方（後方）分割法と　クイック法が早くなっているのは、並べ換えの効果が顕著に現われているからだと思われる。これら４つの方法は、一般的なグラフにたいしたは、有効ではないと思われる。しかし、頂点数に比べ辺数が多いグラフのような複雑なグラフに対しては、ある程度有効であると思われる。






## 謝辞

計算機を自由に使えるように配慮をしていただいた、学習院大学数学科教授飯高　茂氏とそのゼミの皆さんに感謝します。

## 参考文献

［Ｉｍａｉ＆Ｉｒｉ］:Practical efficiencies of existing shortest-path algorithms and a new bucket algorithm,J. of the Operations Research Society of Japan,Vol.27,No.1,pp.43-57
［伊理］（１９８６）：計算幾何学と地理情報処理、共立出版.
［Ｒ．Ｅ．Ｔａｒｊａｎ］（１９８３）:Data Structures and Network Algorithms , Society for Industrial and Applied Mathematics.