RA-002

有向グラフに対する多項式時間省メモリ最短経路アルゴリズム

Polynomial Time Memory Constrained Shortest Path Algorithms for Directed Graphs

今井 達也 †

Tatsuya Imai

1. はじめに

二頂点対最短経路問題（以下，最短経路問題）は，与えられた辺コスト付きグラフおよびその頂点 s,t に対して，s から t への最小コストの経路を求める問題である．

最短経路問題に関する重要な課題の一つとして，アルゴリズムの省メモリ化が挙げられる．しかし，特にグラフ族を限定せずに一般的な辺コスト付き有向グラフを対象にした場合，最悪領域計算量がグラフの頂点数 n に対して $o(n)$ であるような多項式時間アルゴリズムはこれまでに全く見つかっていない．最短経路問題の特殊化の一つである連結性判定問題に関しては，一般的な有向グラフに対する唯一のアルゴリズムが 90 年代に Barnes らによって提案されており [2]，これを少し改良すれば全ての辺コストが単位コストであるような有向グラフに対する多項式時間 $o(n)$ 領域最短経路アルゴリズムが得られる．ただし，このアルゴリズムの領域計算量は n とは無関係な任意の定数 $0 < \epsilon < 1$ について $\omega(n^{\epsilon})$ である．なお，この計算量は NNJAG と呼ばれる計算モデルの下ではほぼタイトな値であることが知られている [5].

本論文では Barnes らのアルゴリズムを辺コストに関して拡張し，以下の三つの提案を行う．

- Barnes らの有向グラフ上の連結性判定アルゴリズムを拡張して，有向グラフ上の最短経路問題のための省メモリアルゴリズムの枠組みを提案する．この枠組みは，省メモリの代わりに経路を構成する頂点の個数に制限がかかってしまうような最短経路アルゴリズムがあるときに，これをサブルーチンとして使用することで二頂点間の真の最短経路を省メモリで計算するアルゴリズムである．
- 一般的な有向グラフに対する具体的なサブルーチンを与える．これによって，ある定数 c が存在して，領域計算量 $O(n/2^{c\sqrt{\log n}})$ の多項式時間最短経路アルゴリズムが得られる．c はある程度の任意性を持つが，n とは無関係な定数でないときは時間計算量が n の多項式程度に収まらない．これらの計算量は Barnes らの連結性判定アルゴリズムの計算量と等価であり，特に全ての辺コストが単位コストの場合にはこのアルゴリズム自体が Barnes らのアルゴリズムと等価なものになる．
- 提案手法の特殊化の一例として，多次元格子グラフに対する多項式時間 $o(n)$ 領域アルゴリズムを得るためのサブルーチンを提案する．例えばグラフが二次元格子のとき，このアルゴリズムの領域計算量は $O(n^{2/3})$ である．

なお，限定されたグラフ族に限った場合でも，多項式時間 $o(n)$ 領域アルゴリズムが得られている例は少ない．Asano らは近年，任意の $\epsilon > 0$ について領域計算量 $O(n^{1/2+\epsilon})$ を達成するような二次元格子グラフのための最短経路アルゴリズム族を提案した [1]．本論文で提案する多次元格子グラフのためのアルゴリズムは，格子の次元が 2 のとき，Asano らのアルゴリズムの中で最も高速なものと等価な領域計算量を達成する．また，Imai らは有向平面グラフのための連結性判定アルゴリズムを提案した [6]．Imai らのアルゴリズムは簡単な変更によって辺コスト付き有向平面グラフのための多項式時間 $o(n)$ 領域最短経路アルゴリズムに拡張できる．

2. 準備

議論を簡単化するため，アルゴリズムに与えられるグラフは多重辺を持たない単純有向グラフであるものとする．また，始点からグラフの各頂点へは唯一の最短経路が存在するものと仮定する．なお，各頂点への経路がただ一つである必要はない．更に，始点から各頂点への最短経路のコストは互いに相異なっているものとする．実際には，この三つの制約を取り払っても同様の（ただし煩雑な）議論を展開することができる．また，簡単のため領域計算量はワード単位で取り扱うものとする．

与えられた有向グラフを $G = (V, E)$，辺コスト関数を $c : E \to \mathbb{Q}^+$，始点および終点を $s, t \in V$ とおく．また，グラフの頂点数 $|V|$ を n とおく．アルゴリズムの目的は s から t へのコスト最小の有向経路を求めることである．グラフ $G = (V, E)$ 上の任意の頂点 $a \in V$ から $b \in V$ への長さ l の有向経路とは，$\forall i \in [0, \cdots l-1], e_i = (v_i, v_{i+1})$ および $v_0 = a, v_l = b$ を満たす G の部分グラフ $(\{v_0, \cdots, v_l\}, \{e_0, \cdots e_{l-1}\})$ である．以下の議論では有向経路を単に経路と呼ぶ．また経路の頂点列ないし辺列を経路と同一視する場合がある．経路のコストは経路に含まれる各辺のコストの和 $\sum_{i=0}^{l-1} c(e_i)$ によって定義される．以下では頂点 $a \in V$ から $b \in V$ へのコスト最小の経路を a から b への最短経路と呼ぶ．後の議論では最短経路に関する次の命題が重要になる．

†東京工業大学 大学院情報理工学研究科

Copyright © 2013 by Information Processing Society of Japan and
The Institute of Electronics, Information and Communication Engineers
All rights reserved.

命題 1 グラフ $G = (V, E)$ の任意の頂点 $v_0 \in V$ から $v_l \in V$ への長さ $l \geq 0$ の最短経路 $(\{v_0, \cdots, v_l\}, \{e_0, \cdots e_{l-1}\})$ を内点 $v_{l'}$（ただし $\forall l' \in \{0, \cdots, l\}$ ）で分割して作られた二つの経路は，v_0 から $v_{l'}$ への長さ l' の最短経路および $v_{l'}$ から v_l への長さ $l - l'$ の最短経路である．なお，$v_{l'}$ は v_0 または v_l 自身でもあり得る．

命題 2 グラフ $G = (V, E)$ の任意の頂点 $v_0, v_l \in V$ について，v_0 から v_l への最短経路が頂点 $v_{l'} \in V$ を経由するという仮定の下では，v_0 から $v_{l'}$ への最短経路に $v_{l'}$ から v_l への最短経路をつなげた経路は v_0 から v_l への最短経路になる．逆に，v_0 から v_l への最短経路が頂点 $v_{l'} \in V$ を経由しないという仮定の下では，v_0 から $v_{l'}$ への最短経路に $v_{l'}$ から v_l への最短経路をつなげた経路は v_0 から v_l への最短経路にはならない．

なお，本論文では経路の長さおよびコストがそれぞれ重要な役割を果たすため，両者を混同しないように注意せよ．また，頂点 $a \in V$ から $b \in V$ への「**長さ X 以下の**」最短経路のコストを $\mathrm{cp}(a, b, X)$，その実際の長さを $\mathrm{lp}(a, b, X)$ によって表すこととする [1]．$\mathrm{cp}(a, b, n)$ および $\mathrm{lp}(a, b, n)$ は a から b への通常の最短経路のコストと長さを表す．a から b への長さ X 以下の経路が存在しない場合には，$\mathrm{cp}(a, b, X)$ および $\mathrm{lp}(a, b, X)$ は便宜的に ∞ を返すこととする．最後に，出力の記述長が作業領域の大きさを上回る場合を考慮して，提案アルゴリズムは他の省メモリアルゴリズムと同様に出力を書き込み限定のメモリに打ち出していくものとする．

3. アルゴリズムの枠組み

本節では，提案アルゴリズムの一般的な枠組みを示す．提案アルゴリズムを実際に動作させるためにはあるパラメータとサブルーチンが必要であり，アルゴリズムの最終的な計算量はこれらに依存する．後で述べるように，本論文で提案する一般的な有向グラフのためのアルゴリズムは Barnes らのアルゴリズムと同様の大きさまでしか領域を削減できないが，特殊なグラフを取り扱う場合にはこのサブルーチンを工夫しパラメータをチューニングすることによって領域計算量を更に削減することができる．

まずパラメータ $\lambda \in \{1, \cdots, n\}$ を一つ固定し，頂点 $a, b \in V$ および λ を入力に取って対 $(\mathrm{cp}(a, b, \lambda), \mathrm{lp}(a, b, \lambda))$ を返すサブルーチン $\mathrm{SP}(a, b, \lambda)$ の存在を仮定する．一言で述べると，提案アルゴリズムは，ある $j \in \{0, \cdots, \lambda - 1\}$ について，始点からの最短経路の長さが $j, j + \lambda, j + 2\lambda, \cdots$ の頂点に対してのみその最短経路のコストと長さを記憶することにより作業領域を節約する，という最短経路アルゴリズムである．SP は保持している頂点から別の頂点への最短経路を補完するために使用される．ここで j, λ に対してアルゴリズムが保持する頂点の集合を

$$V(j, \lambda) := \{v \in V \mid \mathrm{lp}(s, v, n) \equiv j \mod \lambda\} \quad (1)$$

とおく．j の値によって $V(j, \lambda)$ の要素数は変化するが，任意の有向グラフ G および任意の始点 s について，$|V(j, \lambda)| \leq \lfloor n/\lambda \rfloor$ を満たす j は $\{0, \cdots, \lambda - 1\}$ の中に必ず一つ以上存在する．アルゴリズムは，とりあえず j を固定して計算を進めて，$V(j, \lambda)$ の要素数が $\lfloor n/\lambda \rfloor$ を超えることが確定したら計算を中断し別の j で再度計算を始める，という手続きを行うことによって，記憶する頂点の個数を常に $\lfloor n/\lambda \rfloor$ 以下に保ちながら動作する．

固定された j に対してはアルゴリズムは次のように動作する．現在の j で計算を始めてからこれまでに $V(j, \lambda) \cup \{s\}$ の要素であることが判明した頂点の集合を S とおき，S の要素を一つずつ増やしていく．アルゴリズムは始点 s から S の各要素への最短経路のコストと長さを記憶し，S に新たな頂点 u を加える際にはそれらを利用して s から u への最短経路のコストと長さを計算・保持する．以下では $\forall x \in S$ について，アルゴリズムが記憶している s から x への最短経路のコストを $C[x]$，長さを $L[x]$ とおくこととする．S の初期値は $\{s\}$ であり，$C[s] = 0, L[s] = 0$ である．

S に一つ要素を加える際には，まず全ての $u \in V \setminus S$ に対して次の計算を行う．それぞれの $x \in S$ について，SP を用いてコスト $C[x] + \mathrm{cp}(x, u, \lambda)$ と長さ $L[x] + \mathrm{lp}(x, u, \lambda)$ を計算し，そのうち $C[x] + \mathrm{cp}(x, u, \lambda)$ が最も小さくなる x に対する値をそれぞれ c_{su}, l_{su} とおく．これは，s から S のいずれかの頂点 x への最短経路に高々長さ λ 以下の x から u への最短経路を加えたものの中で，コストが最小になる x を求めていることに等しい．アルゴリズムはそれぞれの $u \in V \setminus S$ に対して上記の計算を行い，l_{su} が $l_{su} \equiv j \mod \lambda$ を満たす u の中で，c_{su} が最小のものを S に新たに付け加える．$C[u]$ および $L[u]$ には c_{su} および l_{su} を保存する．u および x を探す際には，常に暫定で最小の要素だけを覚えておけばメモリ消費に関する問題は発生しない．

アルゴリズムは，それ以上 S の要素が増やせなくなるまで上記の手続きを繰り返す．この繰り返しが完了した時点で，S に $V(j, \lambda)$ の全ての要素が収集され，同時に s から $V(j, \lambda)$ の各要素への最短経路のコストと長さが C および L に保存されたことになる（アルゴリズム全体の正しさは補題 1 で証明する）．s から t への経路が一つ以上存在するならば，命題 1 より，$\mathrm{lp}(s, u, n) \equiv j \mod \lambda$ かつ $\mathrm{lp}(u, t, n) \leq \lambda$ を満たす $u \in V$ が，s から t への最短経路上に必ず一つ以上存在する．よって，$S = V(j, \lambda) \cup \{s\}$ および命題 2 より，（S を拡張するときと同様に）s からそれぞれの $u \in S$ への最短経路に u から t への高々長さ λ 以下の最短経

[1] a から b への最短経路（コストが $\mathrm{cp}(a, b, X)$ の経路）が複数ある場合には $\mathrm{lp}(a, b, X)$ はその中の最短の長さと定義する．

Copyright © 2013 by Information Processing Society of Japan and
The Institute of Electronics, Information and Communication Engineers
All rights reserved.

路を付け加えて，その中で最もコストの小さなものを求めることによって s から t への最短経路のコストを計算することができる．後は，ダイクストラ法 [4] などと同様にして s から t への最短経路を逆順に出力していけばよい．以上の手続きをまとめた擬似コードを **Algorithm 1**，**Algorithm 2** に示す[2]．便宜的に $\infty < \infty$ および $\infty \equiv j \mod \lambda$ は false とする．S は集合を，C, L は写像を表す何らかのデータ構造である．上で述べた計算打ち切りにより，3 行目のループは高々 $\lfloor n/\lambda \rfloor$ 回で十分である．

Algorithm 1（入力 G, c, s, t，パラメータ λ）

```
 1: for j = 0, ··· , λ − 1 do
 2:     S ← {s}; C[s] ← 0; L[s] ← 0;
 3:     for i = 1 to ⌊n/λ⌋ do
 4:         u ← null; c_su ← ∞; l_su ← ∞;
 5:         for v ∈ V \ S do
 6:             c_sv ← ∞; l_sv ← ∞;
 7:             for x ∈ S do
 8:                 (cp, lp) ← SP(x, v, λ);
 9:                 if C[x] + cp < c_sv then
10:                     c_sv ← C[x] + cp; l_sv ← L[x] + lp;
11:                 end if
12:             end for
13:             if l_sv ≡ j mod λ then
14:                 if c_sv < c_su then
15:                     u ← v; c_su ← c_sv; l_su ← l_sv;
16:                 end if
17:             end if
18:         end for
19:         if u ≠ null then
20:             if |S| + 1 > ⌊n/λ⌋ + 1 then
21:                 i のループを中断して次の j へ;
22:             end if
23:             S ← S ∪ {u}; C[u] ← c_su; L[u] ← l_su;
24:         else
25:             i のループを終了して 28 行へ;
26:         end if
27:     end for
28:     t に対して 6 ～ 12 行目の手続きを適用し，C[t] ← c_su;
29:     Algorithm 2 を用いて最短経路を出力して終了;
30: end for
```

Algorithm 1 の正当性は以下の補題によって示される．

補題 1 $V(j,\lambda) \cup \{s\}$ の中で s からの最短経路のコストが k 番目に小さい頂点を $v_{j,k}$ とおく．**Algorithm 1** の 3 行目において，現在の i に対し，$S = \{v_{j,1}, v_{j,2}, \cdots, v_{j,i}\}$ が成り立つ．また，このとき $\forall u \in S$ について $C[u] = \mathrm{cp}(s,u,n), L[u] = \mathrm{lp}(s,u,n)$ が成立している．

証明 j を固定して帰納法によって証明を行う．以下では $v_{j,k}$ を単に v_k と書くこととする．

[2]なお，始点からある頂点への最短経路が複数存在する場合には，SP の二番目の戻り値は最短経路の最短の長さである必要がある．また，**Algorithm 1** の 9 行目および 14 行目に最短経路の長さによるタイブレークを加える必要がある．

Algorithm 2（入力 $G, c, s, t, \lambda, S, C, L$）

```
 1: print t;
 2: v ← t;
 3: while v ≠ s do
 4:     for u ∈ S \ {v} do
 5:         (cp, lp) ← SP(u, v, λ);
 6:         if C[u] + cp = C[v] then
 7:             v′ ← v;
 8:             while v′ ≠ u do
 9:                 for u′ ∈ V do
10:                     (cp′, lp′) ← SP(u, u′, λ);
11:                     if (u′, v′) ∈ E and cp′ + c((u′, v′)) = cp
                        then
12:                         print (u′, v′), u′;
13:                         v′ ← u′;
14:                         cp ← cp′;
15:                         break;
16:                     end if
17:                 end for
18:             end while
19:             v ← u;
20:             break;
21:         end if
22:     end for
23: end while
```

$i = 1$ のときは，S の要素は s のみであり $C[s] = 0, L[s] = 0$ なので明らか．

$|V(j,\lambda) \cup \{s\}| \geq k+1$ かつ $k \leq \lfloor n/\lambda \rfloor$ を仮定し（すなわち，ループが $i = k+1$ 番目の 3 行目に到達するものと仮定し），帰納法の仮定が $i = k$ 番目まで成立しているとき，$i = k+1$ 番目が成立することを示す．

まず，$i = k$ のループにおいて，5 行目の v が v_{k+1} のとき，7 行目から 12 行目までのループが終了した時点で $c_{sv} = \mathrm{cp}(s, v_{k+1}, n), l_{sv} = \mathrm{lp}(s, v_{k+1}, n)$ が成立していることを示す．s から v_{k+1} への経路が一つ以上存在するのであれば，命題 1 より，s から v_{k+1} への最短経路は現在の S の中に少なくとも一つ以上の内点を持ち，s から内点への最短経路と内点から v_{k+1} への最短経路に分割することができる．この内点の中で最も L の値が大きいものを $v' \in S$ とおく．アルゴリズムは s から v_{k+1} への最短経路の内点となりうる S の要素を網羅的に探索しており，かつ帰納法の仮定より $C[v'] = \mathrm{cp}(s, v', n), L[v'] = \mathrm{lp}(s, v', n)$ なので，命題 2 より，$\mathrm{lp}(v', v_{k+1}, n) \leq \lambda$ ならば $c_{sv} = \mathrm{cp}(s, v_{k+1}, n), l_{sv} = \mathrm{lp}(s, v_{k+1}, n)$ が成立する．もし $\mathrm{lp}(v', v_{k+1}, n) > \lambda$ であると仮定すると，命題 1 および命題 2 より，v' から v_{k+1} への最短経路上のある内点 $v'' \in V \setminus (S \cup \{v_{k+1}\})$ が存在して $\mathrm{lp}(s, v'', n) \equiv j \mod \lambda$ が成り立つ．また，辺コストが 0 より大きいため，経路のコストは経路を延ばすほど加算的に増えていくことから，$\mathrm{cp}(s, v'', n) < \mathrm{cp}(s, v_{k+1}, n)$ が成り立つ．しかし，これは v_{k+1} が $V(j,\lambda) \cup \{s\}$ の中で $k+1$ 番目にコストの小さな s からの最短経路を持つという仮定に矛盾する．よって，$\mathrm{lp}(v', v_{k+1}, n) \leq \lambda$，すなわち，5 行目の v が v_{k+1} のとき，7 行目から 12 行目までのループが終了した時点では $c_{sv} = \mathrm{cp}(s, v_{k+1}, n), l_{sv} = \mathrm{lp}(s, v_{k+1}, n)$ が

Copyright © 2013 by Information Processing Society of Japan and
The Institute of Electronics, Information and Communication Engineers
All rights reserved.

成立している．また以上より，もし $i=k$ の 23 行目の u が v_{k+1} であるならば，$i=k+1$ のループの先頭において $v_{k+1} \in S, C[v_{k+1}] = \mathrm{cp}(s, v_{k+1}, n), L[v_{k+1}] = \mathrm{lp}(s, v_{k+1}, n)$ が成り立っている．

次に，$i=k$ 番目の 23 行目の u が v_{k+1} であることを背理法によって示す．$i=k$ 番目の 23 行目の u は v_{k+1} ではないと仮定する．上で述べた議論は v_{k+1} が u に選ばれたかどうかには関係が無いため，$i=k$ において 5 行目の v が v_{k+1} のときは，7 行目から 12 行目までのループ終了後に $c_{sv} = \mathrm{cp}(s, v_{k+1}, n), l_{sv} = \mathrm{lp}(s, v_{k+1}, n)$ が成立しているはずである．すなわち，13 行目および 14 行目で，$l_{sv} \equiv j \mod \lambda$ および $c_{sv} \leq \mathrm{cp}(s, v_{k+1}, n)$ を満たす別の頂点 $u \in V \setminus S$ が選ばれたことになる．

- $u \in V(j, \lambda)$ のとき，$\mathrm{cp}(s, v_{k+1}, n) \geq c_{su} \geq \mathrm{cp}(s, u, n)$ より，s から各頂点への最短距離が互いに相異なるという仮定および v_{k+1} が $V(j,\lambda) \cup \{s\}$ の中で $k+1$ 番目にコストの小さい s からの最短経路を持つという仮定に矛盾する．
- $u \notin V(j, \lambda)$ のとき，13 行目の条件分岐を通過したことと $u \notin V(j,\lambda)$ より，今得られている s から u への経路は s から u への最短経路ではない．この経路は現在の S のいずれかの頂点から長さ λ 以下の経路で u に至るものの中で最短なので，これが s から u への最短経路でないならば，s から u への最短経路は現在の S のある頂点から $\lambda+1$ 個以上の頂点を通る経路である．よって前の議論と同様に，命題 1 および命題 2 より，ある内点 $v''' \in V \setminus (S \cup \{u\})$ が存在して，$\mathrm{lp}(s, v''', n) \equiv j \mod \lambda$ が成り立つ．また，同じく経路のコストは加算的なので $\mathrm{cp}(s, v''', n) < \mathrm{cp}(s, u, n)$ が成り立つが，$\mathrm{cp}(s, v_{k+1}, n) \geq c_{su} \geq \mathrm{cp}(s, u, n)$ よりこれは v_{k+1} が $V(j,\lambda) \cup \{s\}$ の中で $k+1$ 番目にコストの小さい s からの最短経路を持つという仮定に矛盾する．

よって，$i=k$ のときの 23 行目において u は v_{k+1} である．

以上より，題意は任意の i で真である． □

また，**Algorithm 2** の正当性については次のことが言える．まず，もし 7 行目において $\mathrm{lp}(u, v, n) \leq \lambda$ であれば，7 行目から 18 行目の手続きは，u から u' への最短経路のコストを動的に計算していることを除いて，単にダイクストラ法と全く同じ処理を行って u から v への最短経路を逆向きに出力しているにすぎない．また，**Algorithm 1** が 29 行目に到達した時点で，$S = V(j,\lambda) \cup \{s\}$ が成立しているため，$\forall v \in \{t\} \cup V(j,\lambda)$ は s からの最短経路の内点 u を S の中に必ず一つ以上持ち，また内点のうち少なくとも一つの要素において $\mathrm{lp}(u, v, n) \leq \lambda$ が成り立つ[3]．ある頂点がそのような内点かどうかを調べるためには，命題 2 よりコストを調べれば十分である．**Algorithm 2** の 3 行目から 6 行目の処理は，現在の v に対して上記の内点を探すという処理を $v=s$ になるまで繰り返しているだけなので，7 行目以降の処理と合わせて，**Algorithm 2** は s から t への最短経路を正しく出力する．

最後に提案アルゴリズムの計算量について述べる．SP の時間計算量・領域計算量を $\mathcal{T}_{SP}(\lambda), \mathcal{S}_{SP}(\lambda)$ とおく．また，S, C, L の大きさはその要素数に比例し，読み書きは定数時間で行えると仮定する．まず，**Algorithm 2** の時間計算量を考える．3 行目の WHILE ループは高々 $|S|$ 回しか反復されず，**Algorithm 2** に到達した j において $S = O(\frac{n}{\lambda})$ なので，$O(\frac{n}{\lambda})$ 回程度しか反復されない．4 行目の FOR ループは手前の WHILE ループ一回に対し最大で $|S|-1$ 回反復するため，5 行目のサブルーチン呼び出しは高々 $O(n^2/\lambda^2)$ 回程度しか行われない．8 行目の WHILE ループは複数のループの内側にあるが，実際には **Algorithm2** の実行一回に対して高々 n 回しか反復されない（でなければ出力経路の長さが n を超える）．9 行目の FOR ループは 8 行目の WHILE ループ一回に対し最大で n 回反復するため，10 行目のサブルーチン呼び出しは高々 $O(n^2)$ 回である．以上をまとめると **Algorithm 2** の時間計算量は $O(n^2 \mathcal{T}_{SP}(\lambda))$ である．また作業領域は $O(\mathcal{S}_{SP}(\lambda))$ で十分である．**Algorithm 1** のうち 28, 29 行目の **Algorithm 2** を除いた部分の時間計算量は 1, 3, 5, 7 行目のループにより

$$O\left(\lambda \times \frac{n}{\lambda} \times n \times \frac{n}{\lambda} \times \mathcal{T}_{SP}(\lambda)\right) \tag{2}$$

すなわち $O(\frac{n^3}{\lambda}\mathcal{T}_{SP}(\lambda))$ となる．28 行目の処理における時間計算量は 7 行目のループにより $O\left(\frac{n}{\lambda}\mathcal{T}_{SP}(\lambda)\right)$ である．$\lambda \leq n$ であるため，**Algorithm 2** の時間計算量と合わせて，**Algorithm 1** 全体の時間計算量は $O(\frac{n^3}{\lambda}\mathcal{T}_{SP}(\lambda))$ となる．領域計算量は S, C, L の使用領域と $\mathcal{S}_{SP}(\lambda)$ を合わせて $O(\frac{n}{\lambda} + \mathcal{S}_{SP}(\lambda))$ となる．

4. 一般的な有向グラフのためのサブルーチン

ある正整数 $\lambda = o(n)$ について，一般的な有向グラフの任意の二頂点 $a, b \in V$ に対して多項式時間かつ $o(n)$ 領域で動作するサブルーチン $\mathrm{SP}(a, b, \lambda)$ について解説する．本節で述べるサブルーチンは Barnes らの連結性判定アルゴリズムのためのサブルーチン（[2] の 3 節）を応用したもので，基本的には総当たり的なアルゴリズムである．Barnes らのものと同様にまずは非再帰版について解説したのち，それを拡張する形で再帰版について述べる．

[3] 2 節の仮定より長さが λ 以下の内点はただ一つであるが，仮定を外して複数存在する場合には別にどれを使ってもよい．

Copyright © 2013 by Information Processing Society of Japan and
The Institute of Electronics, Information and Communication Engineers
All rights reserved.

4.1. 非再帰版サブルーチン

入力グラフ $G=(V,E)$ の各頂点にコスト 0 の自己ループをつけたグラフを $G=(V,E')$ とおく．このアルゴリズムではまず，ある正整数 $k\in\{1,\cdots,n\}$ を用いて，頂点集合 V を k 個の互いに素で大きさが可能な限り均等な集合 $D_1,\cdots,D_k$ に分割する．全ての集合の要素数が $\lfloor\frac{n}{k}\rfloor$ 以上でさえあれば，分け方は何でも良い．サブルーチンは，インデクス列 $i=(i_0,\cdots,i_L)\in\{1,\cdots,k\}^{L+1}$ ごとに，ベルマンーフォード法 [3] と類似の反復法で最短経路のコストと長さを計算する．具体的には，大きさ $\lceil\frac{n}{k}\rceil$ の配列を 2 つ（正確にはコストと長さで 2 つずつ）用意して，j 回目の反復では片方の配列が D_{i_j} の各要素の値に対応し，もう片方の配列が $D_{i_{j+1}}$ の各要素の値に対応しているものとして，ベルマンーフォード法と同様の値更新を行う．これによって，現在の i において $v_0\in D_{i_0},\cdots,v_L\in D_{i_L}$ を満たすような長さ L の頂点列 $v_0,\cdots,v_L$ を用いて作られる経路のうち，D_{i_L} における終点毎の最短経路のコストと長さが計算される．グラフに自己ループを加えるのは，長さ L 未満の経路も同時に考慮するためである．$k=1$ のとき，このアルゴリズムはベルマンーフォード法に単に最短経路の長さの計算を加えたものと等価である．k による V の分割は，単に計算を複数回に分割しているにすぎず，時間を消費することで代わりにメモリが節約されている．この手続きをより詳細に解説した擬似コードを **Algorithm 3** に示す[4]．

Algorithm 3 を実行すると，パラメータ k に対して，2 行目の FOR ループは k^{L+1} 回，9 行目の FOR ループは L 回，12, 13 行目の FOR ループはそれぞれ $O(n/k)$ 回反復される．また，各配列の大きさが $\lceil n/k\rceil$ であることから，配列初期化の時間計算量は $O(n/k)$ である．よって，このアルゴリズムをサブルーチン SP として使用するとき，

$$\mathcal{T}_{SP}(L)=O\left(k^{L+1}\times L\times\left(\frac{n}{k}\right)^2\right) \tag{3}$$

が成立する．またインデクス i および配列 CP_0,CP_1,LP_0,LP_1 を記憶するメモリを合わせて，領域計算量は

$$\mathcal{S}_{SP}(L)=O\left(L+\frac{n}{k}\right) \tag{4}$$

ワードとなる．

4.2. 再帰版サブルーチン

まず，**Algorithm 3** を単純に r 段再帰させることにより，最大長さ L^r の経路を **Algorithm 3** の r 乗の時間と r 倍のメモリで調べることができる．紙面の都合で厳密な擬似コードは省略するが，**Algorithm 3** の 14 行目から 23 行目にかけて，関数 c によるコスト

[4]なお，入力グラフの制約を緩和する際には，**Algorithm 1** と同様に経路の長さに関するタイブレークを加える必要がある．

Algorithm 3 非再帰版 SP(a,b,L)

```
 1: cp ← ∞, lp ← ∞;
 2: for i = (i_0, ··· , i_L) ∈ {1, ··· , k}^{L+1} do
 3:   if a ∉ D_{i_0} or b ∉ D_{i_L} then
 4:     continue;
 5:   end if
 6:   長さ ⌈n/k⌉ の配列 CP_0, CP_1, LP_0, LP_1 を用意;
 7:   CP_0 ← ∞; LP_0 ← ∞;
 8:   CP_0[a] ← 0; LP_0[a] ← 0;
 9:   for l = 0 to L − 1 do
10:     src ← (l + 0 mod 2); dst ← (l + 1 mod 2);
11:     CP_(dst) ← ∞; LP_(dst) ← ∞;
12:     for x ∈ D_{i_l} do
13:       for y ∈ D_{i_{l+1}} do
14:         if (x, y) ∈ E' then
15:           if CP_(dst)[y] > CP_(src)[x]+c((x, y)) then
16:             CP_(dst)[y] ← CP_(src)[x] + c((x, y));
17:             if x ≠ y then
18:               LP_(dst)[y] ← LP_(src)[x] + 1;
19:             else
20:               LP_(dst)[y] ← LP_(src)[x];
21:             end if
22:           end if
23:         end if
24:       end for
25:     end for
26:   end for
27:   if cp > CP_(L mod 2)[b] then
28:     cp ← CP_(L mod 2)[b]; lp ← LP_(L mod 2)[b];
29:   end if
30: end for
31: return (cp, lp);
```

値や長さの定数 0・1 の代わりに，単に自分自身を呼び出して計算した値を足し合わせればよい．このときの時間計算量は

$$O\left(\left(k^{L+1}\times L\times\left(\frac{n}{k}\right)^2\right)^r\right)=O\left(k^{r(L-1)}\times L^r\times n^{2r}\right) \tag{5}$$

となる．

しかし，時間計算量において頂点数 n が r 乗されていることから，サブルーチンの時間計算量を多項式時間に保つためには定数段しか再帰することができない．また，単純に **Algorithm 3** を r 段再帰しただけでは 12 行目から 25 行目のループにおいて，実質的に同じ計算を繰り返している部分が多々あり無駄が残る．そこで，この無駄を解消したアルゴリズムを **Algorithm 4** に示す．**Algorithm 4** は単一の始点から終点への最短経路のコストと長さを計算する代わりに，インデクス i_{in},i_{out} を用いて，$D_{i_{in}}$ のいずれかの頂点から $D_{i_{out}}$ の各頂点への最短経路のコストと長さを同時に計算する．再帰を用いて重複的なインデクス列を枝刈りしている以外は **Algorithm 4** は実質的に **Algorithm 3** と同じことをしているだけなので，**Algorithm 4** に初期化処理を加えた **Algorithm 5** は a から b への長さ L^r 以下の最短経路のコストと長さを正しく返す．**Algorithm 4** を r 段繰り返すとき，時間計算量 $O\left(k^{r(L-1)}\times L^r\times n^2\right)$ となる．また領域計算量は単純

Copyright © 2013 by Information Processing Society of Japan and
The Institute of Electronics, Information and Communication Engineers
All rights reserved.

Algorithm 4 SPR$(i_{in}, i_{out}, L, r, CP_{in}, LP_{in})$

```
1: 長さ ⌈n/k⌉ の配列 CP_out, LP_out を用意;
2: CP_out ← ∞, LP_out ← ∞;
3: for i = (i_0, ··· , i_L) ∈ {1, ··· , k}^{L+1} do
4:    if i_0 ≠ i_in or i_L ≠ i_out then
5:       continue;
6:    end if
7:    長さ ⌈n/k⌉ の配列 CP_0, CP_1, LP_0, LP_1 を用意;
8:    CP_0 ← CP_in; LP_0 ← LP_in;
9:    for l = 0 to L − 1 do
10:      src ← (l + 0 mod 2); dst ← (l + 1 mod 2);
11:      if r = 0 then
12:         CP_(dst) ← ∞; LP_(dst) ← ∞;
13:         for x ∈ D_{i_l} do
14:            for y ∈ D_{i_{l+1}} do
15:               if (x, y) ∈ E' then
16:                  if CP_(dst)[y] > CP_(src)[x] + c((x, y)) then
17:                     CP_(dst)[y] ← CP_(src)[x] + c((x, y));
18:                     if x ≠ y then
19:                        LP_(dst)[y] ← LP_(src)[x] + 1;
20:                     else
21:                        LP_(dst)[y] ← LP_(src)[x];
22:                     end if
23:                  end if
24:               end if
25:            end for
26:         end for
27:      else
28:         (CP_(dst), LP_(dst)) ← SPR(i_l, i_{l+1}, L, r − 1, CP_(src), LP_(src));
29:      end if
30:   end for
31:   for y ∈ D_{i_out} do
32:      if CP_out[y] > CP_(L mod 2)[y] then
33:         CP_out[y] ← CP_(L mod 2)[y];
34:         LP_out[y] ← LP_(L mod 2)[y];
35:      end if
36:   end for
37: end for
38: return (CP_out, LP_out);
```

な再帰を行う場合と同じく $O\left(r(L+\frac{n}{k})\right)$ となる．

最後に，$\lambda = L^r$ とおいて，**Algorithm 5**（単に再帰の最初の段のための初期化を行って SPR を呼ぶアルゴリズム）を **Algorithm 1** に組み込んだときの最短経路アルゴリズム全体の計算量を示す[5]．最短経路アルゴリズム全体の時間計算量 $\mathcal{T}$ および領域計算量 $\mathcal{S}$ は

$$\begin{aligned}\mathcal{T} &= O\left(\frac{n^3}{L^r} \times \left(k^{r(L-1)} \times L^r \times n^2\right)\right) \\ &= O\left(n^{3+2} \times k^{rL-r}\right), \quad (6)\\ \mathcal{S} &= O\left(\frac{n}{L^r} + r(L+\frac{n}{k})\right) \quad (7)\end{aligned}$$

となる．仮に $L = 2$ を使用するとき，時間計算量の k^r が n の定数次多項式 n^c であるためには

$$r = c\frac{\log n}{\log k} \quad (8)$$

である必要がある．このとき領域計算量は種々の定数を除くと

$$O\left(\frac{n}{2^{c\log n/\log k}} + \frac{n}{k\log k/\log n}\right) \quad (9)$$

となる．この式を最小化するのは困難であるため，$c\log n/\log k = \log k$ となる k を求めると，$k = 2^{\sqrt{c}\sqrt{\log n}}$ となる．これを式 (9) に代入すると，$n/2^{c\log n/\log k} = n/2^{\sqrt{c}\sqrt{\log n}}$ となり，$n/(k\log k/\log n) = n/2^{\sqrt{c}\sqrt{\log n}-\log(\sqrt{c}\sqrt{\log n})}$ となる．定数を適宜処理すると，ある定数 $c' = O(\sqrt{c})$ が存在して

$$\mathcal{S} = O\left(\frac{n}{2^{c'\sqrt{\log n}}}\right) \quad (10)$$

となる．

Algorithm 5 再帰版 SP(a, b, L, r)

```
1: 長さ ⌈n/k⌉ の配列 CP_0, CP_1, LP_0, LP_1 を用意;
2: CP_0 ← ∞; LP_0 ← ∞;
3: CP_0[a] ← 0; LP_0[a] ← 0;
4: i_in ← i s.t. a ∈ D_i; i_out ← i s.t. b ∈ D_i;
5: (CP_1, LP_1) ← SPR(i_in, i_out, L, r, CP_0, LP_0);
6: return (CP_1[b], LP_1[b]);
```

[5] 少なくともパラメータをチューニングする前の計算量は Barnes らの連結性判定アルゴリズムと等価である．パラメータチューニングに関しては本論文では簡単な議論を行うに留めるが，必要であれば [2] を参照せよ．

5. 格子グラフのためのサブルーチン

本節では提案アルゴリズムの特殊化の一例として，多次元格子グラフのためのサブルーチンの作成方法を示す．以下では格子グラフの次元数を d とおく．また，頂点のインデクスから頂点の座標が定数時間で計算できるものとして議論を進める．

格子グラフの場合は，単純に，ダイクストラ法 [4] を用いることによって効率の良いサブルーチンを作ることができる．格子グラフでは，任意の頂点 v について，v を始点とする長さが l 以下のいかなる経路を考えても，v からマンハッタン距離が $l+1$ 以上離れた頂点には到達することができない．すなわち，長さ λ 以下の最短経路を求める際には，始点からのマンハッタン距離が λ 以下の高々 $O(\lambda^d)$ 個の頂点を考えれば十分である．このときのサブルーチンの領域計算量は $O(\lambda^d)$ となり，一例として特にヒープなどを使わないダイクストラ法を考えれば時間計算量は $O(\lambda^{2d})$ となる．

以上のサブルーチンを **Algorithm 1** と組み合わせたとき，最短経路アルゴリズム全体の時間計算量 $\mathcal{T}$ および領域計算量 $\mathcal{S}$ は

$$\begin{aligned}\mathcal{T} &= O(\frac{n^3}{\lambda} \times \lambda^{2d}) \quad (11)\\ \mathcal{S} &= O(\frac{n}{\lambda} + \lambda^d) \quad (12)\end{aligned}$$

Copyright © 2013 by Information Processing Society of Japan and
The Institute of Electronics, Information and Communication Engineers
All rights reserved.

となる．$\mathcal{S}$ を最適化することを考えると，$\lambda = O(n^{1/(d+1)})$ のとき $\mathcal{S}$ は最小となり，このときの計算量はそれぞれ

$$\mathcal{T} = O(n^{3+(2d-1)/(d+1)}) \quad (13)$$
$$\mathcal{S} = O(n^{d/(d+1)}) \quad (14)$$

となる．なお，Asano らのアルゴリズム族はある再帰アルゴリズムの再帰段数を変更することによって得られるものであり，再帰の段数が一段のときの領域計算量は $d = 2$ のときの本論文のアルゴリズムと同じく $O(n^{2/3})$ である．

謝辞

本研究はJSPS 特別研究員奨励費（課題番号 24･8506）の助成を受けたものです．

参考文献

[1] Tetsuo Asano and Benjamin Doerr. Memory-constrained algorithms for shortest path problems. In *Proceedings of the Twenty Third Canadian Conference on Computational Geometry*, pages 315–318, 2011.

[2] Greg Barnes, Jonathan F Buss, Walter L Ruzzo, and Baruch Schieber. A sublinear space, polynomial time algorithm for directed¡ e1¿ s¡/e1¿-¡ e1¿ t¡/e1¿ connectivity. In *Structure in Complexity Theory Conference, 1992., Proceedings of the Seventh Annual*, pages 27–33. IEEE, 1992.

[3] Richard Bellman. On a routing problem. Technical report, DTIC Document, 1956.

[4] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[5] Jeff Edmonds, Chung Keung Poon, and Dimitris Achlioptas. Tight lower bounds for st-connectivity on the nnjag model. *SIAM Journal on Computing*, 28(6):2257–2284, 1999.

[6] T Imai, K Nakagawa, A. Vinodchandran, N.V.and Pavan, and O. Watanabe. An $o(n^{\frac{1}{2}+\epsilon})$ space algorithm for directed planar reachability with polynomial running time. In *Proceedings of the 27th IEEE Conference on Computational Complexity (in press)*, 2013.

Copyright © 2013 by Information Processing Society of Japan and
The Institute of Electronics, Information and Communication Engineers
All rights reserved.