Caterpillar Graph における
独立点集合遷移問題についての研究

北陸先端科学技術大学院大学
情報科学研究科
山田　武
平成 28 年 3 月

修士論文

Caterpillar Graphにおける
独立点集合遷移問題についての研究

1310710　　山田　武

主指導教員　上原　隆平

審査委員主査　上原　隆平

審査委員　平石　邦彦

緒方　和博

北陸先端科学技術大学院大学

情報科学研究科

平成28年2月

概要

グラフにおける独立点集合とは，どの 2 点も辺で結ばれていない頂点の部分集合のことをいう．あるグラフに対し，2 つの独立点集合（各々の独立点をトークンと呼ぶ）II_b と II_r が与えられたとき，独立点集合遷移問題とは II_b から II_r へのグラフの独立点集合の系列が存在するかどうかを判定する問題である．ただし，2 つの連続する独立点集合の間では，定められたルールでトークンを変化させるものとする．このルールの違いによりいくつかの種類が存在する独立点集合遷移問題のうち，本論文では，Token Sliding（移動）を扱う．この問題は理想グラフに対して PSPACE 完全であることが知られているが，グラフ理論における様々な問題と同様，グラフクラスを制限することにより多項式時間で解くことが可能となる．本論文では，caterpillar graph におけるこの問題の線形時間アルゴリズムを示す．また，独立点集合の系列が存在すると判定された場合の最短の遷移系列を求める多項式時間アルゴリズムもあわせて示す．

目次

第1章　序論

1.1 背景と目的

グラフ理論は，他の近接する学問分野などと比較して，発生は新しいが，ここ数十年間で急速に発展を遂げた分野である．グラフが持つ最大の特徴として挙げられるのが，頂点と辺のみによって構成されるシンプルな構造と広範な応用性であろう．代表的な例としては，社会に存在する様々な対象とその関係が織り成す構造体及びその構造体に対して与えられた条件付き課題を解くために，構造体をいったんグラフに置き換え，グラフ理論のフィールドでその課題を解明し，それを構造体に還元するというようなものが挙げられる．置き換えの対象としては，交通・通信等のネットワーク，化学式，人間関係など多種多様である．本論文においてはそのようなグラフ上の問題として，グラフの独立点集合にスポットを当てる．一般的には，グラフ理論における独立点集合問題と言えば，そのグラフにおける最大サイズの独立点集合を求める問題である．この問題は，実用面においても有効性が高い．例えば，ある対象物の集合とそれらの対象物のいくつかの対に互いに競合があるとき，その中から競合を回避しつつ，最大数の対象物を選ぶ問題等に応用することができる．本論文で扱う独立点集合遷移問題はその派生型ともいえる問題である．グラフに対し２つの独立点集合II_bとII_rが与えられたとき，独立点集合遷移問題とは，II_bからII_rへのグラフの独立点集合の系列が存在するかどうかを判定する問題である．ただし，2 つの連続する独立点集合の間では，定められたルールでトークンと呼ばれる各々の独立点を動かすものとする．この定められたルールとしては，Token Sliding（移動），Token Jumping（ジャンプ），Token Addition and Removal（追加/削除）の 3 種類があり，本論文では Token Sliding（移動）を取り扱う．以降，本論文において独立点集合遷移問題といえば Token Sliding（移動）に関する独立点集合遷移問題のことを指すものとする．この問題は理想グラフに対して PSPACE 完全であることが知られているが，グラフ理論における様々な問題と同様，グラフクラスを制限することにより多項式時間で解くことが可能となる．筆者はこれまでに proper interval graph および trivially perfect graph に対する独立点集合遷移問題に関し，多項式時間アルゴリズムの作成を行った[1]．本論文に示す caterpillar graph に対する多項式時間アルゴリズムについては，更なる発展がみられ，筆者の属する研究グループにおける tree に対する多項式時間アルゴリズムの開発への布石となった[2]．また，あるグラフクラスに対して，遷移可能性を判定する多項式時間アルゴリズムが作成され，遷移可能であると判定された下で，その最短の遷移系列を求める多項式時間アルゴリズムが存在するのかどうかというのも大変興味深い問題である．本論文では caterpillar graph に対する，この最短遷移系列を求める多項式時間アルゴリズムを示す．この問題は近年考案されたものであり，最大サイズの独立点集合を求める問題などのグラフ理論における他の諸問題と比べかなり新しいものであるが，現在，既に活発な研究

が行われている問題である．

1.2　本論文の構成

本論文では，第 2 章をグラフの基本的な用語の定義，第 3 章を caterpillar graph の独立点集合遷移問題に対する線形時間アルゴリズムの提示，第 4 章を caterpillar graph の独立点集合遷移問題が遷移可能という条件下においての最短遷移系列を求める多項式時間アルゴリズムの提示，第 5 章をまとめとする．

第 2 章　準備

2.1　基本用語

2.1.1　グラフとは

グラフ $G=(V, E)$は頂点の集合 Vと辺の集合 Eからなる．Eの要素は Vの要素の 2 つの組である．図 1 のグラフを例に基本的な用語を説明する[3]．頂点の集合 Vと辺の集合 E は，$V=\{v_1,v_2,\cdots,v_6\}$，$E=\{e_1,e_2,\cdots,e_8\}$である．また，$e_1=\{v_1,v_2\}$，$e_2=\{v_2,v_3\}$等となる．辺 e_8のように，ある頂点 v_1を出て，その頂点 v_1へ戻る辺を loop と呼ぶ．2 つの頂点間に辺が存在するとき，2 点は隣接しているという（v_1と v_2，v_4と v_5等）．また，各々の頂点は辺に接続しているという（v_1と v_2は e_1に接続，v_4と v_5は e_4に接続等）．頂点の次数とは，その頂点に接続している辺の本数のことをいう（v_2の次数は 3，v_5の次数は 2 等）.

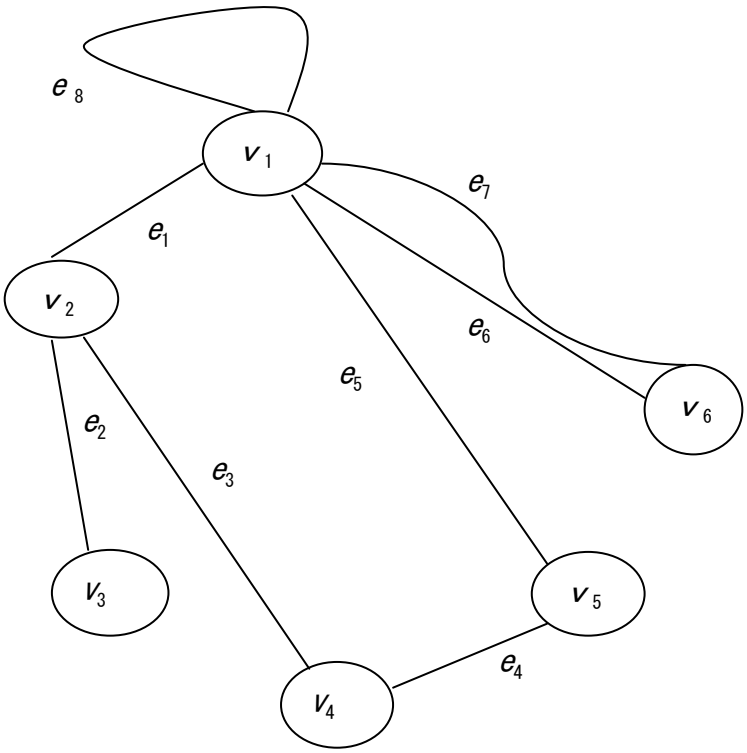

図 1　グラフ

図 2 のように，loop がなく，しかも頂点のどの対も高々1 つの辺で結ばれているとき，そのグラフは単純グラフと呼ばれる．本論文で扱うグラフはすべて単純グラフとする．

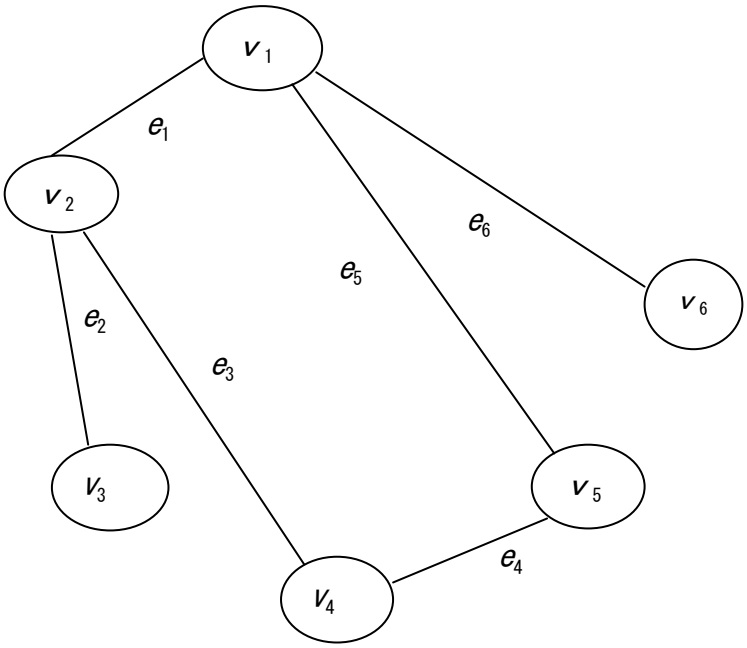

図 2　単純グラフ

単純グラフは，すべての頂点対が辺で結ばれているとき，完全グラフと呼ばれる．n 個の頂点よりなる完全グラフを Kn と書く．

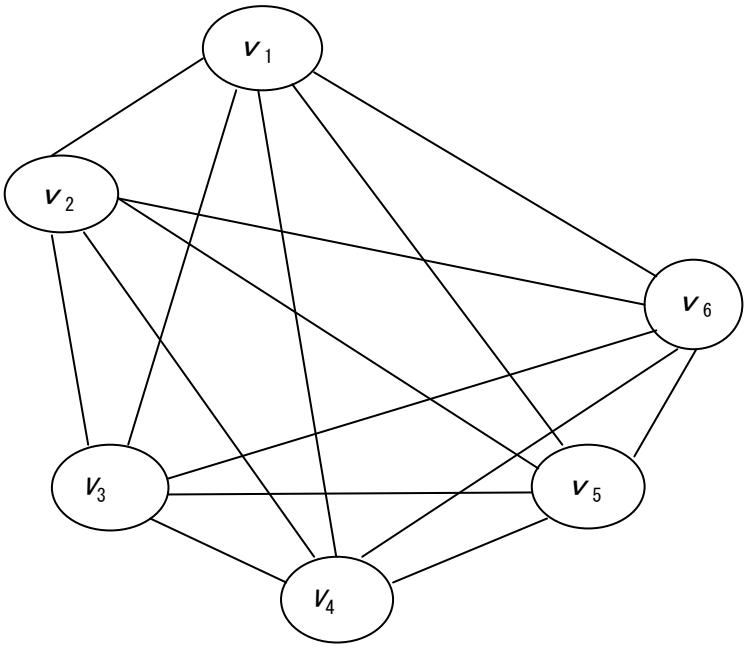

図 3　完全グラフ K_6

グラフにおける歩道 W とは有限個の頂点と辺の交互列であり，$\{v_0, e_1, v_1, e_2, v_2, \cdots, e_i, v_i, \cdots, e_n, v_n\}$を満たすものをいう．ここで，$e_i$ ($1 \leqq i \leqq n$) の両端点は v_{i-1}，v_i であり，列を構成する辺の本数を歩道の長さという．例えば，図 2 において，$\{v_1, e_1, v_2, e_3, v_4, e_4, v_5\}$は長さ 3 の歩道である．歩道において，どの点も高々一度しか現れないとき，特にその歩道を path と呼ぶ．したがって，上記の v_1 から v_5 への長さ 3 の歩道は path でもあり，path に始点と終点を結ぶ辺を加えたものを，特に cycle と呼ぶ．図 2 の $\{v_1, e_1, v_2, e_3, v_4, e_4, v_5, e_5, v_1\}$は長さ 4 の cycle である．どの 2 つの点も path で結ばれているようなグラフを連結グラフ, そうでないグラフを非連結グラフという．図 4 において示したグラフは，例えば v_1 から v_4 への path は存在しないので，非連結グラフである．

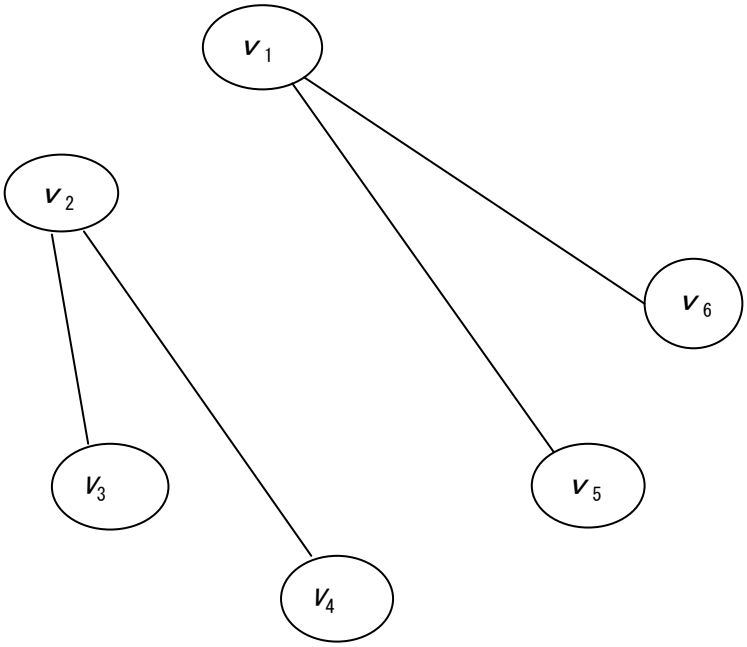

図 4　非連結グラフ

2.1.2　tree

cycle をもたない連結グラフを tree という．一例を図 5 に示す(図 2 のグラフより cycle を構成する 1 つの辺 e_4 を削除したものである).

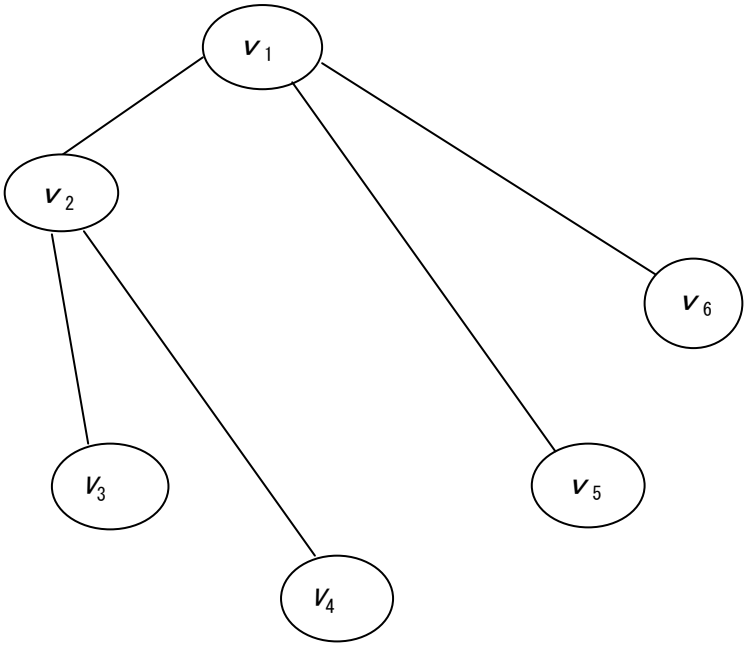

図 5　tree

2.1.3　caterpillar graph

caterpillar graph とは path に次数 1 の頂点がつながった tree である．一例を図 6 に示す．正確には，グラフ G=(V, E)の頂点集合 Vは spine とそれ以外の leaf に分類でき，spine は path(s [1]，…，s [m])を導出し，leaf の要素は次数 1 で spine の要素につながっている. 本論文においては議論を単純にするために，spine の頂点数を 2 以上とする．また，spine の両端の頂点 s [1]，s [m]は必ず 1 つ以上の leaf を持つものとする.

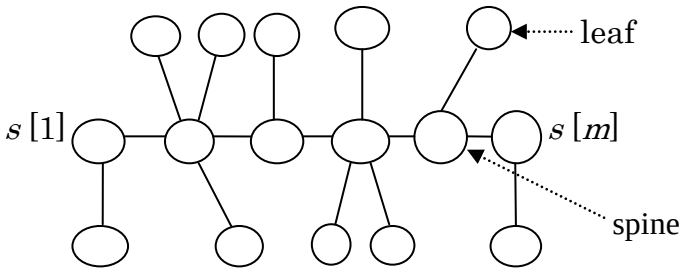

図 6　caterpillar graph

2.1.4　独立点集合

グラフ $G=(V, E)$の Vの部分集合 Sについて，Sのどの 2 点も辺で結ばれていないとき， Sをグラフ Gの独立点集合という(図 7).

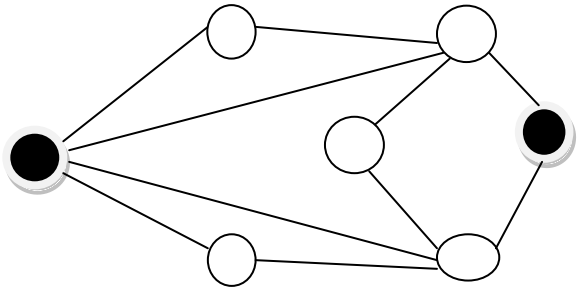

図 7　独立点集合

また，$V-S$のどの頂点を Sに追加しても Sが独立点集合でなくなるとき， Sを極大独立点集合という(図 8).

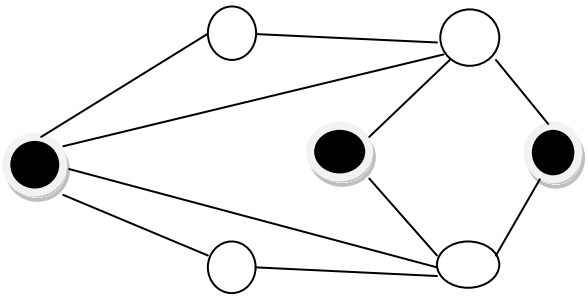

図 8　極大独立点集合

Gにおいて， Sが最大サイズである場合，特に最大独立点集合という(図 9). 最大独立点集合は必ず極大独立点集合でもある.

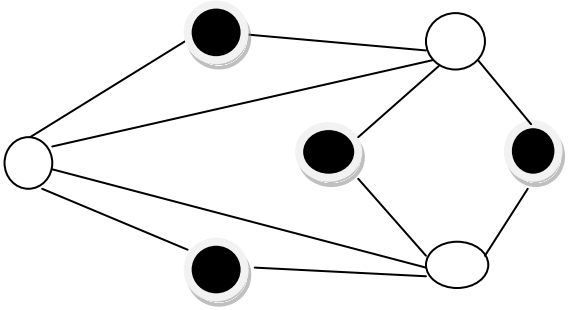

図 9　最大独立点集合(サイズ 4)

2.2　独立点集合遷移問題

独立点集合遷移問題はHearnとDemaineにより考え出されたものである[4]. グラフ上に｜II_b｜＝｜II_r｜である２つの独立点集合II_bとII_rが与えられたとき, 独立点集合遷移問題とは独立点集合の系列＜$\mathrm{II}_1, \mathrm{II}_2, \cdots, \mathrm{II}_\ell$＞が存在するかどうかを決定する問題である．ただし，ここで，以下（a），（b）が成立しているものとする．

（a）$\mathrm{II}_1 = \mathrm{II}_b, \mathrm{II}_\ell = \mathrm{II}_r$, $1 \leq i \leq \ell$のすべてのiについて｜II_i｜＝｜II_b｜＝｜II_r｜

（b）$2 \leq i \leq \ell$のそれぞれのiについて，II_iはII_{i-1}から次の操作により得られる．II_{i-1}に属する頂点u上のただ１つのトークンをその隣接する頂点vに，辺$\{u,v\}$に沿ってスライドさせる．図10に一例を示す.

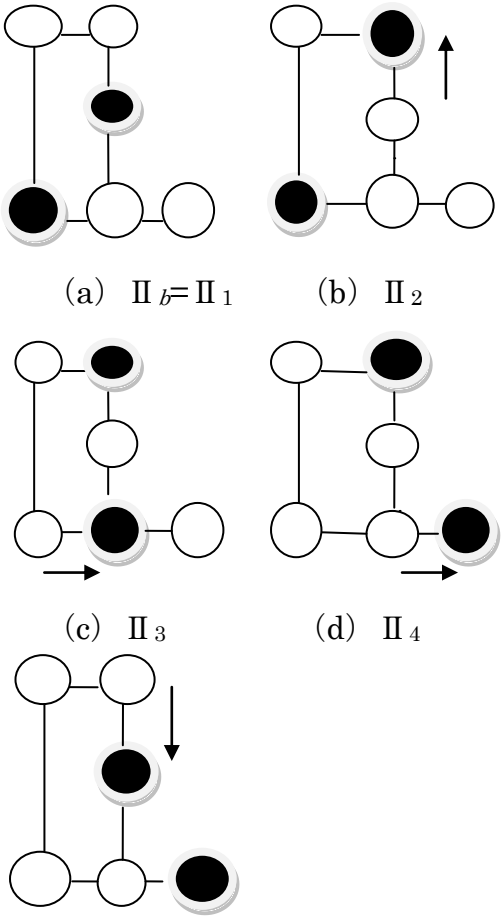

図10　グラフの独立点集合の系列（●：トークン）

第 3 章　caterpillar graph に対する独立点集合遷移問題

本章では次の主定理を証明する．

定理 1

caterpillar graph G と 2 つの独立点集合 II_b と II_r に対する独立点集合遷移問題は，$O(n)$時間及び $O(n)$領域で解くことができる．ここで n は caterpillar graph の頂点数である．さらに，遷移が可能である場合，再配置の列は $O(n^2)$時間及び $O(n)$領域で出力可能である．

3.1　caterpillar graph の限定

補題 1

caterpillar graph G_1 と 2 つの独立点集合 II_{b_1} と II_{r_1} に対する独立点集合遷移問題は，下記(1)～(3)が成立する条件下において，別の caterpillar graph G_2 と 2 つの独立点集合 II_{b_2}，II_{r_2} に線形時間で還元することができる．

(1) G_1，II_{b_1} と II_{r_1} に対する独立点集合遷移問題が遷移可能である⇔G_2，II_{b_2} と II_{r_2} に対する独立点集合遷移問題が遷移可能である．

(2) G_2 の頂点の最大次数は高々3 である．

(3) 還元後の G_2 の spine の両端の頂点の次数は各々 2 である．

言い換えれば，caterpillar graph に対する独立点集合遷移問題は，spine の各頂点に高々1 本の leaf しかついていない caterpillar graph に限定して考えれば十分である．

（補題 1 の証明）

G_1 上で $s[i]$を 4 以上の次数を持った spine の頂点とする．そのとき $s[i]$には少なくとも 2 つの leaf の頂点 $\ell_1[i]$，$\ell_2[i]$がつながっている状態である．今，II_{b_1} に関する 2 つのトークンが $\ell_1[i]$，$\ell_2[i]$にあるとする．すると，この 2 つのトークンは動かせない．また，他のトークンもこの 2 つのトークンによってブロックされているので，$s[i]$を通過することはできない．II_{r_1} もまた，この 2 つのトークンを含むならば，G_1 から $s[i]$とその leaf である $\ell_1[i]$，$\ell_2[i]$を取り除き，新たな 2 つの caterpillar graph の独立点集合遷移問題として取り扱うことができる．

II_{r_1} がこの 2 つのトークンを含まなければ，問題に対する答えが “遷移不可能”

であることは明らかである. 少なくとも 2 つのトークンが，II_{b_1}側の caterpillar graph の 1 つの spine の頂点の leaf として共存するのであれば，それを線形時間で判定できる．そこで，II_{b_2}，II_{r_2}とも，1 つの頂点にトークンを持った leaf は高々1 本しかつながっていないとすることができる．同様の理由により，1 つの頂点につながっている 1 本の leaf 以外はこの問題を考える時, 除外してよい. より正確には，II_{b_1}からII_{r_1}への遷移可能性にかかわらず，1 つの spine の頂点につながっている leaf のうち，使用されるのは 1 本だけであるということである．それゆえ，1 つの頂点から，上記 1 本の leaf 以外は除去できる．特に不要な leaf を取り除き，$s[1]$ 及び $s[m]$の次数は 2 とすることができる．□

以上より，これからは，補題 1 で述べられた caterpillar graph のみを考えていくことにする．つまり，spine の頂点 $s[1]$，$s[2]$，…，$s[m]$を持った caterpillar graph に対して，$s[1]$ 及び $s[m]$の次数は 2，$1<i<m$ である i に対しては $s[i]$の次数は 2 または 3 とする．また，spine の頂点 $s[i]$に leaf がある場合，これを $\ell[i]$とする．

3.2 locked path

次に caterpillar graph の独立点集合遷移問題にとって核となる概念を導入する．caterpillar graph G と G の独立点集合 II に対して，G 上の 1 本の path $\mathrm{P}=(p[1], p[2], \cdots, p[k])$が II によって lock されている(locked path)とは下記の 3 つの条件が同時に成立することである．

(a) k は 5 以上の奇数である．
(b) $\mathrm{II} \cap \mathrm{P}=(p[1], p[3], p[5], \cdots, p[k])$
(c) $p[1]$と $p[k]$の次数は 1 である．つまり両者とも leaf である．

図 11 において，点線で囲まれたエリアが locked path である．locked path の概念を用いれば，キャタピラグラフ上の動かすことのできない独立点集合を特徴づけることができる．

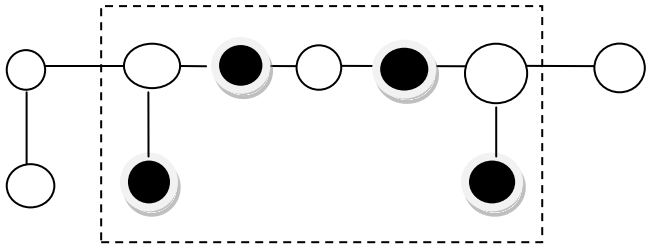

図 11　locked path の一例（ ：独立点集合）

定理 2

caterpillar graph を G，G の独立点集合をⅡとする．このとき，Ⅱに属するトークンをまったく動かすことができない⇔ある h に対して，locked path の集合 P[1]，…，P[h]が存在し，独立点集合Ⅱはその和集合である．

（定理 2 の証明）

明らかにⅡが locked path の和集合であるならばトークンを動かすことはできない．それゆえ，G 上のあるⅡに対して，Ⅱに属するどのトークンも動かすことができないということを仮定し，それらはすべて locked path 上に置かれているということを示す．まず始めに，トークンが$\ell[i]$上にある場合を考える．そのとき，$s[i]$にトークンは存在しない．$s[i-1]$ ，$s[i+1]$の両方にトークンが存在しないならば，$s[i]$にトークンを動かすことができるが，これは矛盾である．ここで，一般性を失うことなく，$s[i+1]$にトークンが存在すると仮定できる．その結果，$\ell[i+1]$，$s[i+2]$にトークンが存在しないことになる．また，トークンを持たない$\ell[i+1]$が存在すると，$s[i+1]$のトークンが$\ell[i+1]$に移動でき，それにより$\ell[i]$のトークンが $s[i]$に移動可能となる．よって，$s[i+1]$は leaf 自体を持たないとすることができる．$\ell[i+2]$にトークンが存在するならば，トークンは locked path($\ell[i]$，$s[i]$, $s[i+1]$, $s[i+2]$，$\ell[i+2]$)上に置かれていることとなり，矛盾は生じない．次に，$\ell[i+2]$にトークンが存在しないと仮定する．その時，$s[i+1]$上のトークンは動かすことができないので，$s[i+3]$上にトークンが存在しなければならない．このプロセスを繰り返すことにより，結果的に，1 つの端点$\ell[i]$を持つ，locked path を得る．次に，トークンが spine の頂点 $s[i]$に存在する場合を考える．$s[i]$が$\ell[i]$を持つならば，トークンを$\ell[i]$に動かすことができる．これは矛盾である．よって，$s[i]$は leaf を持たない．一方，$s[i-1]$，$s[i+1]$の両方にトークンは存在しない．これら，2 つの頂点に関して，最初の場合と同様の手法を取ることができるので，($s[i-1]$, $s[i]$, $s[i+1]$)を path の一部とする locked path を得ることができる．　□

caterpillar graph Gと独立点集合Ⅱについて，Ⅱが locked path を含むならば，その locked path の頂点を通過して，トークンを動かすことはできない．ゆえに，locked path は Gを 2 つの部分グラフに分割すると考えてよい．このとき，各々の部分グラフについて別々に独立点集合遷移問題を考えればよいことになる．

3.3　最左充填独立点集合

caterpillar graph Gと正の整数 kについて以下，サイズが kである最左充填独立点集合$\Pi_k(G)$を定義する．

まず始めに$\Pi_k(G)$に$\ell[1]$を加える．そして各々の i=2，3, 4, …, m-1 について，k個のトークンを次のルールに従って置いていく．

(1) $s[i-1] \notin \Pi_k(G)$ ＆ $s[i]$の次数は 2
　$s[i]$を$\Pi_k(G)$に加える．

(2) $s[i-1] \notin \Pi_k(G)$ ＆ $s[i]$の次数は 3
　$\ell[i]$を$\Pi_k(G)$に加える．

(3) $s[i-1] \in \Pi_k(G)$
　この iに関しては何もしない．

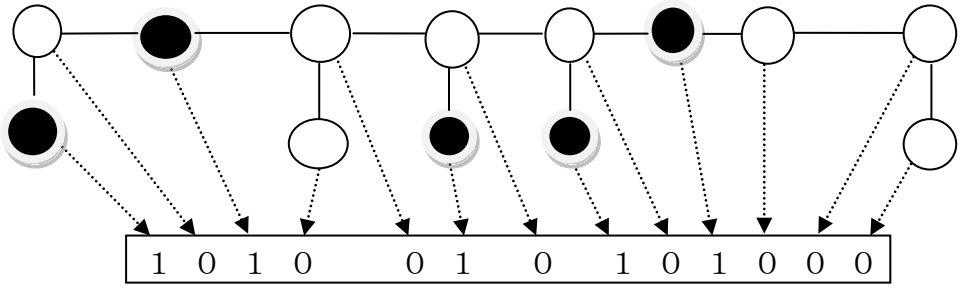

図 12　サイズ 5 の最左充填独立点集合及びその 2 進符号化

定理 3

caterpillar graph Gとサイズが kの独立点集合Ⅱに対して，Ⅱが locked path を含まないとする．このとき，Ⅱは最左充填独立点集合$\Pi_k(G)$に遷移可能である．さらに，遷移の回数は$O(n^2)$である．ここで，nは Gの頂点の数である．

(定理 3 の証明)

まず最初に，Gについて，以下のような全順序を定義する．

$\ell[1] < s[1] < \ell[2] < s[2] < \cdots < \ell[i] < s[i] < \cdots < \ell[m] < s[m]$

(ただし，$\ell[i]$に関しては，存在しない場合がある．)

ここで，2 進数を表す列，$B(\Pi) = b[0]\ b[1]\ \cdots b[n]$を以下のルールにより定義す

る．n は G の頂点の数である．
(0) 各々のビット $b[i]$は全順序で i 番目に現れる頂点に対応する．
(1) トークンを持った頂点は 1 に符号化される．
(2) トークンを持たない頂点は 0 に符号化される．
一例として，図 12 の caterpillar graph のサイズ 5 の最左充填独立点集合は 2 進数列 10100101000 に符号化される．
k 個の頂点からなる最左充填独立点集合が形成する 2 進数列 $B(\Pi_k(G))$と，他の k 個の頂点からなる独立点集合が形成する 2 進数列 $B(\Pi)$について，各々を 2 進数とみなせば，必ず，$B(\Pi_k(G))> B(\Pi)$が成立する．1 を同数 k 個含む 2 つの 2 進数列 B_1 と B_2(｜B_1｜＝｜B_2｜とする)について，以下のルールで決定される距離 dist (B_1, B_2)を導入する．各々の i=1, 2, …, k について，B_1 が i 番目の 1 を $j_1(i)$番目のビットに含み, B_2 が i 番目の 1 を $j_2(i)$番目のビットに含むと仮定する．そのとき，dist (B_1, B_2)を
$\Sigma_{(i=1\cdots k)}\ \delta(j_1(i)-j_2(i))$
によって定義する．
例えば，dist(10100, 00101)=2 となり，dist (B_1, B_2)=0 ⇔ B_1=B_2 である．さらに，2 つの長さ n の 2 進数列 B_1 と B_2 について，各々のビットは上記で導入した距離に対して高々1 しか寄与しないので，dist (B_1, B_2)は $O(n)$である．次に locked path を含まないサイズ k の独立点集合Ⅱと最左充填独立点集合$\Pi_k(G)$を比較する．そのとき，$\Pi \neq \Pi_k(G)$であり，2 進数 $B(\Pi_k(G))$>$B(\Pi)$である．
$B(\Pi_k(G))$= $b[1]$ $b[2]$ …$b[n]$，$B(\Pi)$= $b'[1]$ $b'[2]$ …$b'[n]$で表す．
このとき，$b[i]$=1，$b'[i]$=0，すべての $i'<i$ について $b[i']= b'[i']$であるようなインデックス i が存在する．そのインデックス i について，$B(\Pi)$と $B(\Pi_k(G))$は同数の 1 を含むので, すべての $i \leq i'<j$について $b'[i']$=0 及び $b'[j]$=1 であるような、もう 1 つのインデックス $j>i$ が存在する．そこで，以下、2 つの場合(a), (b)が生じる．
(a)$b'[j]$が spine の頂点 $s[j']$ である場合，$s[j']$と $b[i]$に対応する頂点間の頂点にトークンは存在しない．一方，$b[i]$は spine の頂点 $s[i']$か$\ell[i']$に対応する(ここで，i'は $i'<j'$を満たす)．どちらにせよ，$\Pi_k(G)$は最左充填独立点集合であるから $s[i-1]$にトークンは存在しない．それゆえ，Ⅱ上で s[j']上のトークンを spine の頂点 $s[i']$または$\ell[i']$にスライドさせることができる．$O(n)$回のスライド作業の後に，新たな独立点集合Ⅱ'を得る．ここで，dist(Π',$\Pi_k(G)$)=dist(Π,$\Pi_k(G)$)－1 である．
(b)$b'[j]$が$\ell[j']$に対応する場合，Ⅱは独立点集合であるから $s[j']$にトークンは存在しない．さらに $s[j'+1]$にもトークンが存在しないならば，$\ell[j']$上のトークンを $s[i]$に最初の場合と同様の手法でスライドさせることができる．そして，新た

な独立点集合Ⅱ'を得る．ここで，dist(Ⅱ', $\Pi_k(G)$)=dist(Ⅱ, $\Pi_k(G)$)－1 である．そこで次に，$s[j'+1]$にトークンが存在する場合を考える．仮定により，Ⅱは locked path を含まない．よって，{$\ell[j']$，$s[j']$, $s[j'+1]$, $s[j'+2]$} を含むどんな path も locked path を含まない．それゆえ，$s[j'']$にトークンが存在し，{$s[j''+1]$, $\ell[j''+1]$, $s[j''+2]$}にはトークンが存在しないようなインデックス j''($j''>j'$)が必ず存在する．そうすると，その path P に沿って，トークンを 1 つ 1 つスライドさせることができ，$O(n)$回のスライド作業により，$s[j'+1]$上のトークンを $s[j'+2]$へ動かすことができる．そこで，$\ell[j']$上のトークンを $s[i]$に最初の場合と同様の手法でスライドさせることができる．この作業の後，path P に沿って，トークンの遷移を巻き戻す．$O(n)$回の巻き戻し作業後，新たな独立点集合Ⅱ'を得る．ここで，dist(Ⅱ', $\Pi_k(G)$)=dist(Ⅱ, $\Pi_k(G)$)－1 である．

以上，1 回の過程が $O(n)$，それを高々 n 回繰り返すので，最終的に $O(n^2)$回のスライド作業により $\Pi_k(G)$を得る． □

3.4　定理 1 の証明

最後に本論文の核となる定理 1 の証明に戻る．

（定理 1 の証明）

caterpillar graph に与えられた独立点集合 Π_{b_1}について各々の頂点が locked path の一部であるかどうか $O(n)$時間で判定できる．

(0) 状態 S を“not locked path”で初期化する．

(1) i=1, 2, …, m について，($s[i]$, $\ell[i]$)の状態により状態 S を更新する．

($s[i]$, $\ell[i]$)=(x,y)は次のように決定される．

$x\in$ {0,1}，$y\in$ {0,1,－}

1 はトークンが頂点に存在することを示し，0 はトークンが頂点に存在しないことを示し，－は leaf が存在しないことを示す．

下記の各々の case($s[i]$, $\ell[i]$)について状態 S をどのように更新するのか記述する．

case (0, 1):

S が“not locked path”であるならば，S を“locked path?”にセットする．そして，i を locked path の左端点候補として記憶する．S が“locked path?”であるならば，すぐ隣りの左の点にトークンが存在する場合，locked path が成立するので，左端点候補として記憶した点からこの点までが locked path である．すぐ隣りの左の点にトークンが存在しない場合，いったん，S は“locked path?”をリセットし，この点が左端点候補となり，再び S は“locked path?”の状態となり，右側に進んで行く．

case(0,0)及び case(0,－):
$s[i\text{-}1]$にトークンが存在する場合，S の状態をそのまま，次に送る．$s[i\text{-}1]$にトークンが存在しない場合，“not locked path”により，S をリセットする．S の前の状態がどのような状態であってもリセットを行う．
case (1, 0):
“not locked path”により，S をリセットする．S の前の状態がどのような状態であってもリセットを行う．

case(1,－):
S の状態をそのまま次に送る．

上記の方法により，$O(n)$時間ですべての locked path の頂点を調べることができる．まず始めに，この方法を，(G, II_b)と(G, II_r)に対して同時に適用する．必要とする時間は $O(n)$時間である．そして，両者の locked path が完全一致するかどうかを調べる．完全一致しなければ，アルゴリズムは“遷移不可能”の答えを返す．両者の locked path が完全に一致する場合は，アルゴリズムは caterpillar graph Gをいくつかの部分グラフに分割する．この部分グラフ群は locked path ではない頂点のみから形成される．そして，各部分グラフごとに，遷移問題を解き進める．ここで locked path のトークンが存在する左端点と右端点は必ず leaf である．つまり，locked path を P=($p[1]$，$p[2]$，…，$p[k]$)とし，Gが Pによって G_1と G_2に分割されるのであれば，G_1と G_2の P に対する隣接点は $p[2]$と $p[k\text{-}1]$である．そして，この 2 点にはトークンは存在しない．よって，G_1と G_2は P の部分は除外して，完全に分離して問題を解けばよい．次に，各々の部分グラフについて，アルゴリズムは遷移前後の状態が同数のトークンを有するかを調べる．すべての部分グラフについて，トークンの数が一致すれば，アルゴリズムは“遷移可能”の答えを返し，そうでなければ，“遷移不可能”の答えを返す．上記アルゴリズムの正当性は定理 2 と定理 3 より導かれる．また，アルゴリズムが $O(n)$時間，$O(n)$領域で実行されることも容易にわかる．また，遷移列自体を出力するためには，定理 3 を用いてアルゴリズムに修正を加えれば，修正後のアルゴリズムは長さ $O(n^2)$の列を出力することになる．なお，図 13 の遷移例に見られるように，遷移列の長さは $\Theta(n^2)$ となり得るため，遷移列自体を出力するアルゴリズムは線形時間では実行できない．

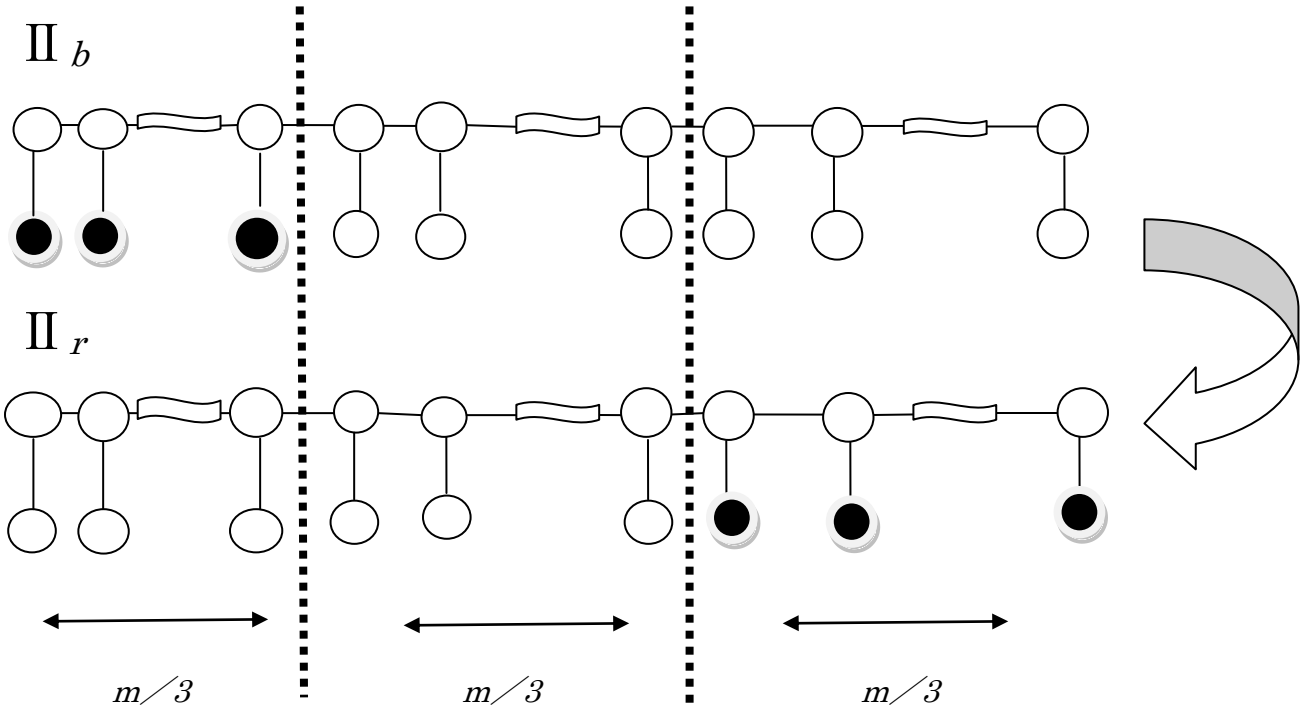

図 13　遷移列の長さが $\Theta(n^2)$の遷移例

第 4 章　最短遷移列

本章では caterpillar graph の独立点集合遷移問題に対して，第 3 章において示したアルゴリズムにより“遷移可能”と判定された場合に，その最短の遷移系列を求めるアルゴリズムを示す．つまり，II_b から最少の数の遷移系列で II_r へたどり着く，その遷移系列を求める．

定理 4

caterpillar graph と 2 つの独立点集合 II_b および II_r に対する独立点集合遷移問題が“遷移可能”であるとき，遷移系列のうち，その列の数が最少の遷移系列を $O(n^2)$時間, $O(n)$領域で求めることができる. ここで n は caterpillar graph の頂点の数である.

4.1　トークンの対応

遷移前後のトークンの対応にとって鍵となるトークンどうしの追い越し(通過)を改めて次のように定義する．pair{$s[x]$, $\ell[x]$}に存在するトークンを X, pair{$s[y]$, $\ell[y]$}に存在するトークンを Y, $x<y$ とする. X の遷移先を pair{$s[z]$, ℓ[z]}, Y の遷移先を pair{$s[w]$, ℓ[w]}とすると，トークンどうしの追い越しとは，$z>w$ が成立することとする．第 3 章においても触れたが，トークンの遷移に関して，トークンどうしの追い越しは発生しない．つまり $z<w$ が必ず成立する．独立点集合遷移問題の前提条件である，互いのトークンは独立ということより，2 つのトークンが同時に pair{$s[x]$, $\ell[x]$}に存在することはできない．このことより，トークンどうしの追い越しは発生しないことは明らかである(図 14 参照). よって，次の補題 2 が成立する.

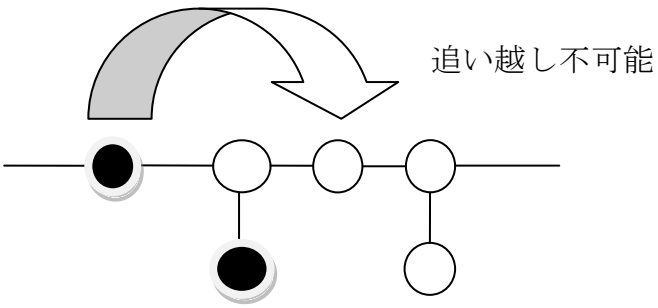

図 14　トークンどうしの追い越し

補題 2

独立点集合遷移問題が“遷移可能”であるとき，遷移前後のトークンの対応は一意に決定される．

遷移前後について，各々，トークンに左から番号づけを行うと，同一の番号を持つトークンが対応関係を成す．遷移前のトークンの集合Π_bを$\{b_1, b_2, \cdots, b_i, \cdots, b_k\}$，遷移後のトークンの集合$\Pi_r$を$\{r_1, r_2, \cdots, r_i, \cdots, r_k\}$，写像 g を $g: \Pi_b \rightarrow \Pi_r$とすると，すべての iについて

$$g(b_i) = r_i$$

が成立する．つまり，遷移前，最左から i番目に位置するトークンは，遷移後も最左から i 番目に位置する．次に各々のトークンの遷移方向に着目する．$b_i \in \mathrm{pair}\{s[x], \ell[x]\}$，$r_i \in \mathrm{pair}\{s[y], \ell[y]\}$とする．各々のトークンの遷移について，$x < y$ である場合を R，$x > y$である場合を Lとする．また，$x = y$ である場合を Cとする．すると，遷移全体を R，L，Cからなる記号列によって表現することができる．ここで，連続する Rまたは Lは，それらをまとめて，改めて Rまたは Lとする．図 15 にその様子を示す．

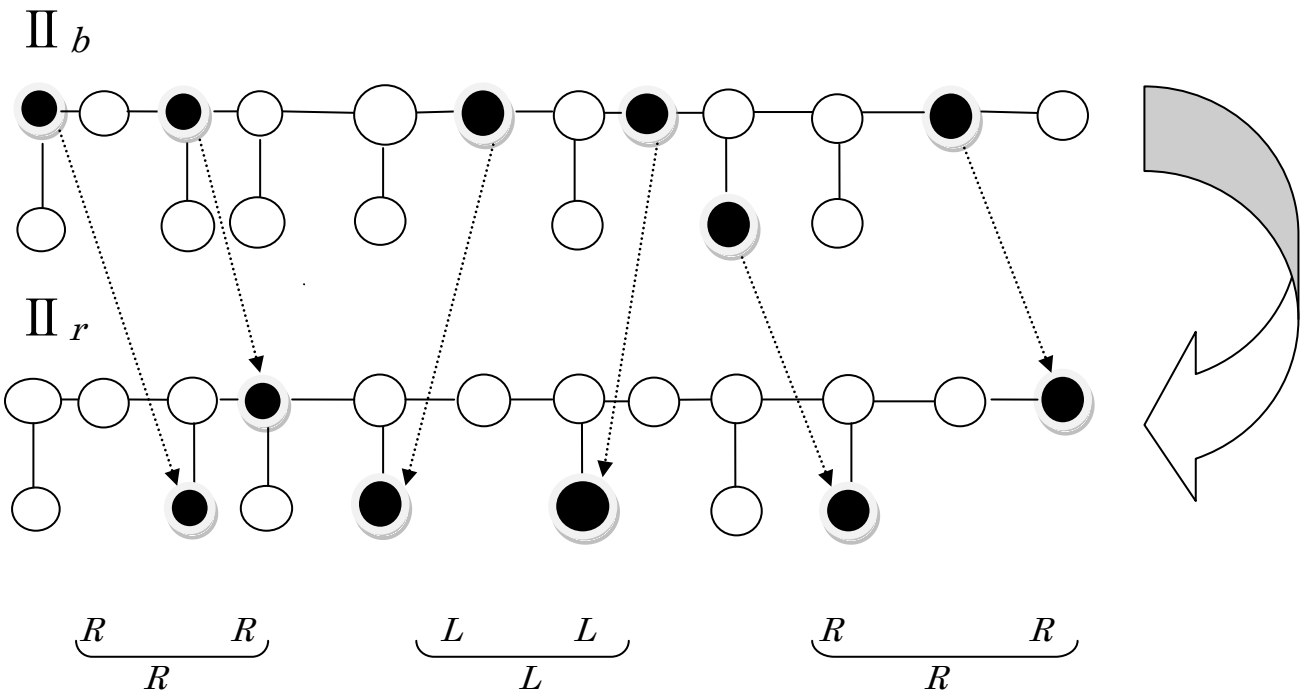

図 15　遷移例および遷移方向による分類

Cについては leaf の有無等も考慮し，下記 $C1$～$C5$ に分類する．図 16 にその様子を示す．

$C1$：遷移前 $\ell[x]$→遷移後 $\ell[x]$

$C2$：遷移前 $s[x]$→遷移後 $\ell[x]$

$C3$：遷移前 $\ell[x]$→遷移後 $s[x]$

$C4$：遷移前 $s[x]$→遷移後 $s[x]$($\ell[x]$有)

$C5$：遷移前 $s[x]$→遷移後 $s[x]$($\ell[x]$無)

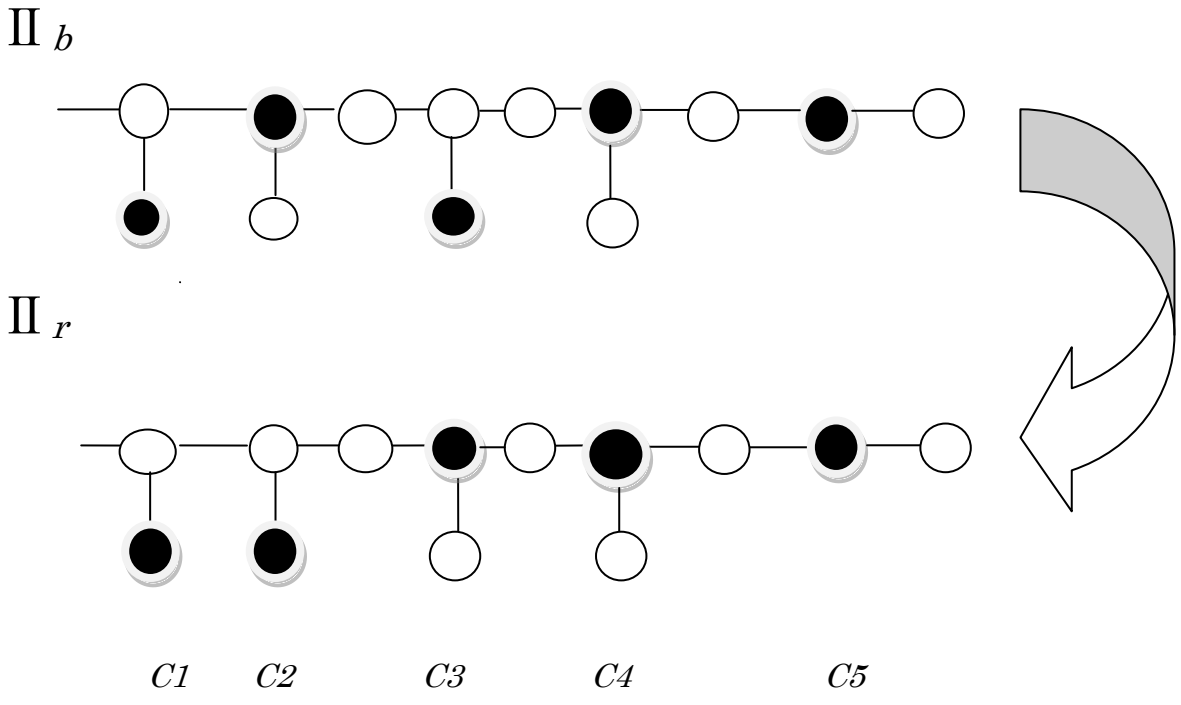

図 16　Cの分類($C1$～$C5$)

4.2　*R*, *L*, *C* におけるトークンの動き

4.1 において定めた *R*, *L*, *C1*～*C5* について，各グループ内での最短遷移を実現するトークンの動きを考える．一般的に，caterpillar graph を含む tree 上の 2 つの頂点間の最短経路は一意に決定されるので，その 2 つの頂点間でのトークンの遷移を考えた場合，トークンがその最短経路上のすべての頂点を一度ずつ通って遷移すれば，それが最短経路であることは明らかである．したがって，下記の定義 1 に示す例外的な場合を除き，各グループにおいて，各々のトークンは上記の最短経路上をすべての頂点を一度ずつ通って遷移すればよい．トークンが複数個存在する場合は，1 つのトークンの遷移が完了後，次のトークンの遷移に移行する．*R* を例にすると，*x* の値が大きいトークンから順次遷移を行う．

定義 1　detour

以下，一般性を失うことなく，一方向に対して議論を進める．

1. $\{n \in \mathbb{N}_0 \mid s[j-2n], \cdots, s[j]\}$ にトークンが存在する．ただし，n が $n \in \mathbb{N}$ の場合，$\{s[j-2(n-1)], \cdots, s[j]\}$は leaf を持たない．

2. $s[j]$に存在するトークンの右隣りの一つ先のトークンの遷移に関連し，上記に示したトークンが下記 3. に示す一連の遷移を行わなければならない．

3. 遷移は$\ell[j-2n]$の存在により 2 つの場合に分けられ，両者の一連のトークンの遷移を detour と定義する．

case(a)　$\ell[j-2n]$が存在する

$\{s[j-2n], \cdots, s[j]\}$

$\{\ell[j-2n], s[j-2n+1], \cdots, s[j-1]\}$

$\{s[j-2n], \cdots, s[j]\}$

case(b)　$\ell[j-2n]$が存在しない

$\{s[j-2n], \cdots, s[j]\}$

$\{s[j-2n-1], \cdots, s[j-1]\}$

$\{s[j-2n], \cdots, s[j]\}$

図 17～図 20 にトークンの具体的な動きを示す．対象としている独立点集合遷移問題が“遷移可能”であるという条件より，対象となるトークンの detour 可能性は明らかである．また，トークンの遷移に関して追い越しは不可能である等のことから，detour が最短遷移の実現に相反しないことも明らかである．detour の判定アルゴリズムは第 3 章において示した locked path の判定アルゴリズムに修正を加えれば容易に構築可能である．

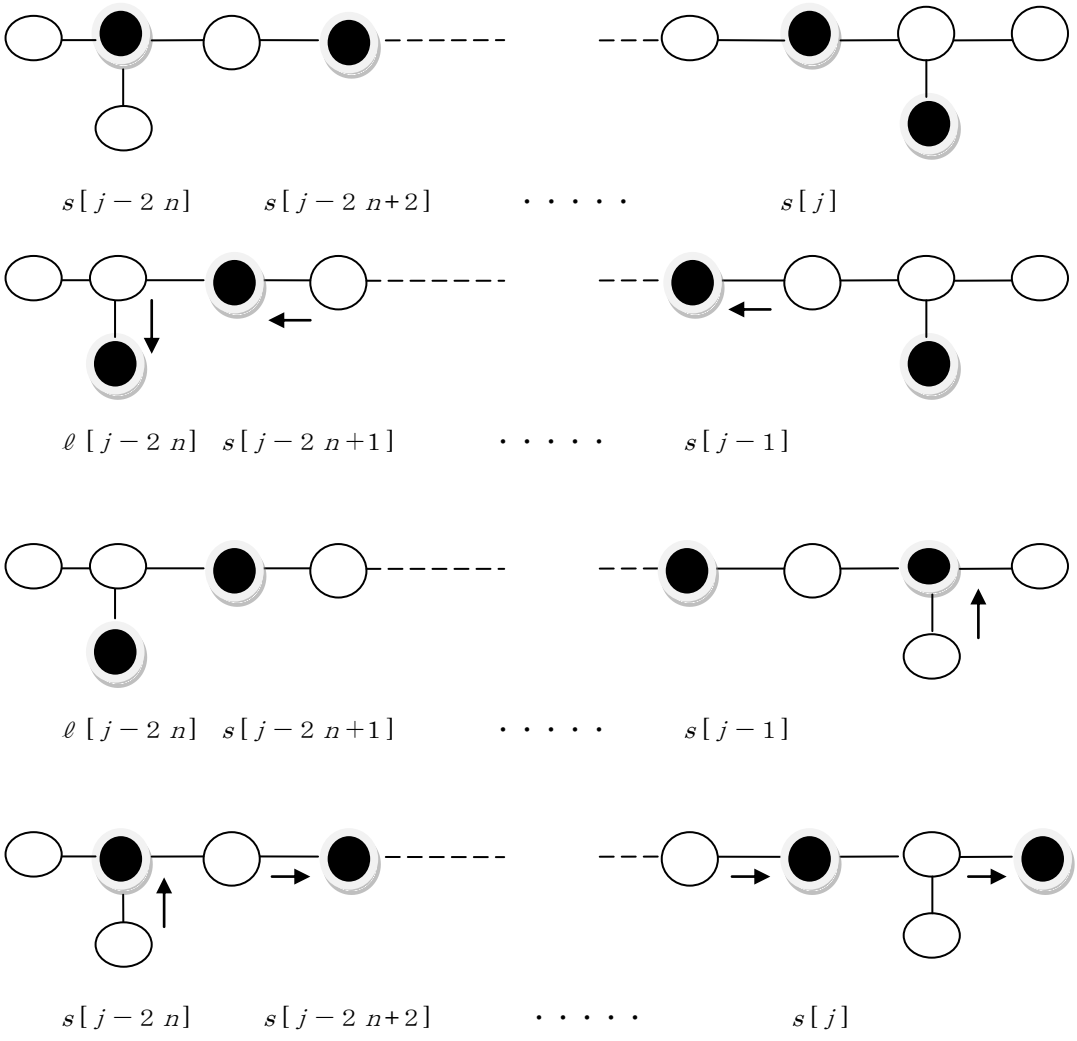

図 17　detour (case(a)その 1)

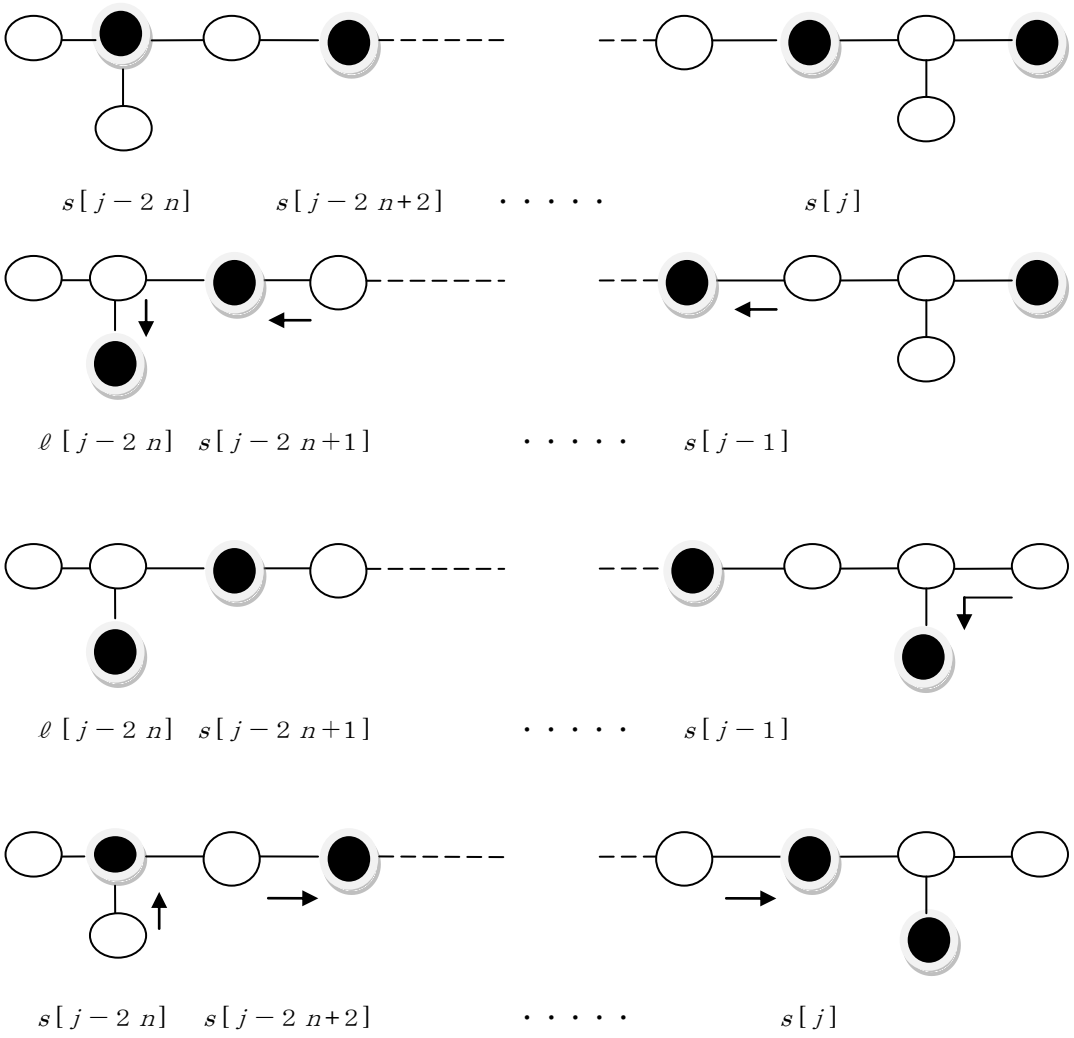

図 18　detour (case(a)その 2)

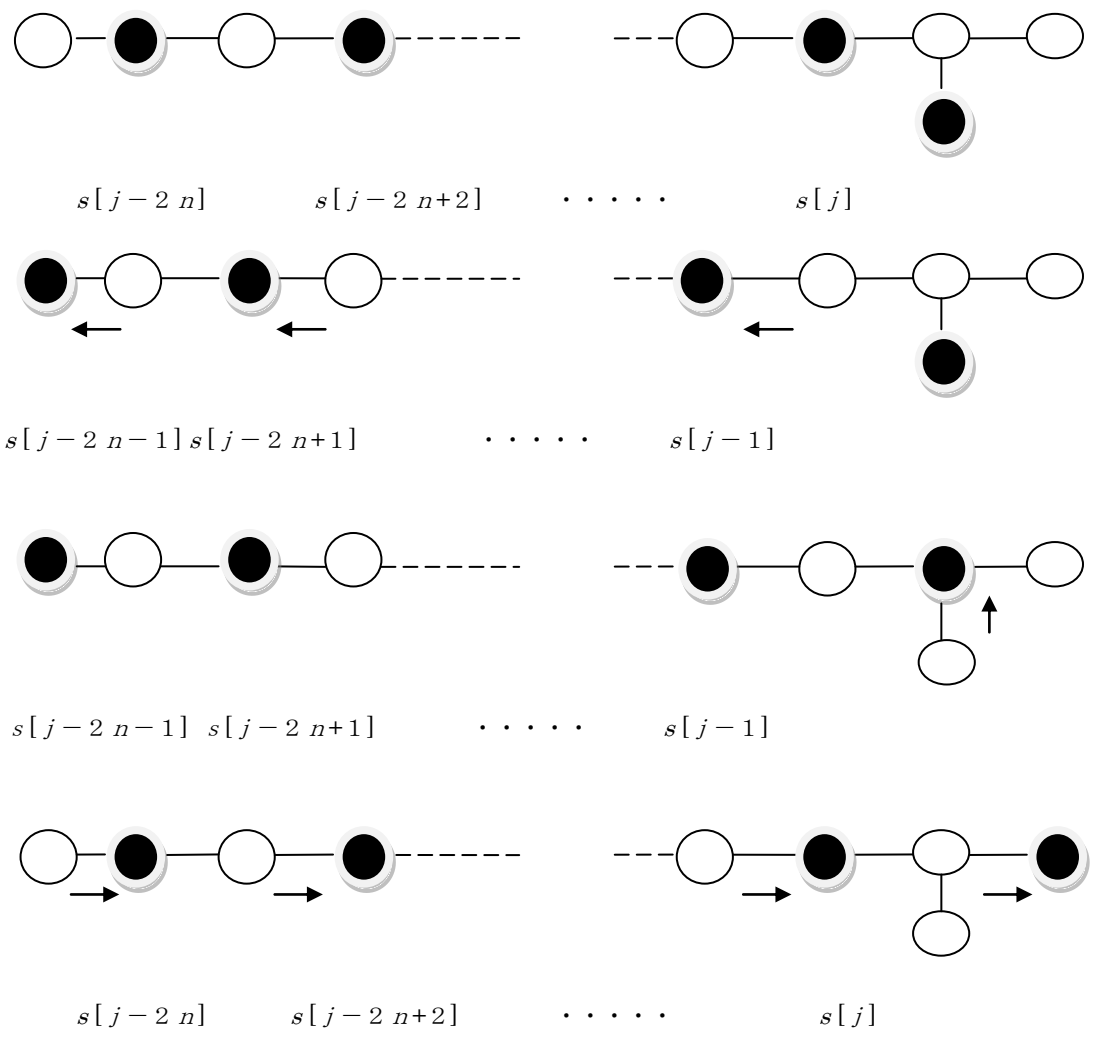

図 19　detour (case(b)その 1)

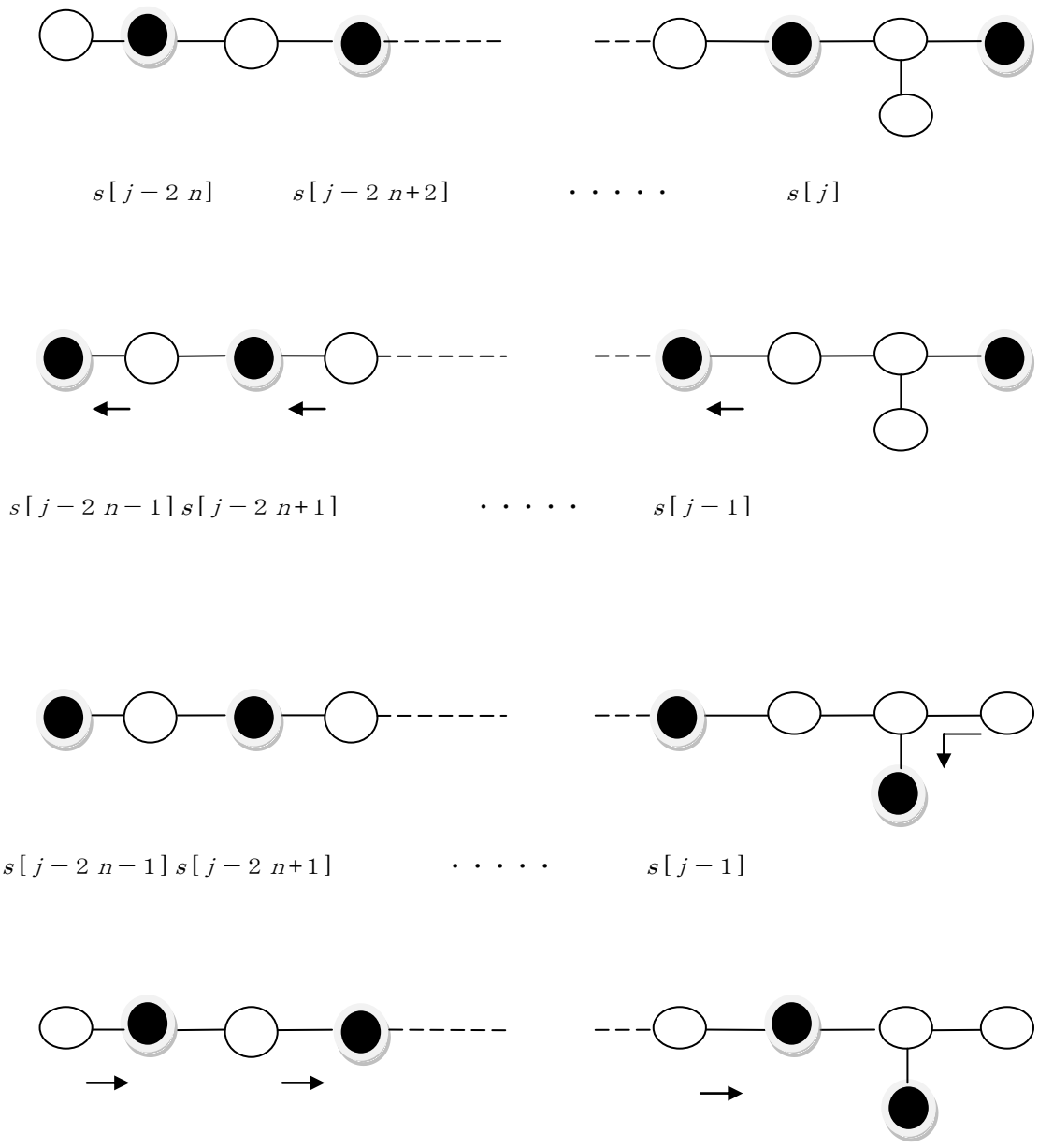

図 20　detour（case(b)その 2）

4.3　*R, L, C* 相互の関係性

次に，最短遷移を実現するために，*R, L, C1*～*C5* のグループ相互の遷移の実行の優先順序について考える．図 21 の遷移例において，トークン②はトークン①より先に動かなければ最短遷移は実行できない．トークン①が先に動くと冗長な動きが生じてしまうことは明らかである．また，トークン③とトークン④については，どちらが先に動いても，最短遷移に関して影響はない．図 21 の遷移例で見ると，*C3* と *R* については比較可能(R が優先)であり，*R* と *L* については比較不能(優先順序は無し)である．このように，最短遷移という観点からみると，遷移を表す *R, L, C1*～*C5* の記号列を半順序集合として捉え，互いの実行の優先順序を考えればよいことがわかる．

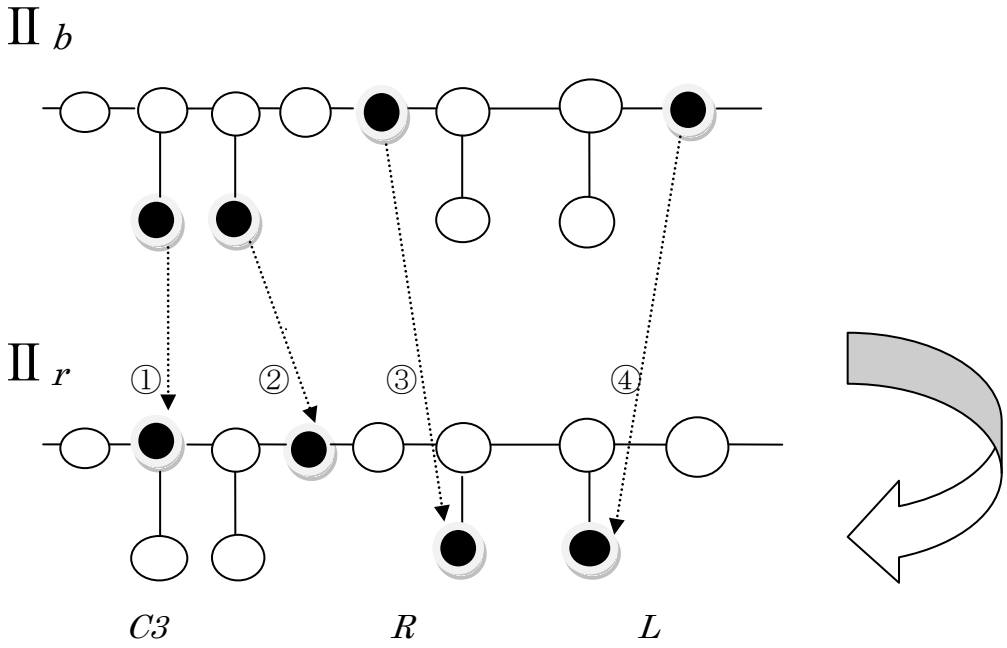

図 21　グループ相互の実行の優先順序

以下，グループ相互の実行の優先順序について，考えられるすべてのパターンについて分析を行う．まず，*C1*～*C5* が出現した場合を考える．

(1)*C1* が出現した場合

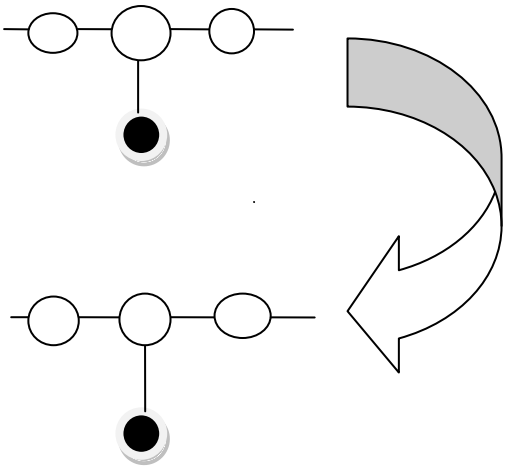

図 22　*C1*

C1 については，左右がいかなる場合でも，トークンの遷移は発生しない．また，*C1* に属するトークンが他のトークンの往来を妨げるので，*C1* に対する左右各々を別々の部分グラフとして問題を考えればよい．

(2)*C2*が出現した場合

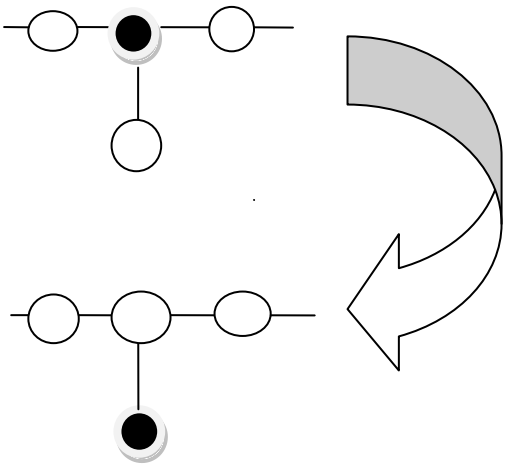

図 23　*C2*

遷移の優先度を，左右の状態に関係なく，左右より高く設定すれば安全である．*s*[*x*]→*ℓ*[*x*]の遷移を行った後は，(1)に帰着する．

(3)*C3*が出現した場合

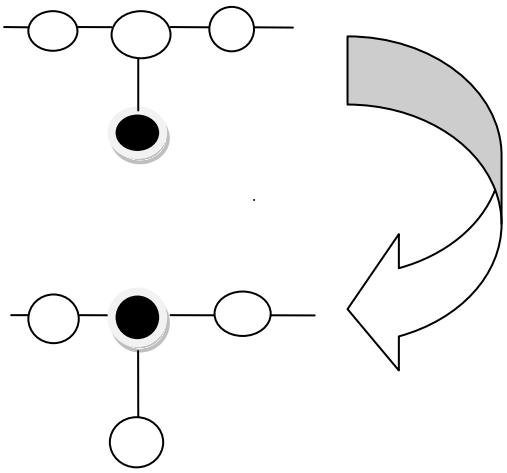

図 24　*C3*

遷移の優先度を，左右の状態に関係なく，左右より低く設定すれば安全である．*C3*に対する左右各々を別々の部分グラフとして問題を考え，左右の部分グラフの遷移がすべて完了した後，*C3*の遷移：$\ell[x] \rightarrow s[x]$を行えばよい．

(4)*C4* が出現した場合

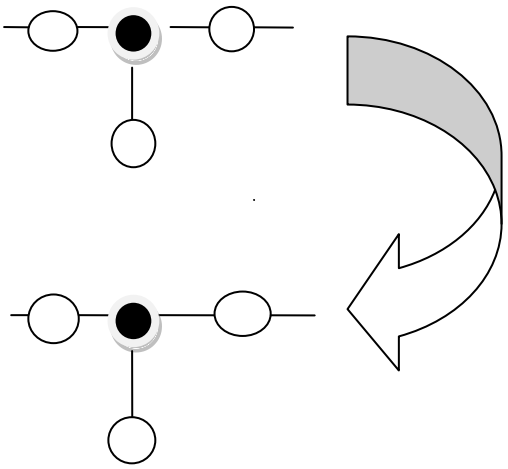

図 25　*C4*

C4 については，基本的に，トークンの遷移は発生しない．また，*C4* に対する左右各々を別々の部分グラフとして問題を考えればよい．ただし，一点だけ注意を要することがある．*C4* が 4.2 で述べた detour の一部を構成している場合である．具体的には，*C4* のトークンが detour の case(a)の $s[j-2\ n]$のトークンに該当する場合である．該当する場合，トークンに $s[j-2\ n]\rightarrow\ell[j-2\ n]\rightarrow s[j-2\ n]$の遷移が発生する．図 26 に示すように，左右両方向に対して対象とする detour が構成されていることも考えられるので，$s[j-2\ n]\rightarrow\ell[j-2\ n]$の遷移は左右に対して優先，$\ell[j-2\ n]\rightarrow s[j-2\ n]$の遷移は左右の遷移後に行うと設定すれば安全である．よって，遷移を 2 つに分解して，各々の遷移に順序を付与すればよい．

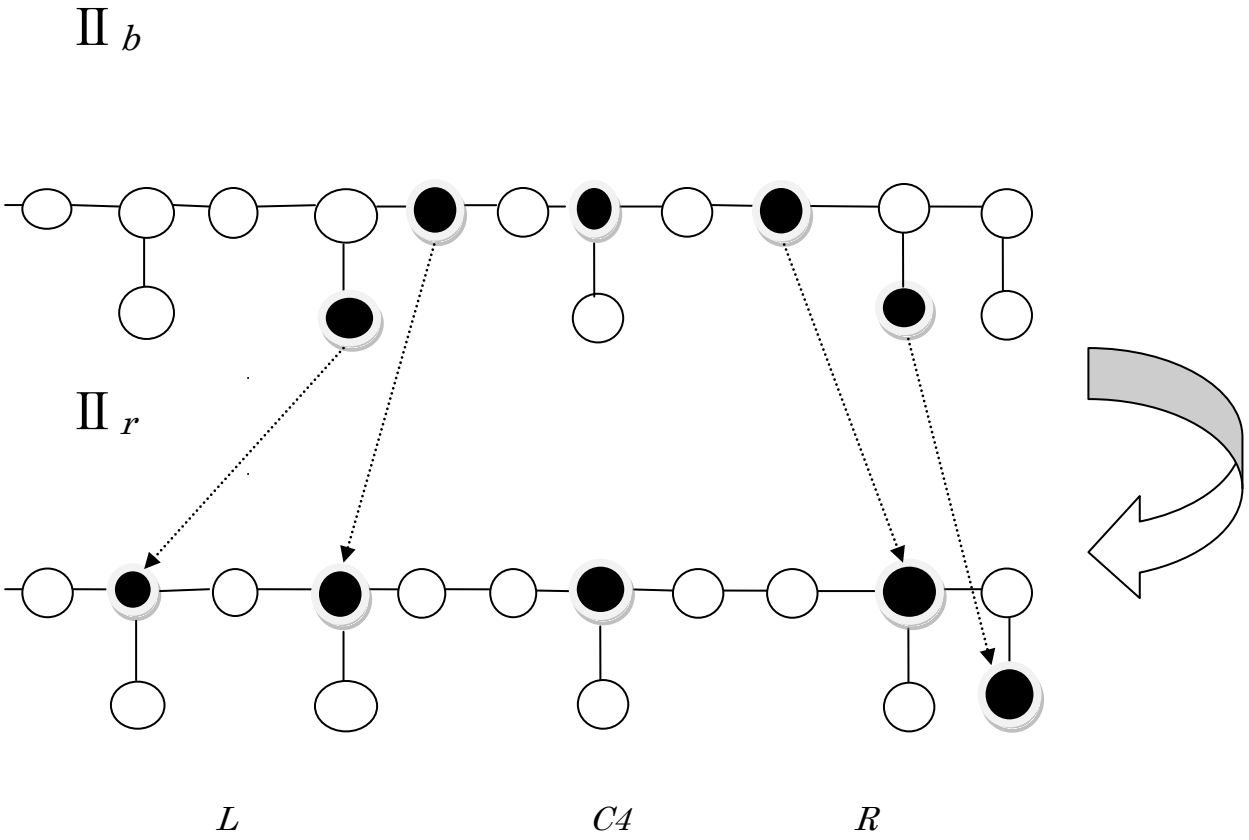

図 26　*C4* が両方向に対して detour の一部を構成する例

(5)*C5*が出現した場合

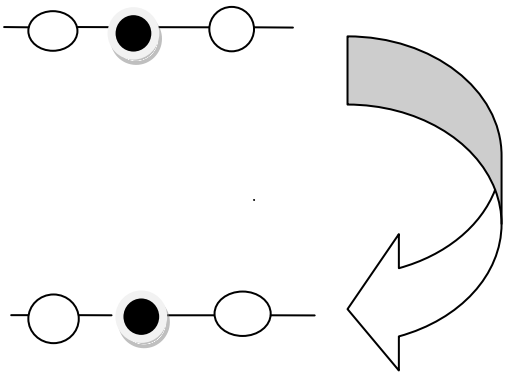

図 27　*C5*

C5 については，基本的に，トークンの遷移は発生しない．また，*C5* に対する左右各々を別々の部分グラフとして問題を考えればよい. ただし, *C4*と同様, *C5*が detour の一部を構成している場合がある．具体的には，*C5* のトークンが detour の case(a)の{$s[j-2\ n+2]$, ・・・, $s[j]$}のうちの一つのトークンに，または，case(b)の{$s[j-2\ n]$, ・・・, $s[j]$}のうちの一つのトークンに該当する場合である．対象とする独立点集合遷移問題が“遷移可能”であるという条件で議論を行っているので locked path は存在しない. したがって, *C4* の様に, 左右両方向に対して detour が検出されることはない．よって，上記に該当する場合，*C5* に対して，ともに detour を構成する他の遷移と同一の遷移方向を付与し(*R* または *L*)，*C5* を含めた全体を，新規の *R* または *L* として扱うことにより問題は解決される．図 28 において示した *C5①*のように，2 つの detour が構成されている場合もあるので，合計で 4 回の遷移を行い，元の位置に戻る場合もあり得るということを付記する．

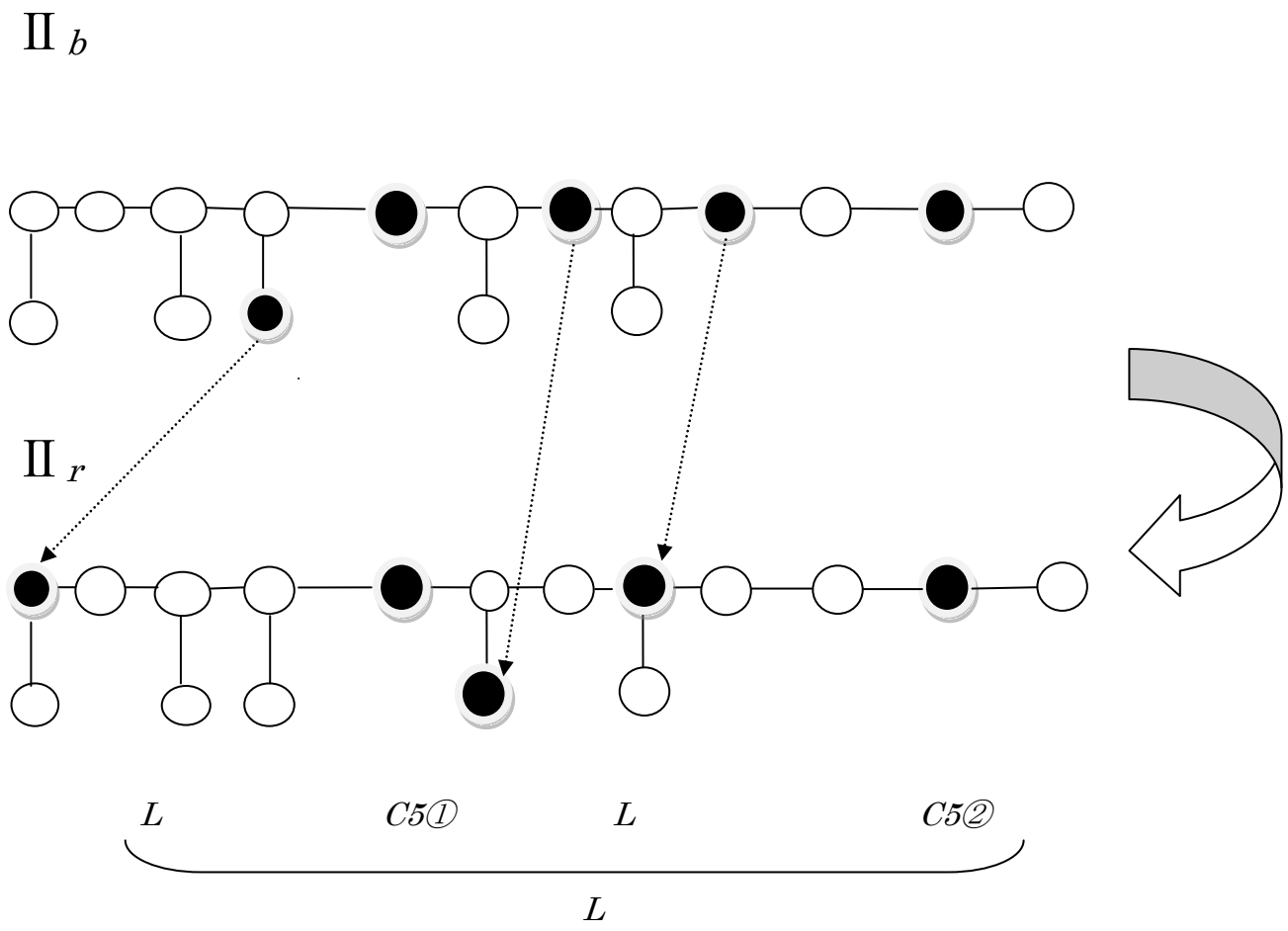

図 28　*C5* が detour の一部を構成する例および遷移の統合
(①，②の番号は 2 つある *C5* を区別するために便宜上付記したもの)

以上，*C1*～*C5* が現われた場合は，基本的に，トークンの遷移に関して，それ自身を除く左右各々を別々の部分グラフとして問題を扱うことができる．したがって，グループ相互の実行の優先順序という面からは，あとは，組合せ *LR* および組合せ *RL* の分析を行えばよい．

(6)組合せ *LR*

Π_b

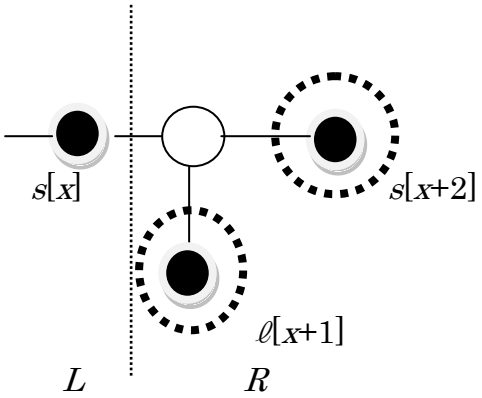

図 29　組合せ *LR*

図 15 の遷移例からもわかるように，Π_b の状態により優先順序を決定すればよい．対称性を考慮し，*L* の最右のトークンが spine に存在する場合を考えれば十分である．その位置を *s*[*x*]とする．その場合の *R* の状態の最も厳しい条件としては，最左のトークンが *ℓ*[*x*+1]または *s*[*x*+2]に存在する時であり，この場合を考えれば十分である．*ℓ*[*x*+1]の時は *L* の遷移を優先すればよい．*s*[*x*+2]の時は *L* または *R* において detour の case(b)が構成され，*s*[*x*]または *s*[*x*+2]に存在するトークンが detour の case(b)の *s*[*j* − 2 *n*]のトークンに該当する場合が比較可能となる．この場合，detour が構成されていない方を優先とすればよい．前提条件より，locked path は存在しないので，*L*, *R* ともに detour の case(b)が構成されていることはあり得ないということを付記する．

(7)組合せ *RL*

Π_r

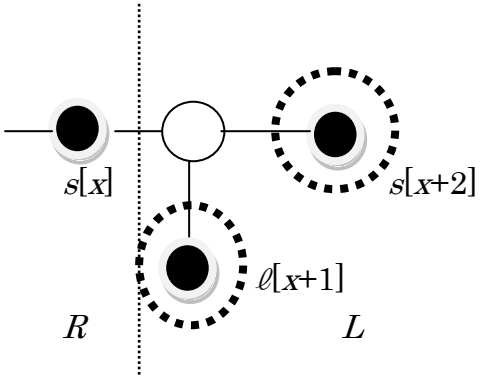

図 30　組合せ *RL*

図 15 の遷移例からもわかるように，Π_rの状態により優先順序を決定すればよい．対称性を考慮し，*R* の最右のトークンが spine に存在する場合を考えれば十分である．その位置を *s*[*x*]とする．その場合の *L* の状態の最も厳しい条件としては，最左のトークンが*ℓ*[*x*+1]または *s*[*x*+2]に存在する時であり，この場合を考えれば十分である．*ℓ*[*x*+1]の時は *L* の遷移を優先すればよい．*s*[*x*+2]の時は *R* または *L* において detour の case(b)が構成され，*s*[*x*]または *s*[*x*+2]に存在するトークンが detour の case(b)の *s*[*j* − 2 *n*]のトークンに該当する場合が比較可能となる．この場合，detour が構成されている方を優先とすればよい．前提条件より，locked path は存在しないので，*R*, *L* ともに detour の case(b)を構成することはあり得ないということを付記する．

4.4　アルゴリズムの構成

4.1～4.3 までの議論をまとめると下記 1.～4.となる．

1．補題 2 より，遷移前後のトークンの対応は一意に決定される．

2．トークンの遷移に関して，遷移の方向性の概念を導入し，その方向に基づきトークンのグループ化(*R, L, C1～C5*)を行う．遷移の実行を，グループ内部での遷移およびグループ相互の関係性というように，2 つの観点から扱う．

3．　caterpillar graph を含む tree 上の 2 つの頂点間の最短経路は一意に決定される．よって，グループ内部でのトークンの遷移は，detour の場合を除き，この最短経路上をすべての頂点を一度ずつ通って遷移すればよい．detour は遷移の実行において必要不可欠であり，最短遷移の実行に相反しない．

4．グループ相互の実行順序については，グループの集合に対して，半順序集合の概念を用いることにより最短遷移が確保される．すなわち，4.3 において見たように，グループどうしが比較可能な場合，最短遷移が実現される実行の順序付けを行えばよい．

以上が最短遷移系列を求めるアルゴリズムの構成の骨子であり，これらのことより定理 4 の実行可能性は明らかである．

第 5 章　まとめ

本論文では caterpillar graph の独立点集合遷移問題に対する線形時間アルゴリズムを示した．また，このアルゴリズムにより遷移可能と判定された場合の最短遷移系列を求める多項式時間アルゴリズムも示した．今後の課題としては tree における最短遷移系列を求めるアルゴリズムの計算時間の解明があげられる．

謝辞

本研究を行うにあたり，長期にわたり懇切丁寧な指導を賜りました上原隆平教授に心より感謝致します.

参考文献

[1] Erik D.Demaine, Martin L.Demaine, Takehiro Ito, Hirotaka Ono,and Ryuhei Uehara: Algorithms for Independent Set Reconfiguration Problem on Graphs. 信学技報, COMP2013-39 ,Vol.113 ,No.371, pp.7-14 (2013)

[2] Erik D.Demaine, Martin L.Demaine, Eli Fox-Epstein,Duc A. Hoang, Takehiro Ito, Hirotaka Ono,Yota Otachi,Ryuhei Uehara, and Takeshi Yamada:Linear-Time Algorithm for Sliding Tokens on Trees. Theoretical Computer Science Vol.600, pp.132-142(2015),October 2015

[3]R.J.ウィルソン著，西関隆夫・西関裕子共訳，グラフ理論入門(原書第 4 版)，近代科学社，2001

[4] Hearn, R.A., Demaine, E.D.: PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. Theoretical Computer Science 343, pp. 72-96(2005)