〔論　文〕

三角形をもたない正則グラフの生成と同型性の判別

朱雀　保正*・吉田　清明*・間瀬　　聖†

Computer construction of triangle-free regular graphs and discrimination of isomorphic graphs

Yasumasa SUJAKU, Kiyoaki YOSHIDA, Kiyoshi MASE

Abstract

An algorithm for constructing triangle-free, regular graphs is presented. Firstly, the methods for generating d-regular graphs using pseudo-random integers and for reducing triangles in the graphs are given. Then, isomorphic graphs are detected and removed. Since the efficient algorithm for discriminating isomorphic graphs is not known, a necessary condition for two graphs to be isomorphic is presented. This necessary condition gives an efficient algorithm to discriminate non-isomorphic graphs. Although sufficiency of the condition is not proved, this algorithm perfectly discriminated all the non-isomorphic graphs from those generated in the period of writing this paper. Finally, the amount of time elapsed in each step is shown.

Key words: triangle-free graph, regular graph, isomorphism, NP-complete, algorithm

1　まえがき

グラフは様々の工学的システムの設計の場で利用されており，いろいろな構造のグラフの性質や構成法が調べられている。そのなかで近年，クラスター（cluster）[1]という概念が注目を集めている。この概念は人のコミュニケーションネットワークがもつ興味深い性質を説明する際に用いられたものであり，ある頂点に隣接する頂点群において，これらの頂点同士がどの程度隣接しているかを表現するクラスター係数(clustering coefficient)[1]が定義されている。この定義によると，クラスターが存在するということは，グラフの枝が三角形を形作っているということである。一方，グラフにおける三角形という概念は従来から扱われており，特に三角形をもたないグラフ（triangle-free graph）が多方面で活発に研究されている[2]-[6]。

システムを設計する際など，三角形をもたないグラフを利用する立場からは，ある与えられた規模のグラフを生成する必要が生じるが，そのようなグラフの構成法は，三角形をもたないグラフの場合，特殊な例を除いては知られていないようである。そこで，本稿ではコンピュータを用いた方法として，ランダムグラフから出発して三角形をもたないグラフを生成する方法を提出する。本方法では，グラフの頂点数と頂点次数（頂点に接続されている枝の数）のみを指定して三角形をもたないグラフを次々に生成していく。このとき頂点次数を各頂点ごとに指定することも可能であるが，本稿では頂点次数がすべての頂点で同じ場合のみを扱う。

グラフの生成にあたっては，すでに生成されたグラフと同型なグラフが得られた場合はこれを捨てることにした。本稿では，同型性判別を高速化する目的で，同型であるための必要条件を提出する。同型性判別の計算複雑度は現在のところNP完全（NP-complete）[7]あるいはそれ以下[8]-[10]と予想されているが，この必要条件は本

* 情報ネットワーク工学科
† 情報ネットワーク工学科学士課程，現在佐賀大学大学院工学系研究科知能情報システム学専攻博士前期課程
平成18年4月28日受理

稿をまとめるにあたって計算したすべてのグラフにおいて，同型性を必要十分に判別している。また，必要条件の計算は瞬時に終了するが，この計算結果を利用すると，グラフが同型である場合に対応の可能性がある頂点の数を減らせることがあり，そのときはどの頂点同士が対応するかを求める計算時間が大幅に短縮できる。

2．グラフの定義

グラフは頂点と枝によって構成されるが，枝に向きがない（矢印がない）グラフを無向グラフという。グラフは，任意の2頂点間がいくつかの枝からなる道（path）でつながっているとき，連結であるという。また，同じ頂点に接続された枝（自己閉路）がなく，2頂点を接続する複数本の枝もないとき，グラフは単純(simple)であるという。さらに，グラフの頂点次数を d としたとき，d がどの頂点でも同じであるようなグラフを d 正則グラフ（d-regular）という。本稿は，連結・単純・正則な無向グラフ（connected simple regular graph）を対象とする。

グラフの接続関係を表現するのに，本稿では隣接行列(adjacency matrix) A を用いる。グラフの i 番目の頂点と j 番目の頂点が1本の無向枝で結ばれている（これを隣接しているという）場合，A の (i, j) 要素と (j, i) 要素は1であり，隣接しない2頂点に対応する要素は0である。よって，無向グラフの隣接行列は対称行列である。また，単純グラフでは，自己閉路がないので A の主対角要素は0である。さらに，d 正則グラフでは，各行，各列毎の要素の値の総和（1の総数）はすべて d である。

隣接行列の積 A^2 の (i, j) 要素の値は，i 番目の頂点と j 番目の頂点の間を2本の枝を経由（あるいは同じ枝を往復）して接続する道の数を表している。たとえば，ある頂点から d 本の枝が出ている場合，これを往復すれば元の頂点に戻るので，この頂点に対応する A^2 の主対角要素の値は d である。同様に A^3 は，3本の枝を経由して接続する道の数を表している。三角形をもたないグラフは，そのすべての頂点が3本の枝を経由して自分に戻る道をもたない。これを式で表現すると

$$\mathrm{diag}\, A^3 = (0,\ 0,\ \cdots,\ 0) \tag{1}$$

となる。ここで，diag は，主対角要素からなるベクトルを表す。

3．三角形をもたないグラフの生成手順

本稿では，頂点数が n で頂点次数が d であるような三角形をもたない正則グラフを生成するが，その手順はつぎのとおりである。n と d が与えられると，まず擬似乱数を用いて主対角要素が0で各行の1の総数が d であるような $n \times n$ の対称行列 $A_{regular}$ を生成する。つぎに，$A_{regular}$ に存在するすべての三角形を拾い出し，正則性を保存するような枝の付け替えを繰り返して，三角形を減らしていくことにより，三角形をもたない d 正則グラフの隣接行列 A_{t-free} を得る。こうして得られた A_{t-free} に対して，連結性の検査をした後，すでに得られているグラフと同型かどうかの判別を行う。以上をまとめると，三角形をもたない正則グラフを生成する手順はつぎのとおりである。

1．d 正則グラフの隣接行列 $A_{regular}$ を生成する。
2．三角形をもたない d 正則グラフの隣接行列 A_{t-free} へと変換する。
3．連結性の検査をする。
4．同型性の判別を行う。

以下では，この手順の各ステップについて説明する。

4．d 正則グラフの隣接行列 $A_{regular}$ の生成

本稿では，パソコンを用いてグラフの生成を行うが，そのためのプログラミングにはMathematicaを用いる。Mathematicaは整数型の擬似乱数*，Real [Integer, {*imin*, *imax*}] を生成できる。ここで，{*imin*, *imax*} は生成する乱数の範囲を表す。隣接行列のどの要素を1にするかを決めるのにこの擬似乱数を用いる。

d 正則グラフの隣接行列 $A_{regular}$ を生成する手順はつぎのとおりである。まず，要素がすべて0の $n \times n$ 行列 $A=(a[1]^T, a[2]^T, ,\cdots, a[n]^T)^T$ を用意する。ここで，$a[i]$ は行列 A の第 i 行を表し，$(\cdot)^T$ は転置を表す。最初に第1行 $a[1]$ の2列目から n 列目までの $(n-1)$ 個の要素

*擬似乱数列を生成する種はSeedRandom [*seed*] の *seed* で与えるが，Mathematicaの ver. 5.0 にはバグがある。すなわち，*seed* の値が k を整数，i を $0 \le i < 64$ を満たす整数とするとき $seed=64\times4k+i$ および $seed=64\times(4k+1)+i$，$seed=64\times(4k+2)+i$，$seed=64\times(4k+3)+i$ において，擬似乱数が同じ値をとってしまう。そこで，整数型の擬似乱数生成に用いる種 *seed′* には，

$$seed' = 64\times4\lfloor seed/64 \rfloor + \mathrm{Mod}[seed, 64] \tag{2}$$

を用いた。ここで，$\lfloor x \rfloor$ は，x を超えない最大の整数を表し，$\mathrm{Mod}[x, y]$ は x を y で割った余りを表す。*seed′* を用いれば周期性は消滅する。

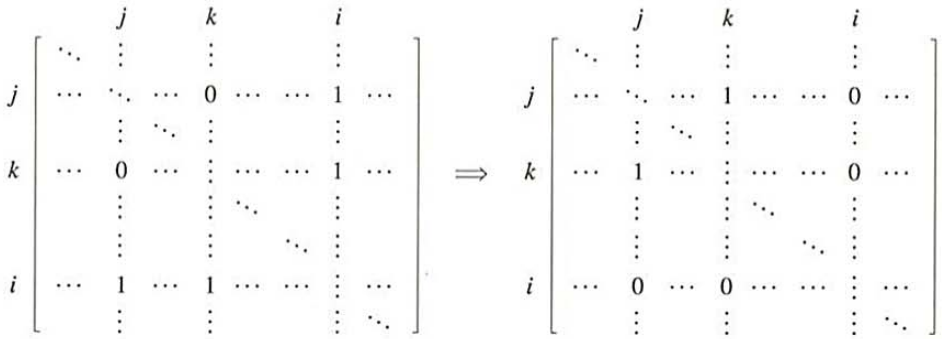

図1　4．1項の手順のステップ3における要素の値の変更法

のうち d 個の要素を0から1に変える。このとき，列の選択を等確率にするために，整数型擬似乱数を用いる。最初の擬似乱数の範囲は $\{imin=1, imax=n-1\}$ とし，以下 $\{imin=1, imax=n-2\}$,…, $\{imin=1, imax=n-d\}$ として，d 個の乱数を生成し，対応する列に1を代入していく。さらに，こうして得られた $a[1]$ の転置 $a[1]^T$ を A の第1列に代入する。これによって，A は対称行列となる。

つぎに第2行 $a[2]$ については，3列目から n 列目までの $(n-2)$ 個の要素のいくつかを0から1に変えるが，注意しなければならないのは，$a[1]^T$ の影響で第1列目が1の場合があることである。そのときは3列目から n 列目までの要素のうち $d-1$ 個の要素を0から1に変え，第1列目が0なら d 個の要素を0から1に変える。

4．1　1の総数が d を超える行への対処法

こうして順次行を進めていくが，途中の i 行目で，1列から $i-1$ 列の範囲だけで値が1の要素数が d 個を超えてしまうことがある。このときは，つぎの手順で i 行の二つの要素を1から0に変更し，i 行よりも上の行の要素の値をこれに伴い変更する。

1．第 i 行で，値が1の列の列番号のリストを作る。ここで，リストの要素数を m とする。

2．範囲が $\{1, m\}$ および $\{1, m-1\}$ の整数型擬似乱数を用いて，2つの列番号の組み合わせをランダムに選ぶ。ここでは，その列番号を (j, k) とする。もちろん，$1 \leq j, k < i$ が満たされている。

3．もしも，$A[j, k]=1$ ならばステップ2へ戻って別の列番号を選ぶ。ただし，$A[j, k]$ は，行列 A の第 j 行 k 列を表す。もし，図1に示すように，$A[j, k]=0$ ならば，次式のように要素の値を変更する。

$$A[j, k]=A[k, j]=1,$$
$$A[i, j]=A[i, k]=A[j, i]=A[k, i]=0 \qquad (3)$$

こうすることで，第 i 行の1の要素数は2だけ減少し，i 行より上の行の1の要素数は d のままになる。

4．第 i 行の1の要素数が d 以下になれば，この手続きを終了する。

5．ステップ1のリストから，上で用いた2つの列番号 j と k を取り除き，m を $m-2$ として，ステップ2へ進む。

4．2　1の総数が d に達しない行への対処法

前項とは逆に，途中の i 行目で，$i+1$ 列から n 列のすべてを1としても，1の要素数が d 個に満たないことがある。このときは，前項の手順の1を0，0を1に置き換えた，つぎの手順を実行する。

1．第 i 行の1列から $i-1$ 列において値が0の列の列番号のリストを作る。ここで，リストの要素数を m とする。

2．範囲が $\{1, m\}$ および $\{1, m-1\}$ の整数型擬似乱数を用いて，2つの列番号の組み合わせ (j, k) をランダムに選ぶ。

3．もしも，$A[j, k]=0$ ならばステップ2へ戻って別の列番号を選ぶ。もし，$A[j, k]=1$ ならば，次式のように要素の値を変更する。

$$A[j, k]=A[k, j]=0,$$
$$A[i, j]=A[i, k]=A[j, i]=A[k, i]=1 \qquad (4)$$

こうすることで，第 i 行の1の要素数は2だけ増加し，i 行より上の行の1の要素数は d のままになる。

4．第 i 行の1の要素数が $d-(n-i)$ 以上になれば，この手続きを終了する。

5．ステップ1のリストから，上で用いた2つの列番号 j と k を取り除き，m を $m-2$ として，ステップ2へ進む。

5．三角形をもたない d 正則グラフへの変換

前節で得られたグラフは，一般に三角形を含んでいる。

この三角形を拾い出すために，つぎの関数を導入しよう。

$$B=A*A^2 \tag{5}$$

ここで，＊は二つの行列の要素ごとの積を表す。行列 B は対称行列であり，つぎの性質をもっている。

1．隣接行列 A が三角形を含まないための必要十分条件は $B=O$ である。ここで，O は零行列を表す。

2．$B[i,j]=m>0$ であれば，番号 i，j の頂点を通る三角形が m 個存在する。

3．番号 (i,j,k) の頂点を結んだ三角形が存在する必要十分条件は，$B[i,j]$, $B[i,k]>0$ かつ $A[j,k]=1$ である。

そこでまず，上の性質2，3を用いて，すべての三角形を列挙する。つぎに各枝がいくつの三角形に含まれているかを数え上げ，その数の多い順に枝を並べ，三角形の数を枝の重みとする。この枝の並びの先頭から順に二つの枝を取り出す。いま，枝の対 $e_1=(i_1,j_1)$，$e_2=(i_2,j_2)$ が取り出されたとし，これらの枝の重みが w_1, w_2 であったとすると，これに対しつぎの操作を行う。

1．$A[i_1,i_2]=A[j_1,j_2]=0$ の場合：枝の対に，新たに枝の対 $e_3=(i_1,i_2)$，$e_4=(j_1,j_2)$ と重み $w_1+w_2-A^2[i_1,i_2]-A^2[j_1,j_2]$ を付与する。

2．$A[i_1,j_2]=A[j_1,i_2]=0$ の場合：枝の対に，新たに枝の対 $e_3=(i_1,j_2)$,$e_4=(j_1,i_2)$ と重み $w_1+w_2-A^2[i_1,j_2]-A^2[j_1,i_2]$ を付与する。

上で付与された重みは，最初の枝の対を除去し，新たな枝の対を付け加えたときに，総計でいくつ三角形が減少するかを与えている。そこで，つぎの操作を行う。

3．重みが最大値である枝の対を除去し，新たな枝の対を付け加える。

この一連の操作を繰り返すことで，三角形を除去する。

上の操作1が成り立つ場合の例を図2に示す。図で枝 (i_1,j_1) と枝 (i_2,j_2) を除去し，枝 (i_1,i_2) および枝 (j_1,j_2) を新たに付け加えた場合，左右にある計3個の三角形がなくなり，上下に新たに計2個の三角形が出現する。よって，差し引き $3-2=1$ 個の三角形が減少するので，操作1の重みは1である。図2では，操作2も成立する。この場合の重みは2である。

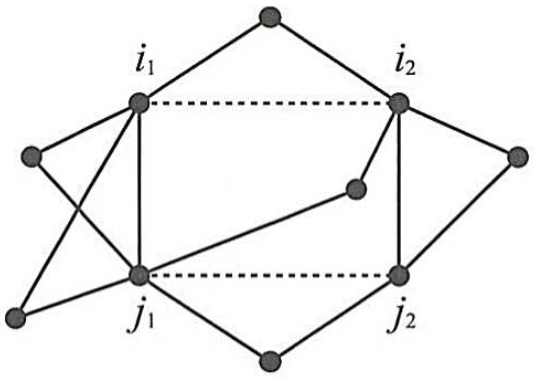

図2　三角形を減少させる枝の付け替えの例

6．連結性の検査

隣接行列 A から，このグラフが連結かどうかを判定するのは容易である。例えば，つぎの有限級数

$$A+A^2+A^3+\cdots+A^p \tag{6}$$

を $p=1$ から $2,3,\cdots,n-1$ と計算していき，値が0の要素の数が減少しなくなるまで繰り返す。最後に0の要素がなければ連結，あれば非連結である。非連結の場合は，その中の連結成分がこの級数から得られる。また，0の数が変化しない最小の p の値は，連結成分の中の任意2頂点間の距離の最大値（直径）を与える。本稿の場合，連結性のみを検査すればよいので，ある要素を含む連結成分のみを取り出すといった効率のよい方法を使うこともできる。

7．同型性の判別

二つのグラフが頂点番号の付け替えで同じグラフになるとき，これらは同型であるという。本稿では，同型なグラフは同じグラフとみなすので，新たに得られたグラフがすでに得られているグラフと同型かどうかを判別する必要が生じる。形が限定されたグラフを除いて，同型性判別には効率のよい方法が知られていないので，基本的には頂点の置換を繰り返すことになる。しかし，その繰り返し回数は $n!$ のオーダーなので計算時間が頂点数 n とともに爆発的に増大する。そこで，本稿では同型であるための必要条件を提出し，これを用いることにする。

7．1　同型性を判別するための必要条件

隣接行列 A から得られる，つぎの集合

$$\{A, A^2, A^3,\cdots, A^{n-d}\} \tag{7}$$

に対し，まず各行列の各行ごとに要素を昇順（または降順）に並べ替え，次に各行列ごとに行を昇順（または降順）に並べ替える。ただし，二つの行の並べ替えにおい

ては，先頭の要素から順に比較して，最初に値が異なった要素を用いて順序を決める。こうして得られた行列の集合を

$$S=\{S_1, S_2, S_3, \cdots, S_{n-d}\} \tag{8}$$

とする。上の集合の行列 S_i の各行は，ある一つの頂点から i 本の枝を経由して，自分自身を含む n 個の頂点の一つ一つに至る道の数を数の少ない順（または多い順）に並べたものになっている。また，S_i における各行の並びは，行の要素の大小関係で決められている。よって，S_i の値は頂点の番号にはよらない。すなわち，

定理 1　同型なグラフにおいては，S は等しい。あるいは，S が等しくないグラフは同型ではない。

定理 1 を用いることにより，多くの同型でないグラフが高速に判別できる。

7．2　S が等しいグラフの同型性判別の高速化

S が等しいグラフの中に同型でないグラフが存在するかどうかは不明なので，S が等しいグラフについては同型性を判別しなければならない。そこで，S の要素（行列）S_i に着目すると，S_i の各行は昇順（または降順）に並べ替えられているので，等しい行があれば連続して並んでいる。よって，等しい行を一つの部分集合とすることにより，S_i をいくつかの部分集合に分けることができる。そこで，この部分集合に属する各行をその頂点番号に置き換えると，S_i から頂点番号群を部分集合とする集合 V_i が得られる。この V_i の集合を

$$V=\{V_1, V_2, V_3, \cdots, V_{n-d}\} \tag{9}$$

とすると，次の定理が得られる。

定理 2　同型なグラフでは，V の中のすべての V_i の各部分集合に属する頂点同士が対応する。

定理 2 を用いると，同型性判別を高速化できる。例えば，n 個の頂点が m 個と $n-m$ 個からなる二つの部分集合に分けられたすると，同型性判別における頂点置換の最大回数は $n!$ から $m!(n-m)!$ となり，その比は

$$\frac{n!}{m!(n-m)!}=\binom{n}{m} \tag{10}$$

となる。ここで，上式右辺は二項係数であり，互いに区別できる n 個から m 個を取り出す組合せの数を表す。たとえば，$n=10$, $m=5$ の場合には，式(10)の値は252となるので，同型でないグラフを判別する計算速度が252倍に高速化されることになる。

この高速化の手法を用いて判別を試みているが，本稿準備中に行った計算においては S が等しいグラフの中に同型でないグラフは発見されなかった。

8．計算機実験結果

以下の計算機実験は，NEC 製デスクトップパソコンで行った。CPU は Pentium 4 (3GHz)，メモリは 512MB，OS は Windows 2000 SP4 である。アプリケーションソフトには Mathematica, ver. 5 を用いた。また，すべての実験結果は，10回の計算の平均値である。

8．1　d 正則グラフの生成

4 節のアルゴリズムにより，d 正則グラフ生成の実験を行った。図 3 に 1 個の正則グラフの生成に要した CPU 時間を示す。d は頂点次数，n は頂点数である。三角形を持たないグラフでは，$n \geq 2d$ が成り立つので，n の最小値は $n=2d$ とした。図 3 によると，CPU 時間はほとんど d には依らず，$n-2d$ のほぼ単調な増加関数である。

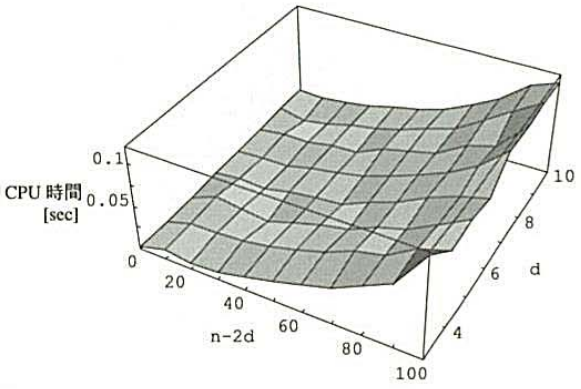

図 3　1 個の d 正則グラフの生成に要した CPU 時間

8．2　三角形をもたない d 正則グラフへの変換

5 節のアルゴリズムにより，前項で得られた d 正則グラフを三角形をもたない d 正則グラフへ変換できる。図 4 に三角形をもたない 1 個の正則グラフの生成に要した CPU 時間を示す。図 4 によると，CPU 時間は d と $n-2d$ の単調増加関数で，増加率が大きく図の大半が平坦に見える。そこで，CPU 時間軸を常用対数にした図を図 5 に示す。この図によると，増加の程度は指数関数よりも緩やかになっている。

参考のために，三角形をもたない正則グラフへの変換手順を始める前の d 正則グラフ 1 個に含まれる三角形の数を図 6 に示す。図 6 によると，三角形の数の初期値は $n-2d$ にはほとんど依存せず，d の単調増加関数であるように見える。図 4 と図 6 を比較すると，三角形数を 0 個にするのに要する CPU 時間は，三角形数の初期値のみの関数ではないことが分る。

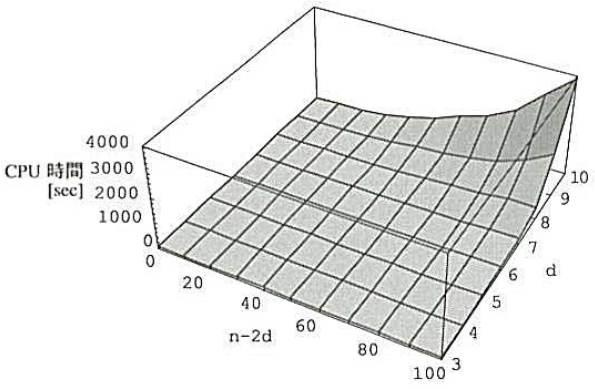

図4　三角形をもたない d 正則グラフを1個生成するのに要したCPU時間

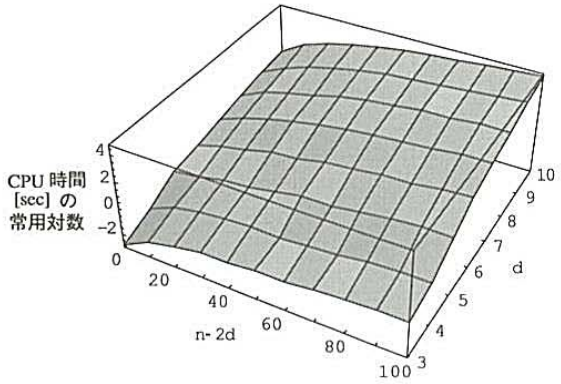

図5　図4のCPU時間軸を常用対数表示した場合

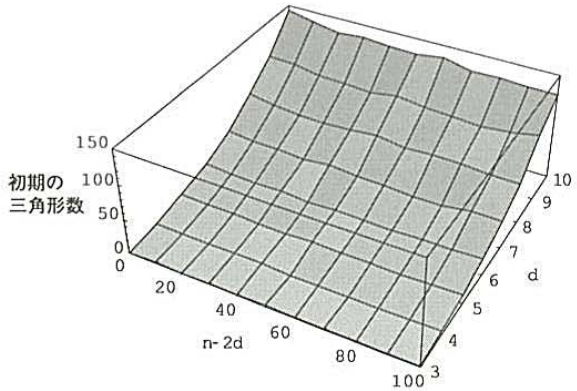

図6　初期の d 正則グラフに含まれる三角形の数

9．むすび

頂点数 n と頂点次数 d を与えて，三角形をもたない d 正則グラフを生成する方法を示し，同型であるための必要条件を与えた。生成に要するCPU時間を測定したところ，三角形を減少させるのに要する時間が支配的であり，パソコンで生成する場合，n が100程度まで，d が10程度までであれば，本方法が計算時間の点で実用できるという結果が得られた。

文　献

[1] 増田直紀，今野紀雄，複雑ネットワークの科学，産業図書株式会社，2005.

[2] J.B. Shearer, "A note on the independence number of triangle-free graphs," Discrete Math., vol.46, pp.83-87, 1983.

[3] S. Locke, "A note on bipartite subgraphs of triangle-free regular graphs," J. Graph Theory, vol.14, no.2, pp.181-185, 1990.

[4] J. Shearer, "A note on the independence number of triangle-free graphs II," Journal of Combinatorial Theory, Series B, 53, pp.300-307, 1991.

[5] J.B. Shearer, "A note on bipartite subgraphs of triangle-free graphs," Random Structures Algorithms, vol.3, no.2, pp.223-226, 1992.

[6] S. Poljak and Zs. Tuza, "Bipartite subgraphs of triangle-free graphs," SIAM J. Discrete Math. vol.7, no.2, pp.307-313, 1994.

[7] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman and Company, New York, 1979.

[8] 戸田誠之助，グラフ同型性判定問題，日本大学文理学部，2001.

[9] 戸田誠之助，"グラフ同型性判定問題の計算量，"信学論 (D-I)，vol.J85-D-I，no.2，pp.100-115，Feb. 2002.

[10] 名古屋孝幸，谷聖一，戸田誠之助，"グラフ同型写像の数え上げ問題に対するアルゴリズム，"信学論 (D-I)，vol.J85-D-I，no.5，pp.424-435，May 2002.