2001年度
青山学院大学理工学部理工学部
卒業論文

研究題目

Scale-Freeネットワーク上での感染伝播

細田　由喜子
羽田野研究室

青山学院大学 理工学部
物理学科 卒業論文

Scale-Freeネットワーク上
での感染伝播

細田由喜子 著

Scale-Freeネットワーク上での感染伝播

細田由喜子

羽田野研究室

平成14年2月20日

概 要

従来規則格子で考えられてきた感染伝播モデルをScale-Freeネットワーク上でシミュレーションし、つながり方の違うネットワークにおける感染状態の比較を行なった。その結果、Scale-Freeネットワークが感染伝播しやすいつながり方をしている事がわかった。

目 次

1　はじめに

世の中には数多くのネットワークが存在している。インターネットやWorld Wide Web (WWW) だけでなく、人間同志のつながりもネットワークと考える事ができる。その多くのネットワークがつながり方に次のような特徴を持つという事が知られている [1]:

- nodeが持つ edge 数 k の頻度分布 $P(k)$ が冪乗になる。

$$P(k) \propto k^{-\gamma} \tag{1}$$

ここで、node とはネットワークの中継点、edge は中継する線の事を言う。このような特徴を持つネットワークを **Scale-Free ネットワーク**と呼ぶ。

本研究では、Scale-Free ネットワーク上の感染伝播を調べた。従来の感染伝播モデルは、規則格子のような単純なもので考えられてきた [2]。しかし、それでは実際の感染病やコンピューターウィルスの感染伝播を再現するのには限界があるように思われる。現実に感染の媒体となる人間同士やコンピューターのつながりは、上述の様に規則格子よりずっと複雑なネットワークのはずだからである。そこで、感染伝播のモデルをより現実的にするため、Scale-Free ネットワーク上での伝播シミュレーションを行なった。

本研究では、つながり方が感染伝播に何らかの関係があるのかどうかを確かめる事を目的としている。そのため、他のネットワークと感染状態の比較をした結果を述べる。まず、第２節では、本研究で用いたネットワークについての詳細を説明する。第３節では、新しい感染伝播モデルについて述べる。第４節では、第３節で行ったシミュレーション結果についての考察、そして第５節では本研究のまとめを述べる。

2　様々なネットワーク

この節では、本研究で用いたネットワークについての詳細を説明する。

2.1　Scale-Free ネットワーク

従来のネットワーク理論では、グラフ理論によるランダムネットワークが主に研究されてきた。ランダムネットワークとは、後述の様に node

数は固定された状態で 、node 間に edge がランダムに引かれているものである。その edge 数の頻度分布はポアソン分布

$$P(k) \propto e^{-ak} \tag{2}$$

に近い分布となる。

しかし、現実の多くのネットワークでは、 式 (1) のような分布となる事が知られている。例として、WWW やインターネット、科学者の共同研究等があげられる [1]。それまでのランダムネットワークの考え方では、分布 (1) を再現する事ができなかった。そこで、新しいネットワークモデルとして、Barabási ら [1] によって Scale-Free モデルが提案されたのである。これには、重要な 2 つの要素が組み込まれている:

成長性: ネットワークには常に新しい node が追加されている。

優先的接続: ネットワークに新しく追加された node が選ぶ接続先は、edge 数の頻度に比例した確率で選ばれる。

例えば、node を論文、その論文が引用されたら edge が引かれるものとすると、

1. インターネット上に常に新しい論文が掲載されていく。
2. 有名な論文は見つけやすく、多くの論文に引用されるが、ほとんどの論文は引用される機会が少ない。

と言い換える事ができる。上の 2 つの要素を取り入れると、実際に式 (1) を満たすネットワークを作成する事ができる [1]。

本研究で用いた Scale-Free ネットワークの作成法を具体的に説明する。

(i) 初期状態として、1 番目と 2 番目の node が 1 本の edge で接続されている状態を作る。

(ii) 3 番目の node をネットワーク内に導入する。この node と edge で接続するための接続先を選択する。
選択確率はそれぞれの node の持つ edge 数に比例させる。具体的には、

♢ 1 番目の node が持つ edge 数は 1

♢ 2 番目の node が持つ edge 数は 1

なので、ネットワークの総 edge 数は 2 となる。よって、

♢ 1 番目の node の選択確率は $\frac{1}{2}$

♢ ２番目の node の選択確率は $\frac{1}{2}$

となる。

(iii) ３番目の node の接続先を決定する。
例えば、１番目の node が接続先として選ばれたとする。

(iv) ４番目の node をネットワーク内に導入する。
接続先の選択確率は、それぞれ

♢ １番目の node の選択確率は $\frac{2}{4}$

♢ ２番目の node の選択確率は $\frac{1}{4}$

♢ ３番目の node の選択確率は $\frac{1}{4}$

となる。

(v) ４番目の node の接続先を決定する。

(vi) 上記のように node がネットワークに導入される度に、(ii) と (iii) を繰り返す。

このようにしてネットワークを作成した後、edge 数の頻度分布を調べると図 1 のようになった。これは、node 数が無限大の極限で、傾き γ が３のフラクタル分布になる [1]。

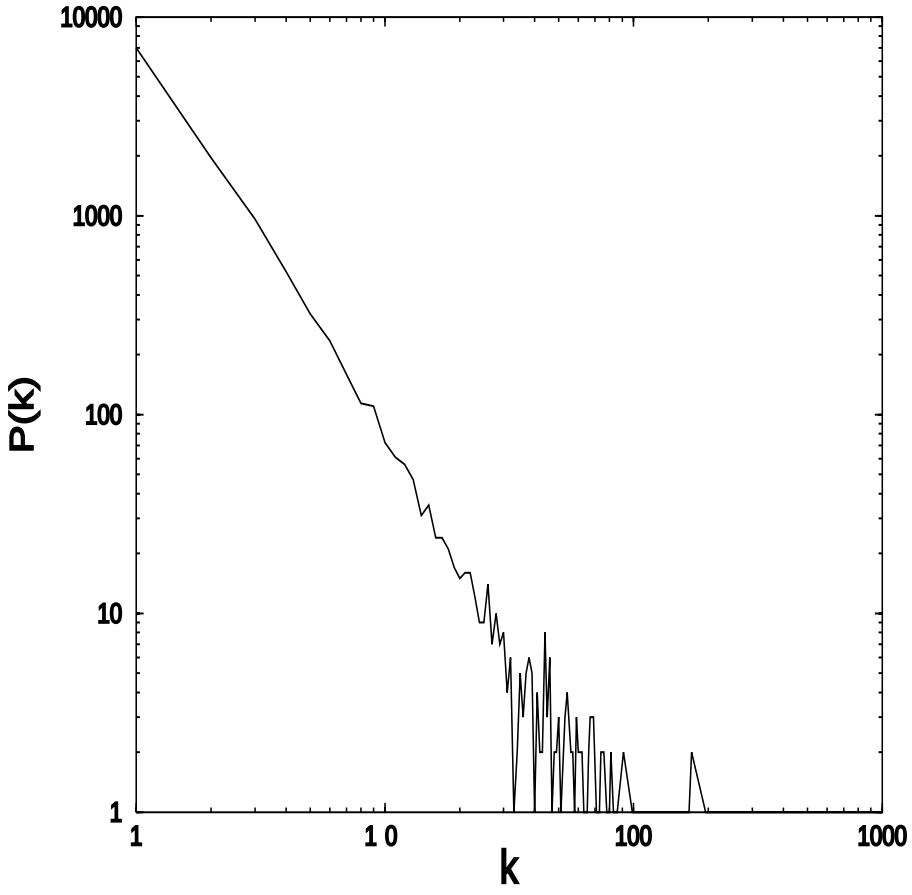


図 1: Scale-Free ネットワークにおける node の持つ edge 数 k の頻度分布 $P(k)$。node 数 12000 のシミュレーション結果。

2.2 ランダムネットワーク

本研究では、 Scale-Free ネットワークの他に、ランダムネットワークと規則格子（１次元鎖、正方格子）上でも比較のために感染伝播のシミュレーションを行なっている。ランダムネットワークと Scale-Free ネットワークの違いは次の通りである:

ネットワーク	Scale-Free ネットワーク	ランダムネットワーク
成長性	node 数増大	node 数は固定
接続法	edge 数の多い node に 優先的に接続される	node 間にランダムに edge がひかれる

これをふまえて、ランダムネットワークの作成を次のように行なった。

(i) 初期状態は全ての node がそれぞれ孤立している状態から始める。
(ii) それぞれの node にランダムに接続先を一つ決定する。
(iii) これを全ての node に対して行う。

以上の操作で、ランダムネットワークを作成した。

2.3 規則格子

比較対象として作成した規則格子は、１次元鎖と正方格子である。各々について、作成プログラムを説明する。

- **１次元鎖**

周期的境界条件を考慮して、全 node で１つの円を描くように接続先を決定する。

今、node が N 個存在すると

♢ １番目の node は２番目の node と接続する。

♢ ２番目の node は３番目の node と接続する。

⋮

♢ $(N-1)$ 番目の node は N 番目の node と接続する。

♢ N 番目の node は 1 番目の node と接続する。

となる。

以上の操作で、1 次元鎖を作成した。

• 正方格子

♢ 同様に周期的境界条件を用いて正方格子を作成した。

3　感染伝播モデル

この節では、新しい感染伝播モデルについて述べる。

冬になると身の回りで風邪にかかる人が続出し、自分も風邪をひく。1 月や 2 月になると、日本全国でインフルエンザの流行が見られる。新種のコンピューターウィルスが出回り、多くの被害が出ている事を耳にする。これらは感染伝播が現実の世界で起こっている例である。

従来の感染伝播モデルが規則格子で考えられてきた、という事は前にも述べた。しかし、実際の人間同士やインターネットは Scale-Free ネットワークになっている。そこで、ネットワーク上での感染伝播のシミュレーションを行ない、現実により近い結果を得ようと考えた。以下では、2 つのプロセス（感染、治癒）を実行する感染伝播モデルを考える [2]。

3.1　感染プロセス

感染プロセスの対象となるのは、健康状態の node である。健康状態の node を選び、それに接続している node がどれか 1 つでも感染している場合、選ばれた node がある確率で感染する。これが、感染プロセスである。

例えば風邪は、身の回りの人間とのつながりでできているネットワーク上で伝播する病気と考える。まず、周りの誰もが健康状態ならば、風邪に感染する可能性は、ほとんど皆無である。しかし、つながりのある人間の誰か 1 人でも風邪に感染した場合は、自分が感染する可能性が出てくる。つまり、感染プロセスの対象者となるのである。

3.2 治癒プロセス

治癒プロセスは感染プロセスと違い、感染状態の node が対象である。感染状態の node をランダムに選び、周囲の状況に関わりなく、ある確率で健康状態にする。これが治癒プロセスである。

感染プロセスの場合、接続している node の状態が大きく関わってきた。しかし治癒プロセスの場合は自然治癒や外部からの治療を施される事を考え、接続状況からは影響を受けないとする。

3.3 モデルの説明

上記の感染プロセスと治癒プロセスを取り入れたモデルを、具体的に説明する。

(i) ネットワークを作る:
以前に説明した手順でネットワークを作成する。
(ii) 初期状態として、ネットワーク内の node をある割合で感染状態にする。
(iii) 治癒プロセスを行なう。ネットワーク内の感染している node 全てが対象となる。治癒率に見合った場合、node は回復し、健康状態に戻る事になる。
(iv) 次に感染プロセスの対象となる node を見つけ出す。対象となるのは、接続している node がどれか１つでも感染している健康状態の node である。
(v) 対象に選ばれた健康状態の node に、感染プロセスを行なう。感染率に見合えば、node は感染状態となる。
(vi) 手順 (iii)(iv)(v) を繰り返す。

以上が本研究の感染伝播モデルとなる。

3.4 シミュレーション結果

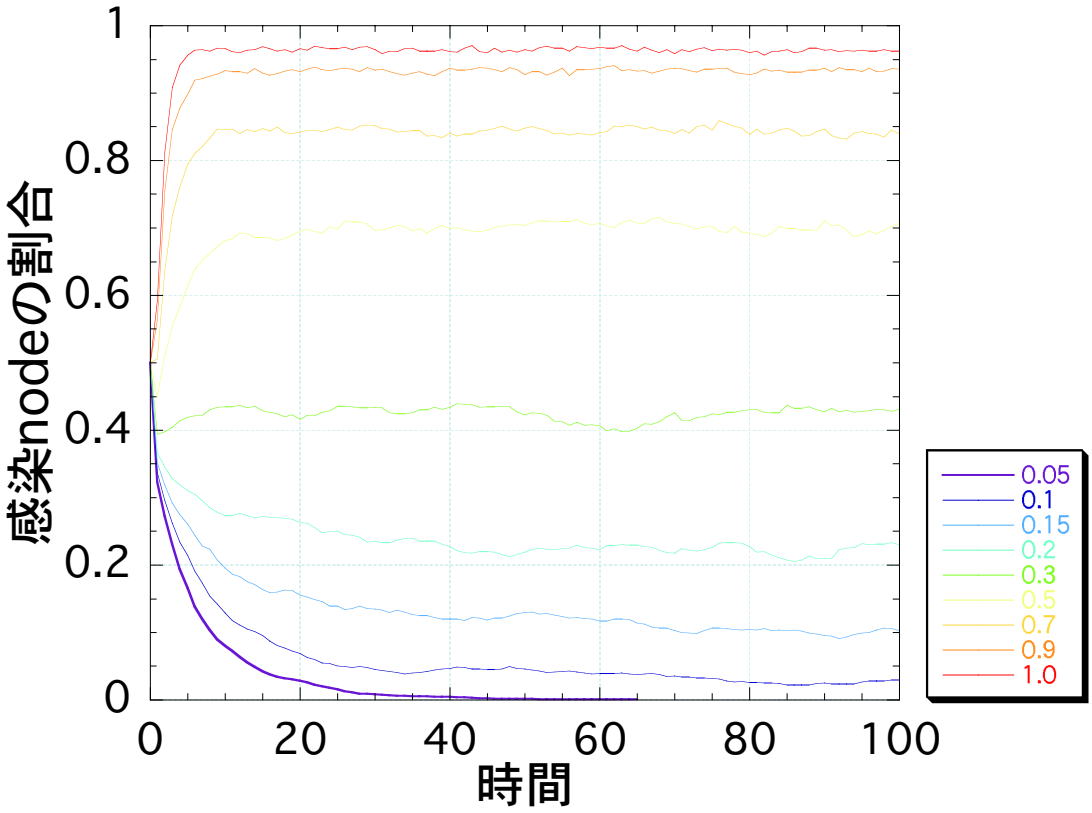
比較対象となる規則格子では、１次元鎖の node が持つ平均 edge 数は 2、正方格子の node が持つ平均 edge 数は 4 である。そこで、Scale-Free ネットワーク、ランダムグラフについてもそれぞれ平均 edge 数 2 のものと、平均 edge 数 4 のものを作成した。以上の６つのネットワークに対し

て感染伝播シミュレーションを行なった。

治癒率は 0.2 に固定し、感染率を変えて感染伝播の様子を調べた。初期状態はネットワーク内の node の約 5 割が感染している状態に設定した。結果を図 2 と図 3 に示す。

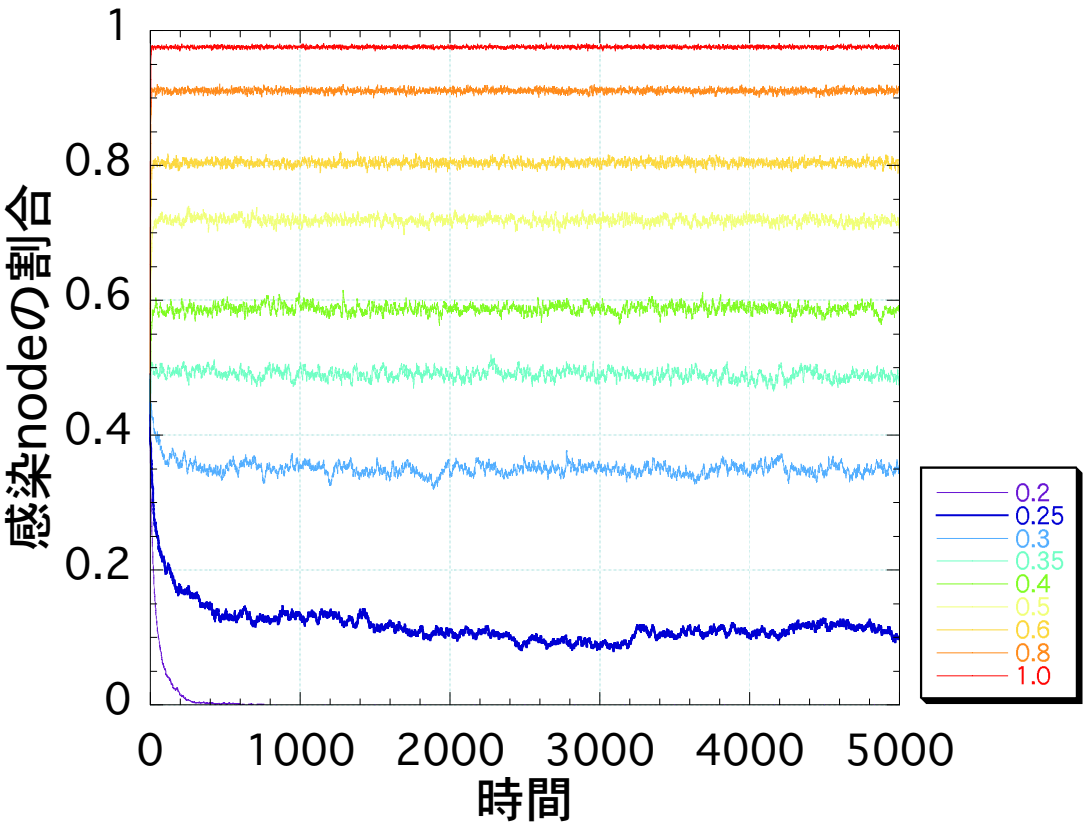
まず、シミュレーション結果の図 2 に注目する。感染率が低すぎる場合は感染が伝播せず、全ての node が健康となる撲滅状態となっている。しかし、ある閾値以上の感染率に設定すると、ある程度の node が感染している平衡状態に落ち着いている。例えば、図 2(a) では、約 20 回を過ぎると平衡状態となる。なお、初期状態を変えても、ほぼ同じ平衡状態を示した。

次に、各々の場合で平衡状態とみなせる時点の感染割合とその時の感染率をプロットしたのが図 4 と図 5 である。いずれも、ネットワークによる差が見られた。

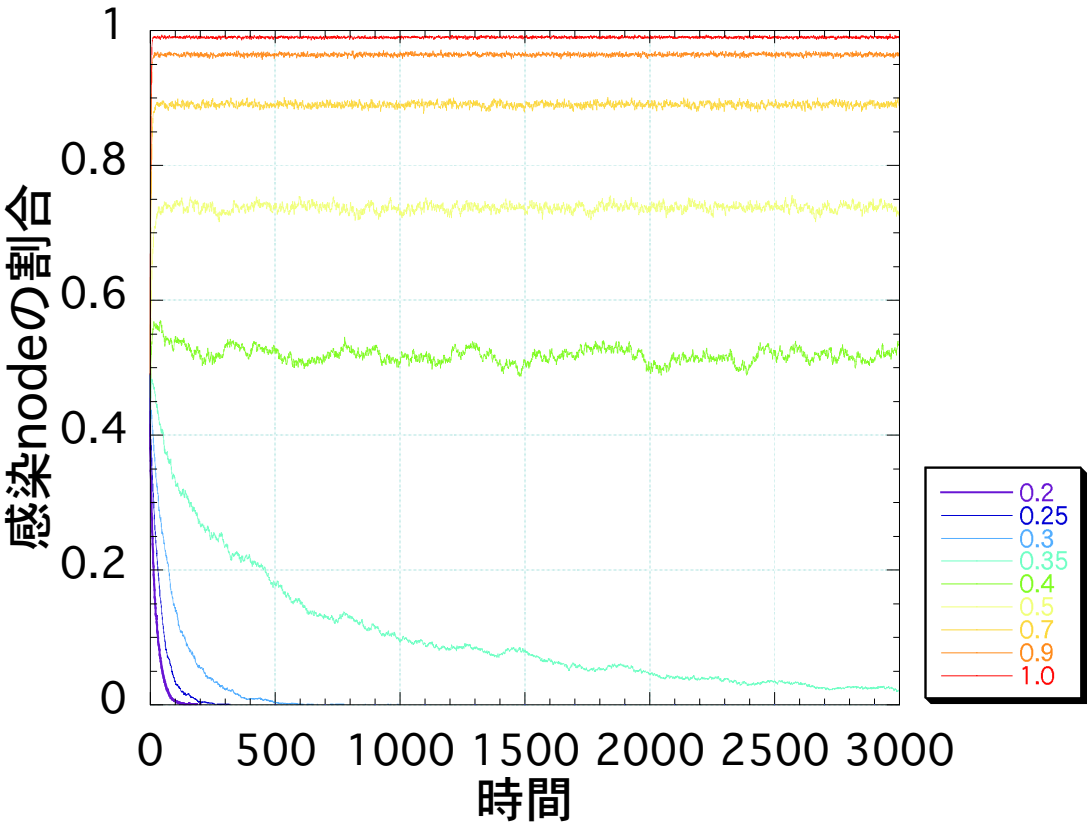


(a)

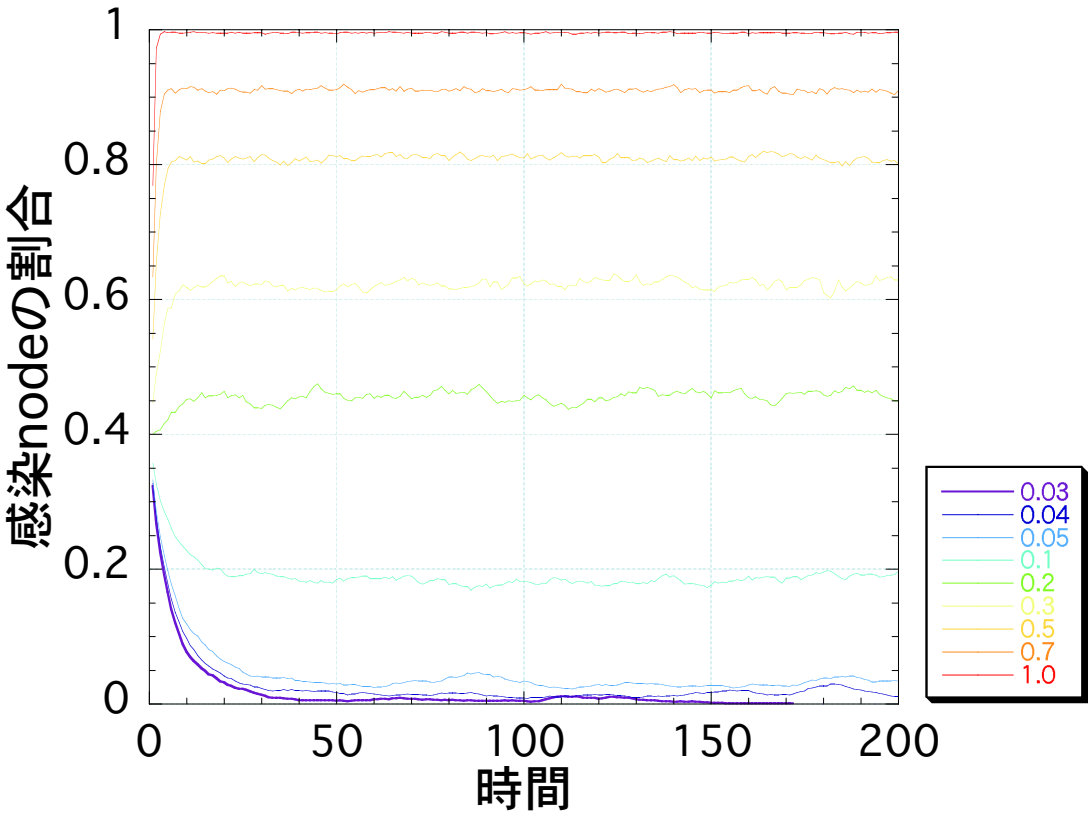
図 2: 感染伝播シミュレーションの結果。node 数 12000、平均 edge 数 2、治癒率 0.2。(a)Scale-Free ネットワーク (感染率 0.05 〜 1.0)；(b) ランダムネットワーク (感染率 0.2 〜 1.0)；(c) 規則格子 (1 次元鎖)(感染率 0.2 〜 1.0)。


感染nodeの割合
時間
0
0.2
0.4
0.6
0.8
1
0
1000
2000
3000
4000
5000
0.2
0.25
0.3
0.35
0.4
0.5
0.6
0.8
1.0

(b)

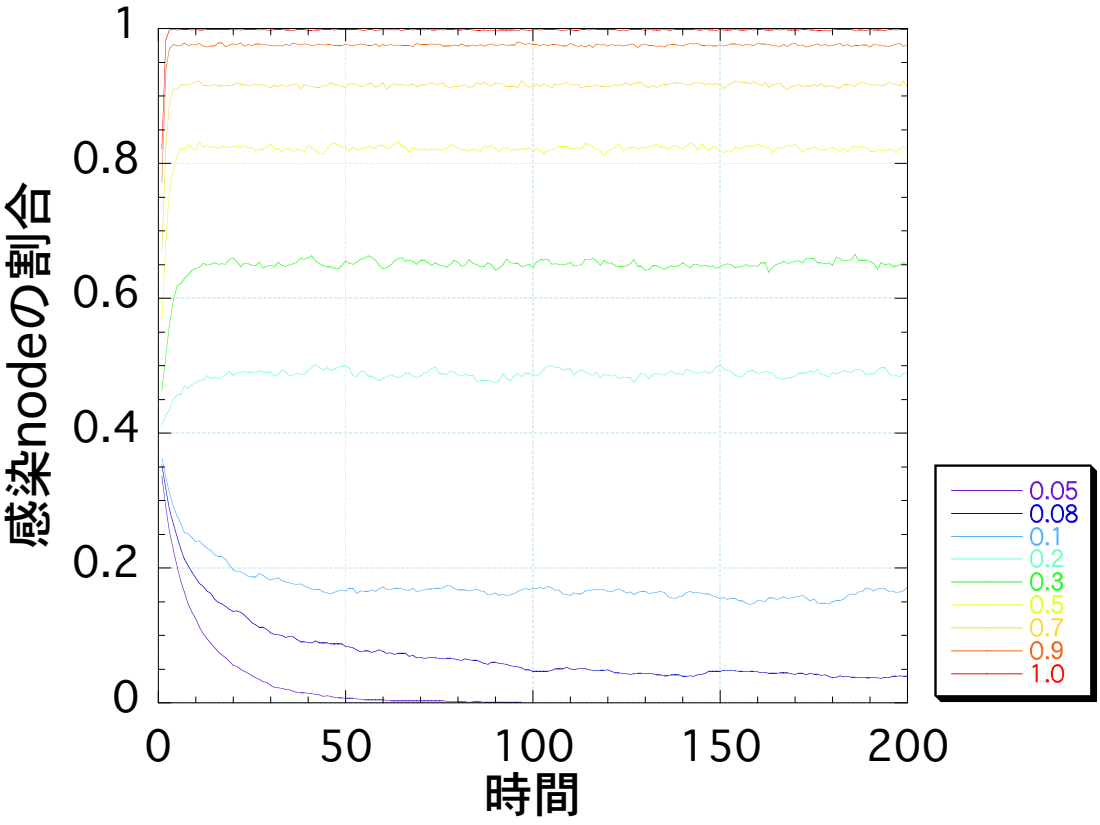

感染nodeの割合
1
0.8
0.6
0.4
0.2
0
0
500
1000
1500
2000
2500
3000
時間
0.2
0.25
0.3
0.35
0.4
0.5
0.7
0.9
1.0

(c)

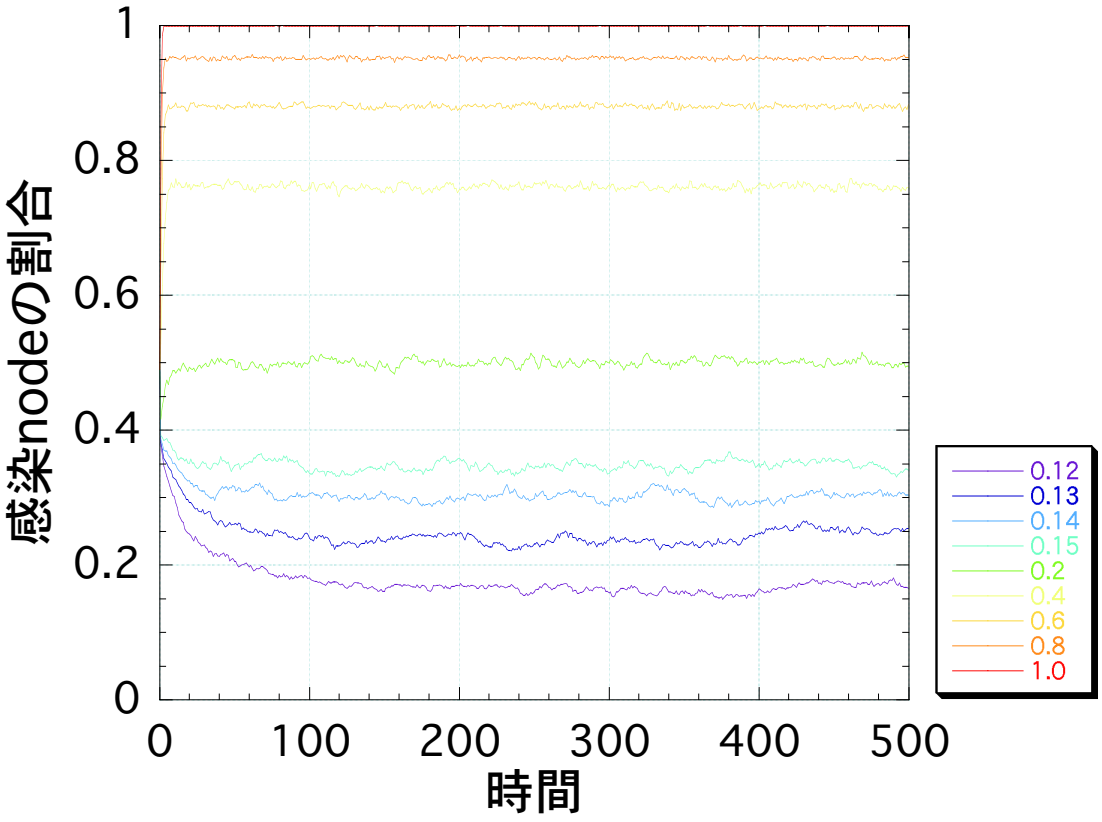


(a)

図 3: 感染伝播シミュレーションの結果。node 数 12000、平均 edge 数 4、治癒率 = 0.2。(a)Scale-Free ネットワーク (感染率 0.03 〜 1.0)；(b) ランダムネットワーク (感染率 0.05 〜 1.0)；(c) 規則格子 (正方格子)(感染率 0.12 〜 1.0)。


感染nodeの割合
1
0.8
0.6
0.4
0.2
0
0
50
100
150
200
時間
0.05
0.08
0.1
0.2
0.3
0.5
0.7
0.9
1.0

(b)


感染nodeの割合
時間
0
0.2
0.4
0.6
0.8
1
0
100
200
300
400
500
0.12
0.13
0.14
0.15
0.2
0.4
0.6
0.8
1.0

(c)

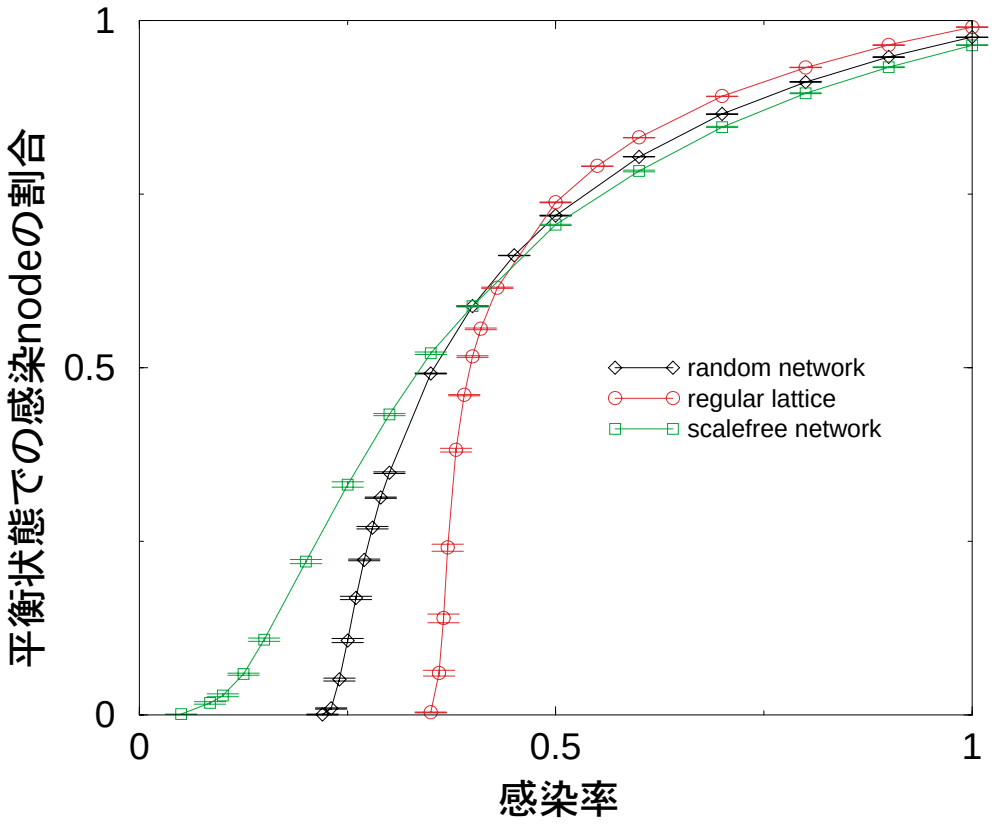


図 4: 平衡状態における感染の割合 (平均 edge 数２)。用いたネットワークは Scale-Free ネットワーク (平均 edge 数２)、ランダムネットワーク (平均 edge 数２)、１次元鎖である。

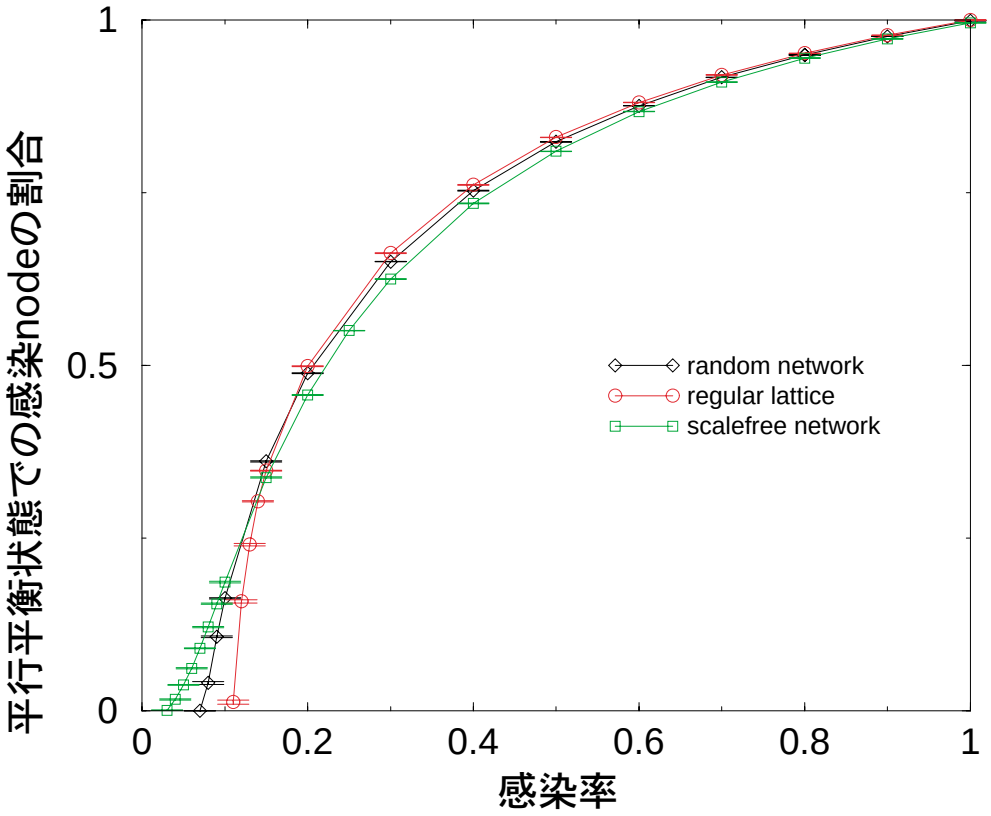


図 5: 平衡状態における感染の割合 (平均 edge 数４)。用いたネットワークは Scale-Free ネットワーク (平均 edge 数４)、ランダムネットワーク (平均 edge 数４)、正方格子である。

4　考察

シミュレーション結果を考察する前に、もう１度目的を述べておく。今回の研究は、ネットワークのつながり方が感染伝播に影響を与えているかどうかを確かめる事である。図 4 と図 5 を見てみると、各々の結果に違いが見られた。これより、ネットワークのつながり方というのは感染伝播に影響を持つと言える。また、edge 数の頻度のバラツキが平衡状態のでやすさと比例している可能性がある事がわかった。そこで、ネットワークのつながり方に注目しながら、シミュレーション結果を考察する。

4.1　平均 edge 数２の場合

まず、図 4 に注目する。感染率 0.4 以下で３つのネットワークのシミュレーション結果に違いが現れている。Scale-Free ネットワークが、他の２つより低い感染率でも、ある程度の平衡状態を示している事がわかる。この後に、ランダムネットワーク、１次元鎖と続く。この結果から考えられる事を述べる。

１次元鎖は１つの円のような形をとっている。感染伝播はこの１本道で行なわれる。そのため、どこかで道がつまる、つまり感染しない node が存在した場合、伝播が進まないと考えられる。また、このネットワークでは １つの node が接続できるのは、２つの node しかない。よって、自身が感染状態にあり、その事で感染プロセスの対象にする事ができる node 数は最大２個までである。この事から、１つの感染 node がネットワークの感染に与える影響が少ない事が考えられる。

これに対して、治癒プロセスは健康 node 全てを対象とする。 感染率が高い場合は、感染プロセスが治癒プロセスの効果を上回るため、感染がネットワーク上に拡がっていく。しかし、低くなるにつれ、感染プロセスによる効果が減少してくる。そして、対象数の多い治癒プロセスの効果がそれを上回り、感染状態の撲滅へと向かう。図 4 では、治癒率 0.2 に対して、感染率 0.3 以下で撲滅状態になっている。これは、２つのプロセスの対象数の大差によるもの、と考えられる。

これに対して、ランダムネットワークと Scale-Free ネットワークは平衡状態がでやすいつながり方をしていると考えられる。特に、後者の方がその傾向が強いと言える。

１次元鎖では１つの感染 node が感染プロセスの対象とできる最大 node

数は２個までだというのは、前に述べた。これに対して、ランダムネットワークと Scale-Free ネットワークは各々の node の接続状況によって、その数は大きく変わってくる。ランダムネットワークの場合は、平均 edge 数が２であるポアソン分布になるように接続されている。今回シミュレーションに用いたものでは、１つの感染 node が１個から８個までを感染プロセスの対象とする可能性を持つ。Scale-Free ネットワークの場合は、その範囲が１個から 200 個近くまで拡大される。 edge 数の頻度分布が冪乗となるという事は、少数派が大きな edge 数を持ち、大多数が１本程度の edge しか持たない事を意味する。よって、どの nodeが感染したかで、２つのプロセスの効果の優劣も大きく変わってくると予想される。

図 4 の結果より、平衡状態のでやすさの順位は Scale-Free ネットワーク、ランダムネットワーク、１次元鎖であった。この事をふまえると、 edge 数の頻度のバラツキが平衡状態のでやすさと比例している可能性が考えられる。

4.2　平均 edge 数４の場合

図 4 と比べると、図 5 では全体的にかなり低い感染率でも平衡状態を示す結果となった。これは edge 総数の増大によるものだと考えられる。図 4 に対して図 5 では、node 数を固定した状態で、平均 edge 数が２倍になっている。つまり、感染ルートが増えたため、より伝播がしやすくなったと考えられる。

感染割合の違いは、感染率 0.2 以下から現れている。ネットワークによる差は図 4 に比べて小さいが、誤差範囲は超えており、また平衡状態のでやすさの順位は図 4 と同様である。この事からも、つながり方によって感染伝播への影響があると言う事ができる。

5　まとめ

本研究では、Scale-Free ネットワーク上での新しい感染伝播モデルをシミュレーション行った。またその研究目的は、ネットワークのつながり方が感染の伝播に何らかの影響を与えるものなのかを確かめる事であった。そこで、Scale-Free ネットワークとはつながり方の違うランダムネットワーク、規則格子 (１次元鎖、正方格子) に対しても、シミュレーションを行った。その結果から２つの事が言える。

◇ ネットワークのつながり方の違いは、感染伝播に何らかの影響を与える。

◇ Scale-Free ネットワークは、規則格子、そしてランダムネットワークよりも感染伝播しやすいつながり方をしている。

今後の課題として、まずシミュレーションを行うネットワーク数を増やす事をあげたい。例えば、平均 edge 数 2 のランダムネットワークは、作成過程で与えられる乱数が異なれば、つながり方に違いが生じる事が予想される。場合によっては、Scale-Free ネットワークにかなり近い頻度分布を示すネットワークを作成する事も考えられる。今回は、各々1パターンで計6つのネットワークでしかシミュレーションを行っていない。

また、新たな感染伝播モデルを提案したいと考えている。その結果が同様になる事を示せば、今回の結果が正しいという可能性が高くなると期待している。

謝辞

私は、研究というものがどういうものかも知らずに研究室に入りました。それから時が経ち、無事に卒業論文を書き終える事ができました。最初から最後まで、私に飽きずに御指導して下さった羽田野先生には大変感謝しております。そして、院生の先輩方には勉学面だけでなく、生活面においても御指導頂きました。そして、一緒に１年間研究し支えてくれた卒研生の皆さん、本当に有難うございました。また、家族の支えもあり、ここまで自分の研究に打ち込む事ができました。私に関わったすべての方々に厚く御礼を申し上げます。

参考文献

[1] A.-L. Barabási and R.Albert, Science **286** , 509 (1999)

[2] 香取眞理,「複雑系を解く確率モデル」,（講談社，1997）

A　プログラムリスト

A.1　Scale - Free ネットワーク (平均 edge 数２)

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#define SIMPLE_SPRNG
#include "sprng.h"
#define SEED 29873754
#define _N 12000        /*node 総数-1*/

main()
{
  int i,j,sum-edge,edges,select;
  int max,No,noconnect;
  static int edge[_N+1],connect[_N+1][_N+1];
  static double select-node,P;

  FILE *scale_node;
  /*接続状況の出力ファイル*/
  scale_node=fopen("one_scale_node.dat","w");

  init_sprng(DEFAULT_RNG_TYPE,SEED,SPRNG_DEFAULT);

/* 初期化 */
  for(i=0;i<=_N;i++)
    {
      edge[i]=0;                  /*node の持つ edge 数*/
      for(j=0;j<=_N;j++)
         {
            connect[i][j]=0;    /*node の接続状況を記録する配列
*/
         }
    }
```

```
  /* 初期条件 */
  edge[0]=1      /*０番と１番が接続している*/;
  edge[1]=1;
  connect[0][1]=1;
  connect[1][0]=1;

/*ネットワーク作成*/
  for(i=2;i<=_N;i++)
    {
      sum-edge=0;
      for(j=0;j<i;j++)
         {
           sum-edge+=edge[j];    /*ネットワーク内の総 edge 数*/
         }

      edges=0;
      select-node=(double)sprng();    /*接続先の選択確率*/
      for(j=0;j<i;j++)
         {
           edges+=edge[j];
           P=(double)edges/sum-edge;       /*j 番目の選択確率*/

           if(select-node<=P)
             {
               select=j;
               break;
             }
          }

      /*接続状況と edge 数の決定*/
       edge[i]=1;
       edge[select]++;
       connect[i][select]=1;
       connect[select][i]=1;
```

```
  }    /*ネットワーク作成終了*/

 /*最大 edge 数の選出*/

max=0;
for(i=0;i<=_N;i++)
  {
    if(edge[i]>=max)
       {
        max=edge[i];
       }
  }

printf("%d\n",max);     /*最大 edge 数出力*/

/*接続状況のファイル出力*/
for(i=0;i<=_N;i++)
  {
    No=0;
    for(j=0;j<=_N;j++)
       {
         if(connect[i][j]==1)
          {
             fprintf(scale_node,"%d\n",j);
             No++;
          }
       }

    for(j=No;j<max;j++)
       {
             noconnect=-1;
             fprintf(scale_node,"%d\n",noconnect);
       }
   }
```

```
  fclose(scale_node);
}
```

A.2　ランダムネットワーク (平均 edge 数２)

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#define SIMPLE_SPRNG
#include "sprng.h"
#define SEED 29473049
#define _N 12000    /*node 総数-1*/

main()
{

  int i,j,select_node;
  int max,No,noconnect;
  static int edge[_N+1],connect[_N+1][_N+1];

  FILE *random_node;
  /*接続状況の出力ファイル*/
  random_node=fopen("one_random_node.dat","w");

  init_sprng(DEFAULT_RNG_TYPE,SEED,SPRNG_DEFAULT);

  /*初期化*/
  for(i=0;i<=_N;i++)
    {
      edge[i]=0;      /*node の持つ edge 数*/
      for(j=0;j<=_N;j++)
         {
            connect[i][j]=0;    /*node の接続状況を記録する配列
*/
         }
    }

    /*ネットワーク作成*/
```

```
for(i=0;i<=_N;i++)
  {

    select-node=(double)sprng()*_N;
    while(connect[i][select-node]==1)
       {
          select-node=(double)sprng()*_N;
       }

     /*接続状況と edge 数の設定*/
    edge[i]++;
    edge[select-node]++;
    connect[i][select-node]=1;
    connect[select-node][i]=1;

  }

 /*最大 edge 数の選出*/
max=0;
for(i=0;i<=_N;i++)
  {
    if(edge[i]>=max)
       {
         max=edge[i];
       }
  }
printf("%d\n",max);     /*最大 edge 数出力*/

/*接続状況のファイル出力*/
for(i=0;i<=_N;i++)
  {
    No=0;
    for(j=0;j<=_N;j++)
       {
        if(connect[i][j]==1)
```

```
            {
              fprintf(random_node,"%d\n",j);
              No++;
            }
         }

      for(j=No;j<max;j++)
         {
               noconnect=-1;
               fprintf(random_node,"%d\n",noconnect);
         }
    }
  fclose(random_node);
}
```

A.3　１次元鎖

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#define SIMPLE_SPRNG
#include "sprng.h"
#define SEED 29473049
#define _N 12001        /*node 総数-1*/

main()
{

  int i,j;
  int max,No,noconnect;
  static int edge[_N],connect[_N][_N];

  FILE *regular_node;
  /*接続状況の出力ファイル*/
  regular_node=fopen("regular_node.dat","w");

  init_sprng(DEFAULT_RNG_TYPE,SEED,SPRNG_DEFAULT);

  /*初期化*/
  for(i=0;i<_N;i++)
    {
      edge[i]=0;
      for(j=0;j<_N;j++)
         {
           connect[i][j]=0;
         }
    }

   /*ネットワーク作成*/
   for(i=0;i<_N;i++)
    {
```

```
  /*接続状況の決定*/
    connect[i%_N][(i+1)%_N]=1;
    connect[(i+1)%_N][i%_N]=1;
  /*edge 数の決定*/
    edge[i%_N]++;
    edge[(i+1)%_N]++;
   }

/*最大 edge 数の選出*/
max=0;
for(i=0;i<_N;i++)
  {
    if(node[i]>=max)
       {
         max=node[i];
       }
  }
printf("%d\n",max);       /*最大 edge 数出力*/

/*接続状況のファイル出力*/
for(i=0;i<_N;i++)
  {
    No=0;
    for(j=0;j<_N;j++)
       {
         if(connect[i][j]==1)
           {
             fprintf(regular_node,"%d\n",j);
             No++;
           }
       }
  for(j=No;j<max;j++)
       {
         noconnect=-1;
         fprintf(regular_node,"%d\n",noconnect);
```

```
        }
    }
  fclose(regular_node);
}
```

A.4 Scale - Free ネットワーク (平均 edge 数４)

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#define SIMPLE_SPRNG
#include "sprng.h"
#define SEED 29873754
#define _N 12000          /*node 総数-1*/

main()
{

  int i,j,max,select,select2,psum,edges,max_edge,No,noconnect;
  static int connect[_N+1][_N+1],node[_N],edge[_N+1];
  double hyouka,P,hyouka2;

  FILE *two_dimen;
  /*接続状況の出力ファイル*/
  two_dimen=fopen("two_dimen_scale.dat","w");

  init_sprng(DEFAULT_RNG_TYPE,SEED,SPRNG_DEFAULT);
  /*初期化*/
  for(i=0;i<=_N;i++)
    {
      edge[i]=0;
      for(j=0;j<=_N;j++)
         {
           connect[i][j]=0;
         }
    }

   /* 初期条件 */
  edge[0]=2;
  edge[1]=2;
```

```
edge[2]=2;
connect[0][1]=1;
connect[1][0]=1;
connect[0][2]=1;
connect[2][0]=1;
connect[1][2]=1;
connect[2][1]=1;

/*ネットワーク作成*/
for(i=3;i<=_N;i++)
  {
    psum=0;
    for(j=0;j<i;j++)
       {
         psum+=edge[j];
       }

    edges=0;
    hyouka=(double)sprng();
    for(j=0;j<i;j++)
      {
         edges+=edge[j];
         P=(double)edges/psum;

         if(hyouka<=P)
           {
             select=j;
             break;
           }
      }

    edge[i]+=2;
    edge[select]++;
    connect[i][select]=1;
    connect[select][i]=1;
```

```
  edges=0;
  hyouka2=(double)sprng();
  for(j=0;j<i;j++)
     {
       edges+=edge[j];
       P=(double)edges/psum;
       if(hyouka2<=P)
         {
           select2=j;
           break;
         }
     }
  while(connect[i][select2]==1)
     {
       edges=0;
       hyouka2=(double)sprng();
       for(j=0;j<i;j++)
         {
           edges+=edge[j];
           P=(double)edges/psum;
           if(hyouka2<=P)
              {
                select2=j;
                break;
              }
         }
     }
  edge[select2]++;
  connect[i][select2]=1;
  connect[select2][i]=1;

}
```

```
 /*最大 edge 数の選出*/
 max=0;
 for(i=0;i<=_N;i++)
   {
     if(edge[i]>max)
        {
          max=edge[i];
        }
   }
 printf("%d\n",max);    /*最大 edge 数出力*/

for(i=0;i<=_N;i++)
   {
     No=0;
     for(j=0;j<=_N;j++)
        {
          if(connect[i][j]==1)
            {
              fprintf(two_dimen,"%d\n",j);
              No++;
            }
        }

     for(j=No;j<max;j++)
        {
          noconnect=-1;
          fprintf(two_dimen,"%d\n",noconnect);
        }
   }
fclose(two_dimen);
}
```

A.5　ランダムネットワーク (平均 edge 数 4)

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#define SIMPLE_SPRNG
#include "sprng.h"
#define SEED 29473049
#define _N 12000        /*node 総数-1*/

main()
{

  int i,j,select_node,select_node2;
  int max,No,noconnect;
  static int edge[_N+1],connect[_N+1][_N+1];

  FILE *two_dimen;
  /*接続状況のファイル出力ファイル*/
  two_dimen=fopen("two_dimen_random.dat","w");

  init_sprng(DEFAULT_RNG_TYPE,SEED,SPRNG_DEFAULT);

  /*初期化*/
  for(i=0;i<=_N;i++)
    {
      edge[i]=0;
      for(j=0;j<=_N;j++)
         {
           connect[i][j]=0;
         }
    }
  /*ネットワーク作成*/
  for(i=0;i<=_N;i++)
```

```
  {
    /*１つ目の接続先の決定*/
    select_node=(double)sprng()*_N;
    while(connect[i][select_node]==1)
       {
         select_node=(double)sprng()*_N;
       }
    connect[i][select_node]=1;
    connect[select_node][i]=1;

    edge[i]+=2;
    edge[select_node]++;

    /*２つ目の接続先の決定*/
    select_node2=(double)sprng()*_N;
    while(connect[i][select_node2]==1)
       {
         select_node2=(double)sprng()*_N;
       }
    connect[i][select_node2]=1;
    connect[select_node2][i]=1;
    edge[select_node2]++;
  }
/*最大 edge 数の選出*/
max=0;
for(i=0;i<=_N;i++)
  {
    if(edge[i]>=max)
       {
         max=edge[i];
       }
  }

printf("%d\n",max);     /*最大 edge 数出力*/
/*接続状況のファイル出力*/
```

```
  for(i=0;i<=_N;i++)
    {
      No=0;
      for(j=0;j<=_N;j++)
         {
           if(connect[i][j]==1)
             {
               fprintf(two_dimen,"%d\n",j);
               No++;
             }
         }

      for(j=No;j<max;j++)
         {
               noconnect=-1;
               fprintf(two_dimen,"%d\n",noconnect);
         }

    }
  fclose(two_dimen);
}
```

A.6　正方格子

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#define _N 12000    /*node 総数*/

main()
{
  int i,j;
  int max,noconnect,No;
  static int edge[_N],connect[_N][_N];

  FILE *two_dimen;
  /*接続状況の出力ファイル*/
  two_dimen=fopen("two_dimen_regu-lat.dat","w");
  /*初期化*/
  for(i=0;i<_N;i++)
    {
      edge[i]=0;
      for(j=0;j<_N;j++)
         {
           connect[i][j]=0;
         }
    }
 /*ネットワーク作成*/
 for(i=0;i<_N;i++)
   {
     connect[i][(i+100)%_N]=1;
     connect[(i+100)%_N][i]=1;

     if(i%100==99)
       {
          connect[i][i-99]=1;
          connect[i-99][i]=1;
       }
```

```
    else
      {
         connect[i][i+1]=1;
         connect[i+1][i]=1;
      }
    node[i]=4;
  }
/*最大 edge 数の選出*/
max=0;
for(i=0;i<_N;i++)
  {
    if(edge[i]>max)
      {
        max=edge[i];
      }
  }
printf("%d\n",max);      /*最大 edge 数出力*/
/*接続状況のファイル出力*/
for(i=0;i<_N;i++)
   {
     No=0;
     for(j=0;j<_N;j++)
        {
          if(connect[i][j]==1)
            {
              fprintf(two_dimen,"%d\n",j);
              No++;
            }
        }
     if(No!=max)
        {
          for(j=No;j<max;j++)
            {
              noconnect=-1;
              fprintf(two_dimen,"%d\n",noconnect);
```

```
            }
        }
    }

 fclose(two_dimen);
}
```

A.7　感染伝播モデル

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#define SIMPLE_SPRNG
#include "sprng.h"
#define SEED 29873754
#define _N 12000      /*node 総数*/
#define s_infe 0.03   /*感染率*/
#define s_cure 0.2     /*治癒率*/
#define _T 300          /*時間*/
#define _P 0.5           /*初期感染率*/
#define _M 240          /*最大 edge 数*/
#define _E 200           /*平衡状態に落ち着いてある程度経った時
間*/

main()
{
  int i,j,inif_State,inif_t,t,sum_State,L;
  static double v_cure,v_infe,V_infe,P_State,sum_Pro,pars;
  static double average_05,a[11],b1,b2,bb,ER;
  static int State[_N+1];
  static int inif_infe,infect[_N+1],af_State[_N+1],link[_N+1][_M];

  FILE *graph_1,*two_dimen;

  /*感染率-感染 node の割合のデータ出力ファイル*/
  graph_1=fopen("two_scale_0.03.dat","w");

  /*接続状況のデータファイル読み込み*/
  two_dimen=fopen("two_dimen_scale.dat","r");

  init_sprng(DEFAULT_RNG_TYPE,SEED,SPRNG_DEFAULT);

  /* 初期化 */
```

```
  for(i=0;i<=_N;i++)
    {
      State[i]=0;          /*node の状態 0なら健康  1なら感染
*/
      infect[i]=0;         /*感染プロセスの選別  1なら対象者*/
      af_State[i]=0;
      for(j=0;j<_M;j++)
         {
           link[i][j]=0;
         }
    }
   /*接続状況の読み込み*/
  for(i=0;i<=_N;i++)
    {
      for(j=0;j<_M;j++)
         {
           fscanf(two_dimen,"%d\n",&link[i][j]);
         }
    }

  fclose(two_dimen);

  /* 感染伝播モデル*/

  average_05=0.0;
  for(L=1;L<=10;L++)
    {
      No=0;
      a[L]=0;

      /*初期状態*/
      for(i=0;i<=_N;i++)
        {
           pars=sprng();
           if(pars<=_P)
```

```
        {
          inif_infe=sprng()*_N;
          State[inif_infe]=1;
        }
  }
/*感染伝播*/
sum_Pro=0.0;
No=0;
for(t=1;t<=_T;t++)
  {
     P_State=0.0;
     /*治癒プロセス*/
     for(i=0;i<=_N;i++)
      {
         if(State[i]==1)
           {
              v_cure=sprng();
              if(v_cure<=s_cure)
                {
                   State[i]=0;
                }
           }
       }
     /*感染プロセス*/
    for(i=0;i<=_N;i++)
       {
         infect[i]=0;
       }

    for(i=0;i<=_N;i++)
      {
         if(State[i]==1)
            {
              for(j=0;j<_M;j++)
                {
```

```
                        if(link[i][j]!=-1 && State[link[i][j]]!=1)
                           {
                             infect[link[i][j]]=1;
                           }
                      }
                  }
            }
           sum_State=0;
           for(i=0;i<=_N;i++)
             {
               if(infect[i]==1)
                 {
                    v_infe=sprng();
                    if(v_infe<=s_infe)
                      {
                        State[i]=1;
                      }
                 }
               sum_State+=State[i];
              }
           /*感染node の割合*/
           P_State=(double)sum_State/(double)(_N+1);

           if(t==_E)
             {
                    sum_Pro+=P_State;     /*平衡状態での感染node
の割合*/
              }
          }

      a[L]=sum_Pro;
      printf("%lf\n",a[L]);
      average_05+=sum_Pro;

   }
```

```
  /*感染 node の割合の平均値*/
  average_05=average_05/10.0;

  /*分散と誤差棒の算出*/
  b1=0.0;
  b2=0.0;
  for(L=1;L<=10;L++)
    {
      b1=(average_05-a[L])*(average_05-a[L]);
      b2+=b1;
    }
  bb=b2/9.0;
  ER=sqrt(bb/10.0);
  /*感染率-感染 node の割合-誤差棒のデータ出力*/
  fprintf(graph_1,"%lf   %lf   %lf\n",s_infe,average_05,ER);

  fclose(graph_1);
}
```