〔駒沢女子大学　研究紀要　第17号　p. 375 ～ 394　2010〕

Artisocによる媒介中心性指数計算とネットワーク表示について

末　木　俊　之*

Betweenness Centrality Caliculation and Network Display with Artisoc

Toshiyuki SUEKI*

キーワード：スケールフリー・ネットワーク　中心性　コミュニティ抽出　幅優先探索アルゴリズム

1．ネットワークの表示と中心性指数

マルチエージェントシミュレーションソフトartisoc textbook[1)]を使いネットワーク上の病気の流行などのシミュレーションを行おうとすると、何らかの実在のネットワークモデル（ノードとリンクの情報）を使うか、人工的に生成したネットワークモデルを使ってシミュレーションすることを考えるであろう。

人工的にネットワークを生成するシミュレーションでランダム・ネットワークを生成するには、『人工社会構築指南 artisocによるマルチエージェント・シミュレーション入門』[1)]第27章で紹介されているシミュレーションが使える。スケールフリー・ネットワークを生成するには、MASコミュニティー（http://mas.kke.co.jp/index.php）[2)]で紹介されている『Barabasi-Albertモデル』artisocシミュレーションをダウンロードして使うことができる。（図１.）は、このシミュレーションを使い人工的に生成させたノード数500個のスケールフリー・ネットワークの例である。

図中の数字は各ノードを一意に区別するためのID番号である。０～499の整数が付与されている。

人工的に任意の場所にノードが生成されているのでリンクが交錯した非常に煩雑な図になっている。このノード配置で病気の流行等のシミュレーションを行っても視覚的に有益な情報が得られるとは思われない。可能であればノードの表示をリンクの交錯がないすっきりとした表示にしたいと考えるであろう。

１つの簡単な表示例として（図２.）のように表示させることを考えてみた。

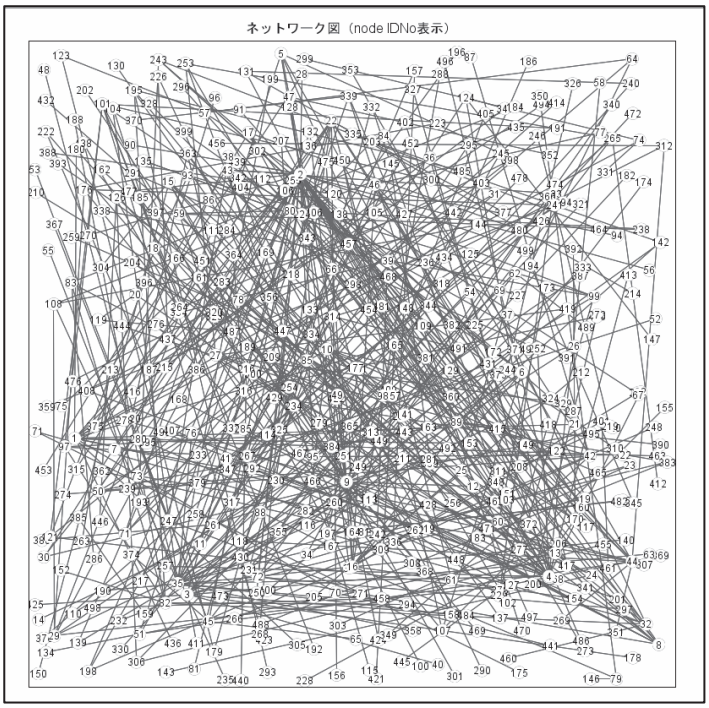
（１）全てのノードを円内に配置する。

（２）重要度の高いノードがより円の中心に来るように配置する。

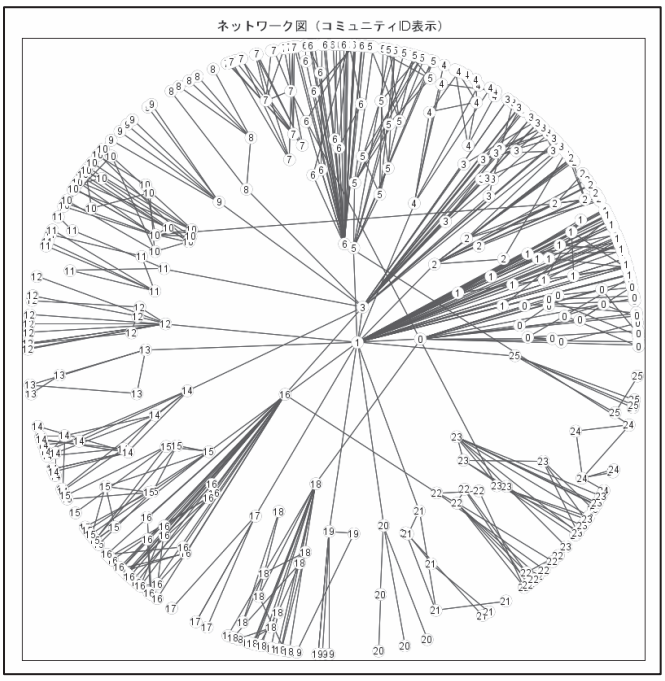
（３）より密なリンクで互いに結合し合っているノードを集めて表示する。そのような集まりをコミュニティと呼ぶとするなら、（図３.）概説図が示すように、（図２.）の例図には26個のコミュニティがあり、同一コミュニティに属するノードを扇型内に集めて表示してある。図上の数字は、各ノードの所属するコミュニティを一意に区別する番号になっている。

これにより、リンクの交錯を減らして、すっきりとしたノード表示にすることができる。

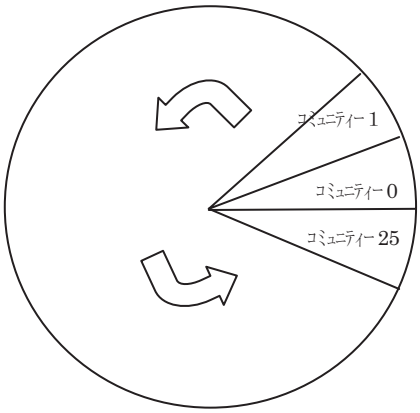
*人間健康学部 健康栄養学科



（図１.）スケールフリー・ネットワーク（ノード数500、リンク数998のネットワーク）の例図

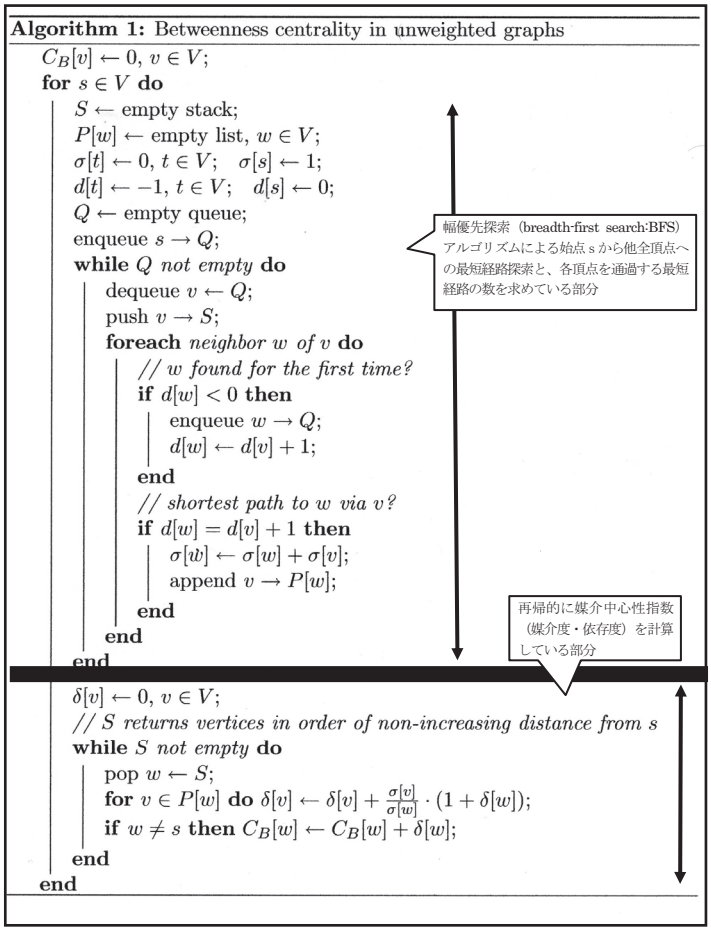


（図２.）スケールフリー・ネットワーク（ノード数500、リンク数998のネットワーク）の例図



（図３.）例図２概説

ノードの重要度を測る指標として、各種中心性があるが、一般的には次数中心性、媒介中心性が考えられる。次数はノードに接続するリンクの数であり、他のノードとつながるリンクの数が多いノードほど重要度が高いと判断するのが１つの考え方である。媒介中心性は、あるノードがそれ以外のネットワーク上のノードペアをつなぐ最短経路に寄与する程度を示す指標である。ノードペア間の情報・影響がネットワークを通して最短経路で伝わると考えた場合に、第三のノードがそれにどれほど寄与しているかを示す。



（図４.）媒介中心性計算アルゴリズム（『A faster algorithm for betweenness centrality』より[5)]）

また、媒介中心性を使うことにより、切断法によるコミュニティ抽出[3]が可能である。ノードをグループ化（コミュニティに分割）して（図２.）のようにコミュニティごとに分割して表示させることができるようになる。

将来的にコミュニティ抽出まで自前のartisocのシミュレーションで実行できる環境を用意することを目指し、今回はまずは媒介中心性を計算するartisocシミュレーションを自前で作成してみた。

２．媒介中心性指数計算方法

媒介中心性指数計算方法については、『ネットワーク科学の道具箱　つながりに隠れた現象をひもとく』[4]、論文『A faster algorithm for betweenness centrality』[5]などを参考にできる。（図４.）は後者の論文に掲載されている疑似プログラム風に表現された媒介中心性指数の計算アルゴリズムである。アルゴリズムの証明も同論文に掲載されている。今回作成したartisocのシミュレーションもこのアルゴリズム通りに作成した。

アルゴリズムは難解なので、ここでは概要だけにする。論文『A faster algorithm for betweenness centrality』[5]に詳細な説明がある。

CB(v) は、グラフ上の任意の頂点vの媒介中心性指数を意味する。（図５.）の式で定義される。

Vは考えているグラフ上の全ての頂点（vertex）の集合を意味する。

$\delta\ st$(v) は、条件s ≠ t ≠ vを満たす始点sと終点tの頂点ペア（s, t）間でsからtへ情報・影響がネットワークを通して最短経路で伝わると考えた場合に、頂点vがそれにどれほど寄与しているかを示す指標である。頂点vの媒介度（依存度）と呼べるものである。

$$C_B(v) = \sum_{s \neq v \neq t \in V} \delta_{st}(v).$$

（図５.）媒介中心性指数の定義（『A faster algorithm for betweenness centrality』より[5]）

頂点vの媒介度（依存度）$\delta\ st$(v) は、（図６.）の式で計算される。

$\sigma\ st$ は、頂点ペア（s, t）間の最短経路の数であり、$\sigma\ st$(v) はそのうち頂点vを通るものの数である。１つの頂点ペア（s, t）で考えた場合の頂点vの媒介中心性指数値（媒介度・依存度）の定義ということになる。

$$\delta_{st}(v) = \frac{\sigma_{st}(v)}{\sigma_{st}}$$

（図６.）（s, t）ペアのみ考えた場合の頂点vの媒介中心性指数（媒介度・依存度）の定義（『A faster algorithm for betweenness centrality』より[5]）

（図６.）の計算をグラフ上で考えられる全ての（s, t）ペアの組み合わせについて計算し総和すると、（図５.）の定義式で示される頂点vの媒介中心性指数が求められる。

（図７.）は、始点sを固定して、s以外の全ての頂点●を終点としたペア（s, ●）の組み合わせで計算する頂点vの媒介中心性指数（媒介度・依存度）の定義式である。

$$\delta_{s\bullet}(v) = \sum_{t \in V} \delta_{st}(v).$$

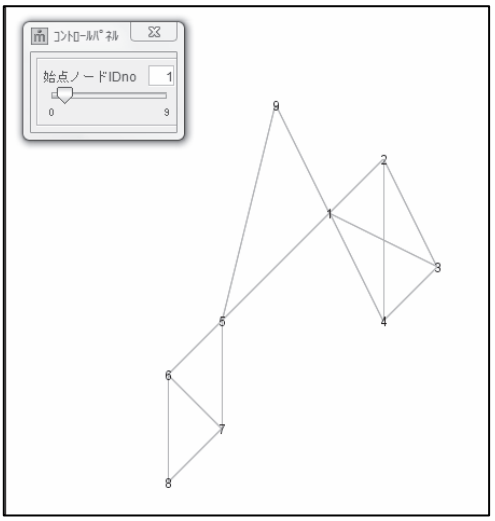
（図７.）（s, ●）ペアで考えた場合の頂点ｖの媒介中心性指数の定義●≠ｓ≠ｖ（『A faster algorithm for betweenness centrality』より[5]）

今回使用した（図４.）のアルゴリズムでは、一番外側のループ（for s ∈ V do）１回の計算処理にて再帰的な計算法により、始点をs、終点をｓ以外の全ての頂点をとする（s, ●）ペアの全ての組み合わせについて、ｓを除いたグラフ上の全頂点の媒介度（依存度）が計算される。

この計算をグラフ上の全ての頂点を始点として繰り返し、各ループごとに計算される各頂点の媒介度（依存度）を各頂点ごとに総和すれば、グラフ上の全頂点の媒介中心性指数が求められることになる。

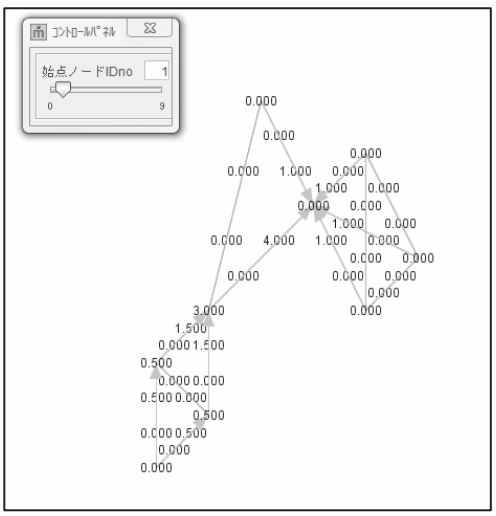
artisocは、シミュレーションの条件（初期条件等）をコントロールパネルで操作・変更可能である。今回のシミュレーションもスライドバーの数値＝０の場合は、媒介中心性指数計算を最後まで完結して実行する動きをするが、０以外の数字を指定して動作させた場合は、スライドバーの数値と一致するIDnoの頂点（ノード）を始点する媒介中心性指数計算のみ実行して終わるようにした。（図４.）アルゴリズムで言えば、（for s ∈ V do）１回分の計算処理のみ実行して終了することになる。

（図８.）のネットワークには、整数型変数IDnoが１～9の一意の値を持つ９個の頂点（ノード）がある。この図は、『Rで学ぶデータサイエンス　８　ネットワーク分析』にて、フリーソフトRにより媒介中心性・コミュニティ抽出計算されている例図[7]）と同じである。左上のスライドバーが１を示しているのでこの条件でシミュレーションを動作させることにより、IDno = １のノードを始点とした全頂点の媒介度（媒介中心性指数）が計算される。



（図８.）ノードIDno表示図

（図９.）は、計算された媒介度を表示する図

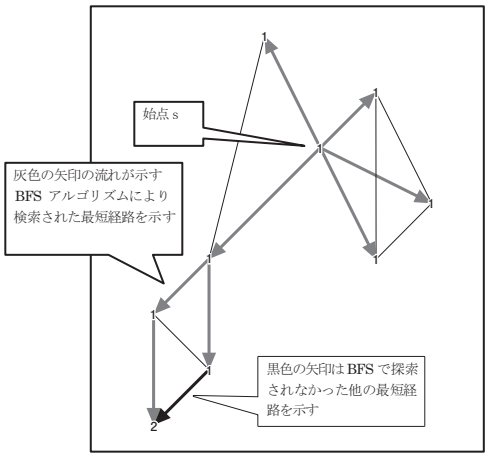


（図９.）IDno = １ノードを始点として計算された媒介度表示図

の例である。各頂点の真上に表示されている数値がその頂点の媒介度である。各辺の上にも２つの数値が表示されている。詳細説明は省略するが、（図４.）のアルゴリズムの計算途中に登場する辺の媒介度である。１つの辺には逆向きの２つの矢印の流れがある。矢印の先に近い位置に表示される数値が、矢印の根本から矢印の方向に情報が流れる場合の辺の媒介度を意味している。

（図４.）における（for s ∈ V do）ループは明確に前半と後半の２つに分かれている。（図４.）の矢印が示すように前半部分では、頂点 s を始点とする他の全頂点への最短経路を探索する。これは幅優先探索（breadth-first search：BFS）アルゴリズムが使われている。ただし BFS では各頂点への１つの最短経路しか探索しない。最短経路は１つだけとは限らないので探索から洩れた最短経路も加える処理部分が追加されている。これにより各頂点を通過する全ての最短経路数が求められるようになっている。

（図10.）は、媒介度（依存度）を計算する計算前半部分が終わった時点の図である。



（図10.）最短ルート数表示図

計算前半で幅優先探索（breadth-first search：BFS）アルゴリズムにより始点 s から他全頂点への最短経路が生成された様子を灰色の矢印の流れで確認することができる。また BFS で探索されなかった他の最短経路が存在する場合には黒色の矢印で表示した。図中の数字は、始点 s から各頂点への最短経路数を示している。左下の頂点へは灰色の矢印１本と黒色の矢印１本の計２本が来ているので最短経路数２になっていることが理解できる。

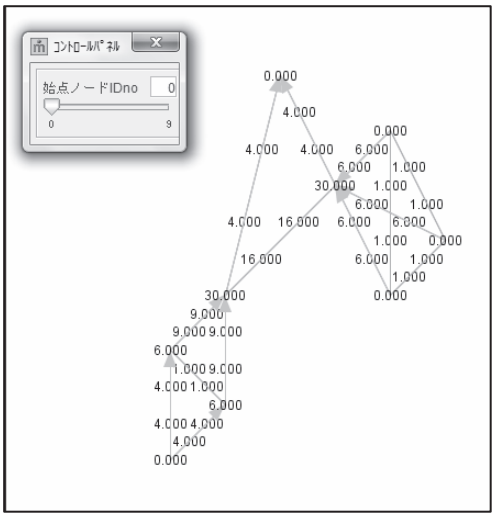
この最短ルート数表示図を見ることにより、媒介度（依存度）計算前半の流れを追うことができる。

アルゴリズム前半の計算処理は始点 s から始まり、灰色の矢印の流れにしたがって処理が進行する。

（図４.）にて、大文字の S で示されるスタックが使われている。データを後入れ先出し（LIFO：Last In First Out; FILO： First In Last Out）の構造で保持するデータ構造である。より新たに格納したものがより早く取り出せる。アルゴリズム前半の最短経路探索部分で始点 s から始まり次々に通過していく頂点を到達順にスタック S に格納している。（図10.）の灰色の矢印の流れがスタック S に保持されているとも言える。

アルゴリズム後半の各頂点の媒介度（依存度）計算の流れは、ちょうど前半の逆になる。後半の計算の流れは、（図９.）の媒介度表示図で確認できる。スタック S から取り出された頂点の媒介度（依存度）が計算される。（図９.）図の灰色の矢印の流れで計算が進行している。最後に始点 s がスタック S から取り出されるとスタックが空になり計算が終了する。

artisoc は、グラフ、ノード・リンク表示図等を同時並行して複数表示可能なので、ノード IDno 表示図（図８.）、最短ルート数表示図（図10.）、媒介度表示図（図９.）などの図を同時



（図11.）媒介中心性指数表示図

に表示させることができる。アルゴリズム前半の動きは、最短ルート数表示図（図10.）で検証でき、アルゴリズム後半の動きは媒介度表示図（図９.）で検証することができる。

（図11.）は、スライドバーの数値＝０でシミュレーションを実行した場合の媒介中心性指数表示図である。最終的に計算された各頂点の媒介中心性指数が表示されている。

２．Universe 変数（グローバルに使う変数）

今回のシミュレーションでは、キューとスタックのデータ構造が必要であった。キューは、データを先入れ先出し（FIFO：First In First Out）のリスト構造で保持するデータ構造で、スタックは、データを後入れ先出し（LIFO：Last In First Out; FILO：First In Last Out）の構造で保持するデータ構造である。

両データ構造共に、データを格納する実体はartisoc のエージェント集合型変数が使える。AddAgt ルーチンを使いエージェント集合型変数にノードを追加することができる。エージェント集合型変数内では、ノードは追加された順番で並んで保持される。先頭データ、最後のデータの位置を示す整数型変数を別に用意して管理すればキュー、スタックのデータ構造が実現できる。

GetAgt ルーチンを使うと、エージェント集合型変数に格納されたノードを格納位置指定で取り出すことができる。

以上、キュー、スタックを実現するための変数などを Universe 変数として用意した。Universe 変数はエージェントに属さない変数で、一般的なプログラミングにおけるグローバル変数として使えるものである。

変数名	型	用途
キュー	エージェント集合型	node エージェントを格納するキューの実体
キュー先頭	整数型	キューに格納されているエージェントの先頭位置を示す
キュー最後	整数型	キューに格納されているエージェントの最後の位置を示す
スタック	エージェント集合型	node エージェントを格納するスタックの実体
スタック先頭	整数型	スタックに格納されているエージェントの先頭位置を示す
スタック最後	整数型	スタックに格納されているエージェントの最後の位置を示す

動作モード	整数型	シミュレーションのステップを管理するための変数。特に深い意味は無い。今回のシミュレーションではエージェントのスクリプトが無いため１ステップの計算処理で終了してしまう。その場合マップ出力表示が上手く表示されないケースがある。そのためわざと２、３ステップ空のステップを入れてからシミュレーションを終了させた。 実際には、最初の１ステップで媒介中心性計算は終わってしまう。
全ノード	エージェント集合型	全 node エージェントを格納する変数
startNode	整数型	コントロールパネルのスライドバーと連動する変数

３．媒介中心性指数計算シミュレーションのエージェント

今回作成したシミュレーションも2008年度研究紀要の電子回路シミュレーション[6]と同様に、頂点（ノード）に関する情報は node エージェントに、ノードとノードをつなぐ情報は link エージェントに保持させた。エクセルシート上で、node エージェント･link エージェントデータを作成し、artisoc に読み込ませた。

（１）node エージェント

エージェントを定義すると、自動的に ID、X、Y、Direction、icon などの変数が用意される。特にシミュレーションでは使用していない変数は、（未使用）と記述する。

変数値はエクセルシート上で作成し、CSV テキストファイル化したものを初期値として読み込ませている。

変数名	型	用途
ID	整数型	aritisocエンジンが自動的に各エージェントに設定する値(未使用)
X	実数型	エージェントの X 座標値
Y	実数型	エージェントの Y 座標値
Layer	整数型	ノードを表示するレイアー（未使用）
Direction	実数型	エージェントの向いている方向（未使用）
state	整数型	node エージェントの状態を表す。（未使用）
IDNo	整数型	プログラム上ノードを識別するための数値
link	エージェント集合型	node エージェントが連結する連結先 node エージェントを格納する 格納された node エージェントに向け自動的に青色の線が描かれるように設定した
Linkagents	エージェント集合型	自分の連結情報を持つ link エージェントを格納する変数
color	整数型	ノードの表示色（今回未使用）
icon	文字列型	表示画像名称（この変数に画像名称が格納されると artisoc が自動的に node エージェント位置に画像表示を行う）（未使用）
前ノード	エージェント集合型	始点とした頂点から最短経路をたどって自分に到達する１つ手前の node エージェントを格納する。
最短ルート数	整数型	始点とした頂点から自分に到る最短ルートの数
距離	整数型	始点とした頂点からの距離（最短距離）

依存度	実数型	ある１つの頂点（ノード）を始点として計算された頂点の媒介中心性指数（媒介度・依存度）
媒介中心性指数	実数型	頂点（ノード）の媒介中心性指数
最短 tree	エージェント集合型	始点とした頂点から始まり BFS 探索（幅優先探索）で検索された自分を通過する最短経路上で自分がつながる次の node エージェントを格納する変数 格納された node エージェントへ向かい自動的にピンク色の線が描かれるように設定した。
別の最短 tree	エージェント集合型	始点とした頂点から始まり BFS 探索（幅優先探索）で検索された自分を通過する最短経路ではない、別の最短経路上で自分がつながる次の node エージェントを格納する変数 格納された node エージェントへ向かい自動的にライム色の線が描かれるように設定した。
枝の数	整数型	頂点（ノード）のつながっている他の頂点（ノード）の数

（2）link エージェント

node エージェントの連結情報を保持するための使われるエージェント。マップ出力において、座標（X，Y）位置に辺の媒介中心性指数を表示するように設定した。

変数値はエクセルシート上で作成し、CSV テキストファイル化したものを初期値として読み込ませている。

変数名	型	用途
ID	整数型	aritisoc エンジンが自動的に各エージェント識別用に（未使用）
X	実数型	エージェントの X 座標値。今回は辺の媒介中心性指数（媒介度・依存度）を表示する位置（X 座標値）として使用。
Y	実数型	エージェントの Y 座標値。今回は辺の媒介中心性指数（媒介度・依存度）を表示する位置（Y 座標値）として使用。
Layer	整数型	ノードを表示するレイアー（未使用）
Direction	実数型	エージェントの向いている方向（未使用）
FromNode	整数型	連結元 node エージェントの IDNo を保持する。
ToNode	整数型	連結先 node エージェントの IDNo を保持する。
cost	整数型	連結先 node エージェントへパケットを転送するコスト値（今回未使用）
媒介中心性指数	実数型	辺の媒介中心性指数（媒介度・依存度）。FromNode から ToNode へ向かう辺が情報を媒介する度合いを示す指数。再帰的に媒介中心性指数を計算する過程で計算される。

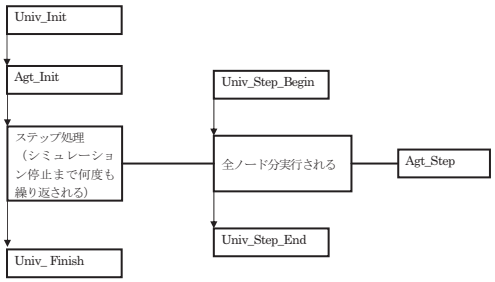
３．シミュレーションのスクリプト

３．１　シミュレーションスクリプトの構成

artisoc のスクリプトは、独立したセクションに分かれている。

artisoc におけるシミュレーションを実行すると、まず Univ_Init セクションに書かれたスクリプトが実行され、次にエージェントの初期化処理 Agt_Init が実行され、その後ステップ処理が繰り返される。

各ステップ処理においては、まず Univ_Step_Begin セクションのスクリプトが実行された後、各ノードの振舞いが記述された Agt_



（図12.）スクリプトの構成外略図

Step セクションのスクリプトが全ノード分実行される。そしてステップ処理の最後に Univ_Step_End セクションのスクリプトが実行される。

シミュレーションの最後には Univ_Finish セクションのスクリプトが実行される。

（図12.）がスクリプトの構成概略図である。上から下に向かう順序で実行されるが、四角の右にある処理部分は、繰り返し実行される部分という意味で図示してある。

今回のシミュレーションでは、エージェントが実行するスクリプトは無い。従って Agt_Init、Agt_Step 部分にスクリプトは無い。

3．2　Univ_Init セクション

全ての link エージェントを読み込み、node エージェントの連結関係データ（FromNode、ToNode ペア）を取り出し、連結元 node エージェントのエージェント集合型変数 link に連結先 node エージェントを格納する。この処理は2008年度研究紀要の電子回路シミュレーション[1)]と同じである。また、マップ上に辺の媒介中心性指数（媒介度・依存度）を表示するための位置（X，Y）座標を計算し link エージェントの X，Y 変数に格納する。

さらに node エージェントのエージェント集合型変数 Linkagents に自分の連結情報を持つ link エージェントを格納する。

そして、全てのノードについて媒介中心性指数＝０、枝の数＝（エージェント集合型変数 link に格納されている node エージェント数）の初期化を行う。

```
1   include "SearchAgt.inc"
2   include "Enqueue.inc"
3   include "Dequeue.inc"
4   include "PushToStack.inc"
5   include "PopFromStack.inc"
6   include "CalcDependency.inc"
7   include "ExportNodeData.inc"
8   include "ExportLinkData.inc"
9
10  Univ_Init{
11  Dim link As Agtset
12  Dim one As Agt
13  Dim FromAgt As Agt
14  Dim ToAgt As Agt
15
16  MakeAgtset（link, Universe.Nodefield.link）  //  link エージェントのみ集める
17  For each one in link
18      if one.FromNode <> 0 then
19          SearchAgt（one.FromNode）  //  リンク元ノードを検索する
20          if CountAgtset（Universe.nodeAgtset）> 0 then
21              FromAgt = GetAgt（Universe.nodeAgtset, 0）
22
23  //  リンク元ノードの LinkAgents 変数に link エージェントを追加する
24              AddAgt（FromAgt.LinkAgents, one）
25
26              SearchAgt（one.ToNode）  //  リンク先ノードを検索する
27              if CountAgtset（Universe.nodeAgtset）> 0 then
28                  ToAgt = GetAgt（Universe.nodeAgtset, 0）
29                  AddAgt（FromAgt.link, ToAgt）  //  リンク元ノードの link 変数にリンク先ノードを追加する
```

```
30
31              //   link エージェントの媒介中心性指数の表示座標の計算
32              one.X = FromAgt.X + ( ToAgt.X - FromAgt.X ) / 3
33              one.Y = FromAgt.Y + ( ToAgt.Y - FromAgt.Y ) / 3
34            end if
35        end if
36    end if
37  next one
38
39
40  MakeAgtset（Universe. 全ノード , Universe.Nodefield.node） //  node エージェントのみ集める
41  For each one in Universe. 全ノード
42     one. 枝の数 = CountAgtset（one.link）
43     one. 媒介中心性指数 = 0      //  各ノードの媒介中心性指数を初期化
44  next one
45
46  universe. 動作モード = 1                //  シミュレーション動作モード （1 ：BFS・媒介中心性指数計算）
47  }
```

3．3　Univ_Step_Begin セクション

コントロールパネルのスライドバーと連動するグローバル変数 startNode の値によって動作を変える。startNode = 0 の場合は、全ノードを始点にして、各ノードの媒介中心性指数計算を行う。startNode が 0 以外の場合は、IDno = startNode である node エージェントを始点とする各ノードの媒介中心性指数計算（媒介度・依存度）だけを実行して終わる。

```
1   Univ_Step_Begin{
2
3   Dim one As Agt
4   Dim startNode As Agt
5
6   if universe. 動作モード = = 1 then
7       //   Universe.startNode 値を IDNo に持つノードが始点の媒介中心性指数計算を行う
8      if Universe.startNode <> 0 then
8
10        SearchAgt（Universe.startNode）
11
12        if CountAgtset（Universe.nodeAgtset） > 0 then
13           startNode = GetAgt（Universe.nodeAgtset, 0 ）
14           CalcDependency（startNode）       //   媒介中心性指数計算
15        end if
16
17     else
18
19        //   全ノードを始点にして、各ノードの媒介中心性指数計算を行う
20        For each one in Universe. 全ノード
21           CalcDependency（one）       //   媒介中心性指数計算
22        next one
23
24     end if
25
26     universe. 動作モード = 2      //  シミュレーション動作モードアップ
27
28  else
29     universe. 動作モード = universe. 動作モード + 1      //  シミュレーション動作モードアップ
30  end if
31
32  }
```

3．4　Univ_Step_End セクション

媒介中心性指数計算は 1 ステップのシミュレーションで終わっているが、その後空のステップを 4 ステップほど実行してからステップ処理を終了させる。

```
Univ_Step_End{

if universe.動作モード = = 4  then    //  シミュレーション動作モード（4：停止）
    ExitSimulationMsgLn（"Simulation Completed"）
end if

}
```

3.5　Univ_Step_Finish セクション

artisoc ではシミュレーションの実行が終わると、エージェント変数内の情報は元の状態（初期状態）に戻ってしまうので、node エージェント（頂点）の媒介中心性指数、link エージェント（辺）の媒介中心性指数等の計算結果を csv 形式のテキストファイルとして書き出してからシミュレーションを終了する。

```
Univ_Finish{
 ExportNodeData（）
 ExportLinkData（）
}
```

3.6　サブルーチン

（1）サブルーチン SearchAgt

同名の SearchAgt.txt テキストファイルとして保存して使用。

インクルード宣言（include "SearchAgt.inc"）により artisoc により自動的に読み込まれて使われる。

IDNo をキーに全 node エージェントから目的の node エージェントを探す。

```
//    IDNo を持つエージェントを全エージェントから検索する
Sub SearchAgt（IDNo As Integer）{

Dim allAgent As Agtset
Dim dest As Agtset
dim one As Agt

ClearAgtset（Universe.nodeAgtset）

MakeAgtset（allAgent, Universe.ElectricCircuit.node）

for each one in allAgent
    if one.IDNo = = IDNo then
        AddAgt（Universe.nodeAgtset,  one）
        Break     // もう調べる必要なし（エージェントが見つかった）
    end if
next one
}
```

（2）サブルーチン Enqueue

同名の Enqueue.txt テキストファイルとして保存して使用。

インクルード宣言（include "Enqueue.inc"）により artisoc により自動的に読み込まれて使われる。

エージェントをキューに格納する。

```
//    node をキューに格納する
Sub Enqueue（one as Agt）{

    AddAgt（Universe. キュー，one）
    Universe. キュー最後= Universe. キュー最後 +  1

}
```

（3）サブルーチン Dequeue

同名の Dequeue.txt テキストファイルとして保存して使用。

インクルード宣言（include "Dequeue.inc"）により artisoc により自動的に読み込まれて使われる。

エージェントをキューから取り出す。

```
1  //    node をキューから取り出す
2  Function Dequeue（）As Integer{
3
4  Dim 結果 As Integer
5  Dim one As Agt
6
7  結果 = 0
8
9  if Universe. キュー先頭 > = Universe. キュー最後 then
10     結果 = 1          //   取り出しデータなし
11 else
12     one = GetAgt（Universe. キュー , Universe. キュー先頭）
13     ClearAgtset（Universe.nodeAgtset）
14     AddAgt（Universe.nodeAgtset, one）
15     Universe. キュー先頭 = Universe. キュー先頭 + 1
16 end if
17
18 return（結果）
19 }
```

（4）サブルーチン PushToStack

同名の PushToStack.txt テキストファイルとして保存して使用。

インクルード宣言（include "PushToStack.inc"）により artisoc により自動的に読み込まれて使われる。

エージェントをスタックに格納する。

```
1  //    node をスタックに格納する
2  Sub PushToStack（one as Agt）{
3
4      Dim TreeFrom As Agtset
5
6      AddAgt（Universe. スタック , one）
7      Universe. スタック最後 = Universe. スタック最後 + 1
8
9  }
```

（5）サブルーチン PopFromStack

同名の PopFromStack.txt テキストファイルとして保存して使用。

インクルード宣言（include "PopFromStack.inc"）により artisoc により自動的に読み込まれて使われる。

エージェントをスタックから取り出す。

```
1  //    node をスタックから取り出す
2  Function PopFromStack（）As Integer{
3
4  Dim 結果 As Integer
5  Dim one As Agt
6
7  結果 = 0
8
9  if Universe. スタック最後 < 0 then
10     結果 = 1          //   取り出しデータなし
11 else
12     one = GetAgt（Universe. スタック , Universe. スタック最後）
13     ClearAgtset（Universe.nodeAgtset）
14     AddAgt（Universe.nodeAgtset, one）
15     Universe. スタック最後 = Universe. スタック最後 − 1
```

```
16  end if
17
18  return（結果）
19  }
```

（6）サブルーチン CalcDependency

同名の CalcDependency.txt テキストファイルとして保存して使用。

インクルード宣言（include "CalcDependency.inc"）により artisoc により自動的に読み込まれて使われる。

1つの頂点（ノード）を始点にする他の全ノードの情報媒介度（媒介中心性指数）を計算する。

```
1   //　ある1つのノードを始点にする各ノードの媒介度を計算する
2   Sub CalcDependency（StartNode as Agt）{
3
4   Dim node As Agt
5   Dim one As Agt
6   Dim FromAgt As Agt
7   Dim ToAgt As Agt
8   Dim 結果 As Integer
9   Dim 依存度加算 As Double
10  Dim link エージェント As Agt
11
12
13  ////////////////////////////////////////////////////
14  //　シミュレーション前半　（BFS）
15
16  //　スタックの初期化
17  ClearAgtset（Universe. スタック）
18  Universe. スタック最後＝－1
19
20  //　キューの初期化
21  ClearAgtset（Universe. キュー）
22  Universe. キュー先頭＝0
23  Universe. キュー最後＝0
24
25  For each one in Universe. 全ノード
26      ClearAgtset（one. 前ノード）　// 前ノードリストのクリア
27      one. 距離＝－1　　　　//　始点からの距離を－1にする
28      one. 最短ルート数＝0　　//　始点からの自分を通過する最短ルート数を0にする
29      one. 依存度＝0　　　　//　始点 s からの自分への依存度をクリア
30  next one
31
32  //　始点ノードの処理
33  StartNode. 最短ルート数＝1　　//　　始点ノードの最短ルート数を1にする
34  StartNode. 距離＝0　　　//　　始点ノードの距離を0にする
35  Enqueue（StartNode）　　//　始点ノードをキューに格納する
36
37  Do While（1）
38
39      結果＝ Dequeue（）　　//　処理すべきノードをキューから取り出す
40      if 結果 ＝＝1 then　　//　処理すべきノードがキューになかった場合はループから抜ける
41          Break
42      end if
43
44      if CountAgtset（Universe.nodeAgtset）＞ 0 then
45          node = GetAgt（Universe.nodeAgtset, 0）
46          PushToStack（node）　　//　ノードをスタックに格納する
47      else
48          Break
49      end if
50
51      For each one in node.link
```

```
52
53          //   ω found for the first time ?
54          if one. 距離 < 0 then
55              Enqueue（one）          //   リンクする次ノードをキューに格納する
56              one. 距離 = node. 距離 + 1
57              AddAgt（node. 最短 tree, one）
58                                      //   ノードの次のノードを最短経路上のノードとしてつなぐ
59
60          end if
61
62          //    shortest path to ω via v ?
63          if one. 距離 = =（node. 距離 + 1）then
64              one. 最短ルート数 = one. 最短ルート数 + node. 最短ルート数
65              AddAgt（node. 別の最短 tree, one）     //   ノードの次のノードを別の最短経路上のノードとしてつなぐ
66
67             AddAgt（one. 前ノード , node）    //   次ノードの前ノードとして追加
68          end if
69
70      next one
71
72  Loop
73
74
75  ////////////////////////////////////////////////////
76  //   シミュレーション動作後半（媒介中心性計算）
77
78  Do While（1）
79
80      結果 = PopFromStack（）      //   処理すべきノードをスタックから取り出す
81
82      if 結果 = = 1 then          //   処理すべきノードがスタックになかった場合はループから抜ける
83          Break
84      end if
85
86       if CountAgtset（Universe.nodeAgtset）> 0 then
87           node = GetAgt（Universe.nodeAgtset, 0）
88       else
89           Break
90       end if
91
92       For each one in node. 前ノード
93           依存度加算 =   （one. 最短ルート数 / node. 最短ルート数）*（1 + node. 依存度）
94
95          //   リンクエージェントの媒介中心性指数依存度加算を加える
96          For each link エージェント in one.LinkAgents
97              if link エージェント .ToNode = = node.IDno then
98                  link エージェント . 媒介中心性指数 = link エージェント . 媒介中心性指数 + 依存度加算
99              end if
100         next link エージェント
101
102     next one
103
104     if node.IDNo <> startNode.IDNo then
105         node. 媒介中心性指数 = node. 媒介中心性指数 + node. 依存度
106     end if
107
108 Loop
109
110
111 }
```

（7）サブルーチン ExportNodeData

同名の ExportNodeData.txt テキストファイルとして保存して使用。

インクルード宣言（include "ExportNodeData.inc"）により artisoc により自動的に読み込まれて使われる。

node エージェントデータを csv 形式で "input.txt" テキストファイルに書き出す。

```
1  //   node 情報を   input.txt ファイルに書き出す
2
3  Sub ExportNodeData（）{
4
5  Dim node As Agt
6  Dim 接続している node As Agt
7  Dim エージェント As Agt
8
9  OpenFileCSV（"input.txt",1,3）
10
11 For each node in Universe. 全ノード
12
13    // 1 つの node 情報を 1 行の csv 形式で書き出す
14    WriteFileCSV（1,node.ID,False）              // ID
15    WriteFileCSV（1,node.X,False）               // X
16    WriteFileCSV（1,node.Y,False）               // Y
17    WriteFileCSV（1,0,False）                  // Layer
18    WriteFileCSV（1,0,False）                  // Direction
19    WriteFileCSV（1,0,False）                  // state
20    WriteFileCSV（1,node.IDNo,False）            // IDNo
21    WriteFileCSV（1,0,False）                  // cost
22    WriteFileCSV（1,0,False）                  // color
23    WriteFileCSV（1," ",False）                // icon
24    WriteFileCSV（1,node. 最短ルート数 ,False）    // 最短ルート数
25    WriteFileCSV（1,node. 距離 ,False）          // 距離
26    WriteFileCSV（1,node. 依存度 ,False）         // 依存度
27    WriteFileCSV（1,node. 媒介中心性指数 ,False）  // 媒介中心性指数
28    WriteFileCSV（1,node. 枝の数 ,True）          // 枝の数
29
30 next node
31
32 CloseFileCSV（1）
33
34 }
```

（8）サブルーチン ExportLinkData

同名の ExportLinkData.txt テキストファイルとして保存して使用。

インクルード宣言（include "ExportLinkData.inc"）により artisoc により自動的に読み込まれて使われる。

link エージェントデータを csv 形式で "output.txt" テキストファイルに書き出す。

```
1  //   link 情報（媒介中心性指数）を   output.txt ファイルに書き出す
2
3  Sub ExportLinkData（）{
4
5  Dim 全 link As Agtset
6  Dim link As Agt
7
8  OpenFileCSV（"output.txt",1,3）
9
10 MakeAgtset（全 link, Universe.Nodefield.link）  //   link エージェントのみ集める
11
12 For each link in 全 link
13
14    // 1 つの link 情報を 1 行の csv 形式で書き出す
15    WriteFileCSV（1,link.FromNode,False）        // リンク元ノード IDNo
16    WriteFileCSV（1,link.ToNode,False）          // リンク先ノード IDNo
17    WriteFileCSV（1,link. 媒介中心性指数 ,True）   // 媒介中心性指数
18
```

```
19  next link
20
21  CloseFileCSV（1）
22
23  }
```

4．結び

今回作成したシミュレーションも2008年度の電子回路シミュレーション[6]と同様に、ルールの総計が200行までという制限のある artisoc textbook で作成可能であった。エクセルシート上で、node エージェント・link エージェントデータを作成し、artisoc に読み込ませ、ノード数に合わせてコントロールパネルに変更を加えれば、いろいろなネットワークで媒介中心性指数計算を行うことができる。

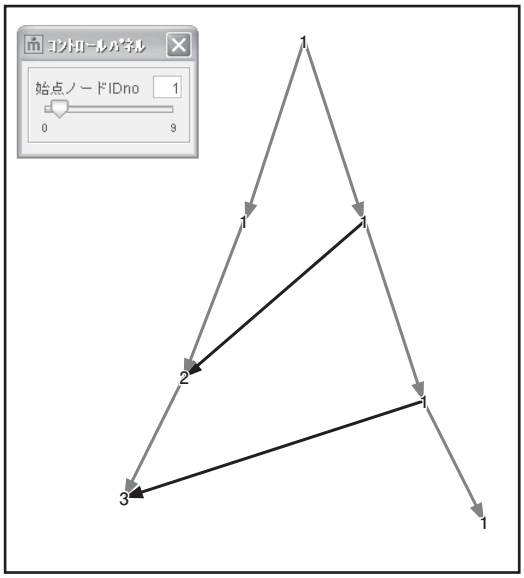
ただし、node エージェントデータをエクセル等で生成する場合は、各ノードの IDno は１以上の整数でなければならない。IDno ＝ 0 は、コントロールパネルのスライダーが対応しない作りになっている。

今回のシミュレーションも、学生各自が授業で使用するパソコンで artisoc が使える環境になっていれば、シミュレーションを自分で動かして、演習することが可能である。コントロールパネルのスライドバー値＝ 0 で動かすと、全頂点の完全な媒介中心性指数を計算させることができる。

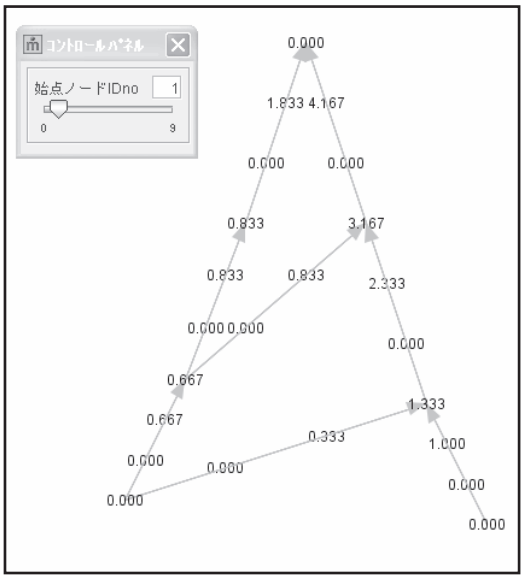
媒介中心性指数の計算アルゴリズムを説明する／学ぶ目的で使うならば、（図８.）のノード IDno 表示図を参考に、始点とする頂点を選択し、コントロールパネルのスライドバーをその頂点 IDno に合わせてからシミュレーションを動作させることにより、１つの頂点を始点とする最短経路で情報が流れるケースでの各頂点の依存度（情報媒介度）を計算させる。そして、アルゴリズム前半の最短ルートと各頂点の最短ルート数を求める部分を（図10.）の最短ルート数表示図によって検証することができる。アルゴリズム後半の媒介度を再帰的に計算する部分は、（図11.）の１つのノードを始点として計算された媒介度表示図で処理の流れを検証することが可能である。

参考にした『ネットワーク科学の道具箱　つながりに隠れた現象をひもとく』にも１つの頂点（一番上の点）を始点にした各頂点の依存度を計算した例[9]（図15.）がある。今回作成したシミュレーションでも同じ計算結果になることが確認できた（図14.）。（図13.）は最短経路と各頂点を通過する最短経路数の図である。

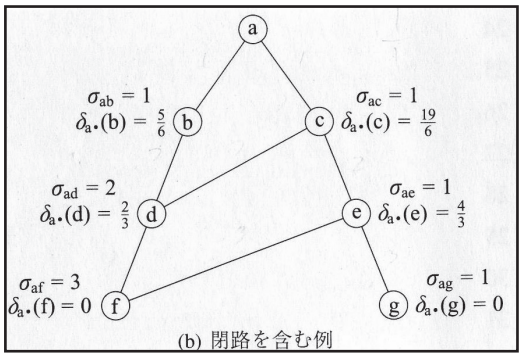
次に黒幕と手下関係を表現するようなネットワーク（図16.）の場合にも今回作成したシミュレーションで媒介中心性指数を計算してみた。（図16.）のノード IDno 表示図で言えば、中心で五角形の頂点になっているノード番号１、２、３、４、５のノードがいわゆる黒幕と呼ぶよう



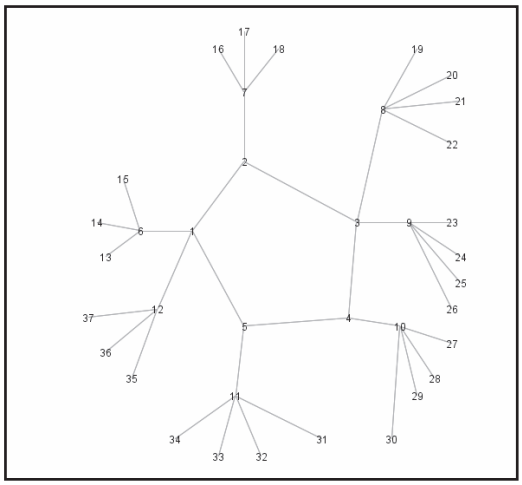
（図13.）最短ルート数表示図



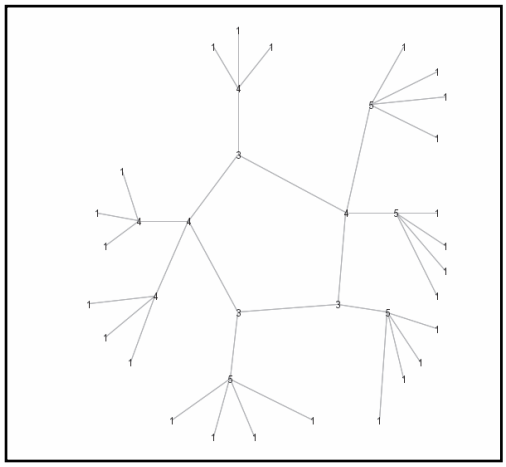
（図14.）媒介度表示図



（図15.）『ネットワーク科学の道具箱　つながりに隠れた現象をひもとく』、P.50の例図[9)]



（図16.）ノードIDno表示図



（図17.）次数（枝の数）表示図

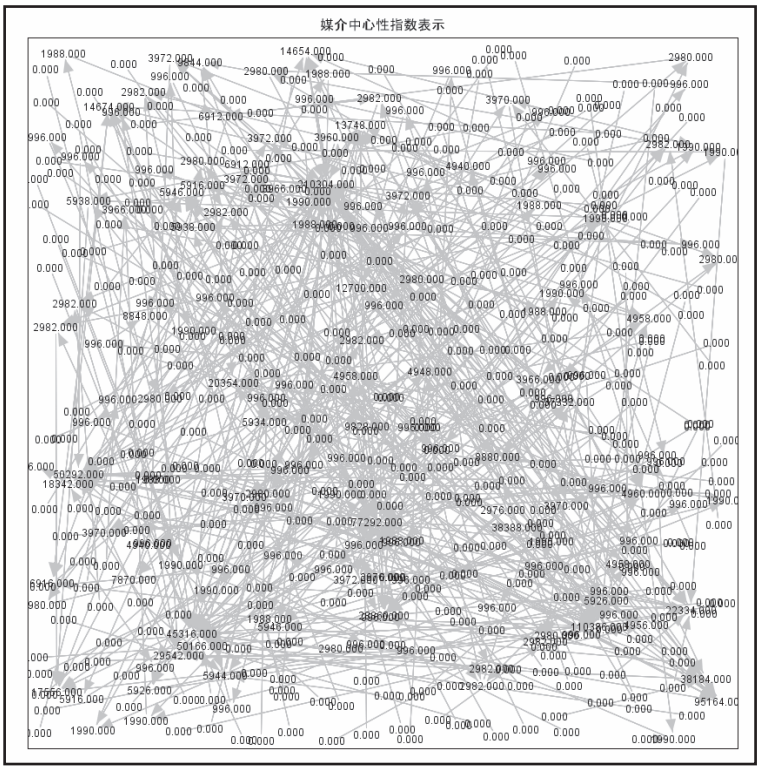
なノードに相当するであろう。

（図17.）図は、ノード（頂点）の次数（枝の数）を表示している。黒幕は必ずしも次数は高くない。黒幕の手下に相当するノード番号６、７、８、９、10、11，12のノードのほうが次数が高いことがありうる。黒幕の手下が多くの実働部隊に相当するノードを持っていれば次数は高くなる。ノードの次数で考えると、黒幕よりもその手下のほうが重要なノードのように見える。

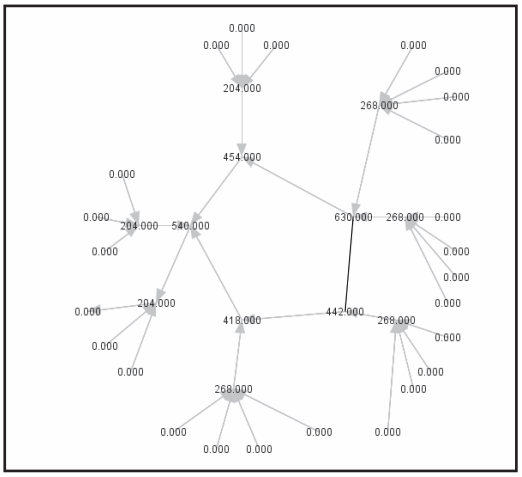
次に、同じネットワークで媒介中心性指数を計算した図（図18.）で見ると、黒幕に相当するノードの媒介中心性指数が断然大きいことが分かる。媒介中心性で考えれば、手下のノードより黒幕に相当するノードのほうが重要度が高い。

このような簡単なネットワーク図を作って、媒介中心性指数と次数を計算して比較対照し説明することも今回のartisocによるシミュレーションで簡単にできるようになった。

最後に、このシミュレーションを使い人工的に生成させたノード数500個のスケールフ



（図19.）スケールフリー・ネットワーク（ノード数500、リンク数998のネットワーク）の媒介中心性指数表示図



（図18.）媒介中心性指数表示図

リー・ネットワークで媒介中心性指数を計算した結果図（図19.）を掲載する。普段使用している Windows Vista パソコンでは、計算に 3 分程度かかっている。

今回、媒介中心性指数を計算する自前の artisoc シミュレーションが利用可能になった。次は、媒介中心性指数を使いコミュニティ抽出を行う自前の artisoc のシミュレーションを作成してみようと考えている。

［参考文献・出典］

1）山影進（2007）人工社会構築指南 artisoc によるマルチエージェント・シミュレーション入門、有限会社書籍工房早山、東京

2）http://mas.kke.co.jp/index.php、MAS コミュニティートップホームページ

3）コミュニティ抽出とは、『相対的により密なリンクで互いに結合し合っている部分集団を塊として、複数取り出す発見的な操作を意

味する。』林幸雄・大久保潤・藤原義久・上林憲行・小野直亮・湯田聰夫・相馬亘・佐藤一憲著（2007）ネットワーク科学の道具箱　つながりに隠れた現象をひもとく、p50-56、近代科学社、東京
4）林幸雄・大久保潤・藤原義久・上林憲行・小野直亮・湯田聰夫・相馬亘・佐藤一憲著（2007）ネットワーク科学の道具箱　つながりに隠れた現象をひもとく、p46-50、近代科学社、東京
5）U. Brandes. A faster algorithm for betweenness centrality. Journal of Mathematical Sociology, Vol. 25, No. 2. P. 163-177,（2001）, http://ella.slis.indiana.edu/~katy/L579/brandes.pdf.
6）末木俊之（2008）artisoc で作成する初等情報教育用電子回路シミュレーション、駒沢女子大学研究紀要、p. 87-102
7）鈴木努著（2009）R で学ぶデータサイエンス　8　ネットワーク分析、p81-86、共立出版株式会社、東京
8）林幸雄・大久保潤・藤原義久・上林憲行・小野直亮・湯田聰夫・相馬亘・佐藤一憲著（2007）ネットワーク科学の道具箱　つながりに隠れた現象をひもとく、p46-50、近代科学社、東京
9）林幸雄・大久保潤・藤原義久・上林憲行・小野直亮・湯田聰夫・相馬亘・佐藤一憲著（2007）ネットワーク科学の道具箱　つながりに隠れた現象をひもとく、p50、近代科学社、東京