# 大規模ネットワーク解析・可視化プログラム

# Pajek

北陸先端科学技術大学院大学
知識科学研究科

佐藤　恵介

2006 年 11 月 7 日

# 目次

表現の注意点

*拡張子の*：*は任意の文字列を表す。サブサブサブメニューの先頭は*で表現されているので違いに注意する。

（用語：解説）：用語に対する定義を解説で説明する。

・：メインメニュー

＞：サブメニュー

＜：サブサブメニュー

*：サブサブサブメニュー

１や２など：サブサブサブサブメニュー

# 1.ネットワークに関する諸概念

ネットワークとは、頂点（ノード）の部分集合とラインの部分集合の 2 つの集合により成り立っている。頂点の部分集合をユニット（Unit)、ラインの部分集合をユニット間の関係（Relations）という。それらは、グラフという。

ラインは、有向と無向に分けられる。頂点やラインの追加的な情報は属性として付属している。（例：名前、ラベル、タイプ、値など）[11]

ネットワークは以下のように表される。

$N=(V,L,P,W)$

グラフ $g=(V,L)$、$V$ は頂点の集合、$A$ は有向線の集合、$E$ は無向線の集合、$L$ はラインの集合で $L=E\cup A$。$n=card(V)$（card は要素）$m=card(L)$。

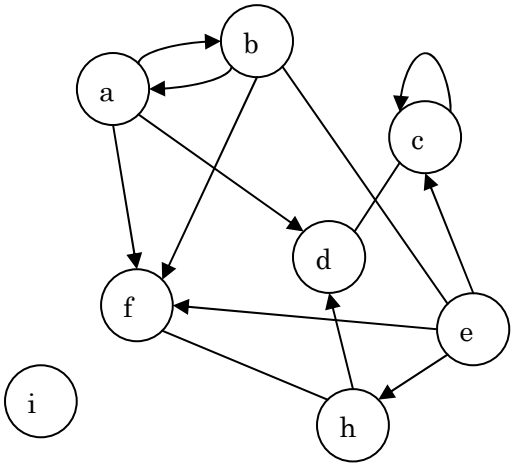
$P$ 頂点の値関数/ 属性：$p:V\rightarrow$ A（有向線集合 $A$ とは違う）

$W$ ラインの値関数/ 重さ：$w:L\rightarrow B$

グラフ(Graph)について

Actor（アクター）：頂点、ノード

Relation（関係）：ライン、無向線、有向線、リンク、タイ




図：グラフ

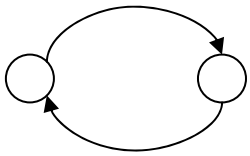
arc（a,d）に注目すると、a は initial vertex,d は terminal vertex という。

edge（c:d）に注目すると、c と d は end vertex という。

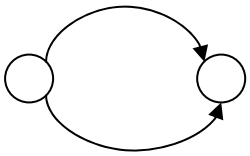
i のように、どの頂点ともつながっていない頂点のことを isolated vertex という。

c の頂点にある自分から出発して自分に帰るラインのことを Loop という。

以下のようなものもある。



Opposite arcs



Parallel arcs

2-mode ネットワーク：Biparite（valued）graphs のこと。詳細は以下。
Temporal networks：動的ネットワーク。時間により変化
P-Graphs：系図
Multiple or multi-relational network：同じ頂点集合のネットワーク

# 2.Pajekの概要

## 2.1Pajekとは

本書は、windows 用ネットワーク分析ソフト UCINET に組み込まれている大規模ネットワーク可視化ツール Pajek のマニュアルである。

UCINET：UCINET は社会ネットワークとその他の類似データ解析を目的とするメニュー方式プログラムです。プログラムには、中立性と結合性測度、サブグループの検出と位置付けメソッド、確率論モデル（P1)、類似・非類似の測度、マトリックス相関（QAP）と多重マトリックス回帰（MRQAP）によるネットワーク仮説検定などの機能が含まれます。
これらのネットワーク解析機能に加えて、UCINET は、多次元スケーリング、クラスター分析、一致分析、および回帰分析などの多変数解析機能を備えています。さらに、データ変換と管理用のツール（線グラフの作成、グラフ転換、ノードと結線による事象マトリックス、値付き、または分割グラフからの多重グラフ作成、グラフのプール化、相関によるセミグループ）を備えています。これに加え、UCINET は完備したマトリックス代数演算言語を持っています。
[1] NetScience UCINET6 https://www.netscience.ne.jp/software/ucinet/

Pajek のネットワークデータの構造は*.net 形式に従う。（*.net は、アスキーコードで書かれたファイル）*.net ファイルを読み込むことにより、pajek 内にネットワークが構成される。pajek を使うには*.net ファイルを作る事が始めである。*.net ファイルの作成は、テキストエディタから手書きで作ることも出来る。よって、生成されたプログラムに依存す

ることなく、*.net ファイルを作れば Pajek で解析できるという意味でどのようなプログラムによって生成されたネットワークでも使うことが出来る汎用性が特徴である。

Pajek で対象となるネットワーク
・普通ネットワーク（有向、無向、有向無向混合）
・マルチリレーションネットワーク
・2 部グラフネットワーク
・動的ネットワーク

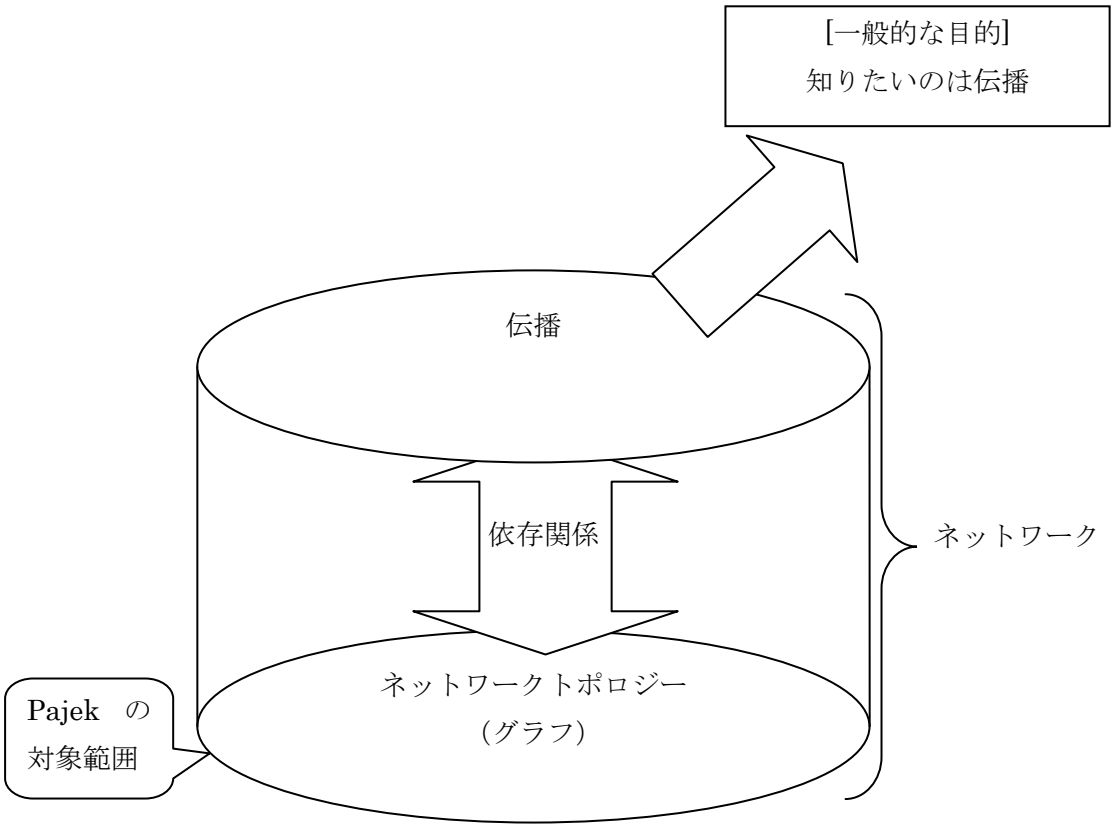
グラフ：頂点と頂点の関係の数学
ネットワーク：グラフ上を伝播対象が伝播する存在
伝播対象：HIV、風邪ウイルス、電力、（物理・化学などの）現象、状態（倒産など）、電波うわさ、パケット、感情など
数理生態学：拡散方程式や SIR・SIS モデルによる８面格子状（４角形のグリットの上下左右と斜めの８つへの伝播）伝播を対象にする
拡散（伝播）ルール：拡散や伝播のルールを決めた方程式・アルゴリズム
ネットワーク生態学：さまざまなグラフ上（ネットワーク上）の伝播を対象にする。実際対象のグラフトポロジーで様々な伝播規則を用いるケースが多い
グラフトポロジー：グラフの形、グラフとも言う
グラフのリワイヤー：グラフの張りなおし
頂点特性：頂点が伝播を制御することもある。頂点が伝播対象を保持したり、抑制したりなど
伝播初期状態：伝播要素の初期状態
伝播決定要素：グラフのトポロジー、拡散（伝播）ルール、頂点特性、伝播初期状態の４つ
グラフ評価：グラフ理論より、オイラー路（一筆書きできるか？）など
ネットワーク評価：対象グラフを伝播させてみる
類似度測定：グラフトポロジーは基本的に行列データなので、統計的手法によりネットワークの類似度などの測定が可能（グラフ理論の１つ）

## Pajek で伝播を扱うことはできない。

Pajek の Network analysis とは、グラフ評価（graph Analysis)のことで、伝播については扱わない。ただ、

- 伝播していく様子を可視化する
  （*.tim によるスナップショットの時系列可視化）
- 伝播の過程のグラフトポロジーの解析をする

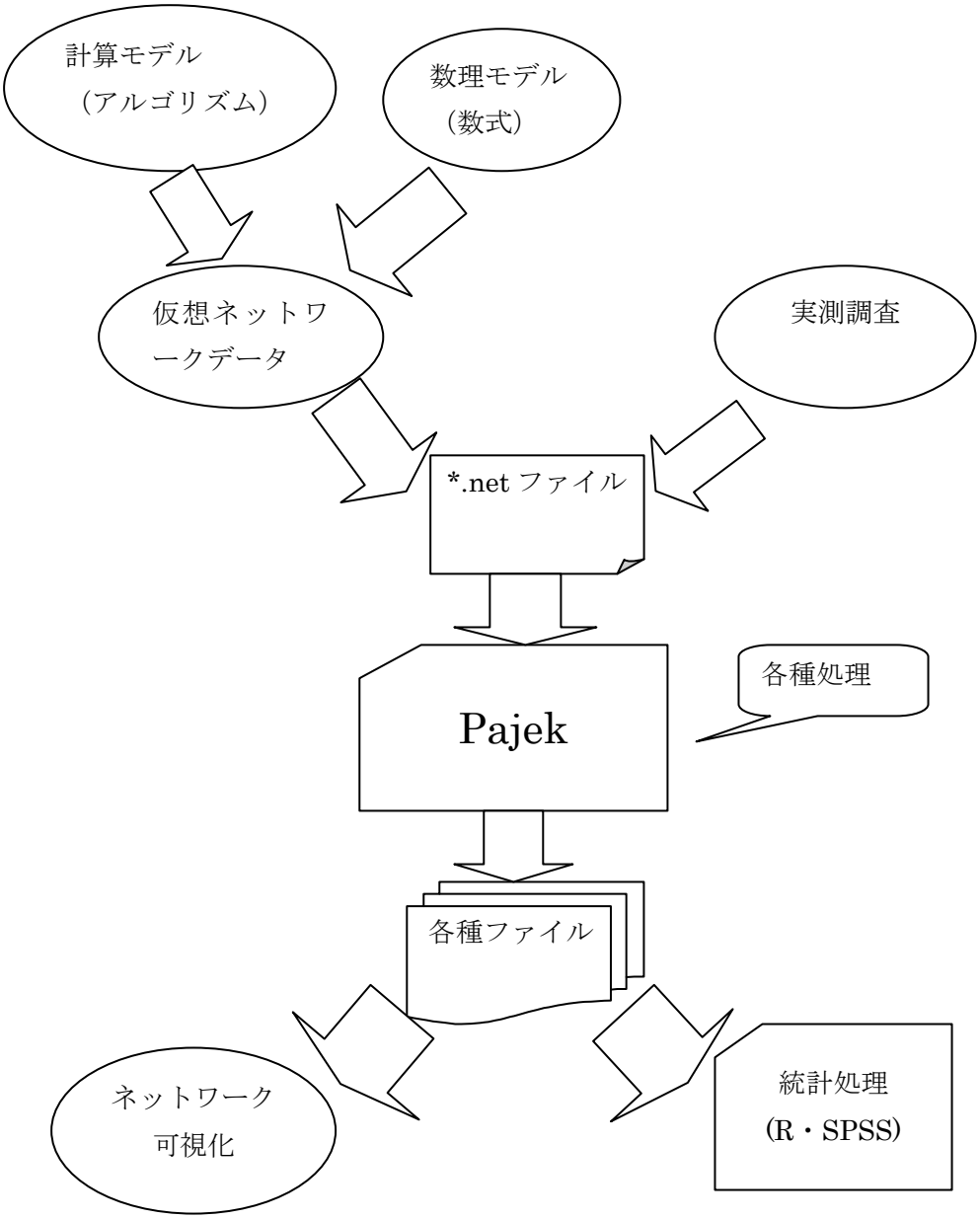
（次数の数、次数分布、Core など）
ことは Pajek で扱うことが出来る。
ネットワークの本質、「伝播」である。一般的にネットワークで一番知りたい本質は、対象がどのように伝播していくかであるからである。ネットワークトポロジー（グラフ）は、『伝播の仕方の構造』でしかない。つまり、Pajek でネットワークトポロジーを解析・可視化してもネットワークがわかったとはいえない。




図：ネットワークと Pajek の対象範囲

## 2.2Pajekの使用

Pajek の使用するためには、どのような対象のネットワークでも、まず、*net ファイルを用意しなければならない。それが、Pajek を使用する第一ステップである。




図：Pajek の処理の流れ

## 2.3Pajekのインストールとその方法

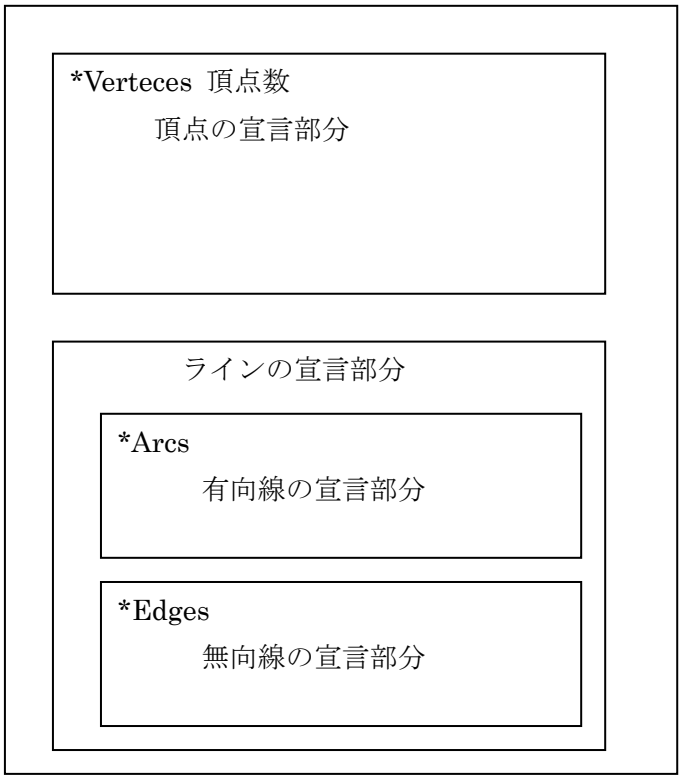
Pajek は以下の URL でダウンロードできる「フリーソフト」である。

Pajek 関連サイト
http://vlado.fmf.uni-lj.si/pub/networks/pajek/

以上のURLから、ダウンロードし、インストールすればPajekを使用することが出来る。

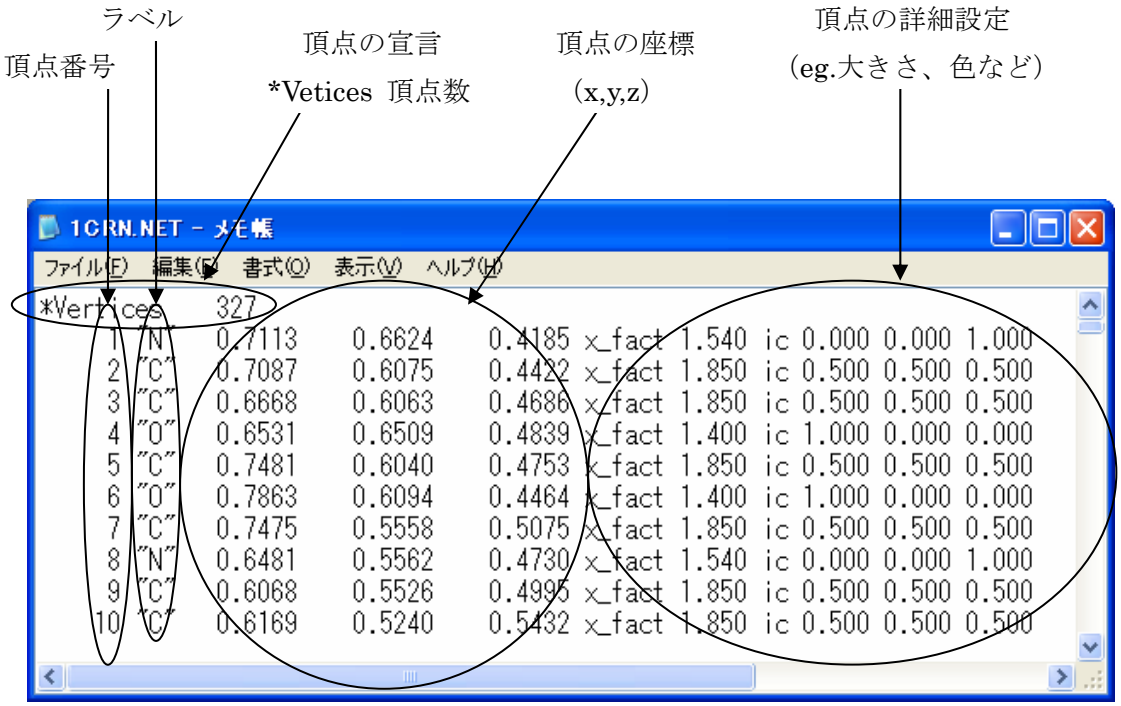
## 2.4 各種ファイルの説明

### 2.4.1*.netファイルの説明




図：*.net ファイルの大まかな構成

*.net ファイルは、上図のように頂点数の宣言部分と、ラインの宣言部分（有向線の宣言部分と無向線の宣言部分）に分かれる。




図：*.net ファイルの構成（Vertices 宣言部分）

１行目『*Vertices　頂点数』
読み込む際に、１行目の宣言頂点数とそれ以降に表示されている頂点データの数が合わないと、読み込みの際にエラーが発生する。
２行目~N 行目『頂点番号　“ラベル”　横（x）座標　縦（y）座標　奥（z）座標』
（N は指定頂点数）
z 座標は省略可能。

注意点：*.net ファイルに半角シフトでなく全角シフトが入るとエラーが発生するので注意する。

さらに以下のコマンドで頂点を詳細に設定できる。
・形
・大きさ
・色

| 設定項目 | コマンド | 意味 | 例 |
|---|---|---|---|
| 形の設定 | elipse | 楕円形にする | |

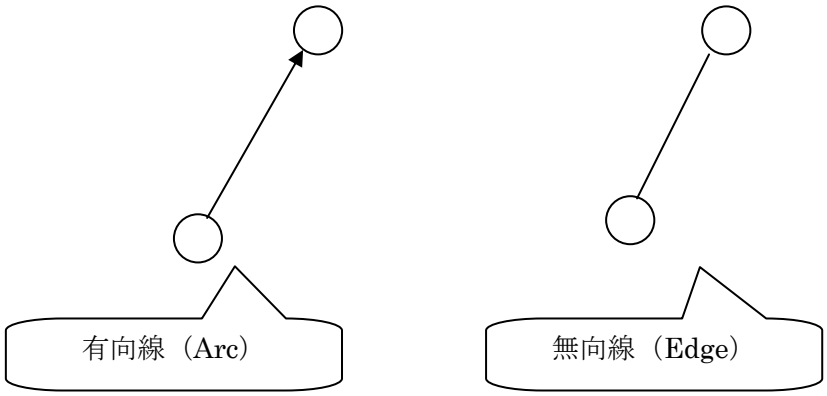
| | | | |
|---|---|---|---|
| | box | 四角形にする | |
| | diamond | 菱形にする | |
| | triangle | 三角形にする | |
| | empty | ノードを表示しない | |
| 大きさの設定 | x_fact | x 軸方向に何倍拡大するか | x_fact 3：x 方向に 3 倍拡大 |
| | y_fact | y 軸方向に何倍拡大するか | x_fact 5：y 方向に 5 倍拡大 |
| 色の設定 | ic | ノードの内側の色を設定 | ic Red：ノード内を赤色にする |
| | bc | ノードの枠の色を設定 | bc Blue：ノードの枠を青色にする |

表：頂点の設定項目のコマンド

色の設定は、RGB（Red Green Blue）で指定も出来る。
例：ic 0.500 0.500 0.500

先頭『*Arcs』
それ以降の指定『出頂点番号　入頂点番号　重み』
で、有向線を指定することが出来る。*Arcs 全体を省略することが可能である。

先頭『*Edges』
それ以降の指定『出頂点番号　入頂点番号　重み』
で、無向線を指定することが出来る。* Edges 全体を省略することが可能である。




図：有向線と無向線

さらに以下のコマンドで頂点を詳細に設定できる。

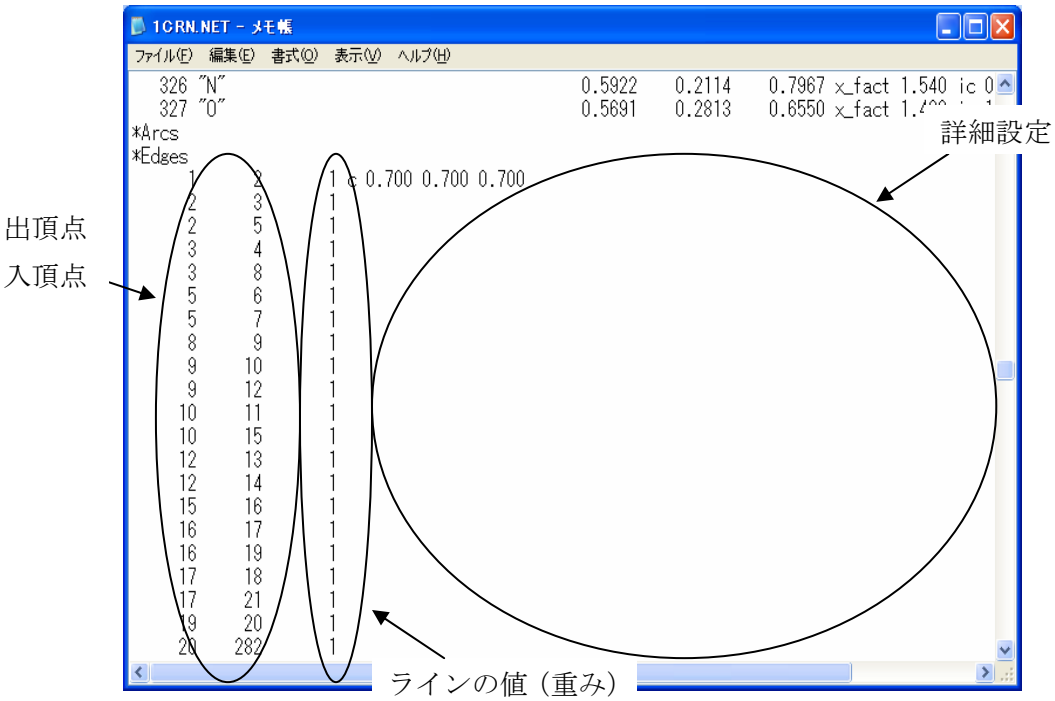
・ラベル
・色
・形

| 設定項目 | コマンド | 意味 | 例 |
|---|---|---|---|
| ラベルの設定 | l | ラベルを付ける | l nakama |
| 色の設定 | c | 紐帯の色を設定する | c Green |
| 形の設定 | p Solid | 紐帯を実線にする | |
| | p Dots | 紐帯を点線にする | |

表：ラインの設定項目

色の設定は、RGB（Red Green Blue）でも設定できる。
例：c 0.700 0.700 0.700

紐帯の実線・点線のデフォルト設定は、
・ラインの値（=重み）が正→実線
・ラインの値（=重み）が負→点線
となっている。




図：*.net のラインの設定

拡張子を.net に指定すれば、Pajek の Mein スクリーンで読み込め、そのファイルを Pajek で解析、可視化できる。

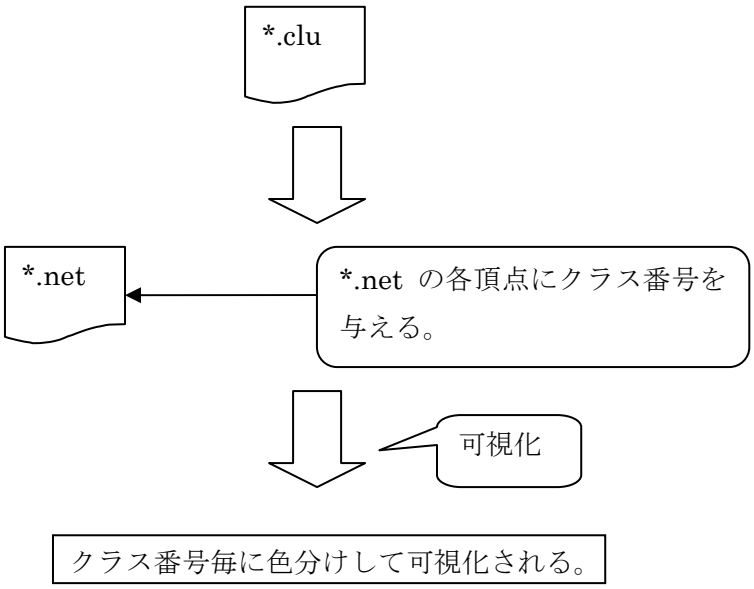
図：*.clu のクラス番号と色分けの色の設定2.4.2*.clsファイルの説明

頂点に対するクラス分けの数字が入る。そのため

・整数

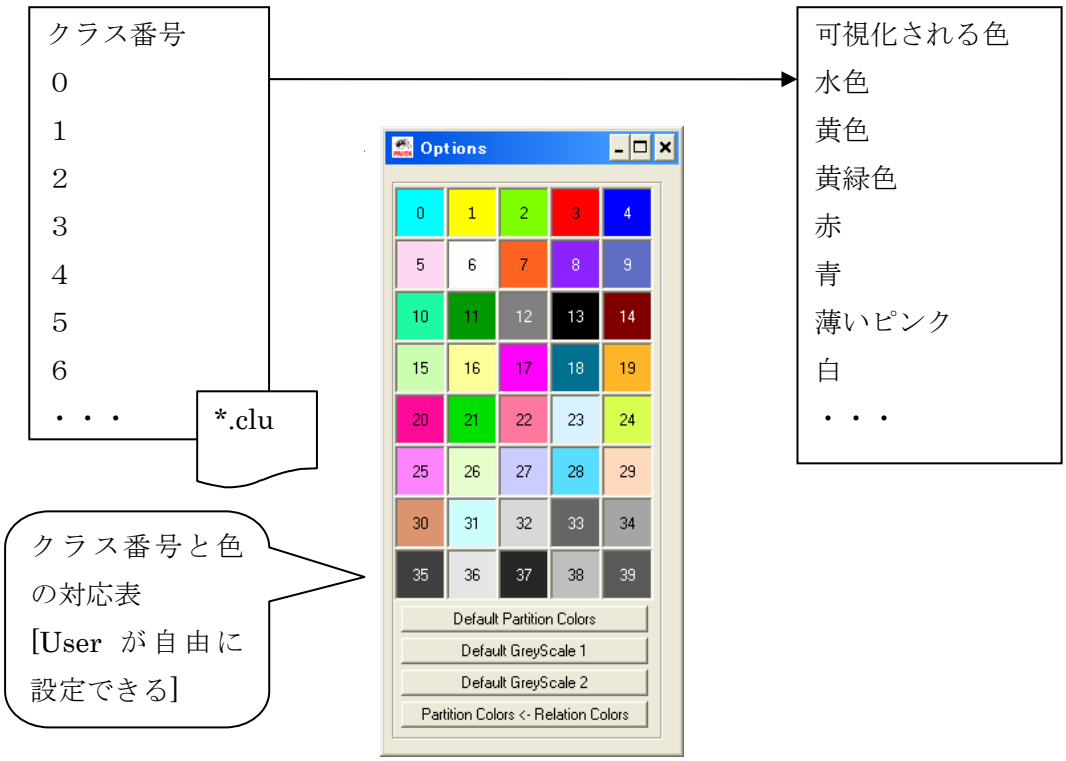
・離散数（連続でない）

である。

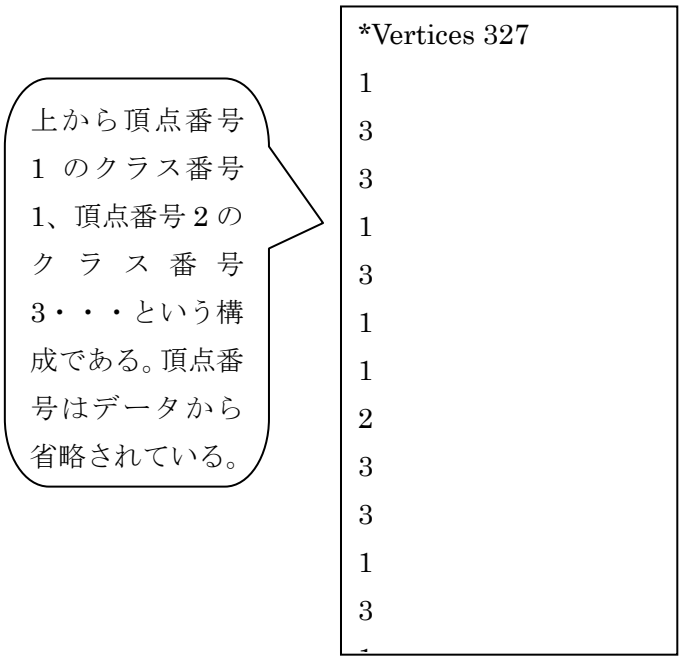



図：*.clu の役割

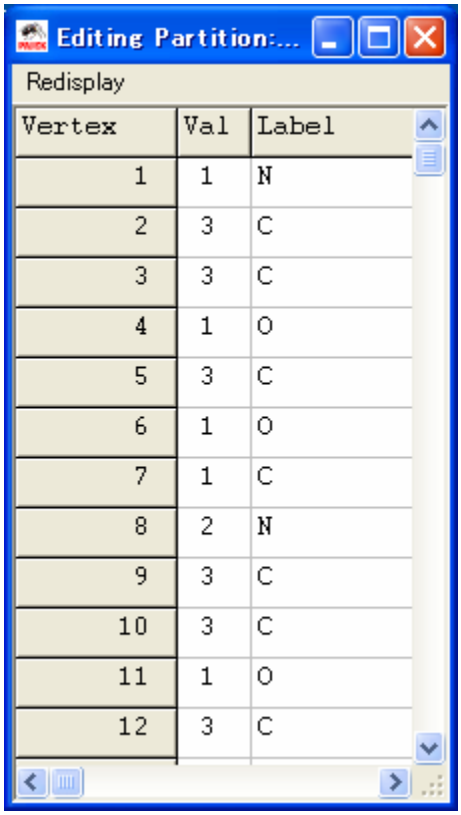
クラスと可視化の色の対応は Draw window の Options>Colors>Partition Colors によって確認できる。



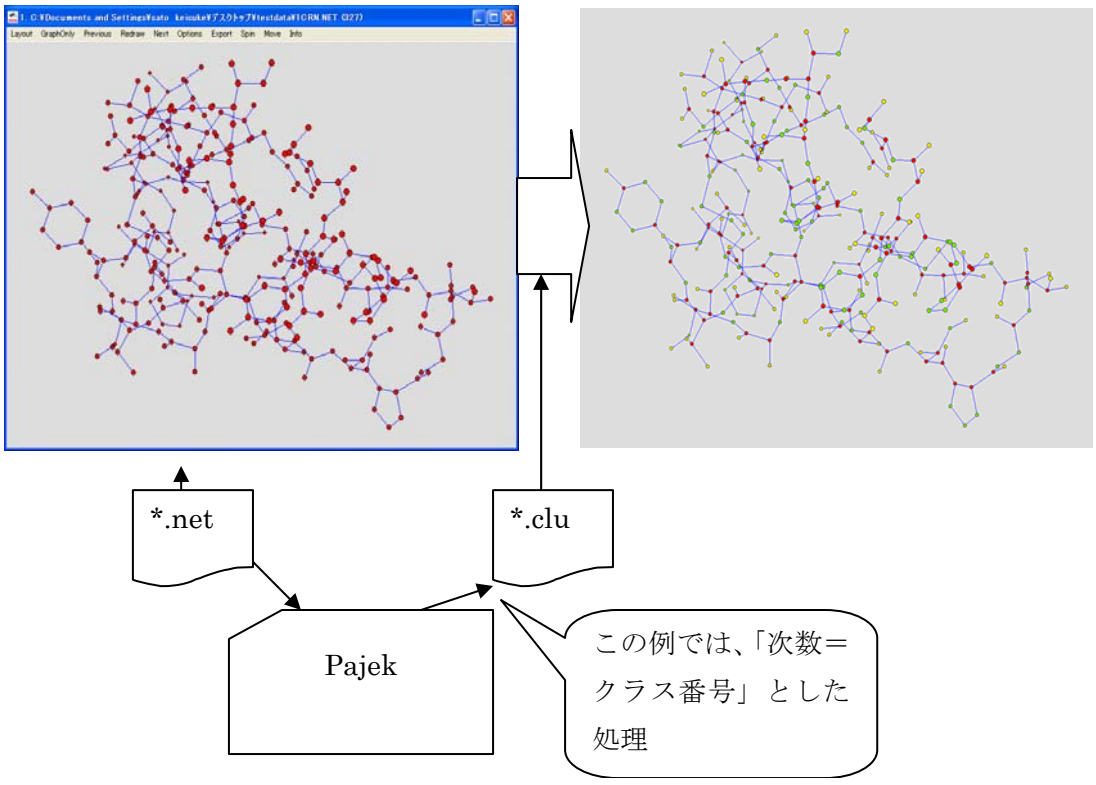

図：*.clu のクラス番号と色分けの色の設定




図：cls ファイルの構成




図：*.cls のメインスクリーンの Edit の表示




図：クラス番号毎の色分けの例

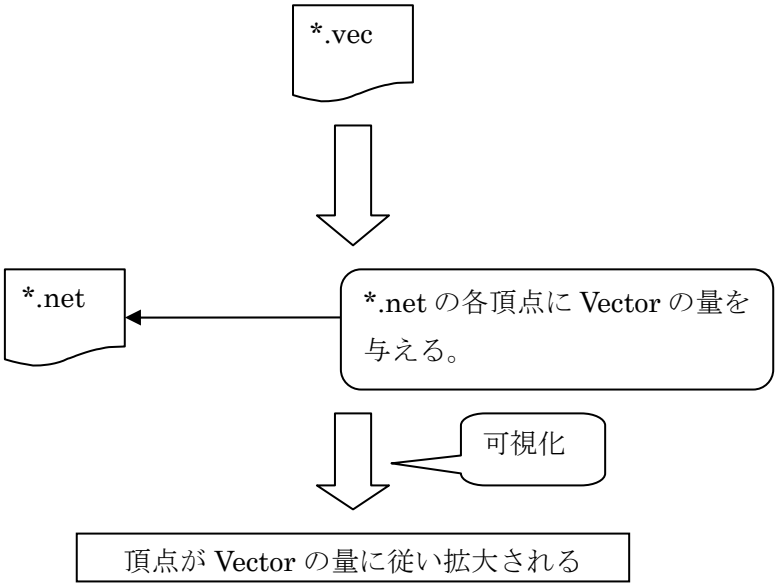
### 2.4.3*.vecファイルの説明

頂点の大きさを決定する値である。

・自然数

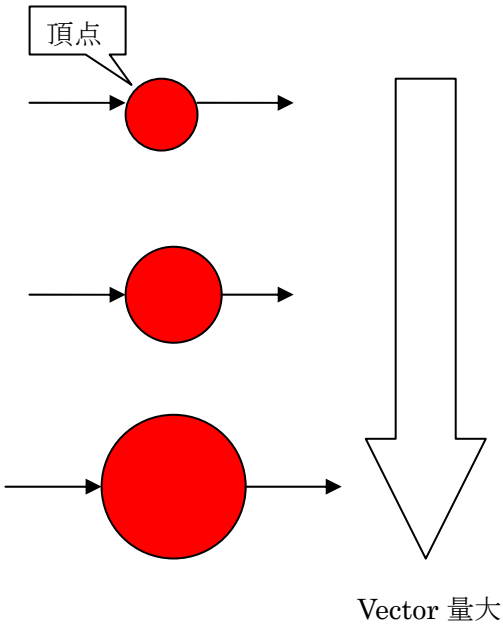
・連続数

である。




図：*.vec ファイルの役割

*.Vec のデータを「Vector 量」と呼ぶならば、Vector 量と可視化時の頂点の大きさには以下の関係がある。

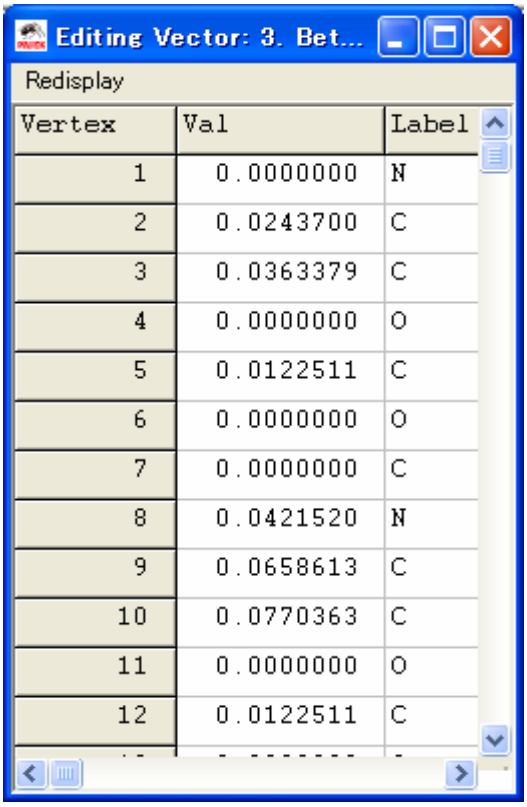



図：Vector 量と可視化時の頂点の大きさ

上から頂点 1 の Vector 量、頂点 2 の Vector 量・・・という構成になっている。頂点番号はデータから省略されている。

```
*Vertices 327
0.00000000000000000
0.02436998584237848
0.03633789523360075
0.00000000000000000
0.01225106182161397
0.00000000000000000
0.00000000000000000
0.04215195847097688
0.06586125530910807
0.07703633789523361
```

図：*.vec ファイルの構成

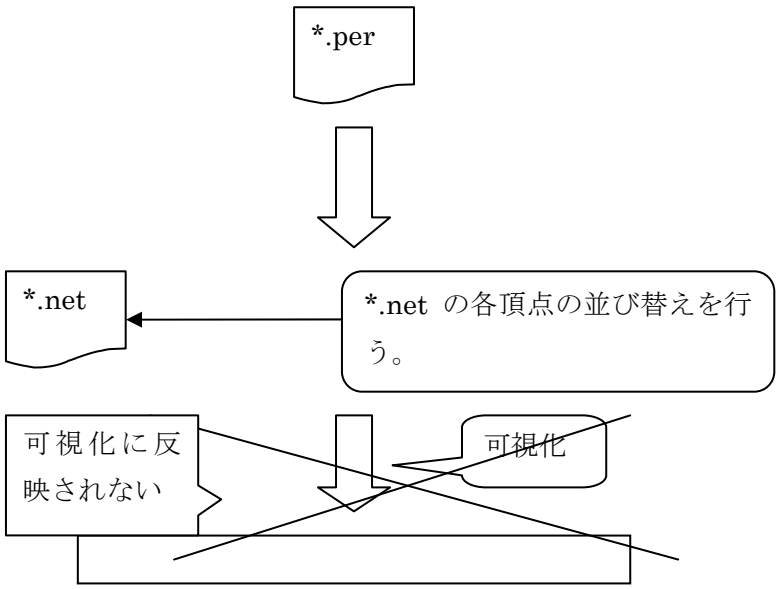



| Vertex | Val | Label |
|---|---|---|
| 1 | 0.0000000 | N |
| 2 | 0.0243700 | C |
| 3 | 0.0363379 | C |
| 4 | 0.0000000 | O |
| 5 | 0.0122511 | C |
| 6 | 0.0000000 | O |
| 7 | 0.0000000 | C |
| 8 | 0.0421520 | N |
| 9 | 0.0658613 | C |
| 10 | 0.0770363 | C |
| 11 | 0.0000000 | O |
| 12 | 0.0122511 | C |

図：*.vec のメインスクリーンの Edit の表示

### 2.4.4*.perファイルの説明

*.per は頂点の並び替えのためのファイルである。このファイルは可視化に反映されない。

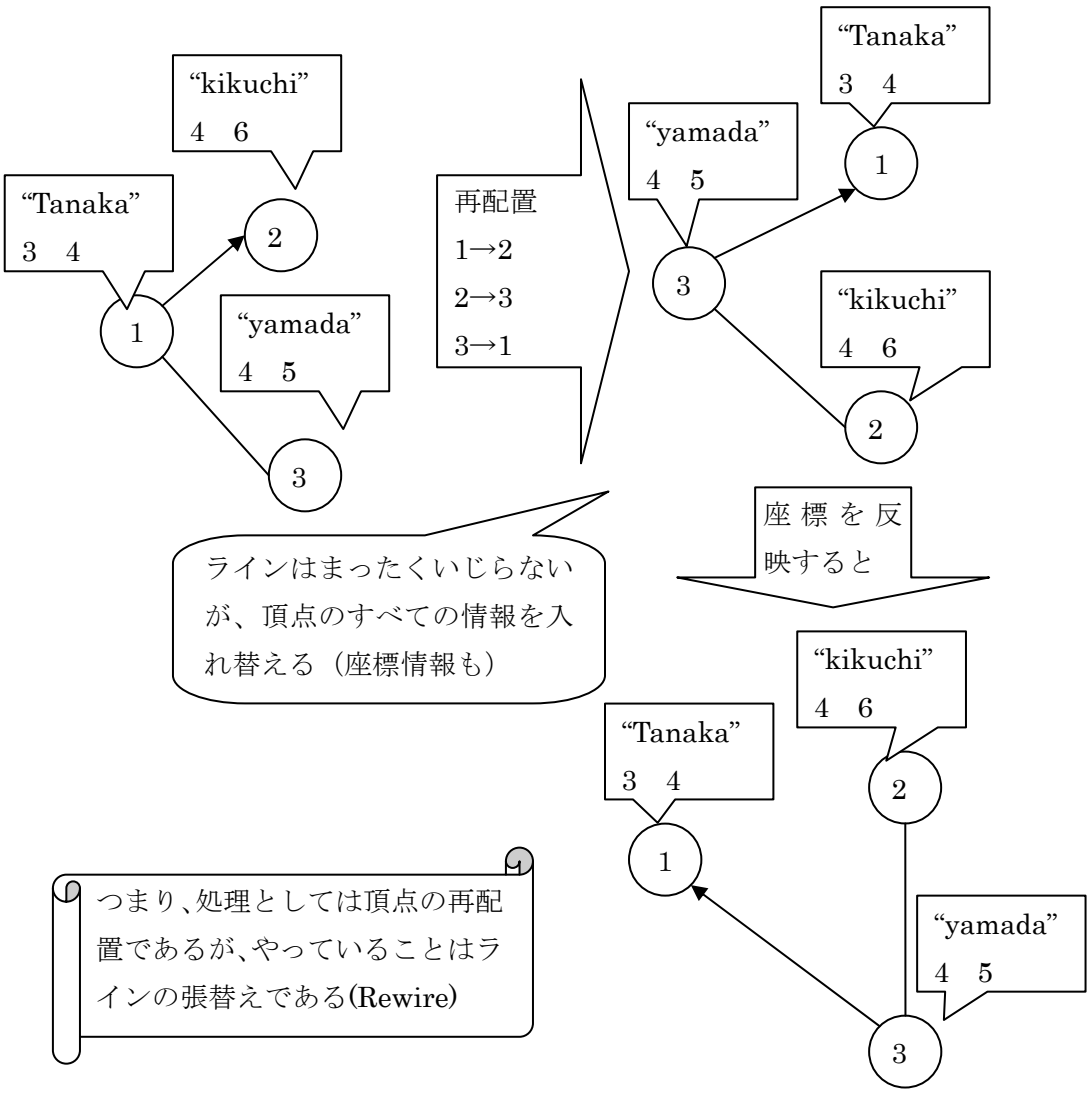



図：*.per ファイルの役割

頂点の再配置とは、下図の如く、言い換えるとラインの張替えである。したがって、*.per はラインの張替えのためのファイルであるといっていい。

頂点は張替えられたあとも、ラインの接続関係のデータはそのままであることには注意する。

ラインの張替えは、ネットワークサイエンスでよく用いられている手法・手段であり、Pajek では、「*.per」がこれを受け持つ。

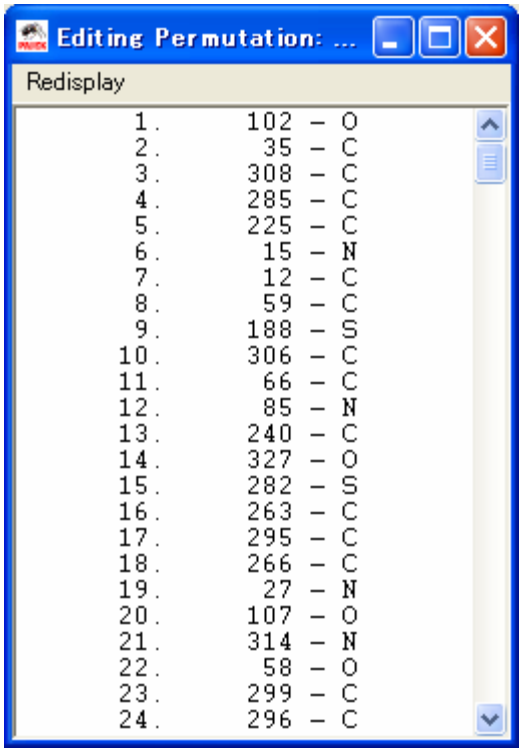



図：頂点再配置とラインの張替え

上から頂点番号 1 の再配置頂点、頂点 2 の再配置頂点である。つまり、1→102、2→35・・・となる。張替え元の頂点番号はデータから省略されている。

```
*Vertices 327
102
35
308
285
225
15
12
59
188
306
66
```

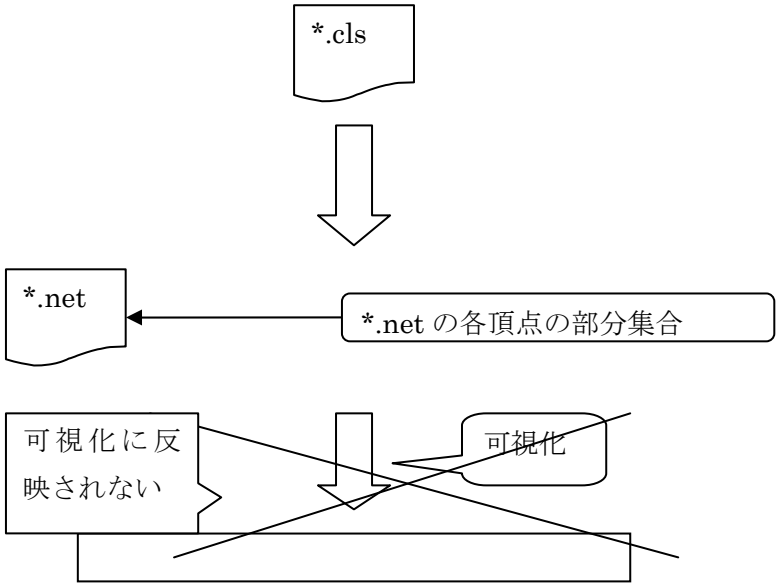
図：*per のファイル構造



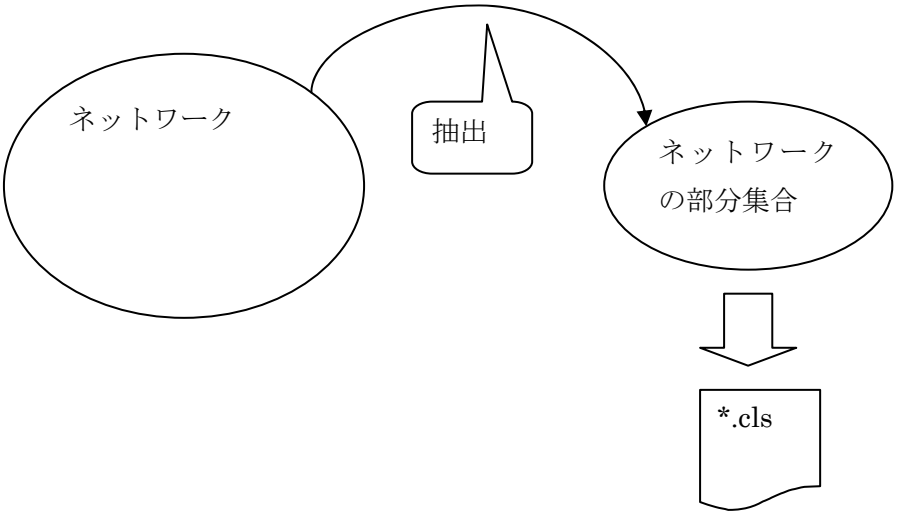

図：*.per のメインスクリーンの Edit の表示

### 2.4.5*.clsファイルの説明

全頂点の部分集合が入る。つまり。クラスターの情報が入ったファイルである。

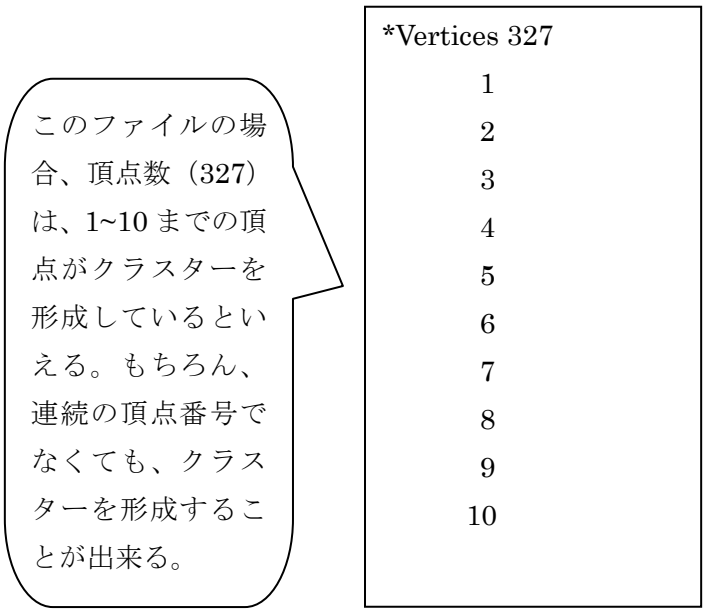



図：*.cls の役割



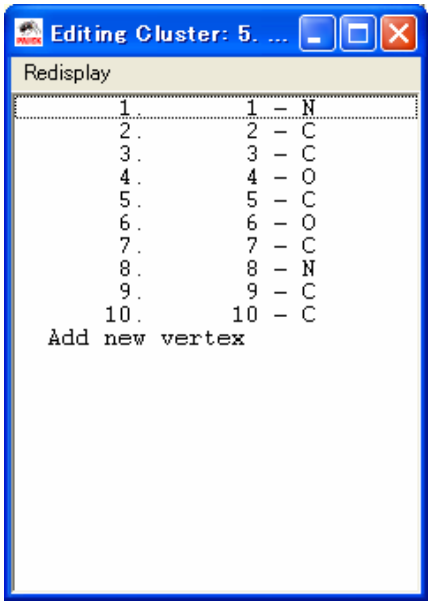

*.cls ファイルの生成

つまり、対象ネットワークを「クラスターかそれ以外か」に分けると考えてよい。




図：*cls のファイル構造

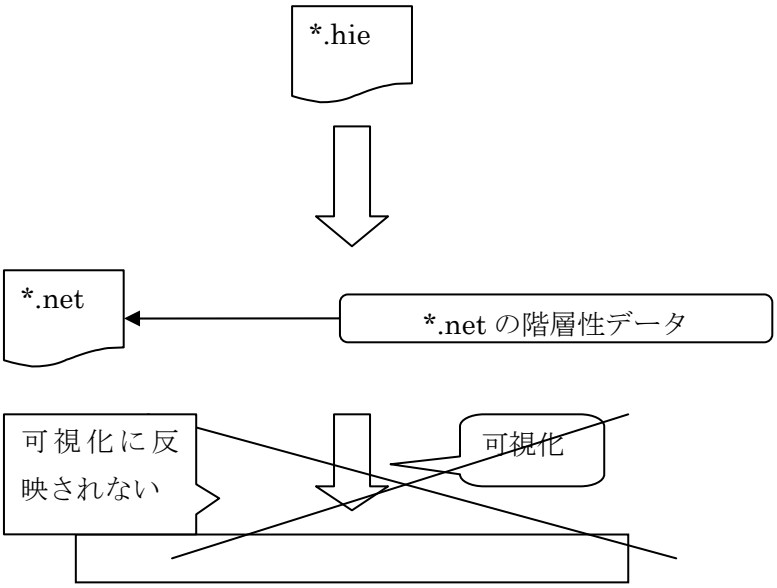
Edit 表示をダブルクリックすると、その頂点番号のデータを消すことが出来る
Add new vertex をクリックすると新しい頂点の情報を追加できる。



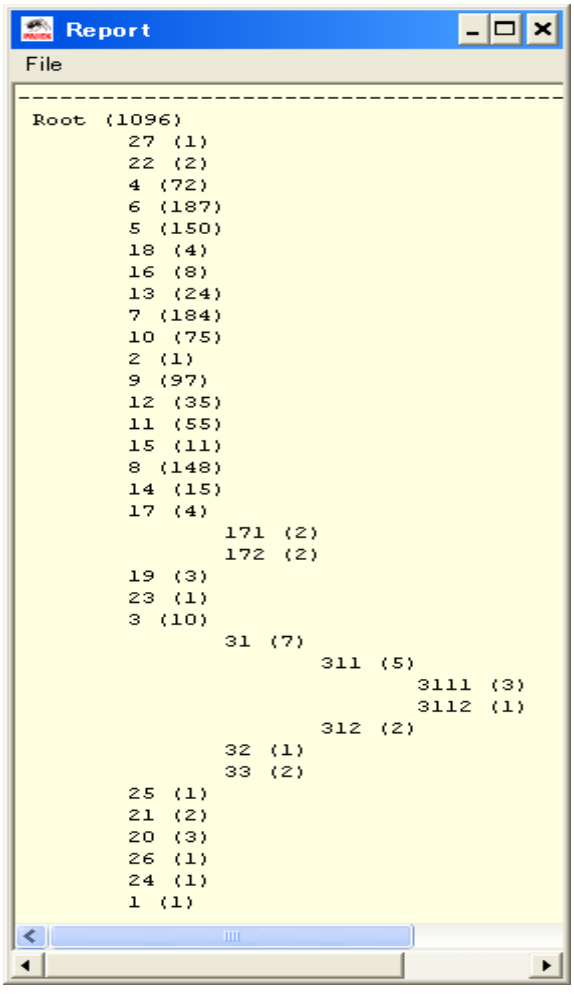

図：*.cls のメインスクリーンの Edit の表示

### 2.4.6*.hieファイルの説明

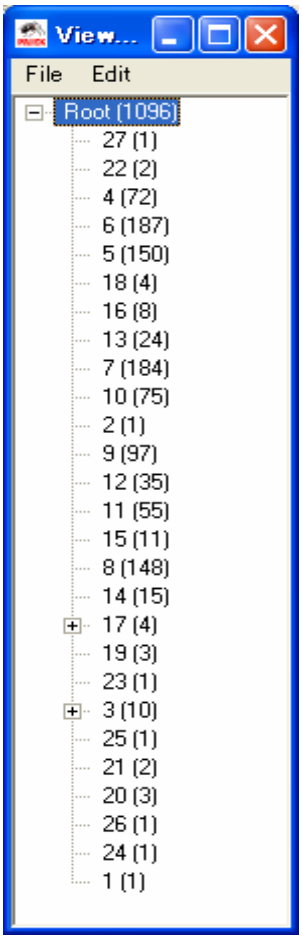
階層ネットワークの階層のデータが入っているファイルである。




図：*.hie ファイルの役割



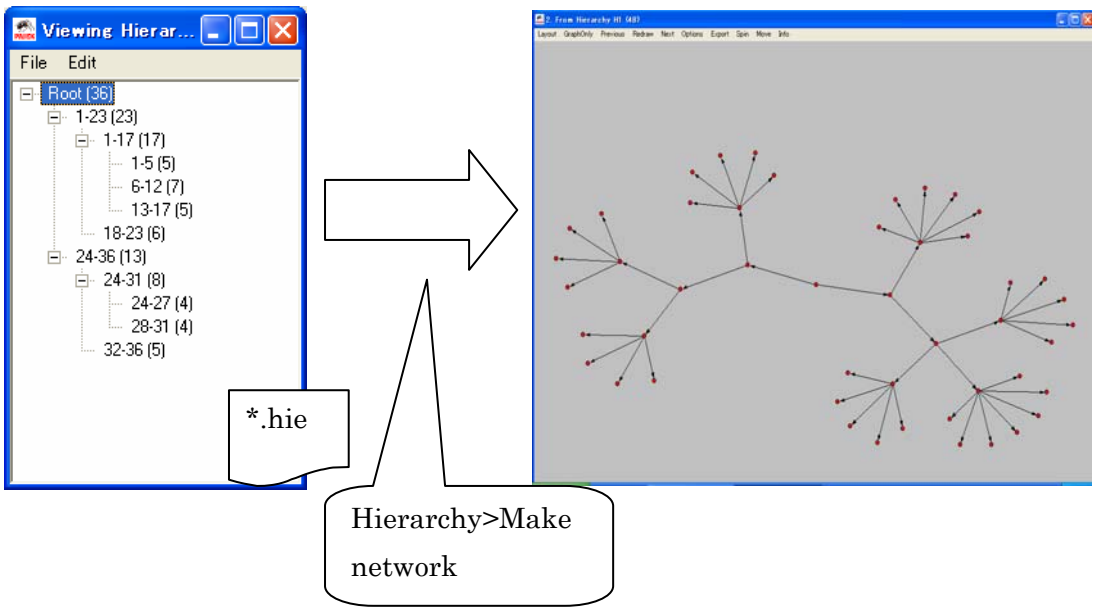

図：*.hie のメインスクリーンの Report の表示




図：*.hie のメインスクリーンの Edit の表示

```
+"Root" 0 0.000 1096 ""
  +"27" 0 1.000 1 ""
    *  1
  +"22" 0 1.000 2 ""
    *  2
    *  87
  +"4" 0 1.000 72 ""
    *  3
    *  27
    *  37
    *  42
    *  78
```

図：*hie のファイル構造




図：*.hie から階層ネットワークを作成した例

### 2.4.7*.pajファイルの説明

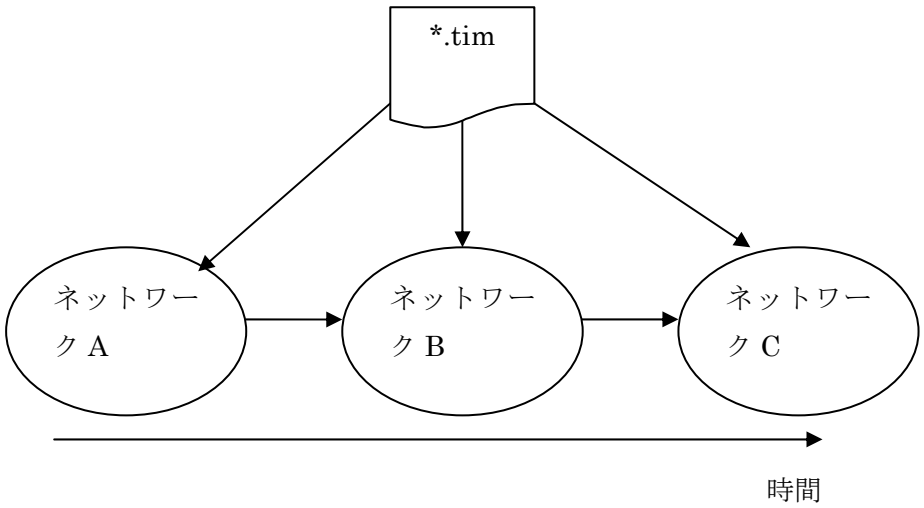
* .net から*hie まで、Pajek に関するファイルを一つにまとめたもの。

### 2.4.8*.logファイルの説明

過去に行われたセッションを記録し、再現することが出来る。

### 2.4.9*.timファイルの説明

ネットワークの成長タイムイベントを指定することが出来る。時間的にネットワークが変化するので、「動的ネットワーク」と呼ぶこともある。




図：動的ネットワークと*.tim

「タイムイベントを可視化で反映する」には、以下の手順を踏む。

① File>Time Events Network>read
② *.tim ファイルを読み込む
③ Net>Transform>Generate in Time>All
④ Select Last Time point で可視化するステップの最後（どのステップまで可視化するか）を指定。より正確に表現するなら、どのステップまでネットワーク（*.net）にタイムを生成するか？（Generate in Time）を指定する。
⑤ Select Step を入力する。何ステップ毎に可視化したいか（Generate in Time）したいかを決定する。
タイム（ステップ）毎の可視化=*.net にタイムを生成させる
⑥ 可視化する（Draw）
⑦ Draw メニューの Next や Previous で、現時点の可視化するステップを指定できる。

| 設定項目 | コマンド | 意味 |
|---|---|---|
| タイム | TI t | t ステップ目のイベントを以下に書くことを宣言 |
| | TE t | ‥‥‥‥‥‥‥‥‥‥‥‥‥‥を以上で終了する宣言 |
| 頂点 | AV v n s | 頂点を作る |

| | | |
|---|---|---|
| (Vertices) | HV v | ・・・・を見えなくする |
| | SV v | ・・・・を見えるようにする |
| | DV v | ・・・・を消す |
| | CV v s | ・・・・の特徴を変える |
| 有向線(Arcs) | AA u v s | 有向線を引く |
| | HA u v | ・・・・・・を見えなくする |
| | SA u v | ・・・・・・を見えるようにする |
| | DA u v | ・・・・・・を消す |
| | CA u v s | ・・・・・・の特徴を変える |
| | CD u v | ・・・・・・の方向を変更する |
| 無向線<br>(Edges) | AE u v s | 無向線を引く |
| | HE u v | ・・・・・・を見えなくする |
| | SE u v | ・・・・・・を見えるようにする |
| | DE u v | ・・・・・・を消す |
| | CE u v s | ・・・・・・の特徴を変える |
| | EP u v s | ・・・・・・を両向有向線に変更する |
| 両向有向線 | PE u v s | 両向有向線を無向線に変更する |
| | AP u v s | ・・・・・・・・・・を引く |
| | DP u v | ・・・・・・・・・・を消す |
| 有向線と無向線 | CT u v | 有向線と無向線の変更 |

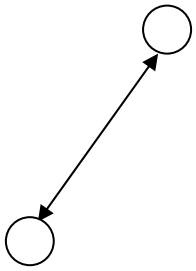
表；タイムイベントのコマンド

| | | |
|---|---|---|
| タイム | t | 時間(ステップ)番号 |
| 頂点 | v | 頂点番号 |
| | n | ラベル |
| | s | 特徴 |
| ライン | u | ラインから出る頂点番号 |
| | v | ラインが入る頂点番号 |
| | s | 特徴 |

表：タイムイベントコマンドの値記号の意味

特徴とは、重さ・形・色などの詳細な属性のこと。

ラインは、無向・有向・両向有向・無向と有向の変換の 4 つのこと。



図：両向有向線

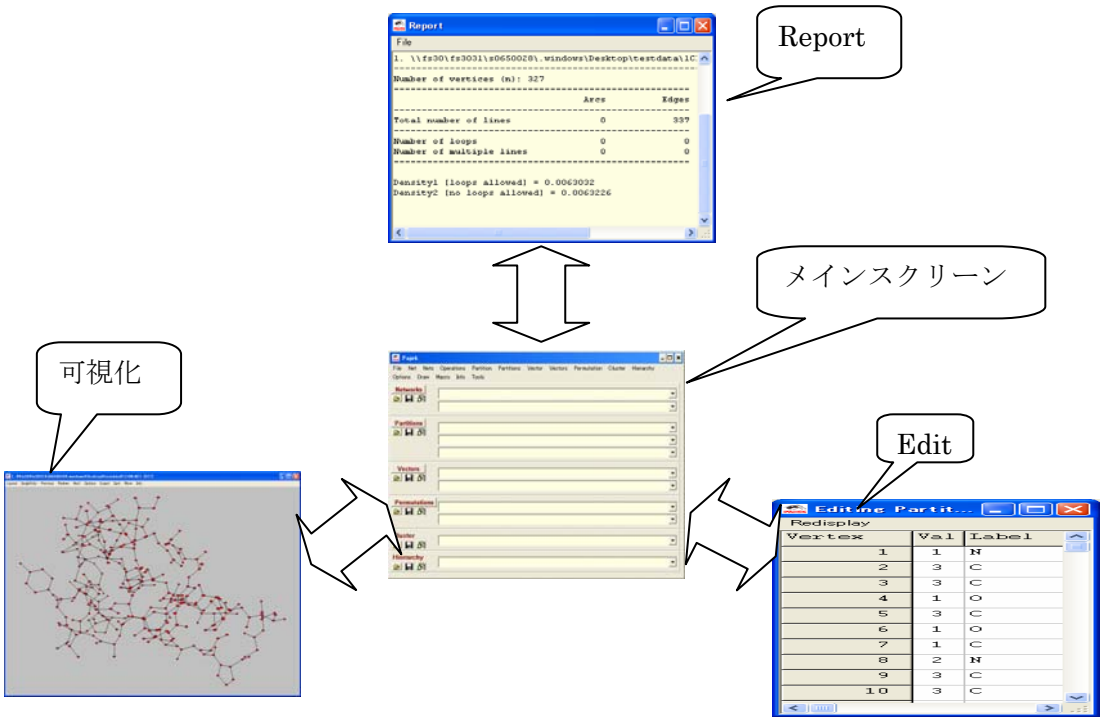
時間 1 での、
ネットワーク
の変化

```
*Vertices 1000
TI 1
AV 1 v1
AV 2 v2
AV 3 v3
TI 2
AA 1 2
AA 2 1
TI 3
AA 1 2
AA 3 1
TI 4
DA 2 1
AA 2 3
```

図：*.tim のファイル構造

## 2.5Pajekの各種ウインドウ

Pajek は、ウインドウベース（ウインドウで構成されている）プログラムである。主に、以下の 4 つのウインドウにより構成されている。




図：Pejke、4 つのウインドウ

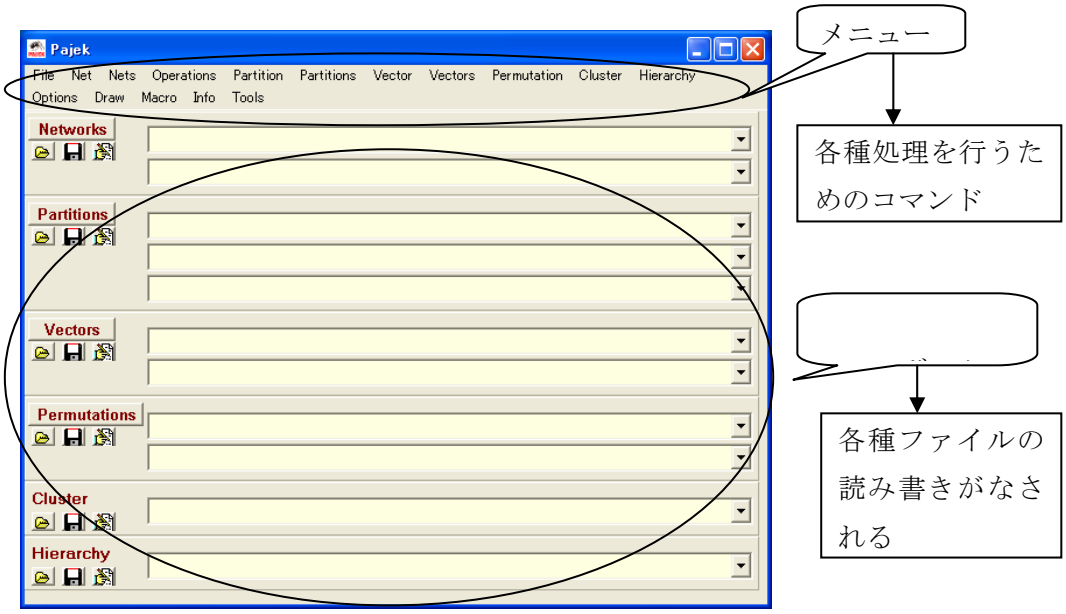
1、メインスクリーン：ファイルの読み書き・コマンド入力という中心的ウインドウ

２、可視化：ネットワークを可視化するためのウインドウ。

３、Report：処理情報を表示するウインドウ。

４、Edit：ファイル情報を表示するウインドウ。

## 2.6Main Screenの構造




図：メインスクリーンの構造

1.Network のボックス

*.net を読み込み、読み込まれたファイル名を表示する。* .net はネットワーク情報である。Pajek の処理のため、中心的なファイルで、これを元に様々なファイル(*.clu、*.vec など)が生成される。

2.Partitions ボックス

頂点の分類のためのファイル*.clu が読み込まれたら表示されるボックス

（Partition なのに、拡張子が*.clu であることに注意）

3.Vectors ボックス

*.vec ファイルが読み込まれたら表示するボックス。*.vec ファイルは、

頂点の大きさに反映される。

4.Permutations ボックス

頂点番号の再配置のための*.per ファイルが読み込まれたら表示するボックス。

このファイルは視覚化に反映されない。

5.Clusters ボックス

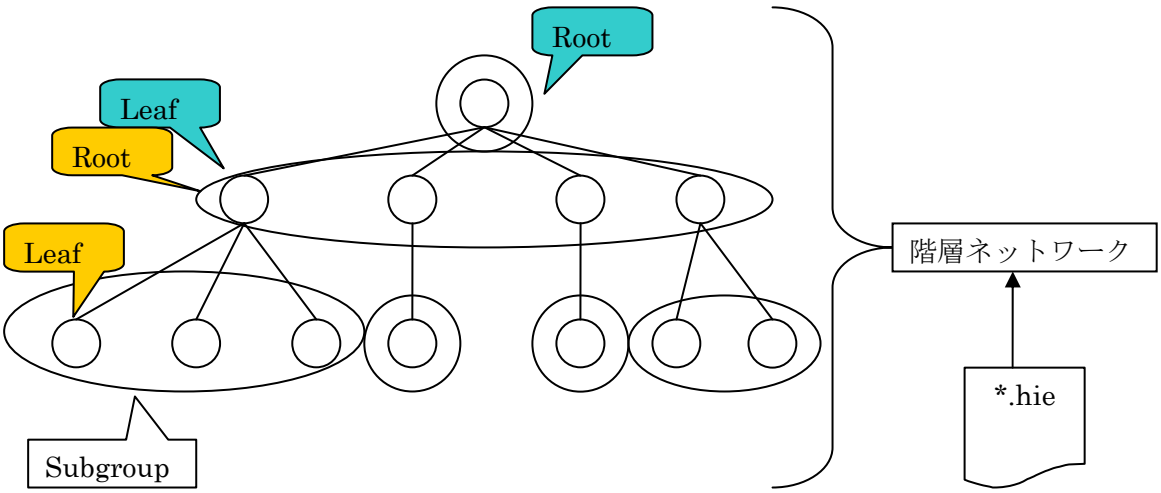
頂点の部分集合（サブネットワーク、つまり、クラスター）を作るための*.cls ファイルが読み込まれたら表示するボックス。

このファイルには、ラインの情報がない。

このファイルは視覚化に反映されない。

6.Hierarchies

階層的に頂点を整える*.hie ファイルが読み込まれたら表示するボックス。

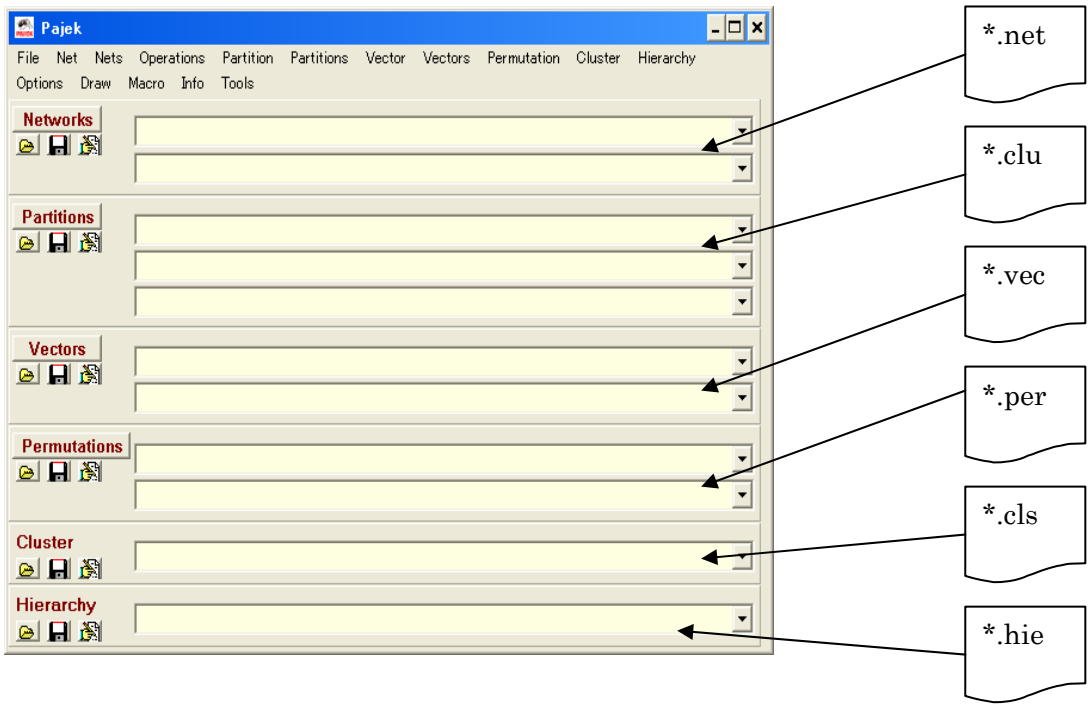



図：階層ネットワーク

Subgroups：階層ネットワークになったときの頂点の部分集合のこと

Root・Leaf：枝葉の関係。一階層上位の 1 つの頂点を Root（枝）とし、それが所有している下位の 1 または複数の頂点を Leaf（葉）という。この関係はどの階層の頂点でも成り立つ。

以上をまとめると、メインスクリーンの各ボックスと読み込まれるファイルの種類（拡張子）は、下図のようになる。




図：メインスクリーンの各ボックスと読み込まれるファイルの種類（拡張子）

# 3.メインスクリーン上のツールバーの各種メニュー

## 3.1File

Pajek の処理に必要な各種ファイルの基本的処理（Read,Edit,Save,Change label,Dispose）を行う。

>networks

<Read

*.net ファイルの読み込みを行う。*.net はアスキーコードで書かれたファイルなので、テキストエディタで編集できる。

<Edit

編集したい頂点を選択し、その頂点のリンクを追加・削除できる。

つまり、対象ノードのリンクを編集できるが既存のリンクは修正できない。

修正したい場合は、一度リンクを消去し、それから追加しなければならない。

<Save

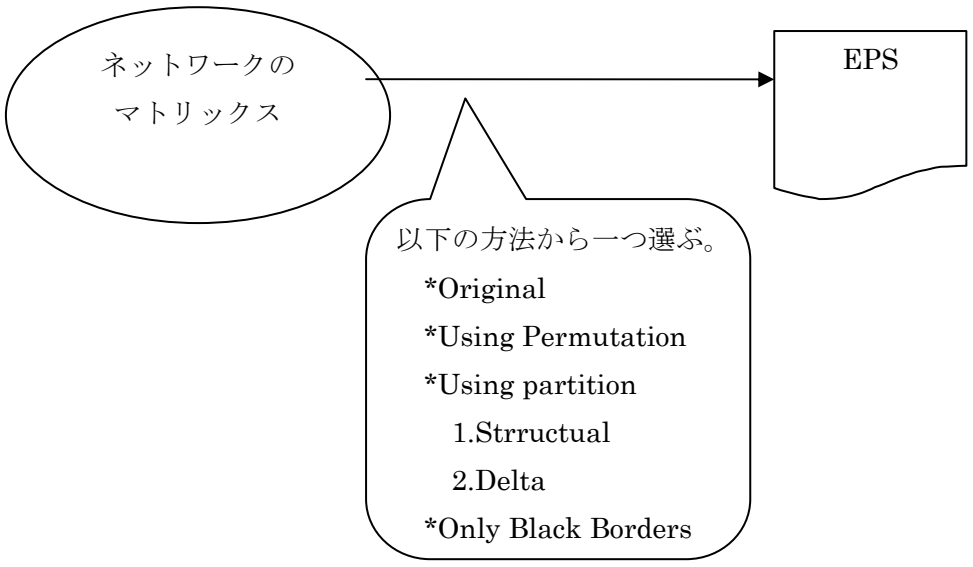
*.net ファイルをセーブする。

<Export Matrix to EPS

ネットワークのマトリックス（行列）を EPS ファイルとして書き込む。

EPS（Encapsulated PostScript）：画像ファイルフォーマットのひとつ

ここでいうマトリックス（行列）とは、ネットワークの視覚化情報をコンピュータ上であらわすデータ構造のことである。以下の、サブメニューは、EPS への書き込み方である。



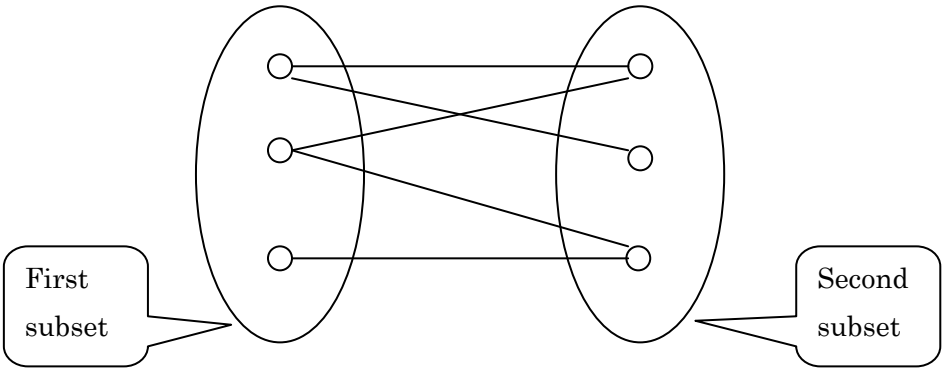

図：Export Matrix to EPS

*Original

初期状態で EPS ファイルに書き込む。1-mode なら 1-mode で 2-mode なら 2-mode で書き込む。

1-mode：2 部グラフでないネットワーク（通常のネットワーク）

2-mode：2 部グラフ（同じ頂点部分集合にリンクがないネットワーク）




図：2 部グラフ(2－Mode)の例

*Using Permutation

Permutation の.per を反映した EPS ファイルの作成

*Using partition

Partition の.cls を反映した EPS ファイルの作成

1.strructual

クラスター：Partition によって生成されるグループのこと

濃度：クラスター内の頂点の数/ネットワーク全頂点数

で表される。クラスターがネットワークでどれだけの位置（数、大きさ）を占めるかを表現できる。各クラスターに 1 つずつ属性としてつく数字で、クラスターを表す指標のひとつである。ネットワークには複数のクラスターが存在することが普通であるから、1 つのネットワークに複数の濃度の値が存在する。

濃度を設定すればどのような頂点がクラスターわけされるか決まる。ため、（クラスター分類）＝（濃度の決定）である。

クラス：クラスターのこと。また、2 部グラフにおいては、2 部グラフの一方と他方のひとつずつを表す。

濃度はクラス間のラインの取りうる最大値に従い標準化される（決定される）。

クラス間のライン：クラスとクラスをつなぐラインのこと。

ネットワークは、濃度を反映した、つまり、クラスター分類したネットワーク（Dense Network）に整形される。

2.Delta
濃度はクラス内にある隣接ノードの最大入力・出力値を持つ頂点（複数）に従い標準化される（決定される)。ネットワークは、まばらネットワーク（Spare Network）に整形される。

*Only Black Borders
すべての、チェックされた行列の中にあるマトリックスは、Black borders を持つことになる。違う言い方をすれば、white and light squares を持つことになる Dark squares は、Black borders を持つことになる。

<Change label
*.net ファイルの名前変更
<Dispose
*.net ファイルを消去する

>Time Events Network
ネットワークのタイムイベントを読み込む。
<Read time Events
ctrl+T でショートカットでき、タイムイベントファイルを読み込む。
<save time Events
ctrl+I でショートカットでき、タイムイベントファイルを書き込む。

>Partition
Pertition ボックスに関する.cls ファイルの（Read,Edit,Save,Change label,Dispose）の処理を行う。
<Read
<Edit
<Save
<Change label
<Dispose

>Permutation
Permulation ボックスに関する.per ファイルの（Read,Edit,Save,Change label,Dispose)の処理を行う
<Read

<Edit
<Save
<Change label
<Dispose

>Cluster

Cluster ボックスに関する.clu ファイルの（Read,Edit,Save,Change label,Dispose)の処理を行う

<Read
<Edit
<Save
<Change label
<Dispose

>Hierarchy

Hierarchy ボックスに関する.hie ファイルの（Read,Edit,Save,Change label,Dispose)の処理を行う

<Read
<Edit
<Save
<Change label
<Dispose

>Vector

Vector ボックスに関する.vec ファイルの（Read,Edit,Save,Change label,Dispose)の処理を行う

<Read
<Edit
<Save
<Change label
<Dispose

>Pajek Project File

<Read pajek
*.paj：networks から vector まですべてのデータが含まれているファイル
*.paj を読み込む。

<save pajek
*.paj を書き込む

>Repeat session

セッション：Pajek 上で過去に行った処理

Log ファイル：Pajek 上で過去に行った処理が適宜記憶されているファイル

*.log ファイルを選択させる。過去のセッションを、*.log ファイルを選択させることで再現する。

メインスクリーンの Networks2 段目にログファイルが読み込まれる。

>Show Report Window

*.net ファイルに関する簡単なレポートを Window に表示する

>Exit

pajek を終了する。

## 3.2Net

1 つのネットワーク(*.net)に対する情報追加を行う。また、ネットワークの変形に関するさまざまな処理を行う。

>Transform

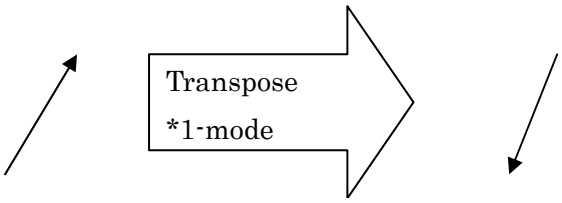
<Transpose

選択したネットワークの置換を行う。

*1-mode

1-mode（network）：普通のネットワーク。2-mode でないネットワーク。

1-mode ネットワークの有向線の向きを変える。




図：Transpose *1-mode

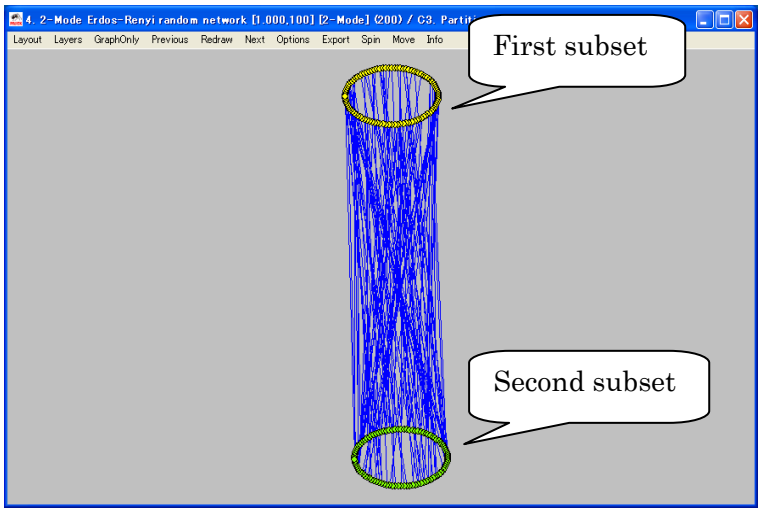
*2-mode

Rows（行）と Cols（列）を置換する。

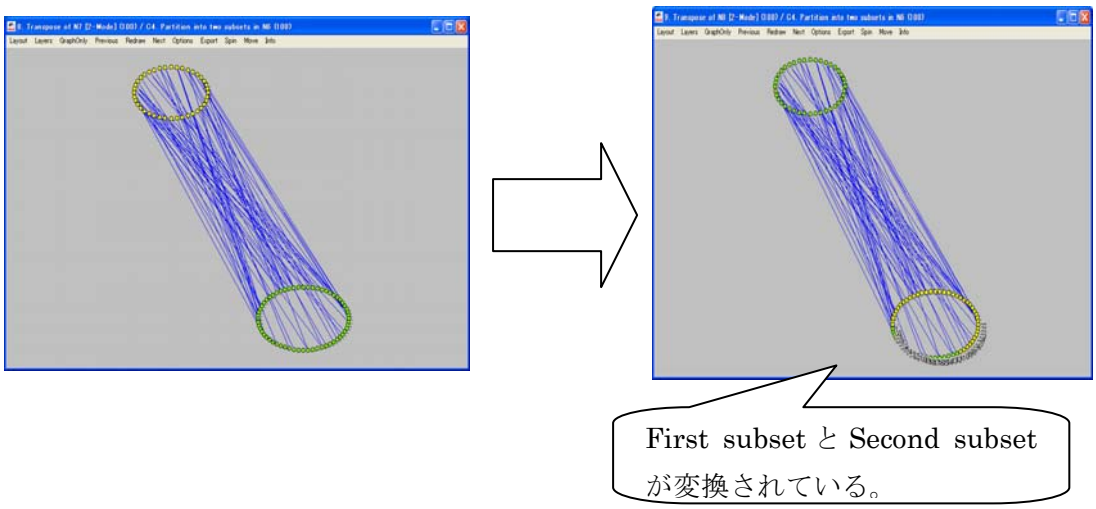
2-mode：2 部グラフ

Row：行

Col：column の略、つまり、列




図：2-mode のランダムネットワーク



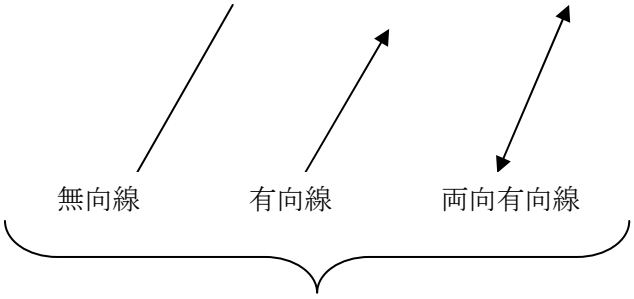

図：Transpose *2-mode

<Remove

ネットワークの構成要素の“消去”を行う。

Remove で行う処理と同等の操作は*.net ファイルの arcs や edges のラインを手動消去することである。このメニューは、膨大な数の頂点・ラインをメニューで消去することが出来る。

ライン：以下の図を参照




ライン

*Selected vertices

選択した頂点を消去する。

手順

①『Selected vertices to remove』のコマンドボックスが出るので、消去したい頂点番号を入力する

②入力した頂点が*.net ファイルから消去される。

*All Edges

選択したネットワークのすべてのエッジ（無向線）を消去する。

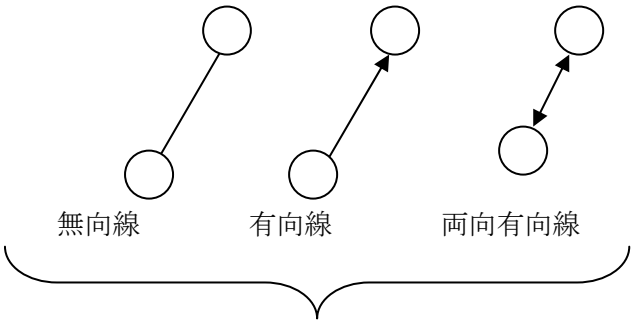
*All Arcs を押す

選択したネットワークのすべての Arcs（有向線）を消去する。

*Multiple Lines を開く

選択したネットワークの Multiple Line を消す。

Multiple line : 特定の有向線(Arc)や無向線(Edge)、すなわち、特定の「順序（秩序、配列）ある」または「順序（秩序、配列）ない」頂点のペア、一回以上起きるとき、これらは、Multiple line である。




ある頂点間の特定のライン
（= Multiple Lines）

つまり、特定のライン（Multiple Lines）を消すということである。

1.Sum Values

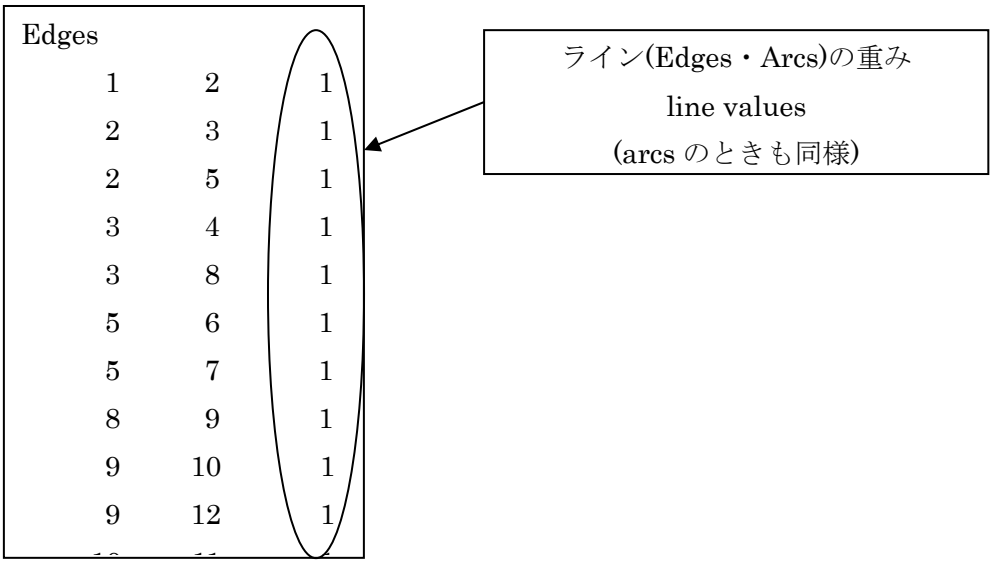
Values：edge や arc の重みの値のこと

すべての消去されたラインの値は、一致した 2 頂点間の消去されていないラインの値が加えられる。

重みを考慮したくなくなったとき

edge や arc の頂点間の線は消さずに、重みだけ消す。

重みはすべて 1 になる




図：line values

2.number of lines を押す

オリジナルのネットワークの 2 頂点間の値を新しいネットワーク 2 頂点間に一致させる。

Sum Values とどこが違うか？

3.Min value

すべての 2 頂点間の最小値を選ぶ

繰り返し行うと、少ない値から消えていく

4.Max value

すべての 2 頂点間の最大値を選ぶ

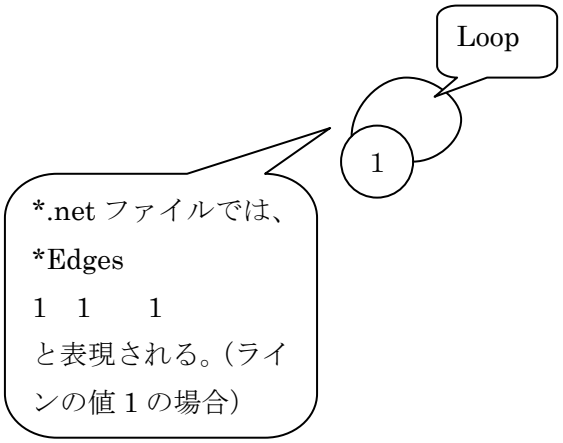
繰り返し行うと、少ない値から消えていく

5.Single Line

重みを 1 にする。

*loops

ループ(loop)：ある頂点から出て同じ頂点に入る。




図：Loop と*.net ファイルの構成

ループを消す。

ループのないネットワークにしたいときの処理。

*Lines with Value

1.lower than

特定の値より低い値（line values）のラインをすべて消去する。

2.higher than

特定の値より高い値（line values）のラインをすべて消去する。

3.within interval

特定の値間のラインをすべて消去する。

*All Arcs form each Vertex expect

各々の頂点が予測するすべての有向線。有向線ネットワークでなければならない。

1.K with Lowest Line Values

Sort=分類する、種類

ラインを出ラインの値（重み）に従った昇順で頂点間を分類する。選んだ数字の最小ラインを保つ。

K：指定した数字

入力手順

①How many lowest output arcs to keep,0 - keep

2.K with Highest Line Values

ラインを出ラインの値（重み）に従った降順で頂点間を分類する。選んだ数字の最大ラインを保つ

<Add

図：Add *Source and Sink

ネットワークの構成要素の“追加”を行う。

*Vertices

頂点の追加。同等処理は、*.net ファイルの Verteics のテーブルに、新しい頂点情報を書き込むことである。

*Source and Sink

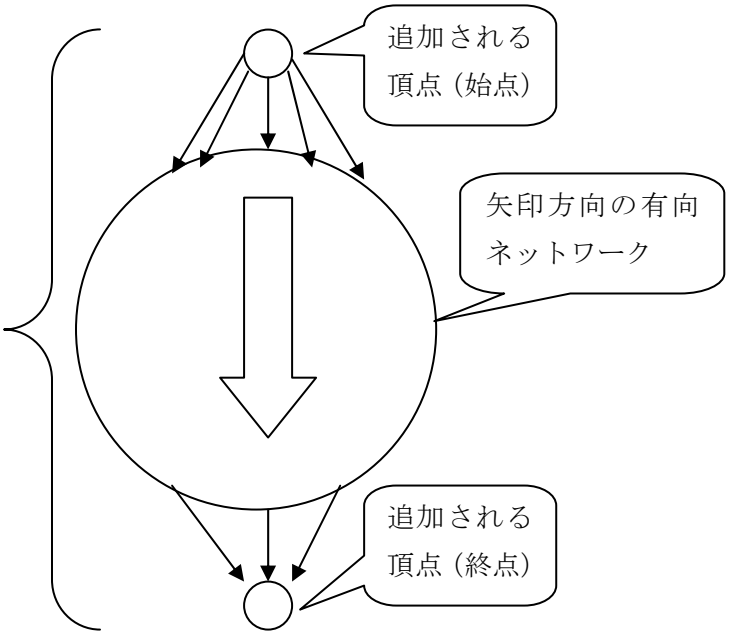
sink：流す

もし、非環状のネットワークなら（acyclic のネットワーク以外は受け付けないメニューである。）、固有のはじめと終わりの Vertex を追加する。下例では、始めが 101、終わりが 102 になる。）新しいネットワークは、２つの人工の２頂点を持つ。（頂点 101 と 102 のこと）

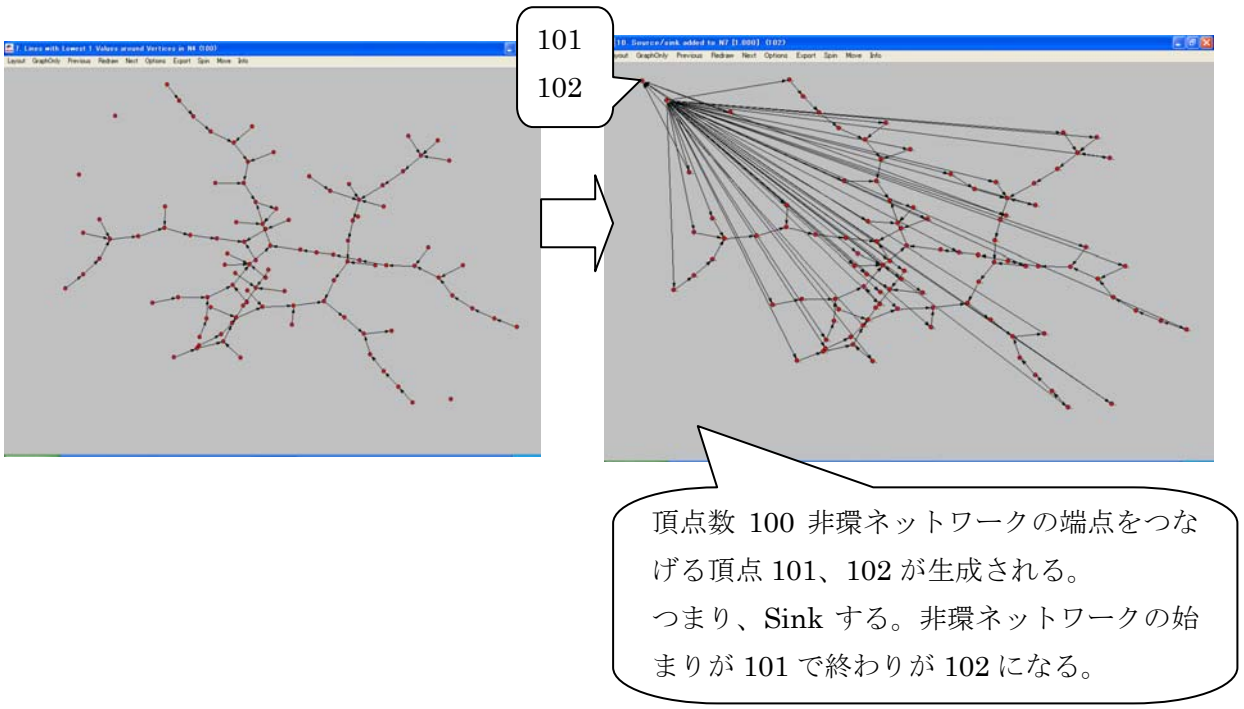
入力項目

①Values on lines added（加えられる頂点から追加されるラインの値）

この状態が、Sink（流す）状態である。




図：Add *Source and Sink




図：Add *Source and Sink の例

*Default Vertex Labels

現在の頂点ラベルからデフォルトのラベルへ再配置する。デフォルトのラベルとは、『v 頂点番号』である。

例：各ラベルが以下のように変更される。

*Vertices 3

1 “c”→”v1”

2 “n”→”v2”

3 “p”→”v3

*Vertex Labels from File

別の*.net ファイルを読み込み、そのラベルを、現在 pajek に読み込まれている*.net の頂点ラベルに書き換える。

*Line Labels as Line Values

ラインのラベルを、ラインの値（重み）と共に再配置する。

*Sibling Edges

Sibling（兄弟の一人）

Sibling Edges :

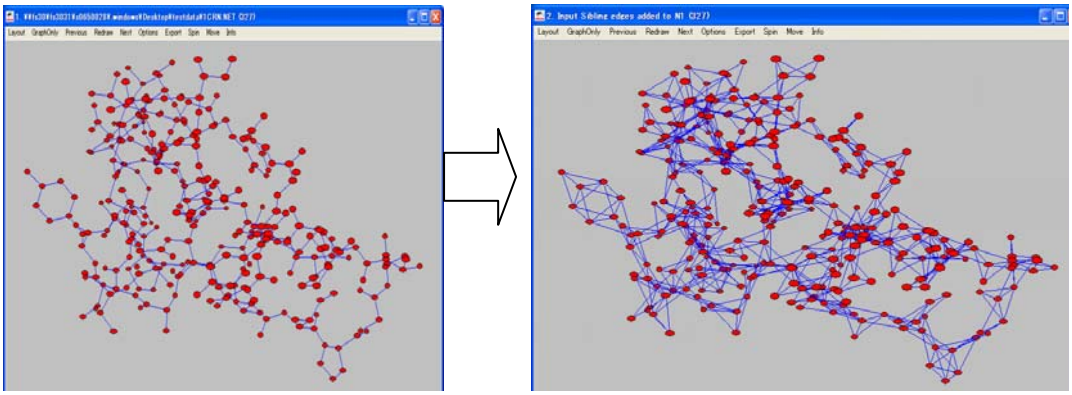
兄弟無向線の共通した頂点を加える。

1.input

先祖の有向線

2.Output

子孫の有向線



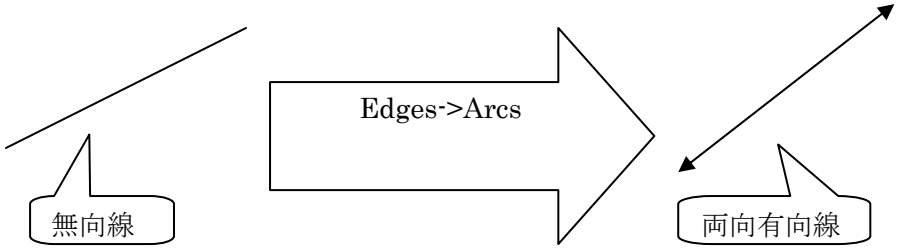
図：*Sibling Edges input の例

<Edges->Arcs

Convert all edges to arcs (in both directions) (make　directed network)

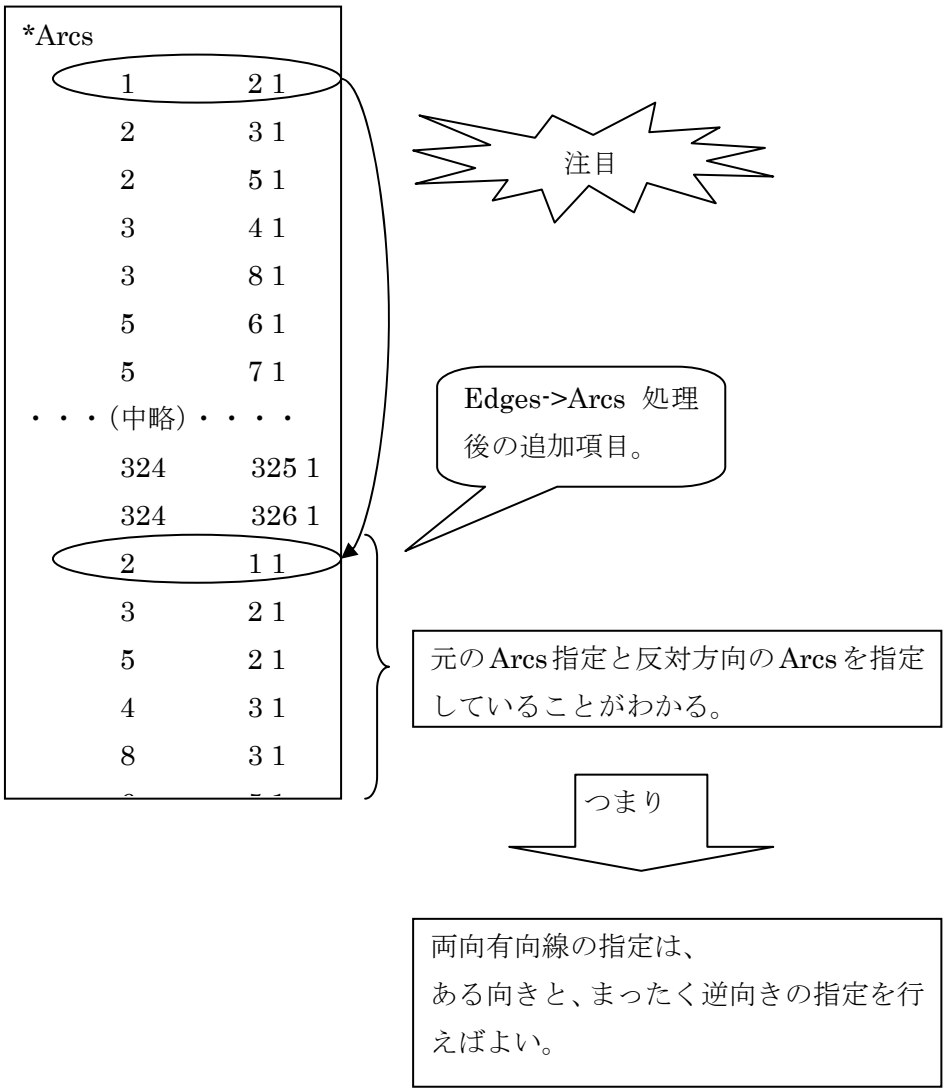
Both directions=両向

無向線を両向有向線にする。

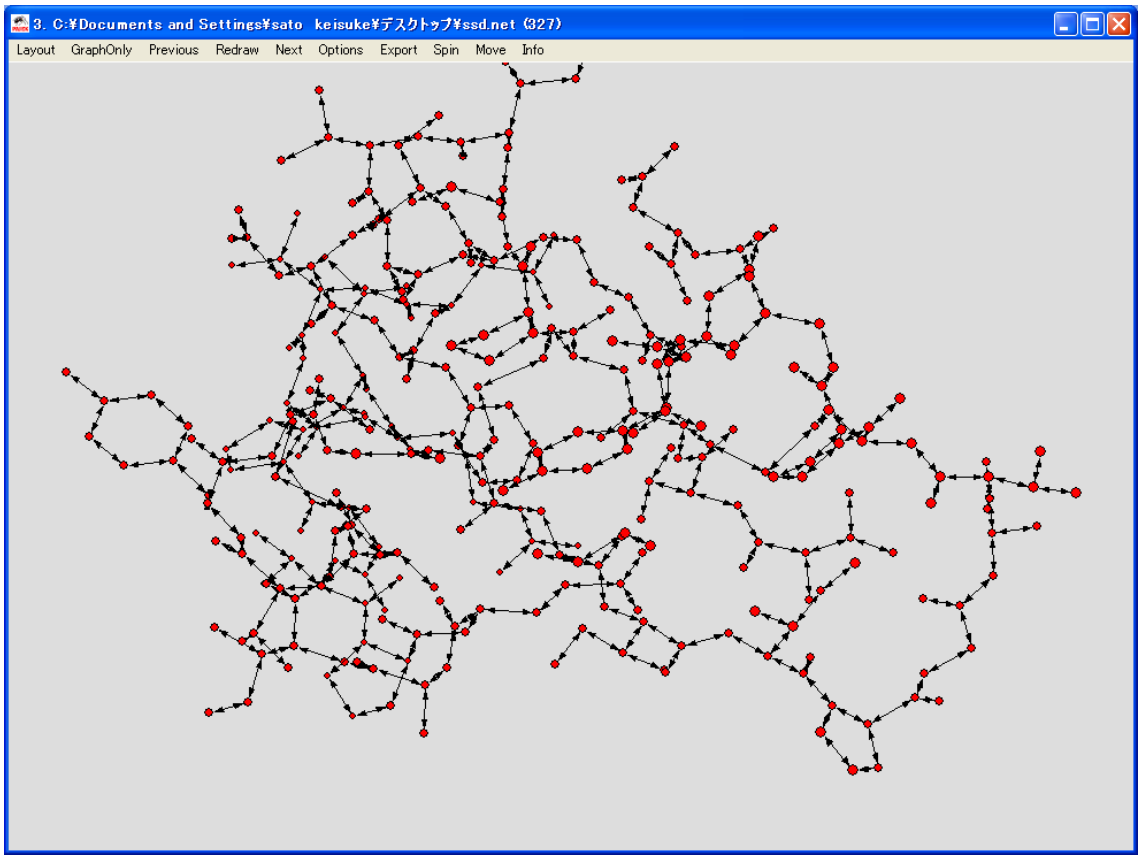



図：Edges->Arcs

.net ファイルには、以下のようになる。




図：両向有向線と*.net の関係

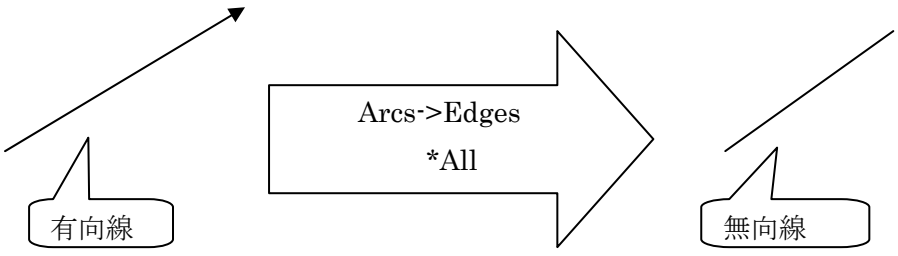



図：無向線から両向有向線に変換した例

<Arcs->Edges

*All

すべての有向線（Arcs）から無向線（Edges）へと変換する。つまり、有向ネットワークから無向ネットワークを作る



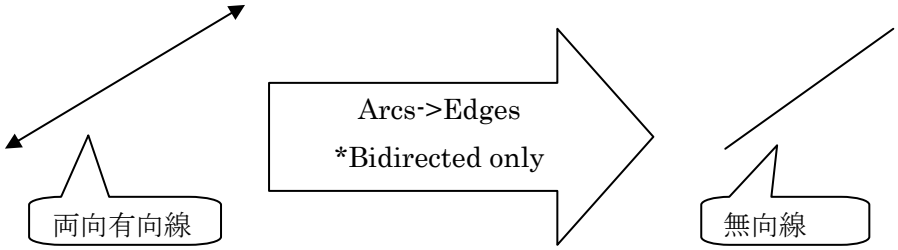

図：Arcs->Edges　*All

*Bidirected only

Bidirect：both direct

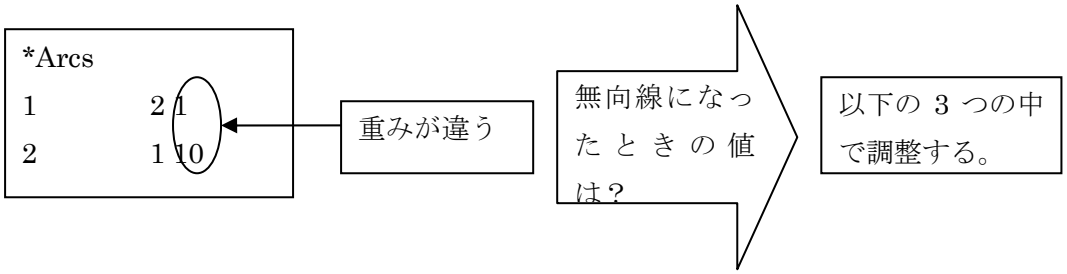
Bidirect：両向有向

両向有向線のみ無向線とする。




図：Arcs->Edges　*Bidirected only

両向有向線は、2 つの逆向きの有向線を重ね合わせた形であるが、以下のように 2 つの有向線のラインの値（重み）が違うこともある。




1.Sum Values

新しい無向線は、両向有向線の値の合計値である。

上例では、1-2 の無向線の値（重み）は 1+10=11 となる。

2.Min Value

新しい無向線は、両向有向線の値の最小値である。

上例では、1-2 の無向線の値（重み）1 となる。

3.Max Value

新しい無向線は、両向有向線の値の最小値である。

上例では、1-2 の無向線の値（重み）10 となる。

<Line Values

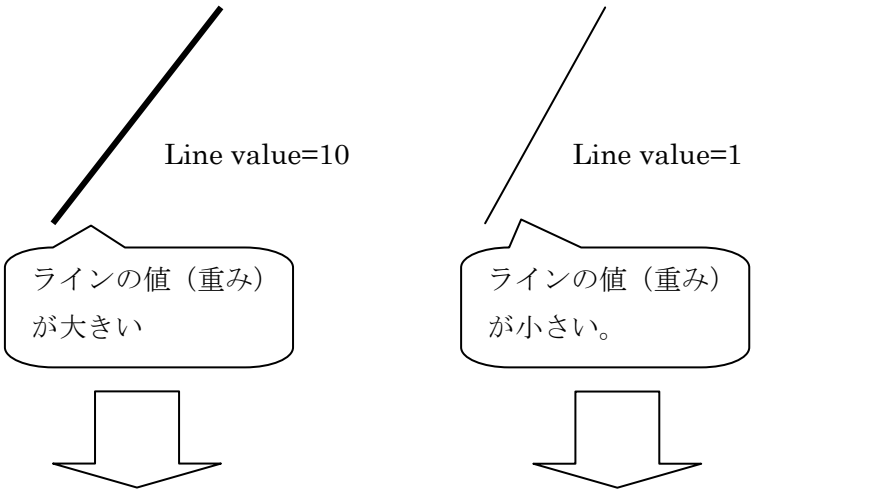
ラインの値（重み）を変化させる。

＃ラインの値（重み）について＃＃＃＃＃＃＃＃＃＃＃＃＃＃＃＃＃＃＃＃＃＃

ラインにも値（重み）がつくことがある。そのラインに伝播がされればされるほど多くの値（重み）がつくと考えてよい。例えば、道路でも交通量の激

しい道路と少ない道路が存在し、それらを同列に考えるよりも値（重み）を付けて考えた方がより実質的である。可視化についてもラインの値（重み）により、ラインが細くなったり太くなったり（あるいはより黒く、からより白くとの色分けがされたり）とラインの値（重み）を反映した可視化も存在する。

Draw window : Options>Lines>Different width あるいは Gray scale をチェック。（詳細は、これらのコマンドの説明しているページへ）




図：ラインの値（重み）と伝播

他にも、

ラインの値が大きい=伝播の確率が高い

という意味もある。

これらのように、ラインの値（重み）は重要な意味を持つ。

＃＃＃＃＃＃＃＃＃＃＃＃＃＃＃＃＃＃＃＃＃＃＃＃＃＃＃＃＃＃＃＃＃＃＃＃

*Recode

ラインの値を選ばれた間隔（頂点間）とこの方法でのラインの値（重み）の再設定に従い確率的に分配する。

入力項目

①Dividing values or #Clusters

$1 \le x_1 \le x_2 \ldots \le x_n \ldots \le 100$ の n を入力。つまり 1~100 のラインの値（重み）の n 個のクラスに分割する。

②Generate new network with line values recorded in this way?

Yse/no

Yes：この方法で作られたラインの値（重み）を新しいネットワーク（*.net）に反映する。

No：反映しない。

```
Lowest value of line:  1.0000
Highest value of line: 100.0000

       Line Values           Frequency      Freq%  CumFreq  CumFreq%
-------------------------------------------------------------------------
 (          ...    1.0000]        673    99.8516     673   99.8516
 (    1.0000 ...    3.0204]          0     0.0000     673   99.8516
 (    3.0204 ...    5.0408]          0     0.0000     673   99.8516
 (    5.0408 ...    7.0612]          0     0.0000     673   99.8516
 (    7.0612 ...    9.0816]          0     0.0000     673   99.8516
 (    9.0816 ...   11.1020]          0     0.0000     673   99.8516
 (   11.1020 ...   13.1225]          0     0.0000     673   99.8516
 (   13.1225 ...   15.1429]          0     0.0000     673   99.8516
 (   15.1429 ...   17.1633]          0     0.0000     673   99.8516
 (   17.1633 ...   19.1837]          0     0.0000     673   99.8516
 (   19.1837 ...   21.2041]          0     0.0000     673   99.8516
 (   21.2041 ...   23.2245]          0     0.0000     673   99.8516
 (   23.2245 ...   25.2449]          0     0.0000     673   99.8516
 (   25.2449 ...   27.2653]          0     0.0000     673   99.8516
 (   27.2653 ...   29.2857]          0     0.0000     673   99.8516
 (   29.2857 ...   31.3061]          0     0.0000     673   99.8516
 (   31.3061 ...   33.3265]          0     0.0000     673   99.8516
 (   33.3265 ...   35.3469]          0     0.0000     673   99.8516
 (   35.3469 ...   37.3674]          0     0.0000     673   99.8516
 (   37.3674 ...   39.3878]          0     0.0000     673   99.8516
 (   39.3878 ...   41.4082]          0     0.0000     673   99.8516
 (   41.4082 ...   43.4286]          0     0.0000     673   99.8516
 (   43.4286 ...   45.4490]          0     0.0000     673   99.8516
 (   45.4490 ...   47.4694]          0     0.0000     673   99.8516
```

| | | | | |
|---|---|---|---|---|
| (  49.4898 ...  51.5102] | 0 | 0.0000 | 673 | 99.8516 |
| (  51.5102 ...  53.5306] | 0 | 0.0000 | 673 | 99.8516 |
| (  53.5306 ...  55.5510] | 0 | 0.0000 | 673 | 99.8516 |
| (  55.5510 ...  57.5714] | 0 | 0.0000 | 673 | 99.8516 |
| (  57.5714 ...  59.5918] | 0 | 0.0000 | 673 | 99.8516 |
| (  59.5918 ...  61.6122] | 0 | 0.0000 | 673 | 99.8516 |
| (  61.6122 ...  63.6326] | 0 | 0.0000 | 673 | 99.8516 |
| (  63.6326 ...  65.6531] | 0 | 0.0000 | 673 | 99.8516 |
| (  65.6531 ...  67.6735] | 0 | 0.0000 | 673 | 99.8516 |
| (  67.6735 ...  69.6939] | 0 | 0.0000 | 673 | 99.8516 |
| (  69.6939 ...  71.7143] | 0 | 0.0000 | 673 | 99.8516 |
| (  71.7143 ...  73.7347] | 0 | 0.0000 | 673 | 99.8516 |
| (  73.7347 ...  75.7551] | 0 | 0.0000 | 673 | 99.8516 |
| (  75.7551 ...  77.7755] | 0 | 0.0000 | 673 | 99.8516 |
| (  77.7755 ...  79.7959] | 0 | 0.0000 | 673 | 99.8516 |
| (  79.7959 ...  81.8163] | 0 | 0.0000 | 673 | 99.8516 |
| (  81.8163 ...  83.8367] | 0 | 0.0000 | 673 | 99.8516 |
| (  83.8367 ...  85.8571] | 0 | 0.0000 | 673 | 99.8516 |
| (  85.8571 ...  87.8775] | 0 | 0.0000 | 673 | 99.8516 |
| (  87.8775 ...  89.8980] | 0 | 0.0000 | 673 | 99.8516 |
| (  89.8980 ...  91.9184] | 0 | 0.0000 | 673 | 99.8516 |
| (  91.9184 ...  93.9388] | 0 | 0.0000 | 673 | 99.8516 |
| (  93.9388 ...  95.9592] | 0 | 0.0000 | 673 | 99.8516 |
| (  95.9592 ...  97.9796] | 0 | 0.0000 | 673 | 99.8516 |
| (  97.9796 ... 100.0000] | 1 | 0.1484 | 674 | 100.0000 |
| Total | 674 | 100.0000 | | |

図：*Recode によって出力される結果 Report の例

*Multiply by

継続的にラインの値を増やす

入力項目

①Multiply line values by

継続的に増やしていくラインの値（重み）の数

*Add Constant

継続的にラインの値（重み）を加えていく。

入力項目

①Constant to add to line values

*Absolute

Absolute line values

*Absolute+Sqrt

Square root of line values

Sqrt：Square root

*Truncate

ラインの値（重み）を切り捨てる。

*Exp

Exponent line values

*ln

Natural logarithm of line values

*Power

Selected power of line values

Power：

*Normalize

ラインの値（重み）を標準化する。他のネットワークのラインの値（重み）と比較しやすくなる。

1.Sum

ラインの値の合計値を１とし、その前提の下で、すべてのラインの値を再計算する。

2.Max

最大値のラインの最大値を１とし、その前提の下で、すべてのラインを再計算する。

<Reduction

以下の対象を縮小する。

*Degree

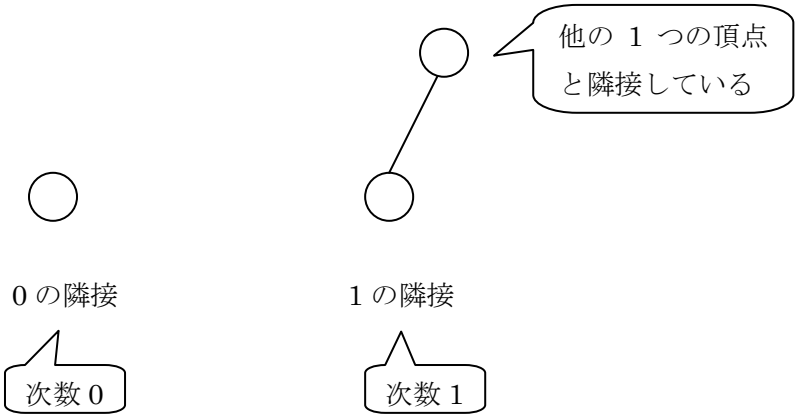
選んだ値よりも小さいすべての頂点を（縮小的に）消す。

1. Input
2. Output
3. All degree

操作は、選ばれたネットワークを限定的にすることが出来る。

*Hierarchical

0 か 1 の隣接（Neighbor)をもつすべての頂点を（縮小的に）消す。




図：0 の隣接と 1 の隣接

結果は、簡素なネットワークと消された頂点の階層となる。

オリジナルのネットワークは最後に元通り(restore)にする。

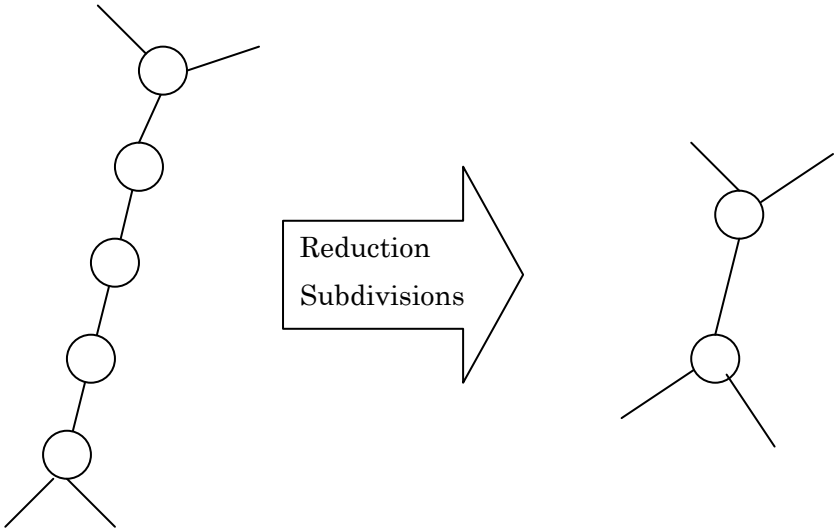
*Subdivisions

Subdivisions：副分割、副分配、副分類

正確に 2 隣接をもつすべての頂点を消す。（2 つのラインと共に）

2 隣接とは、上記の 0 や 1 の隣接と同じく 2 つの頂点とつながった、次数 2 の頂点のことである。

代わりにそれら 2 つの近接に直接、ラインを加える。



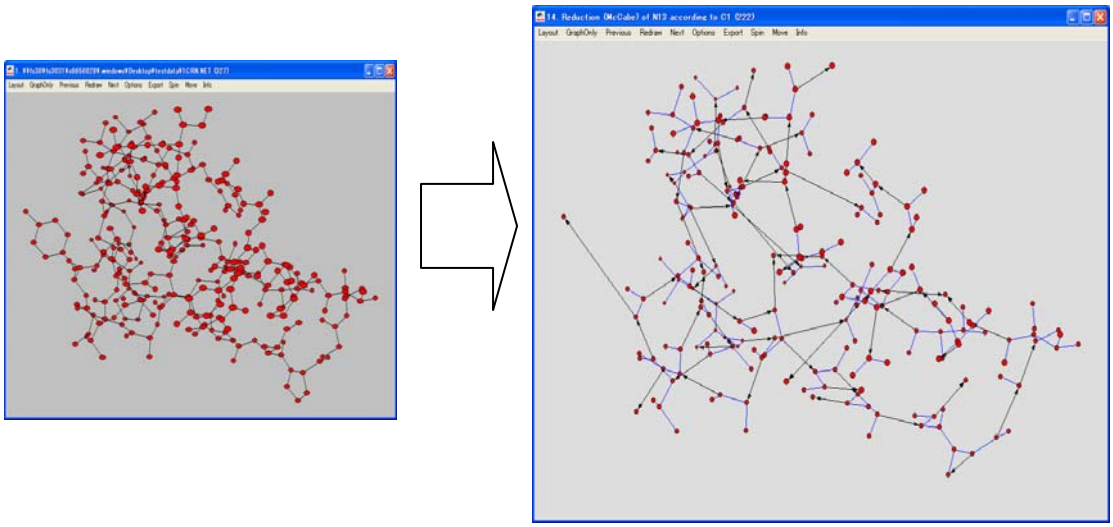

図：Reduction *Subdivisions

結果、可視化するのにちょうどよい簡素なネットワークになる。

だだ、オリジナルのネットワークは最後に元通りにならない。

*Design(McCabe)

すべてのネットワークの部分を McCabe（グラフを描く際のプログラム）に従い縮小する。同サイズの Partition が必要。



図：All degree の partition に従った Mc Cabe 描写

有向線が追加され、頂点数も 327 から 222 へと減少する。

*Structural(flow graph)

<Generate Time

読み込まれた*.tim ファイルに従うステップ時の状態を*.net ファイルへ。

特定の時間かその間を取る。

－：頂点間の限界(Interval of limit)より少ないか多いかで分ける

‘：頂点間を分ける

＊：無限大

例：Vertices 3

1 “a” [5-10,12-14]

2 “b” [1-3,7]

3 “e” [4-*]

Edges

1 2 1 [7]

1 3 1[6-8]

頂点 a は、5~10 と 12~14 のタイムで働き（Active）頂点 b は、1~3、7 のタイムで働き、頂点 e はタイム 4 から働く。ライン 1 から 2 はタイム 7 で働き、ライン 1 から 3 はタイム 4 から働く。（3 番目の数字は重

み)

指定されたタイムのときのみ、現在のネットワーク（temporal network）に表示される。

*All

すべての特定の時間のネットワークを生成する。

*Only Different

変更点が存在したときのみ、ネットワークを生成する。

*Interval

選択された Interval（時間）を与えたラインや頂点のあるネットワークを生成する。

<1-mode to 2-mode

1-mode から 2mode ネットワークを生成する。

<2-mode to 1-mode

2-mode から 1-mode ネットワークを生成する。

*Rows

結果は、ネットワーク間の行要素（アクター)間の関係を入れたネットワークとなる。

ラインの値（重み）が、2 アクター共通のイベントの数を数える。

*Columns

結果は、列要素（イベント）間の関係を入れたネットワークとなる。

ラインの値（重み）が、互いのイベント間を構成するアクターの数を数える。

*Include loops

それぞれのアクターのイベントから合計値（互いのイベントのアクターの数の合計値）にループの値を入れたいとき、このメニューを使用する。

*Multiple lines

頂点が存在する間に Multiple lines 存在する箇所に、値のない 1-mode ネットワークを生成する。

*Normalize 1-mode

2-mode から 1-mode になった 1-mode ネットワークを標準化する。

オプションには「Include loops のチェック」と「Multiple lines のチェックをしない」がある。

$$\text{Geo}_{ij} = \frac{a_{ij}}{\sqrt{a_{ii}a_{jj}}}$$

$$\text{Input}_{ij} = \frac{a_{ij}}{a_{jj}}$$

$$\text{Output}_{ij} = \frac{a_{ij}}{a_{ii}}$$

$$\text{Min}_{ij} = \frac{a_{ij}}{\min(a_{ii}, a_{jj})}$$

$$\text{Max}_{ij} = \frac{a_{ij}}{\max(a_{ii}, a_{jj})}$$

$$\text{MinDir}_{ij} = \begin{cases} \frac{a_{ij}}{a_{ji}} & a_{ii} \le a_{jj} \\ 0 & \text{otherwise} \end{cases}$$

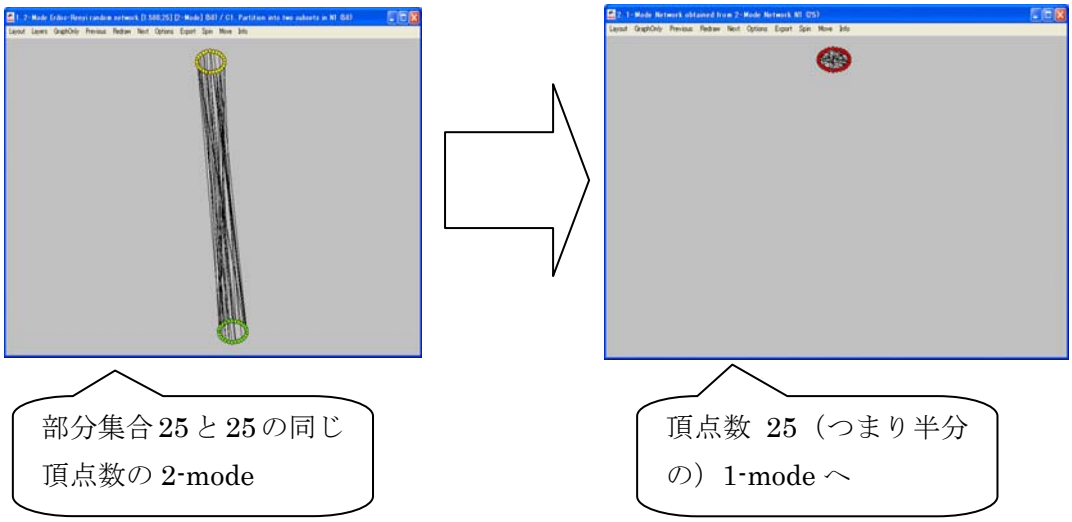
$$\text{MaxDir}_{ij} = \begin{cases} \frac{a_{ij}}{a_{jj}} & a_{ii} \le a_{jj} \\ 0 & \text{otherwise} \end{cases}$$

2-mode から 1-mode になった 1-mode ネットワークは、見やすくない。（ほとんどまばらでない。）そこで、以下のようにまばらにすることも出来る。

Net>Transform>Remove>Lines with value>lower than

*Rows=Cols

同じ頂点数の部分集合（First subset と second subset）をもつ 2-mode を 1-mode に変換する。当然、2 つの部分集合は同じ頂点数の 2-mode ネットワークでないと処理を受け付けない。




図：2-mode to 1-mode *Rows=Cols

<Multiple relation

*Extract Relation(s)

選択された multiple relations network から、選択された関係リスト（List of relations）かそれを引き出す。

入力項目

①Relation Number(s)の選択

*Canonical Numbering

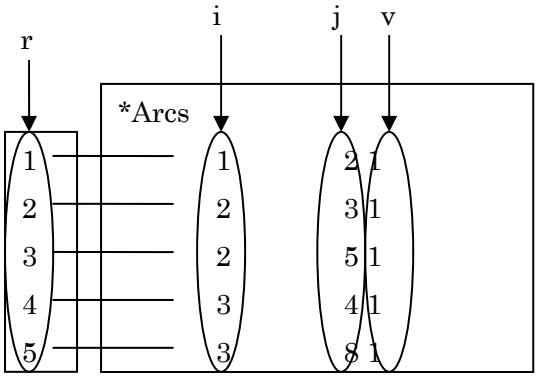
1,2,...と順次的に関係の数を数え上げる。

*Generate 3-Mode Network

1-mode か 2-mode の相互関係 (multirelational network)ネットワークから 3-mode ネットワークを生成する。

Multirelational network の互いのライン r ： i j v

(i から j へ行く、値 v の relational number r)




図：*.net ファイルと r ： i j v の関係

N：1-mode(first mode)の濃度

M：2-mode(second mode)の濃度

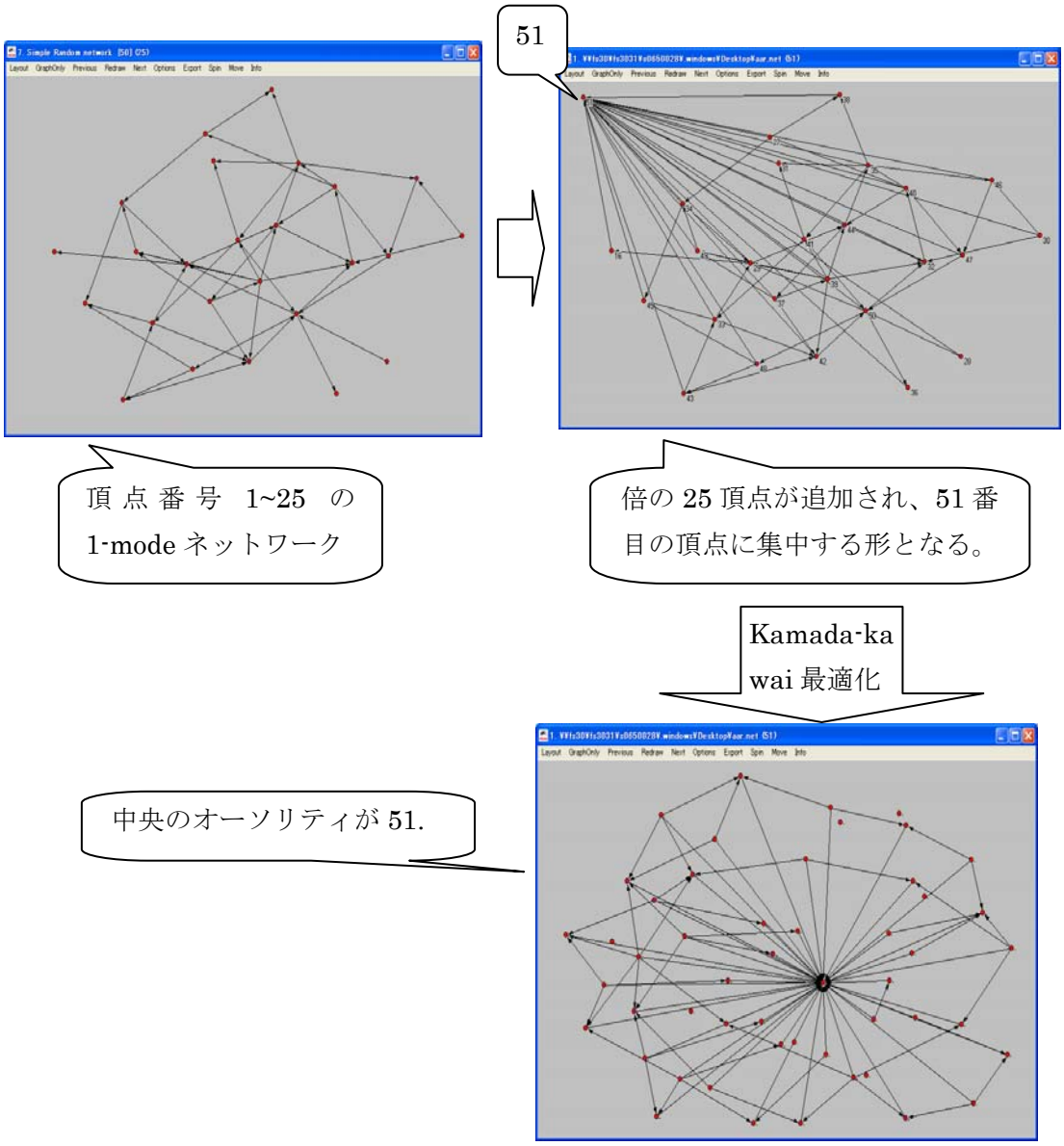
・1-mode network

```
i N+j v
i 2N+r v
N+j 2N+r v
```

・2-mode networks:

```
i j v
i N+M+r v
j N+M+r v
```

以上の関係式から、新しいラインが創出される。




図：Multiple relation　*Generate 3-Mode Network

*Line Values->Relation Numbers

ラインの値（重み）と同じ関係番号を保存する。（ラインの値は、完全な小数点以下切捨ての値：実数の整数化）つまり、ラインの値（重み）を関係番号にする。（Line Values→Relation Numbers）

*Relation Numbers->Line Values

関係番号と同じラインの値（重み）を保存する。つまり、関係番号をラインの値（重み）にする。（Relation Numbers→Line Values）

*Change Relation Number/Label

関係番号をラベルと一致した値の新しい関係番号に置き換える。

<Sort Lines-

ラインの並べ替えを行う。

*Neighbors around Vertices

近隣の回りのラインの並び替えを行う。

他の最終頂点（End-vertex）に従った昇順の中で、互いの頂点の並び替たラインに結合された、互いの頂点の並び替たラインに向かって、近隣の周りの頂点のラインを並び替える。

*Line Values

ラインをラインの値（重み）に従い、昇順か降順に並び替える。

1．Ascending

昇順に並び替える。

2．Descending

降順に並び替える。

>Random Network

選択された項目に従い、ランダムネットワークを生成する

<Total No. of arcs

vertice の数と arcs の数を聞き、有向線ランダムネットワークを生成する。

<Vertices Output Degree

ネットワークの定義数（=頂点数）と与えられた距離の互いの頂点の出次数を反映した有向線ランダムネットワークを生成する。

入力項目

①頂点数

②Min number of arcs from vertex

③Max number of arcs from vertex

④Multiplu line を除くか？

<Erdos-Renyi

Erdos-Renyi のモデルに従い 2-mode ネットワークを生成する。

すべての潜在的なラインに含まれている顕在化する確率 $p$

確率 $p$ の代わりにもっと直感的なものに、以下の次数の平均が使われる。

$$\overline{\deg} = \frac{1}{n}\sum_{n \in V} \deg(V)$$

補足：グラフ $g = (V, L)$、$V$ は頂点の集合、$A$ は有向線の集合、$E$ は無向線の集合、$L$ はラインの集合で $L = E \cup A$。$n = card(V)$（card は要素）$m = card(L)$

それは、$p = \frac{m}{m_{\max}}$を保つ。またシンプルなグラフ向けである。

または、$\overline{\deg} = \frac{2m}{n}$

*Undirected

無向ネットワークを生成する。頂点数と平均距離を聞いてくる。

平均距離：ある頂点がどのくらいの範囲の頂点とリンクできるかという距離の平均。平均距離が大きいほど完全グラフに近くなる。

平均距離の最大値=完全グラフ

1. General

普通の形のネットワーク。

2. Bipartite

2 部からなるネットワーク。

3. 2-Mode

2-mode。

*Directed

1. General

普通の形のネットワーク。

2. Acyclic

Acyclic のネットワーク。

3. Bipartite

4. Acyclic Bipartite

<Scale Free

スケールフリーの無向、有向、無環ネットワークを生成する。

エンドポイントのエッジは以下の確率に従った値にランダムにすべての頂点間が選択される。

$$\Pr(v) = \alpha \frac{in\deg(v)}{|E|} + \beta \frac{out\deg(v)}{|E|} + \gamma \frac{1}{|V|}$$

そのとき、

$\alpha + \beta + \gamma = 1$

さらに、

$\sum_{v \in V} \Pr(v) = 1$

*Undirected

入力項目

①頂点数

②ライン数

③平均距離

④Initial Erdos-Renyi graph:頂点数

⑤Initial probability of lines(0~1)

⑥Alpha（α）

*Directed

①頂点数

②ライン数

③平均距離

④Initial Erdos-Renyi graph:頂点数

⑤Initial probability of lines(0~1)

⑥Alpha（α）

⑦Beta(β)

*Acyclic

①頂点数

②ライン数

③平均距離

④Initial Erdos-Renyi graph:頂点数

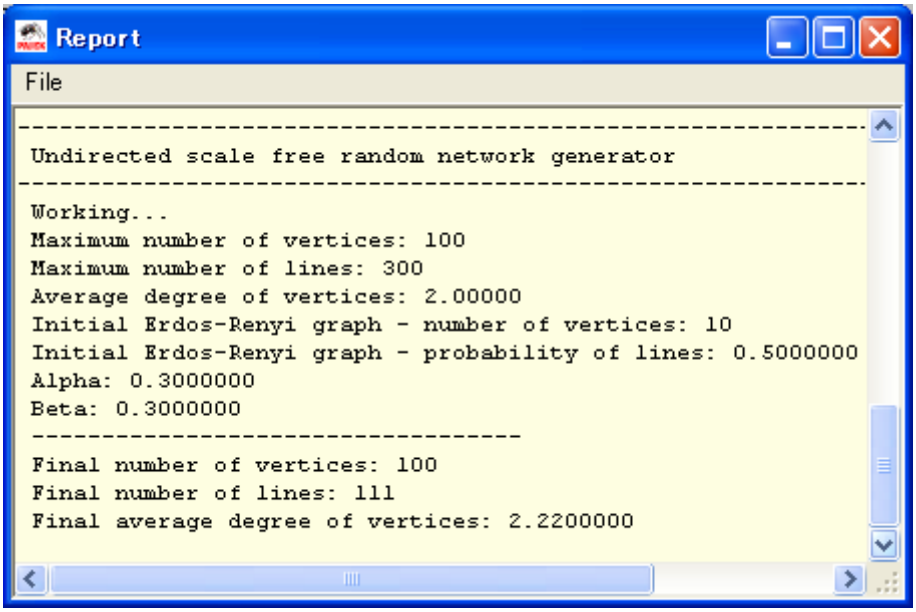
⑤Initial probability of lines(0~1)

⑥Alpha（α）

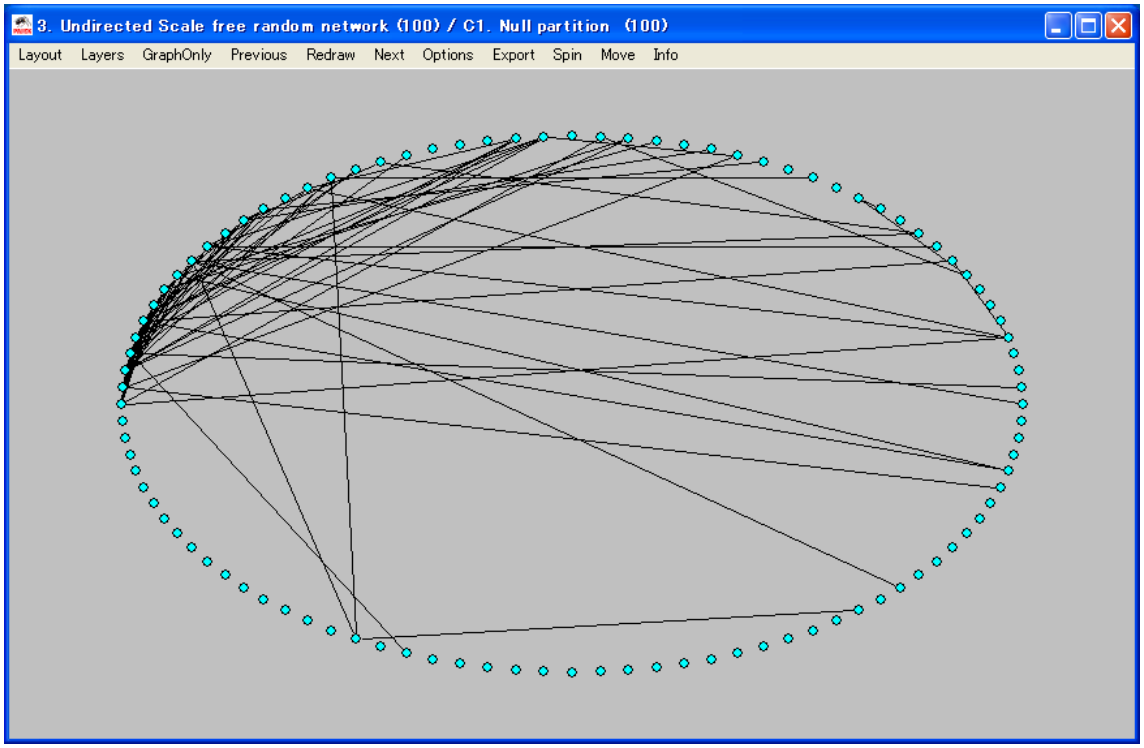
*Adding

・free

「選択する」か「解除する」かのチェック項目。




図：スケールフリーの設定 Report




図：以上の設定によって可視化されたグラフ

<Extended Model

アルバート・バラバシの Extended Model を作る。

入力項目

①頂点数

②孤立頂点の数

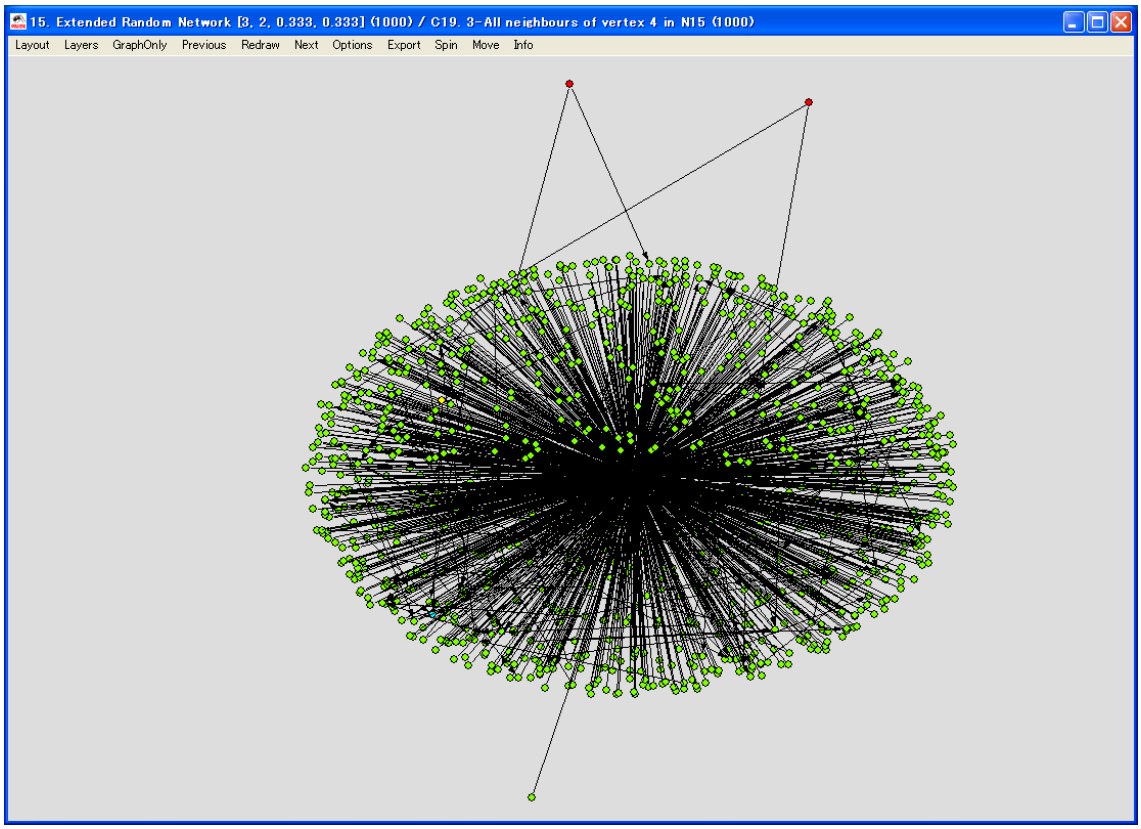
$m0 \geq 1$ の m を入力(1 以上 10 個単位)

③How many lines to add in each step

④新しいラインを追加する確率(0~1)

⑤リワイヤ（ラインの張替え）する確率

⑥multiple line を除去するか？




図：Extended Random Network [3, 2, 0.333, 0.333] (1000)の頂点 4 からのパス数で Partition したもの（Net>Partition>k-neighbor の ALL)。1000 頂点のほとんどが 2 パスでいける（2 パスの近隣）黄緑色であることがわかる。

>Partitions

ネットワークを Partition（クラス分け）する。出力は、*.clu ファイルがメインスクリーン Partitions ボックスに出力される。

<Degree

頂点の次数(Degree)で Partition する。このように、頂点の次数による Partition を便宜上「Degree Partition(次数によるパーティション)」と名づける。

Partitions と Vector のボックスにファイルが生成される。

Partition と Vector には、「Degree Partition」されたデータが入るが、

Partition はクラスごとの色分け、Vector は次数の多くなるに従い、頂点が大きくなるネットワーク可視化を担当する。

*Input

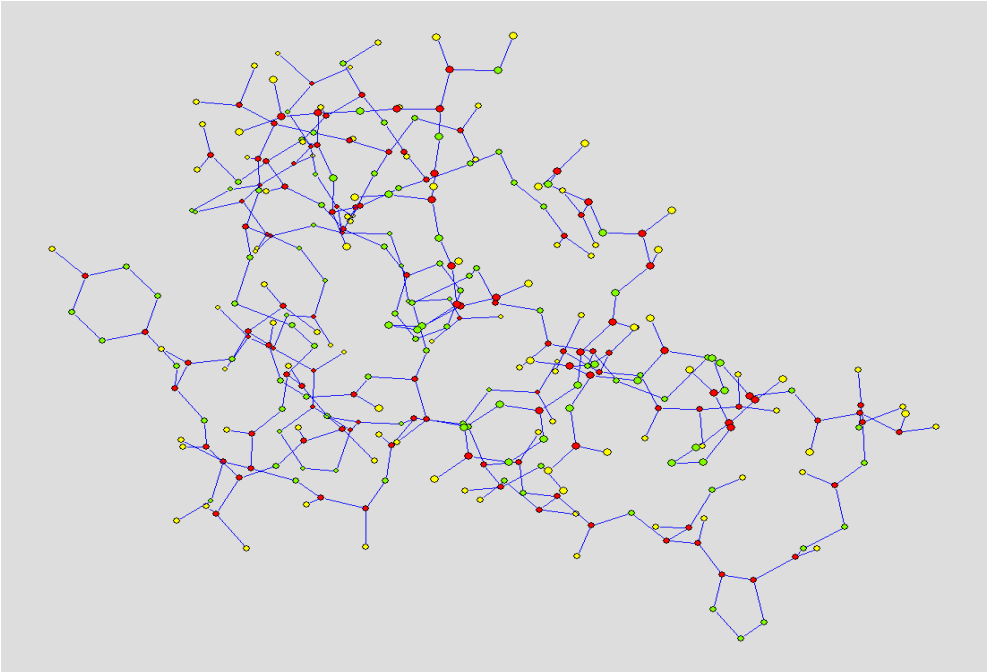
入次数による Degree Partition を生成する。

*Output

出次数による Degree Partition を生成する。

*All

次数（入次数、出次数の区別なく）による Degree Partition を生成する。

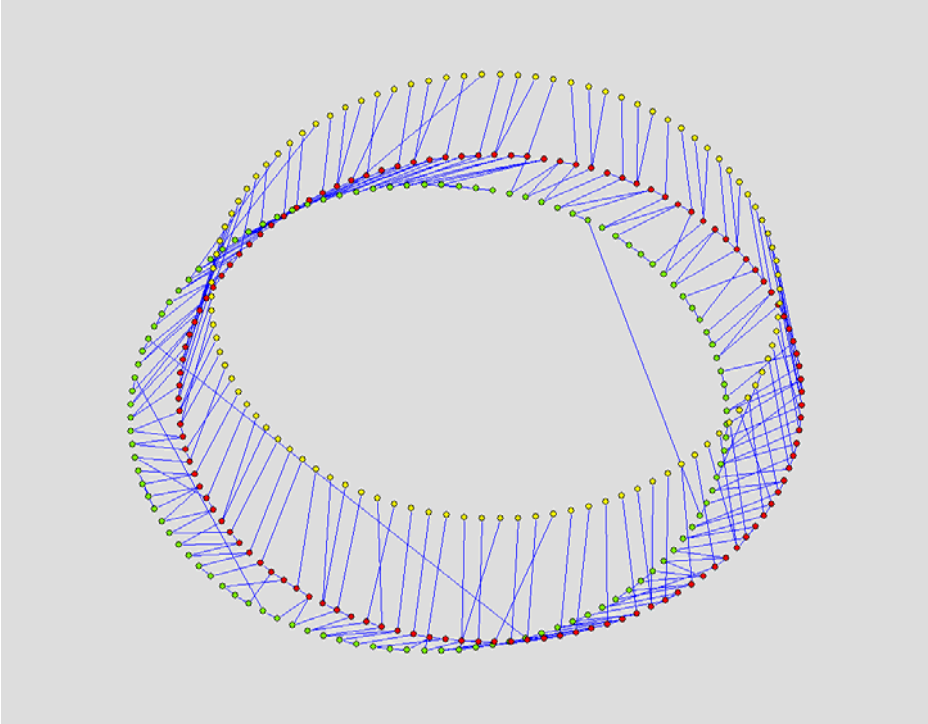


図：初期可視化状態の Degree Partition（ALL）の可視化例

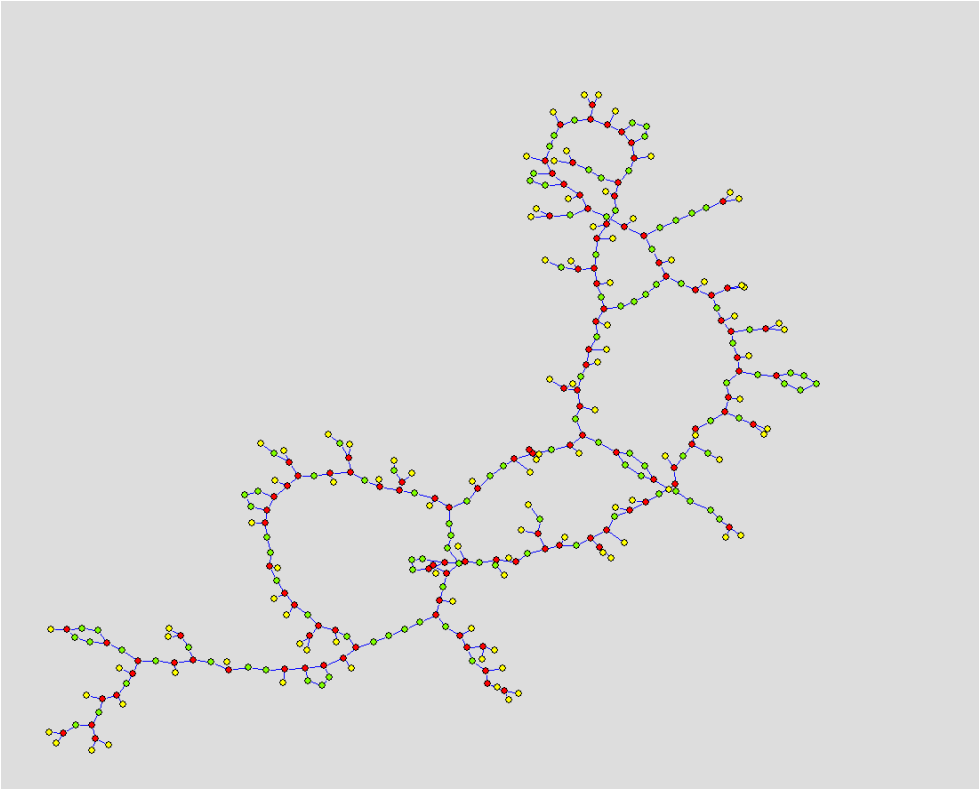
色毎にパーティションされる。

次数と色の対応は、Partition の色の対応設定に依存する。

Draw window の Options>Colors>Partition Colors でみることが出来る。

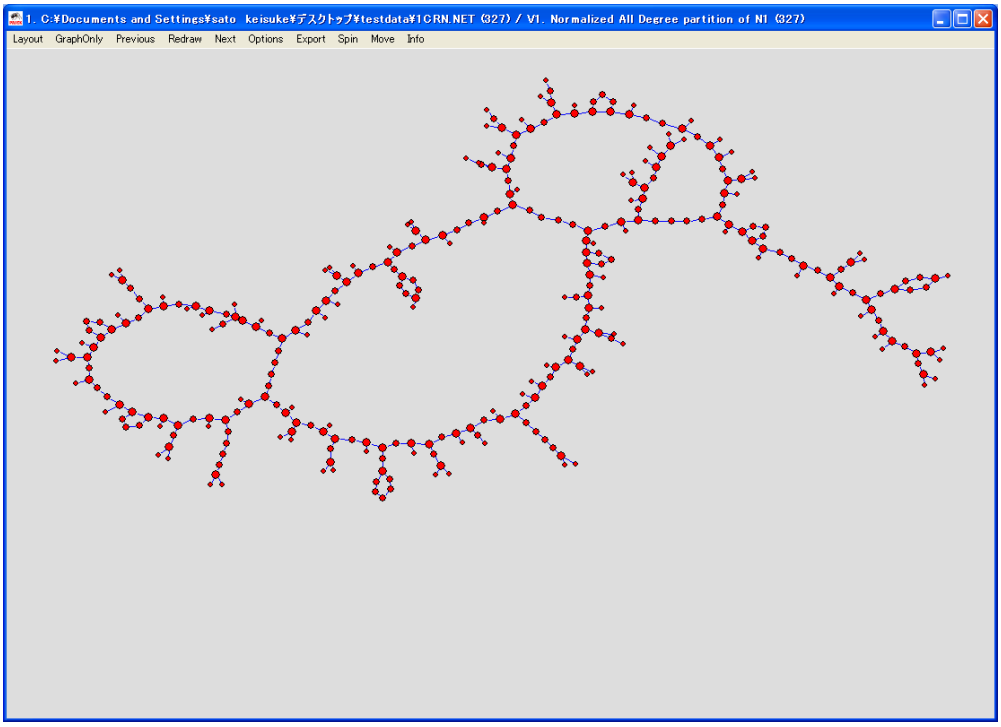


図：Circle－Partition による Degree Partition（ALL）の可視化例
次数でクラスわけされた Petition ごとに、円状に表示
クラスターを飛びえたリンクは、次数の違うもの同士のリンクとなる。



図：kamada-kawai 最適化による Degree Partition（ALL）の可視化例
次数ごとに色分けされていることが確認できる。

*.clu ファイルと共に、*.vec ファイルも生成される。これは、次数が大きいほど Vector が大きい値をとるファイルである。




図：Degree Partition（ALL）で生成された Vector ファイルによる可視化例

<Domain

Domain：領域

互いの頂点のため、入力、出力及びすべての隣接に従ったそれの領域を計算する。結果は、

１．領域のサイズの Partition－頂点に到達できる数

２．標準化された領域のサイズの Vector－標準化は頂点の合計数を 1 にする。

３．領域の~から~ までの平均距離の Vector

の 1 つの Partiton と 2 つの Vector である。

近接誘因指標（Proximity Prestige index）は分けている標準化された平均距離の領域のサイズによって計算できる。

入力項目

①Maximal distance. distance (0-No limit)

0 から限界数（頂点数）までの最大距離を入力。

*Input

*Output

*All

<Core

Core（コア）には以下の 3 つが存在する。

1．K-Core：最大 k 個以上の次数を持つ最大サブグラフ

2．M-Core：Island のこと（Pajek では M-Core といわず、Island という。）
最大 m 以上のラインの値（重み）を持つ最大サブグラフ

3. P-Core：

K-Core は、処理的にいえば次数による Partition である。では、次数による Paritition（Degree Paritition）とはどこが違うだろうか？以下で明確にする。

Degree Paritition

| 次数 | クラス番号 |
|---|---|
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | 4 |
| 5 | 5 |
| 6 | 6 |
| 7 | 7 |
| 8 | 8 |

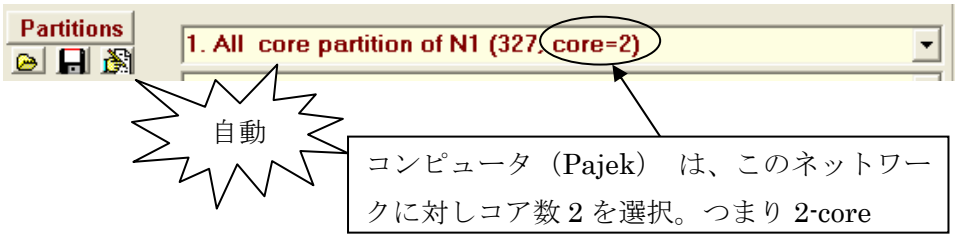
k-core

| 次数 | クラス番号 |
|---|---|
| 1 | 1 |
| 2 | 1 |
| 3 | 1 |
| 4 | 2 |
| 5 | 2 |
| 6 | 2 |
| 7 | 2 |
| 8 | 2 |

クラス番号に明確な線引きがされる。この場合、次数 3 と 4 の間にある。定義より、これは 4-core。

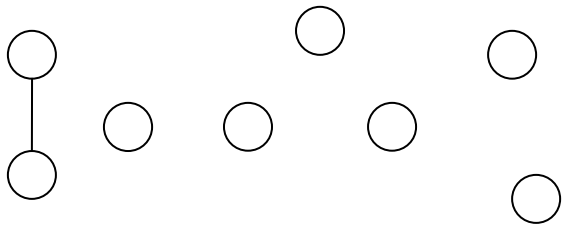
図：Degree Partition と k-core の違い

以下のように考えるとわかる。つまり、K-Core は水面下に入るものと水面上に出るものという風に考えることが出来る。上図では、次数 3 以下が水面下に（クラス番号 1）次数 4 以上が水面上に（クラス番号 2）出ると考えることが出来る。この水面の上か下かの基準 k は、コンピュータによって自動で決定されえるため、User は設定できない。

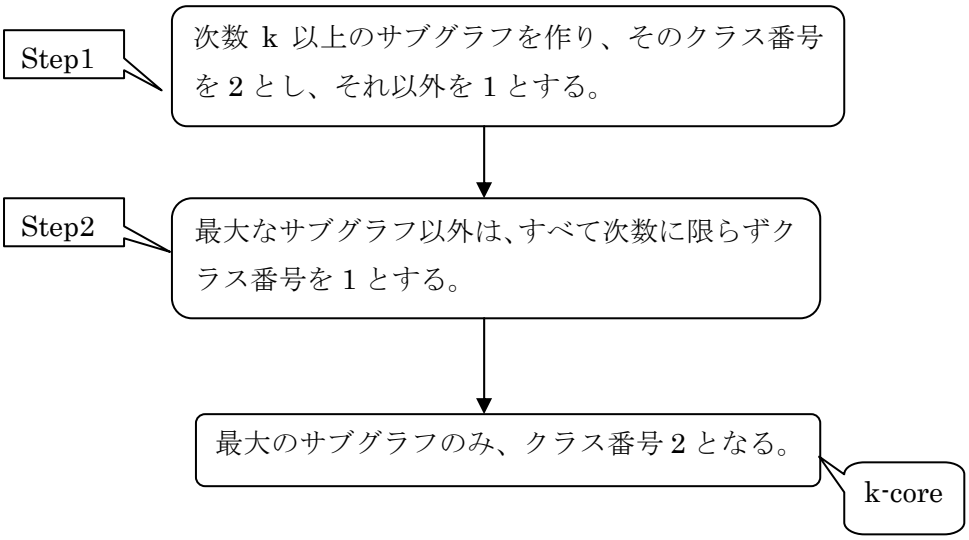



図：自動的に決定される k-core

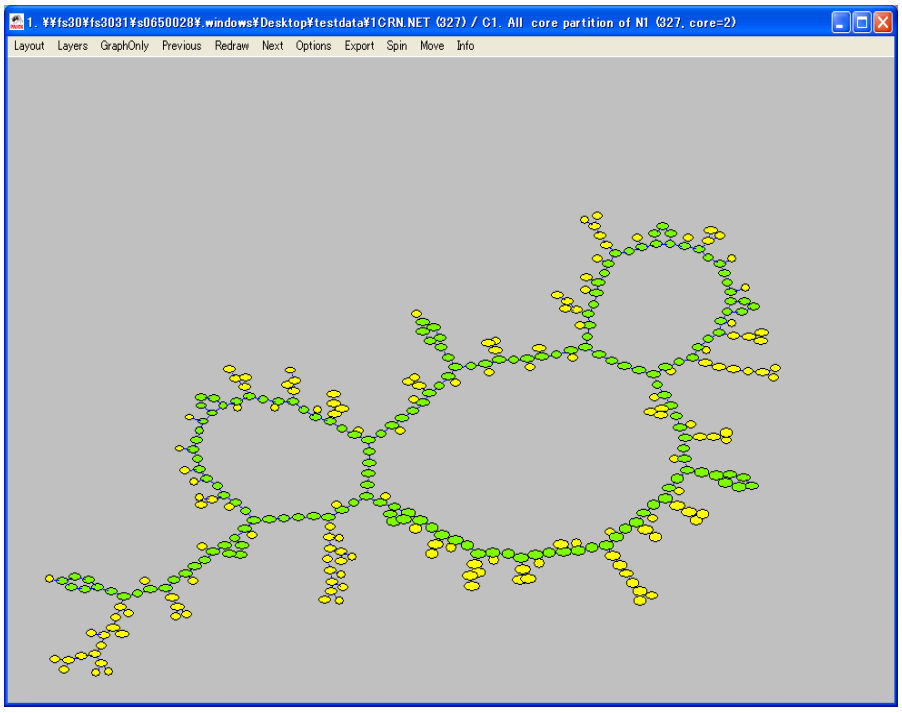
さらに、最大サブグラフという制限があり、最大でないサブグラフの次数 k 以上のサブグラフはすべて対象から除かれる。（クラス番号が 1 となる）



つまり、以下のような処理過程となる。




図：k-core の形成過程




図：K-Core(ALL)による Partition 可視化（kamada－kawai 最適化）2-core

黄色：クラス番号 1

黄緑色：クラス番号 2

このネットワークの 2-core は黄緑色全体のサブグラフである。

（次数 2 のサブグラフでも、クラス番号 2 となっているところが散在するが、これは、Step2 が存在しているためである。）

ちなみに、特定のクラス番号とそれ以外を 0 と 1 で区別するコマンドがある。

Degree partition をつくり特定の次数のみを 1 で表すことも出来る。

Partition>Binarize

また、特定のクラス番号を別のクラス番号へと変換するコマンドもある。

Partition>Fuse cluster

Core と Island の違いは、

Core：次数に注目

Island：ラインの値（重み）に注目

である。

*Input

線が頂点の中に来るもの（入次数）

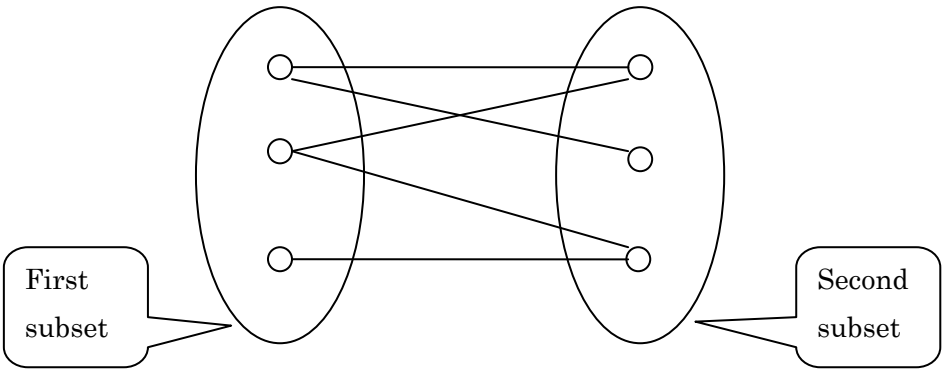
*Output

線が頂点から出て行くもの（出次数）

*All

頂点の隣接数（入次数も出次数も関係ないただの次数）

*2-Mode




図：2 部グラフ(2－Mode)の例

First subset の最小次数の全頂点の集合：k1

Second subset の最小次数の全頂点の集合：k2

処理手順

①k1 と k2 を作る

②k1 と k2 の頂点には、Partition 番号 0 が割り振られる

③k1 と k2 以外の頂点には Partition 番号 1 が割り振られる

④Partition 番号 1 のサブネットワークが 2-Mode の Core である

*2-Mode Review

k1、k2Rows Cols Comp に従い処理

Comp：comprehensive

Comprehensive：包括的な

*2-Mode Border

Compute only border values of k1 and k2 for a given 2-mode network

与えられた 2-mode の k1、k2 を唯一のボーダーの値とした計算

<Valued Core

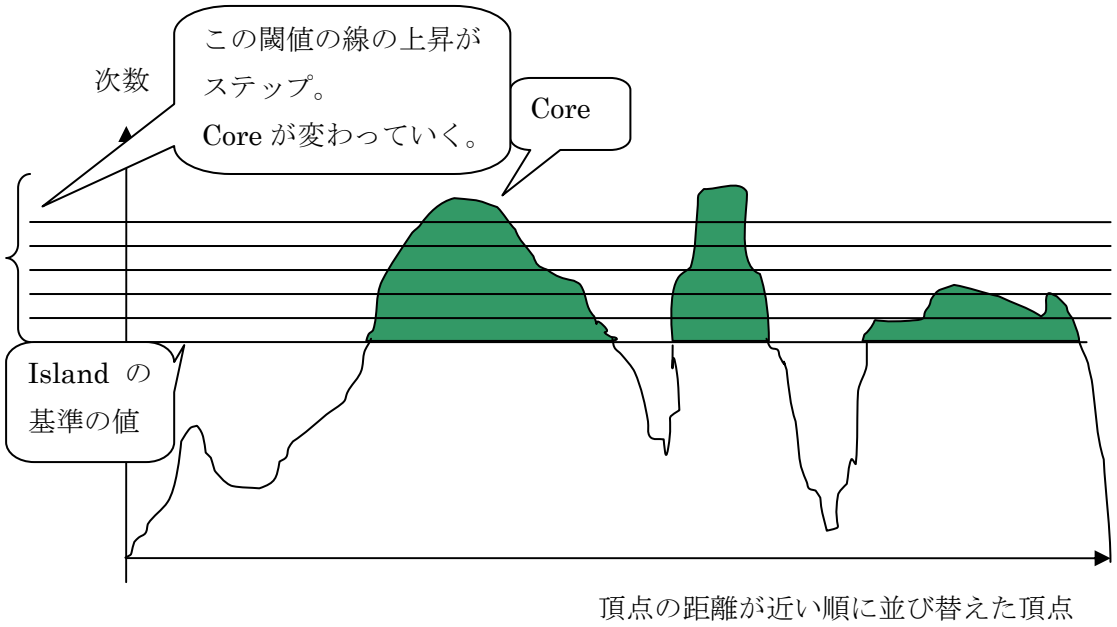
*First Threshold and Step

はじめに閾値を選び、閾値を上げていきながらステップする

聞いてくる値

①First threshold：

②Step(>0)：上昇させていく閾値のステップ




図：Core 生成の原理

1. Input

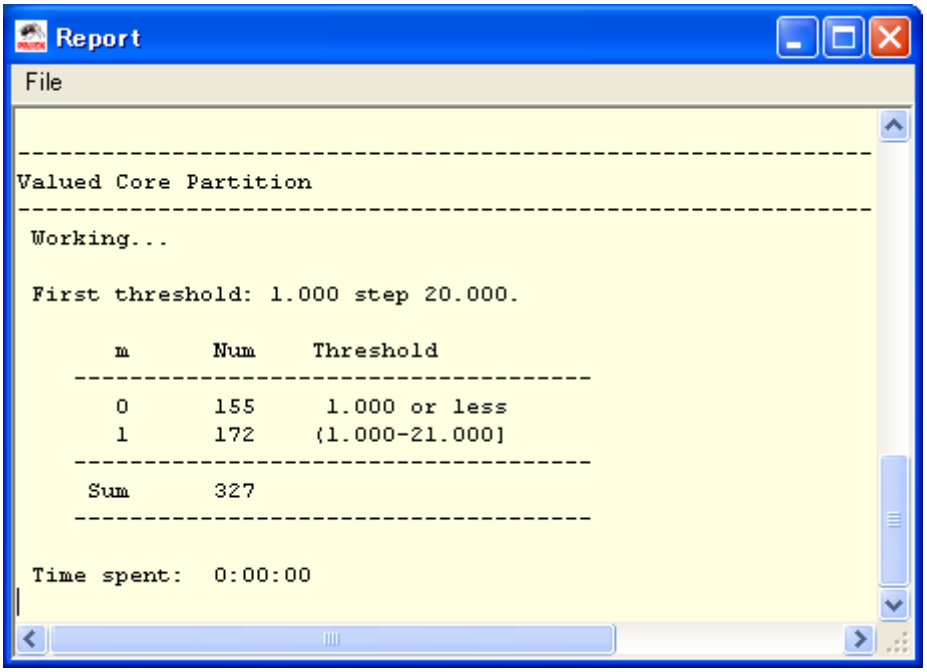
入次数による Valued core

2. Output

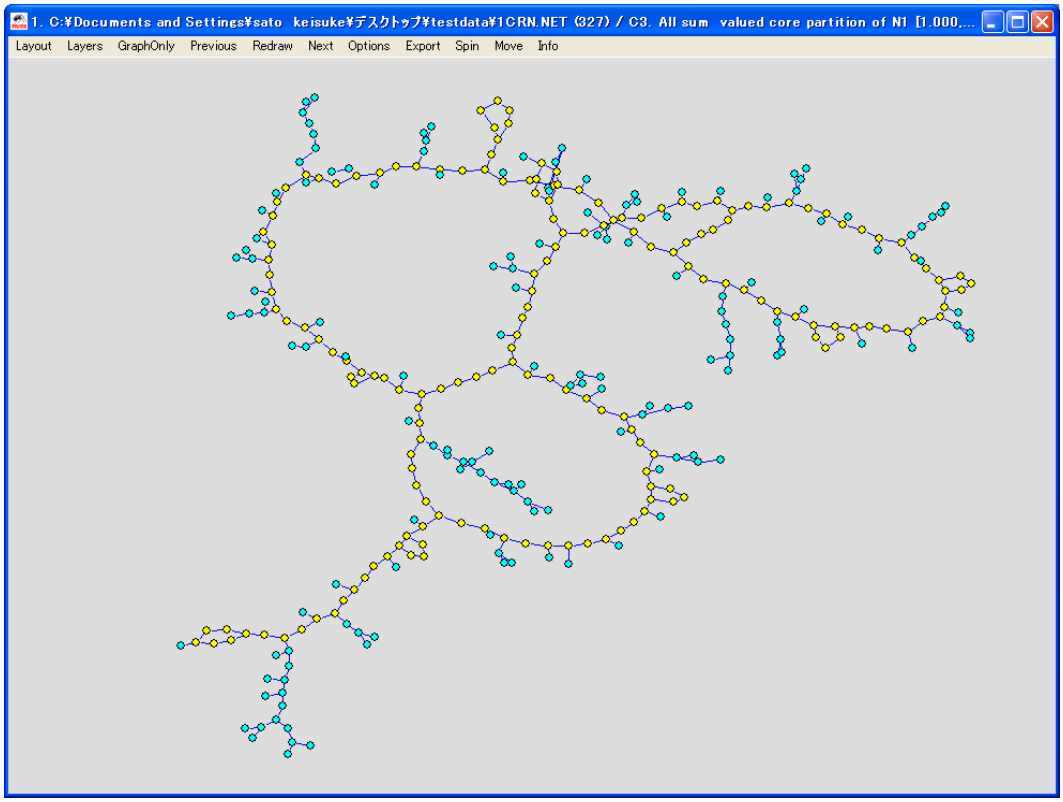
出次数による Valued core

3. All

無向線による Valued core




図：ALL で First threshold：1・Step：20 の結果




図：ALL で First threshold：1・Step：20 の結果の可視化

*Selected Thresholds

Thresholds (increasing numbers) are given using vector

1. Input

2. Output

3. All

*Use max instead of sum

<Depth

Depth：深さ、奥行き

*Acyclic

Acyclic：非環状の

Acyclic ネットワークでないと受け付けないメニュー

頂点の奥行きに従った、Acyclic ネットワークの Partition

*Genealogical

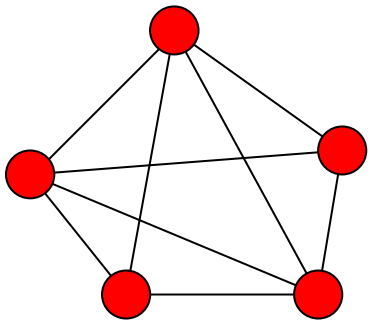
Genealogical：系図の

頂点の Layer に従った系図のネットワークの Partition

<p-Cliques

Clique：グラフ理論上で、すべてのノードが互いに直接的な紐帯で連結しているグラフ

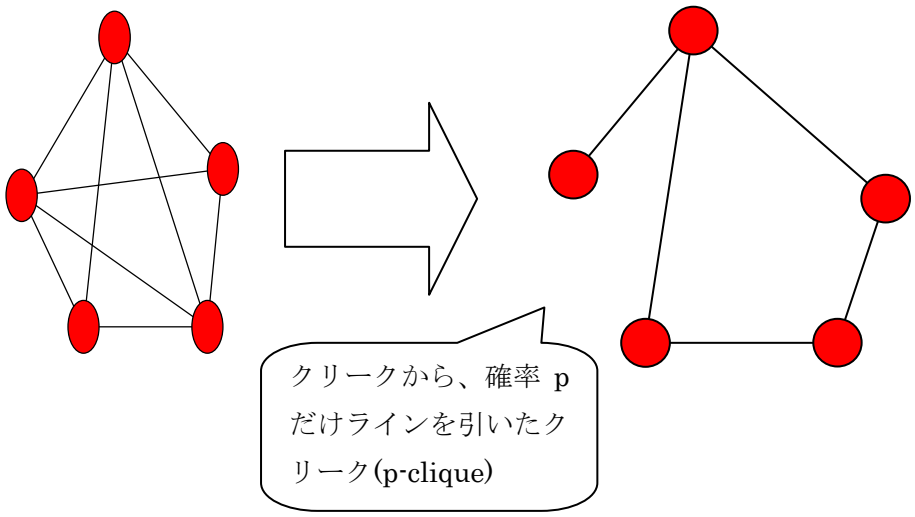
クリークとは、つまり完全グラフのことである。



図：clique

P：number between 0 and 1（0 から 1）

p-Cliques：少なくとも、クラスター内に確率 p（0~1）の近隣を持つ頂点を持つクラスターを Partition する




図：p-clique

つまり、確率 1（100%）の p-clique は、完全にラインを引いた p-clique で、つまり、clique（完全グラフ）である。

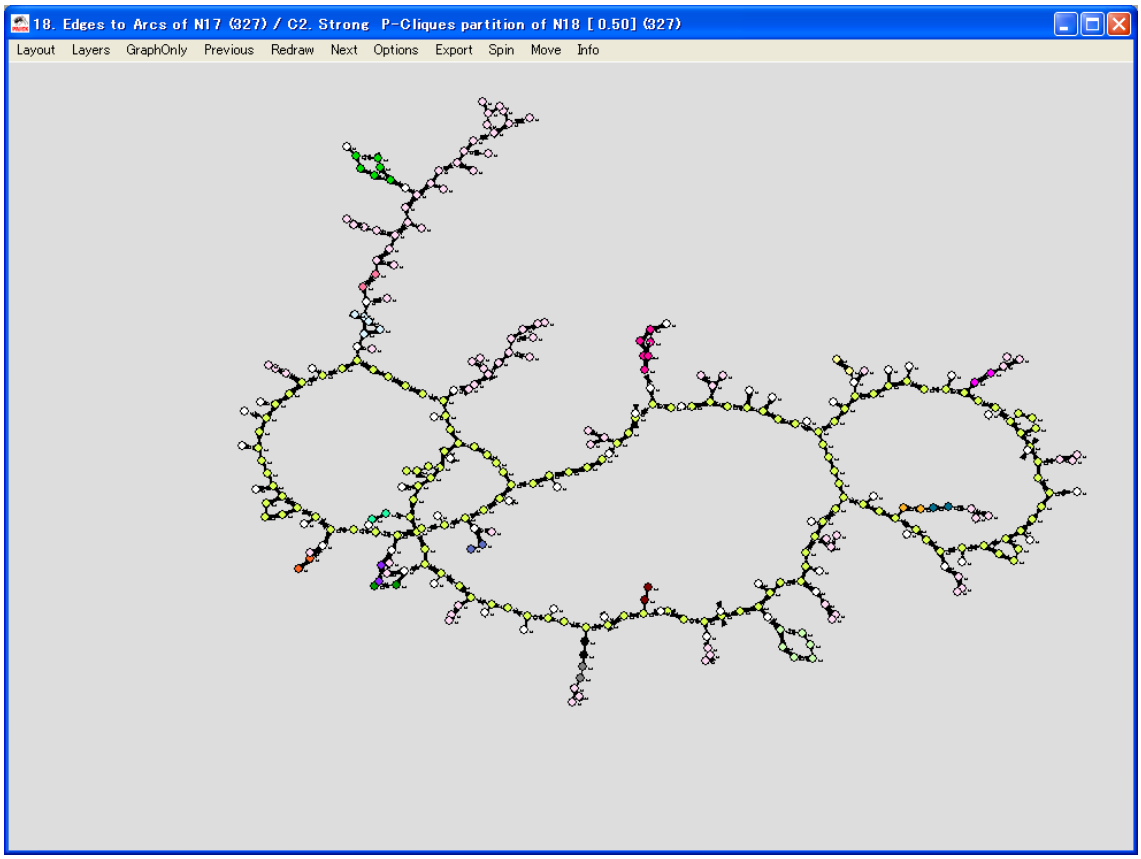
p-Cliques に従い、Partition する。Partition ボックスに*.cls ファイルが生成される。

入力項目

①確率

*Strong

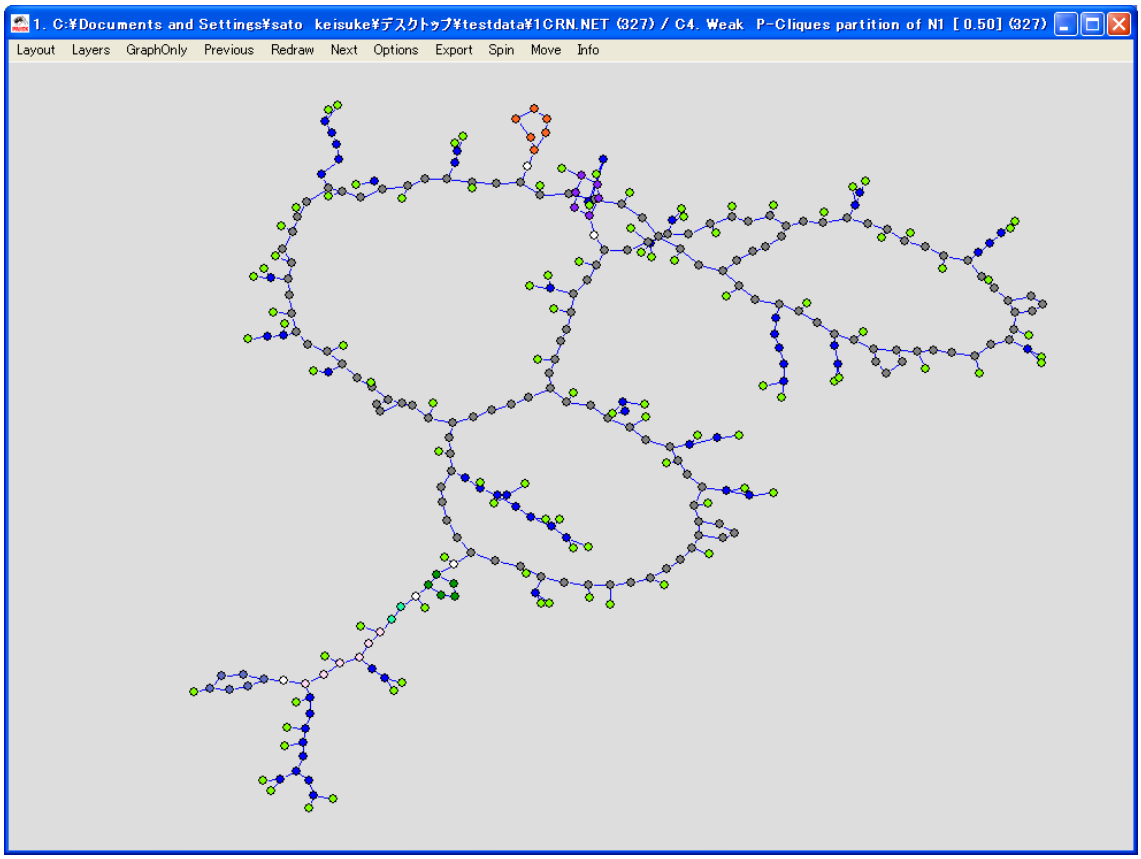
for directed network（有向線を持つネットワーク）




図：確率 0.5 の Strong p-Clique

*Weak

for undirected network（有向線を持たない、つまり無向線を持つネットワーク）




図：確率 0.5 の Weak p-Clique

<Vertex Labels

同じ頂点ラベル=同じクラスとする Partition

<Vertex Shapes

同じ頂点の形＝同じクラスとする Partition

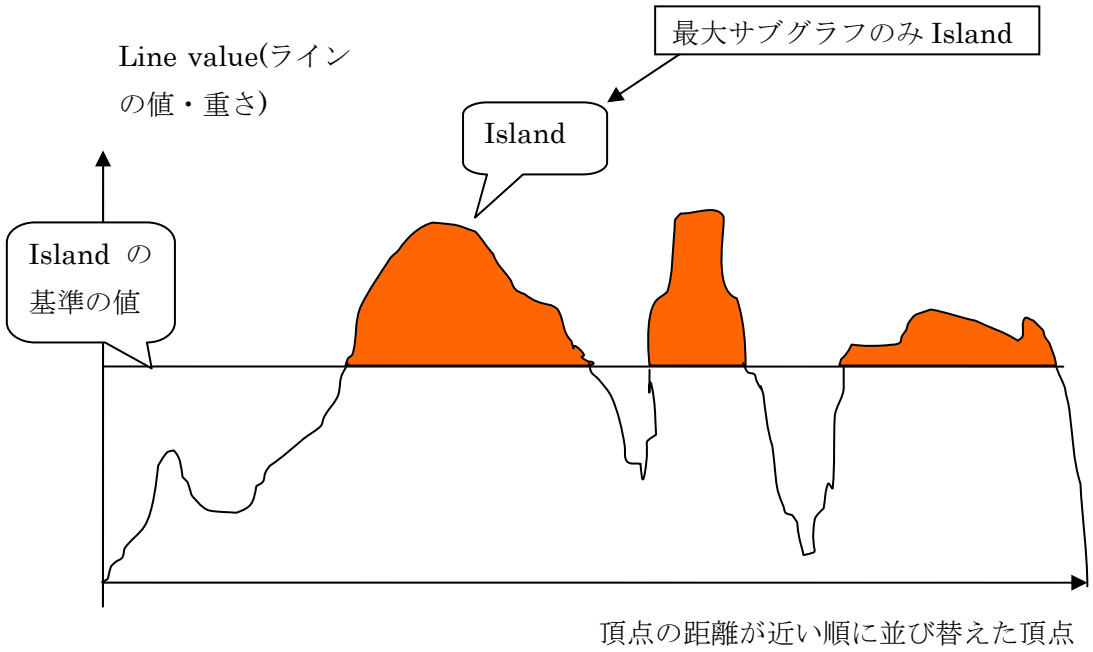
頂点の形＝.net で指定できる

<Islands

重さ：ラインの重さがあるネットワークの頂点の Partition

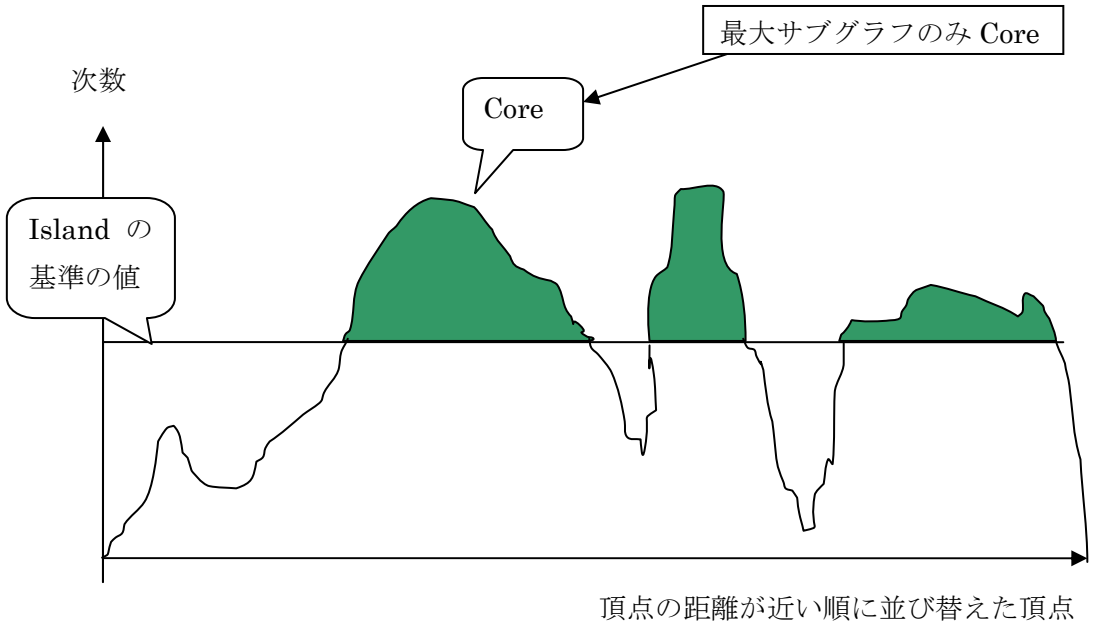
（Weight：Partition vertices of network with values on lines）

Islands：ラインの値（重み）による最大サブグラフ（別称：M-Core）




図：Island 生成の原理

Island に似た概念である Core も同様で、y 軸の「Line value(ラインの値・重さ)」の代わりに「頂点の次数」になる。




図：Core 生成の原理

*Line Weights

入力項目

① Island の最小サイズ

② Island の最大サイズ

*Line Weights[Simple]

入力項目

③ Island の最小サイズ

④ Island の最大サイズ

*Generate network with Islands

[選択項目]

<Bow-Tie

「Bow-Tie」のパーティション形式に従った Partition。

1-LSCC

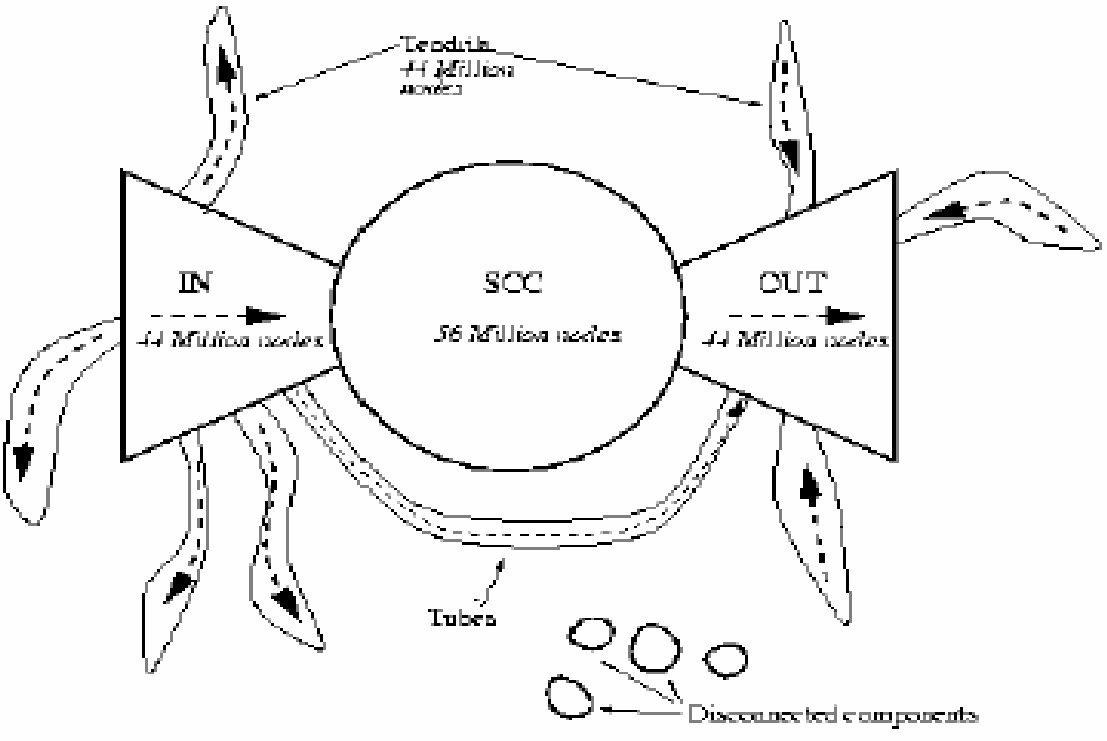
2-IN

3-OUT

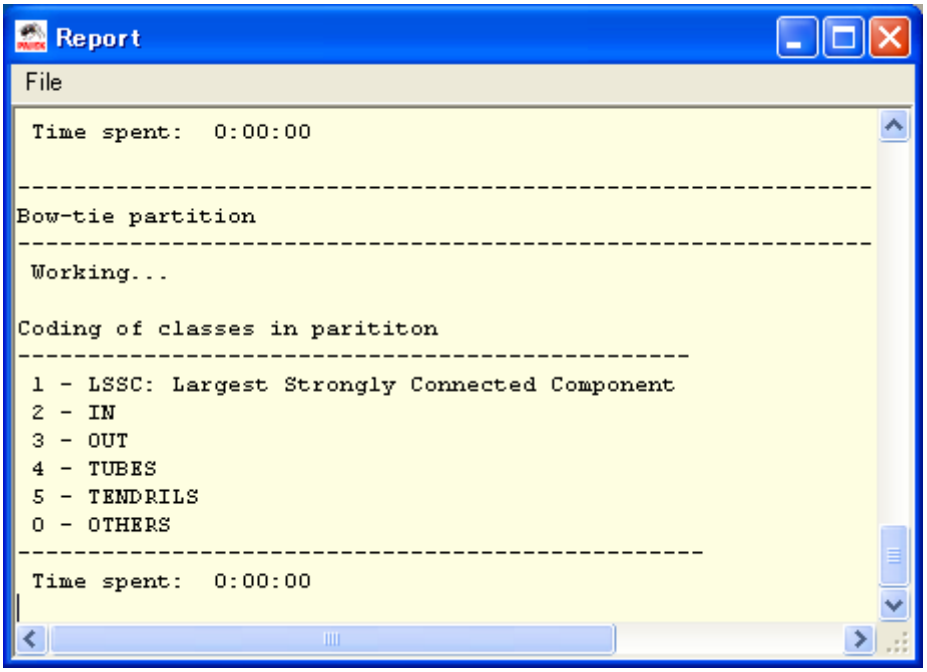
4-TUBES

5-TENDRELS

0-OTHERS




図：Web の Bow-Tie グラフ[manual 18]




図：Bow-Tie 後に表示される Report

有向線ネットワークを巨視的に見ると、bow-tie の形式となっている。

1、有向線ネットワークの中心部分

2、有向線ネットワークの中心部分へ向かう有向線部分

3、有向線ネットワークの中心部分から出て行く有向線部分

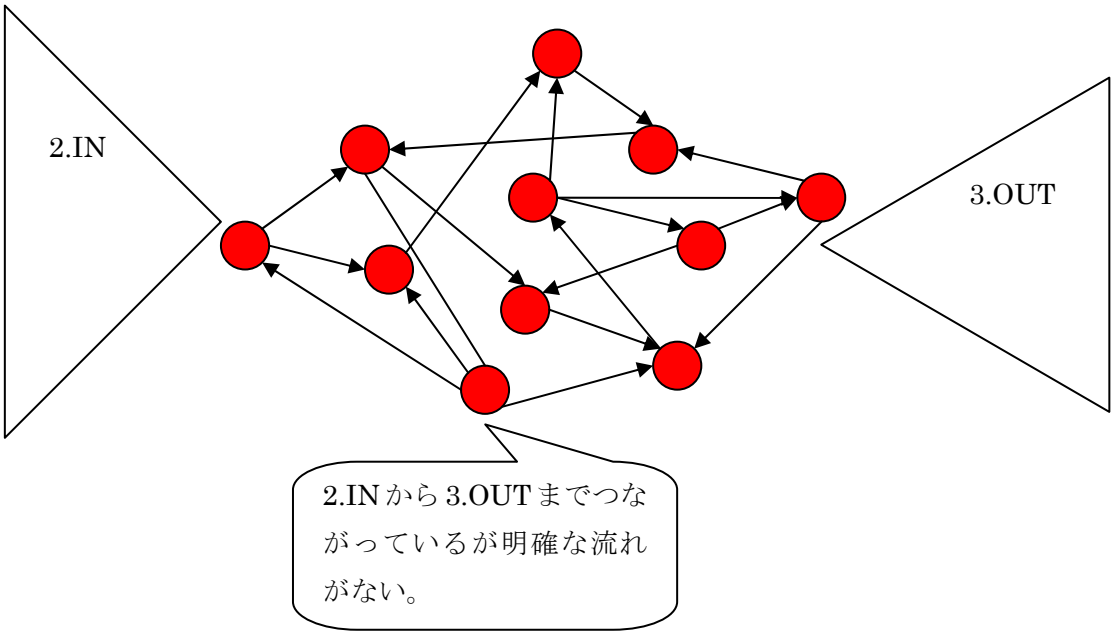
4、2 と 3 が中心部分を通らず直接つながっている部分

5、1,2,3 を関係なく流れが出来ている部分（支流）

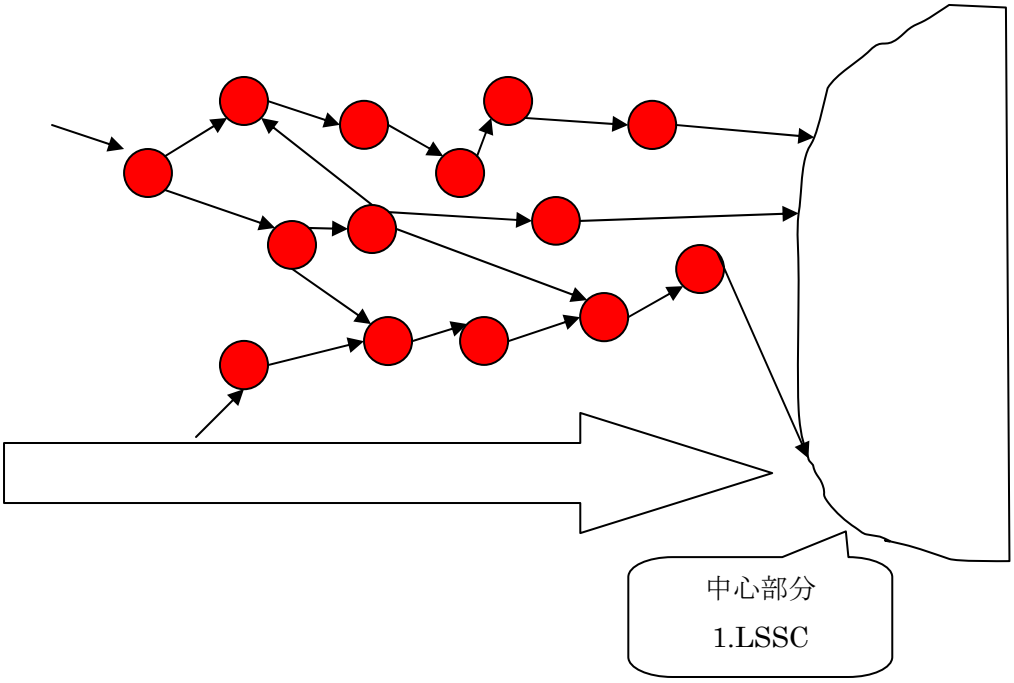
0、まったく孤立している部分

有向線ネットワークは、巨視的に見ると以上のように流れがある。1,2,3 を本流として捕らえ、1 を湖、4 は湖を通らない流れ、5 は支流、0 は沼と捉えるとわかりやすい。

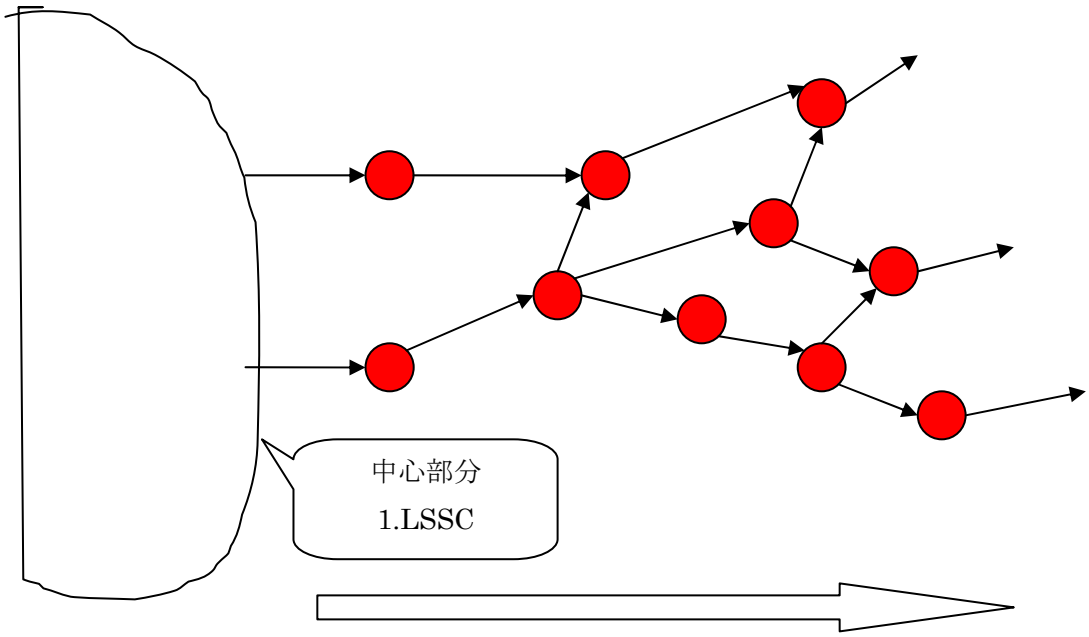
では、実際どうなっているかを下記で見ていく。



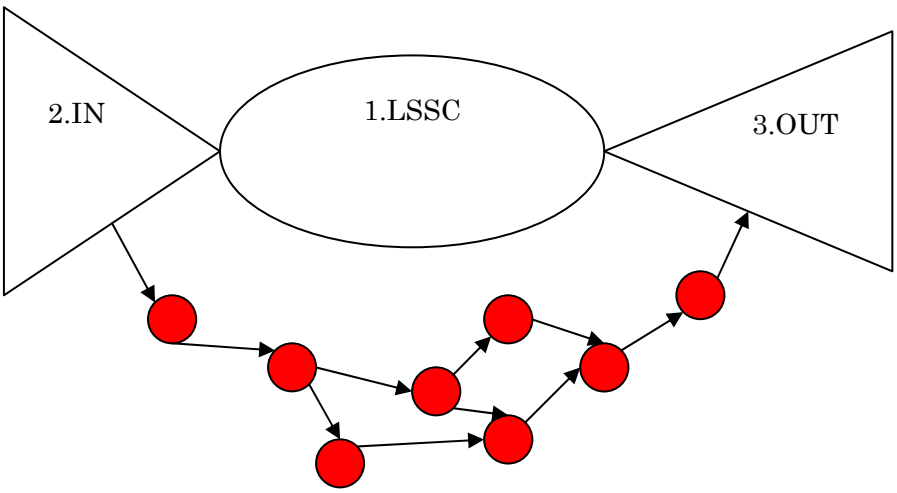

図：1.SSCL



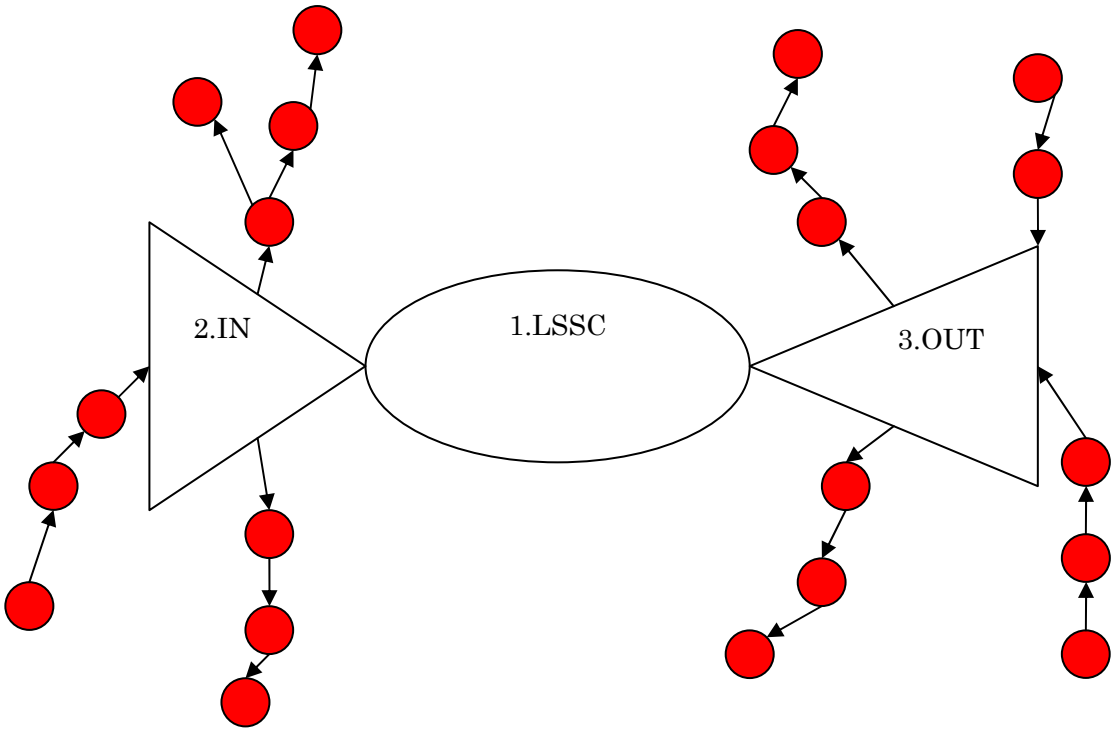

図：2.IN



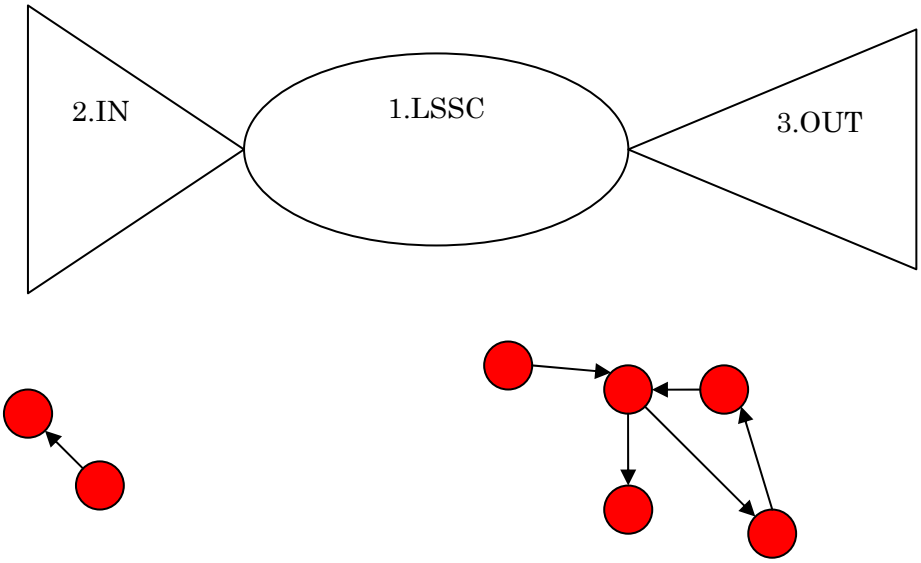

図：3.OUT



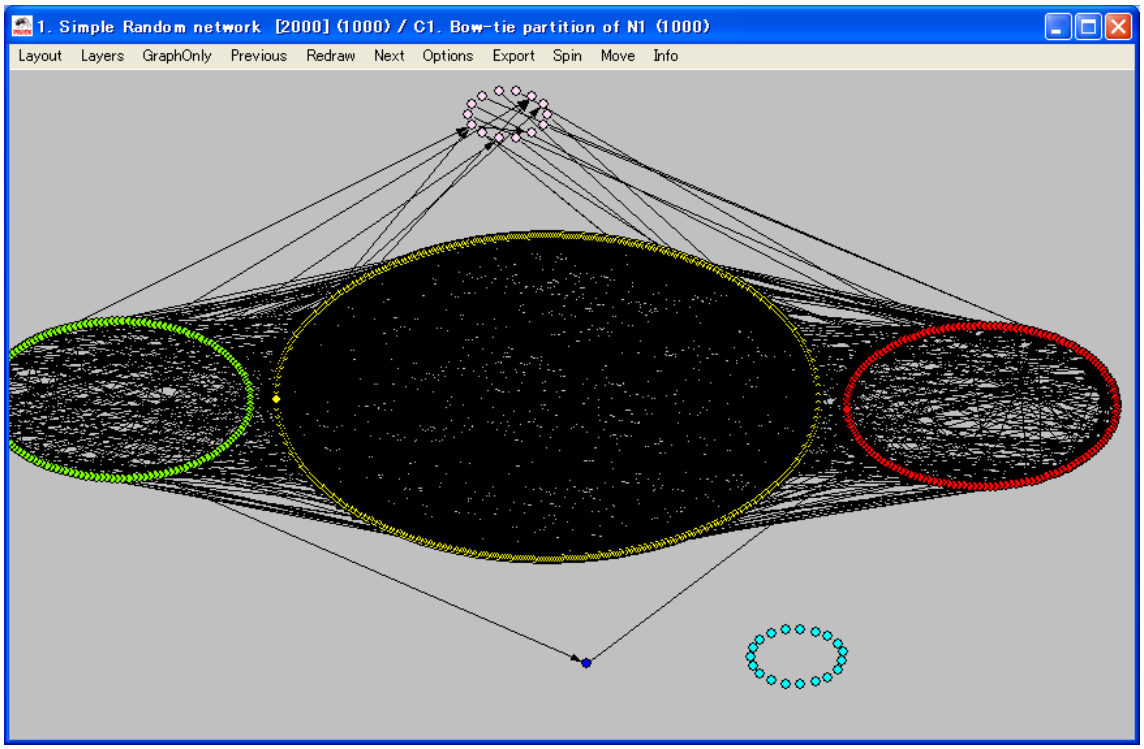

図：4.TUBES




図：5.TENDRILS



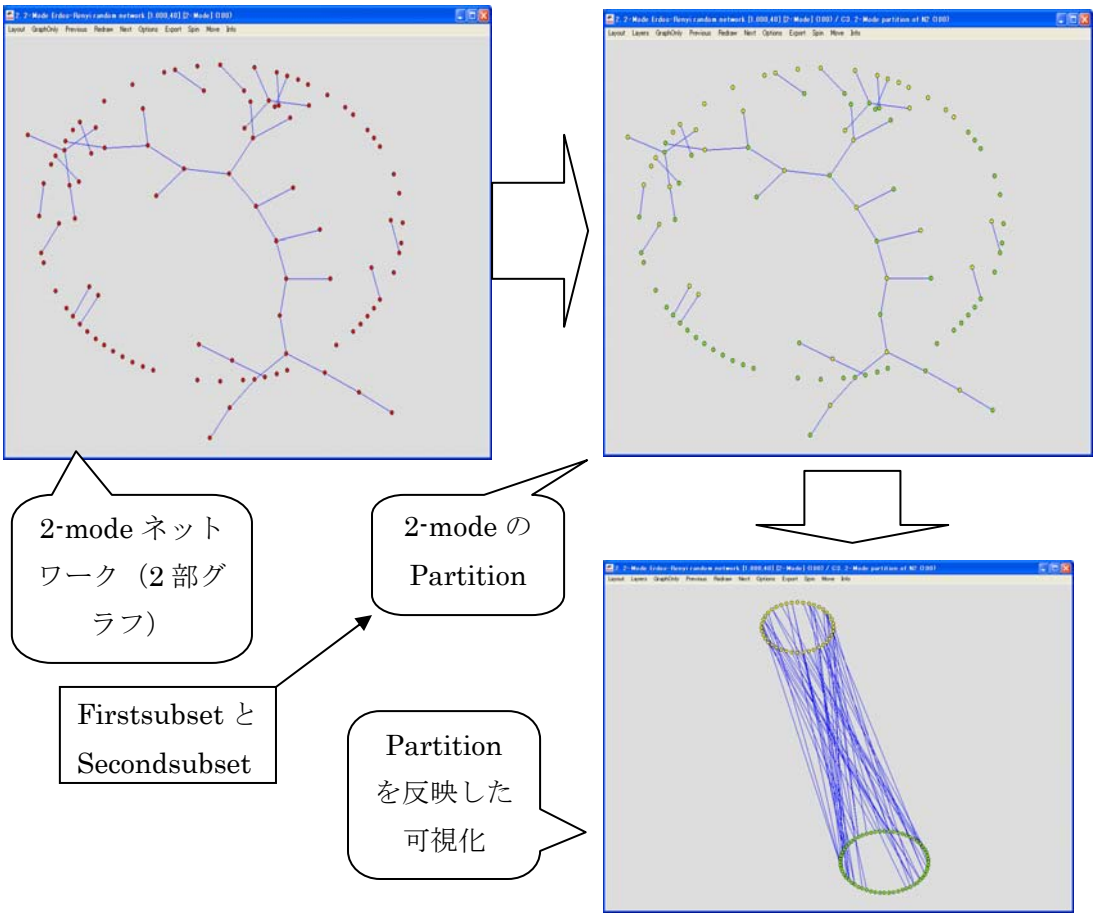

図：0.OTHERS




図：Bow-Tie 後に生成された Partition ファイルで可視化したネットワーク

<2-Mode

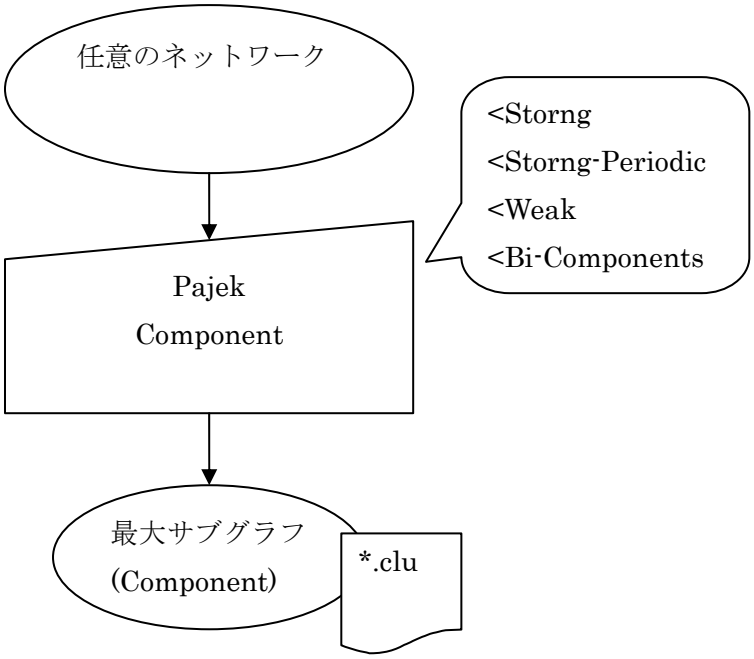
2-Mode（2 部グラフ）ネットワークの 2 つの部分集合（First subset と Second Subset)をパーティションする




図：Partition 2-mode

>Components

Components：ネットワークの最大サブグラフ（1 つ）である。




入力項目

①Component の最小サイズ

Components：最大サブグラフ

<Storng

選択されたネットワークの「Storng Components」

<Storng-Periodic

選択されたネットワークの「Storng-Periodic Components」

強い結合の Components は、期間に従って分け進められる。

<Weak

選択されたネットワークの「Weak Components」

<Bi-Components

選択されたネットワークの「Biconnected Components」

結合点(Articulation points)は、幾つかのクラスに所属している。その結果は、Partition に記録されない。Biconnected Components は、Hierarchy に記録される。Components 内の頂点の最小値を選ぶ。それに加えて、Parititoin に属している結合点が生成される。互いの頂点に属した Biconnected Components の数が与えられる。Paritition は、正確な Biconponents の Paritition に所属している頂点により成り立っており、Biconponents の外にある頂点と結合点は、共に、以下のものが生み出される。

Biconponents の外にある頂点：クラス番号 0

Biconponents：クラス番号 1~ Biconponents の数と結合点はクラス番号

99999998 を取る。

>Hierarchical Decomposition

<Clustering

階層的クラスター分類の手続き。入力は、異なるネットワーク（行列）で、それは、Operation>Dissimilarity かインプットファイルを読み込むことによって得られる。

*Run

階層的クラスター分類を作る。結果は、同種に集められた（nest された）クラスターと、EPS に収められたデンドログラムを含む階層である。

*Options

階層的クラスターをつくる方法を選択する。

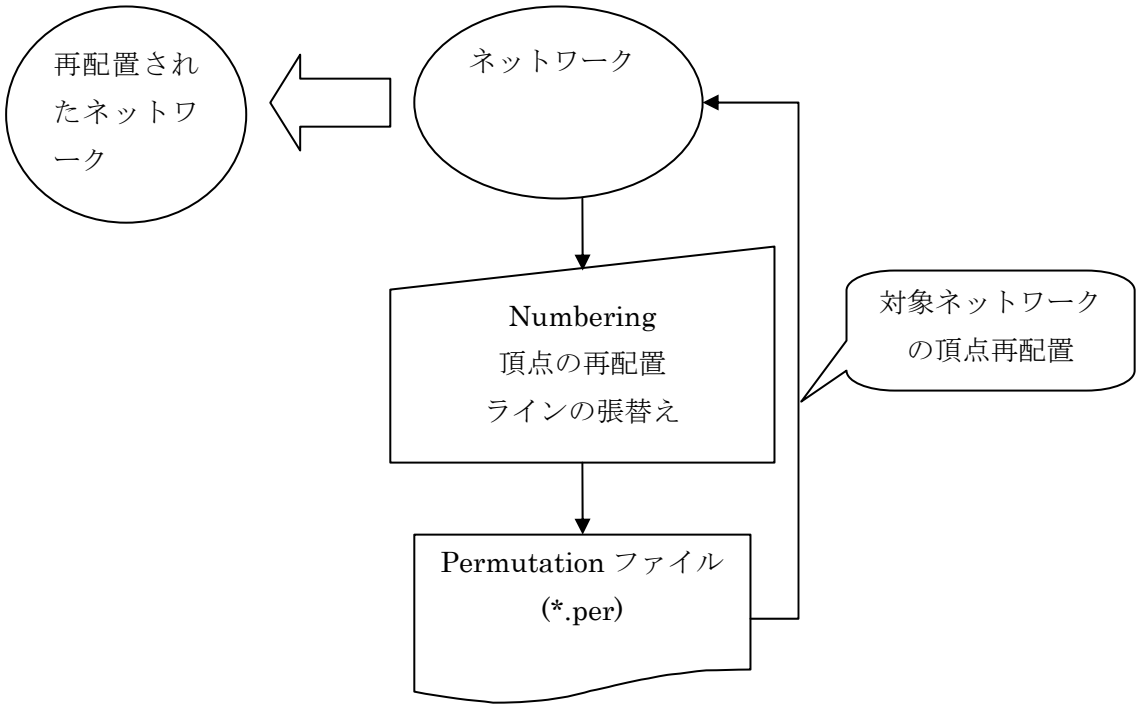
（general,minimum,maximum,average,ward.squared ward）

<Symmetric-Acyclic

ネットワークの相対的非環分解。結果は、同種に集められた（nest された）クラスターを含む階層である。

>Numbering

Permutation に*.per ファイルが生成される。*.per ファイルは頂点の再配置（=頂点番号の張替え）であり、Numbering とは頂点の再配置するメニューである。




図：numberring の処理手順

<Depth First

Depth first numbering of selected network

*Strong

taking directions of arcs into account

有向線の方向を計算する

*Weak

forget directions(or undirected network)

方向を考慮しない。つまり、無向線向け処理。

<Breadth First

Breadth=幅、広さ

Breadth first numbering of selected network

*Strong

taking directions of arcs into account

有向線の方向を計算する

*Weak

forget directions(or undirected network)

<Reverse Cuthill-McKee

RCM numbering=?

<Core+Degree

>Citation Weights

Citation=引用

もしも、ネットワークが Citation network として表せるなら、ラインの値（重み）（つまり、Citation）と頂点（の一部）は以下を出力とした計算ができる。

ラインの値（重み）: Citation

結果は、

ⅰ）重要な Citation で表されたラインの値（重み）のついたネットワーク

ⅱ）メインパス上の頂点による Binary(0 か 1)Partition

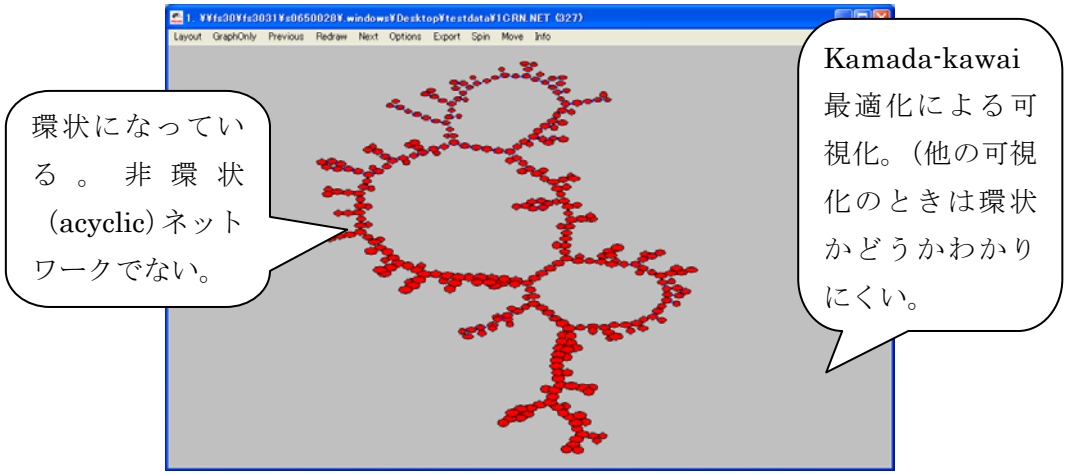
ⅲ）メインパスのみにより構成されたネットワーク

ⅳ）重要な頂点（の一部）の Vector

重さは、標準化（使用されている流れか最大値）またはログをとることも出来る。

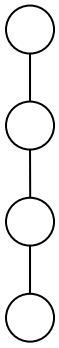
<Search Path count(SPC)

非環状ネットワークでないと処理できない。水源から排水溝への計算を行う方法。




図：環状ネットワーク

Path とは以下のように直線上につながったものである。



図：Path

<Search Path Link count(SPLC)

互いの頂点を水源として考慮した方法。

<Search Path Node pair(SPNP)

method.

>k-neighbors

対象頂点（自分で選んだ）から最大ステップを指定し、どこまでの頂点までいけるか？がわかる。

k-neighbors を選択した後の手順

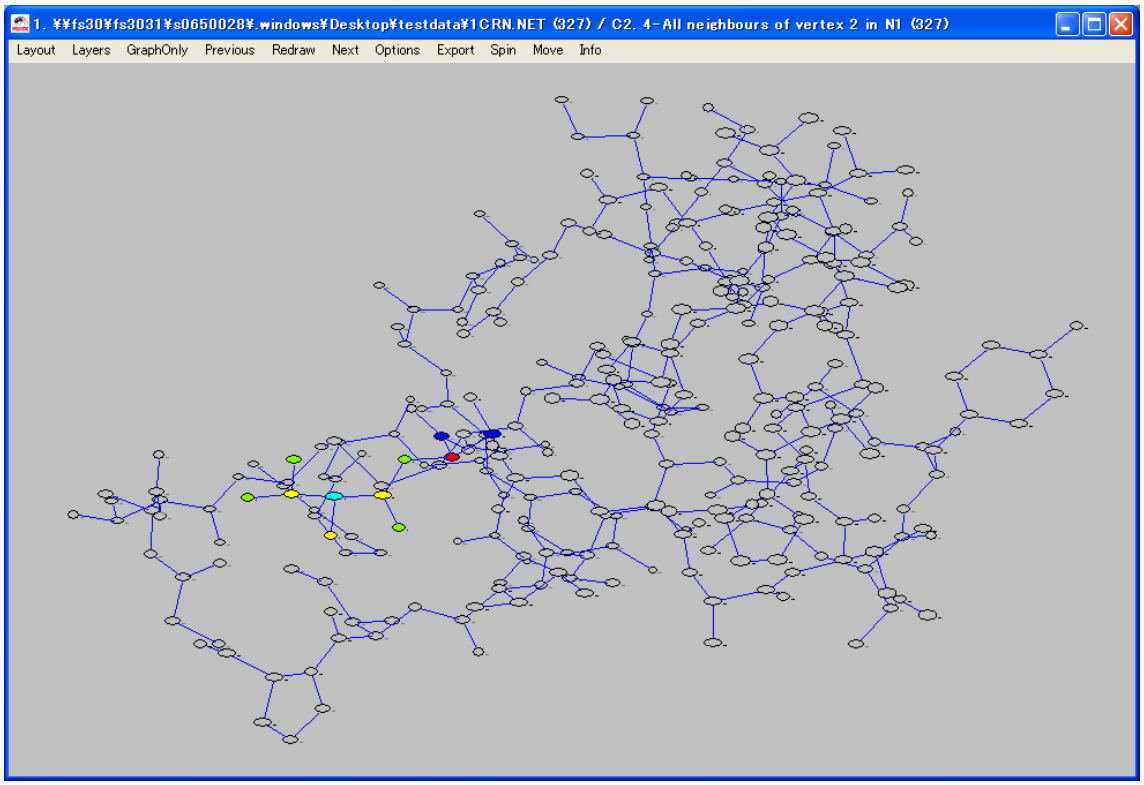
①対象頂点を選ぶ

②最大ステップ数（k-step）を選択する

③メインスクリーン Partition ボックスに*.vec が生成

④Partition と共に可視化（Draw>Draw-Partition）

⑤下図のような可視化が出来る




図：頂点 2（水色）から 4 ステップ（最大値）でいける頂点

1 ステップでいける－黄色

2－ステップでいける－黄緑

3－ステップでいける－赤

4－ステップでいける－青

この配色は、Partition の色別設定に従う。

Options>Colors>partition Colors>for vertices

で変更可能

可視化のとき、ctrl＋N を押せば頂点番号、ctrl＋L を押せばラベルが表示されて、どの頂点まで到達するかがわかる。

（Options>Make Vertices Using）

k-neighbors：

Select All vertices…

<Input

最大 k-step の選ばれた頂点に届くことが出来る入次数の頂点を探す。

..form which we can reach selected vertex in at most k-steps

<Output

最大 k-step の選ばれた頂点に届くことが出来る出次数の頂点を探す。

..that can be reached form selected vertex in at most k-steps

<All

..Input+output

<From Clusters

計算は、選ばれた距離に従ったクラスター内の互いの頂点距離となる

>Paths between 2 vertices

2 頂点間のパスに関して・・・

<One Shortest

2 頂点間の最短パスを求める。

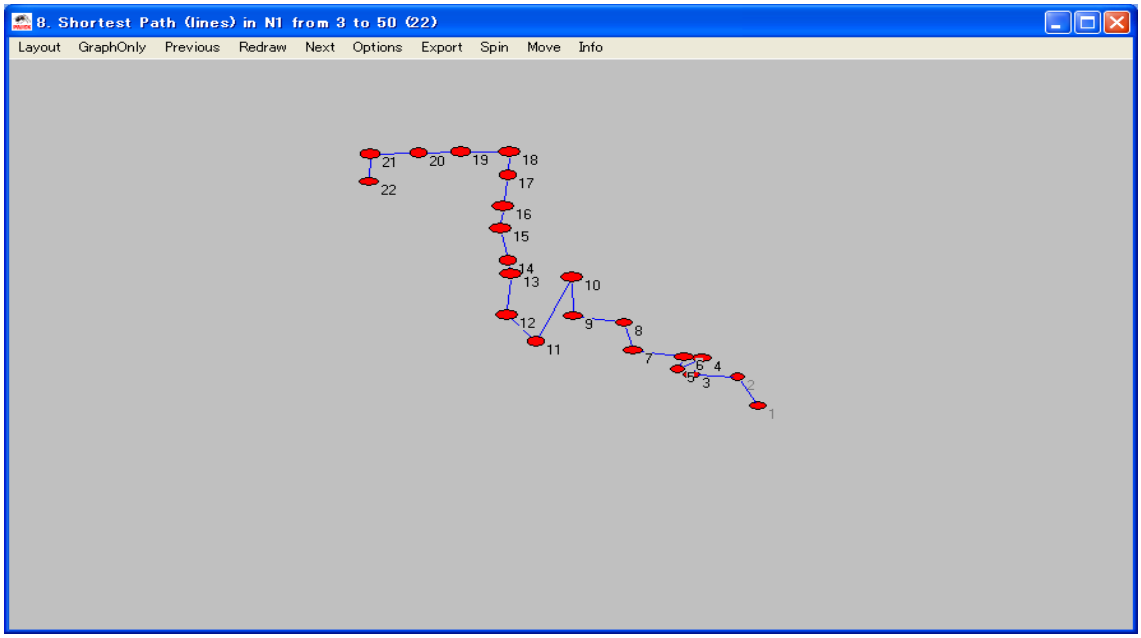
入力項目

① From

② To

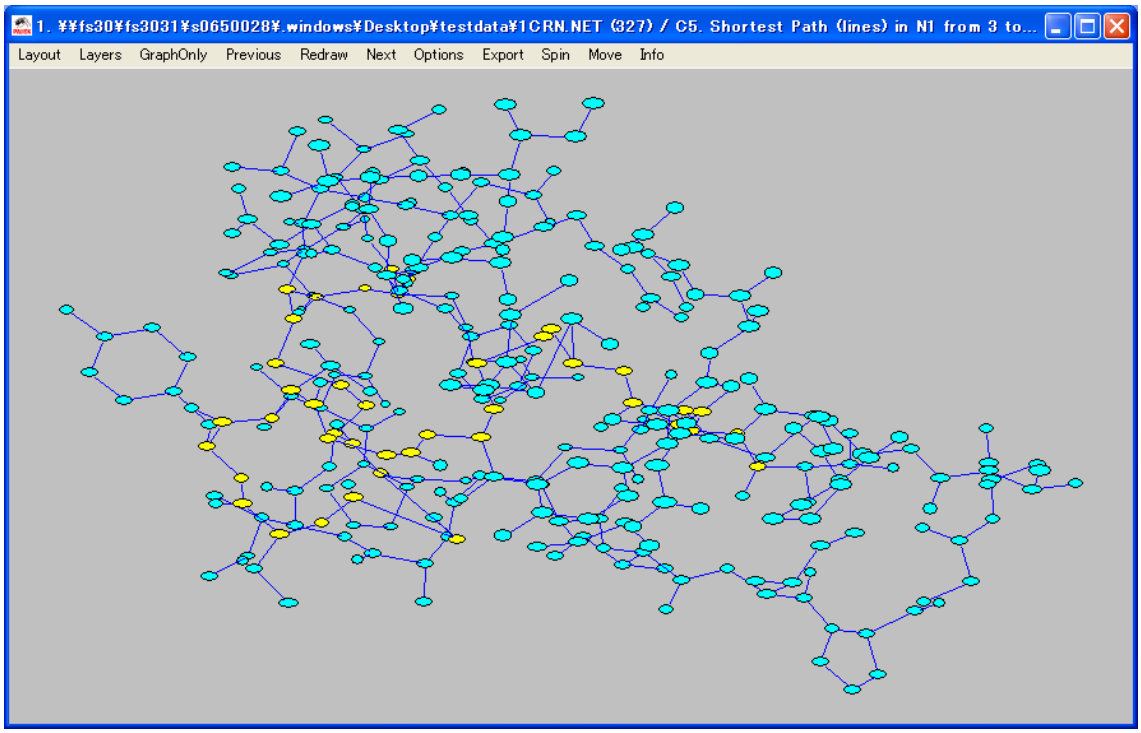
③ Forget value of lines?(yes/no)

④ Identity vertices in source of network

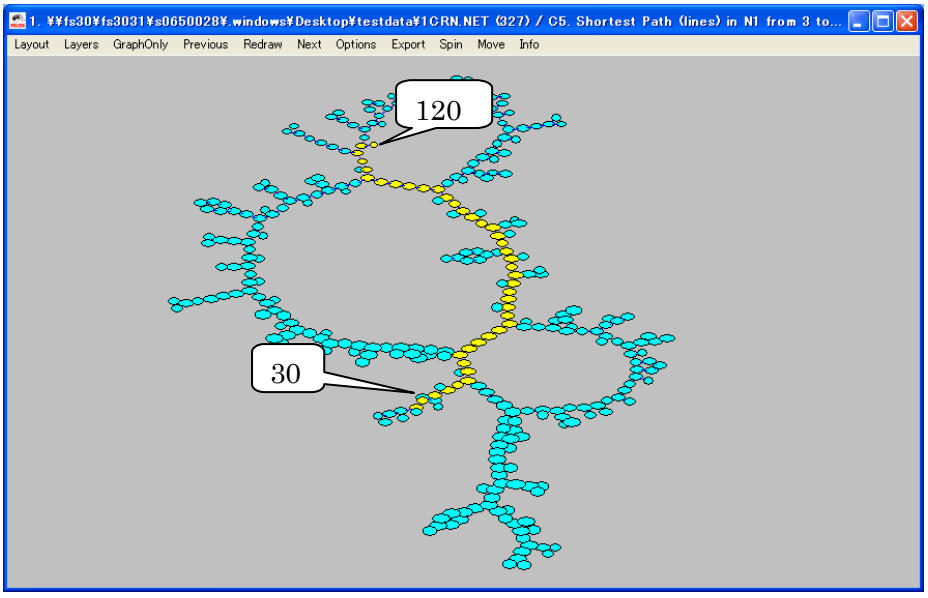



図：頂点番号 30 から 50 までの最短パス。

番号は、パス数である。新しいネットワークファイルとなっているため、Darw のみでは、どの頂点を通っているかはわからない。しかし、元のネットワークファイルと生成された Partition ファイルを使って可視化すればどの頂点を通って最短パスを形成しているかがわかる。




図：元ネットワーク内の 30 から 120 までの経路（黄色）




図：元ネットワーク内の 30 から 120 までの経路（黄色）kamada-kawai 最適化

<All shortest

すべての 2 頂点間の最短パスを求める。

<walks with Limited Length

最長レンジを探す。

入力項目

① From

② To

③ Forget value of lines?(yes/no)

④ Identity vertices in source of network

⑤ Input maximum length of walk allowed(>0)

<Diameter

Diameter を探す

Diameter：直径

最長最短パスを探す

Diameter：最長最短パス?

<Geodesics Matrices

small network のみの処理！

Geodesics：測地線 ・最短パス長のこと

最短パスのマトリックスを計算する

そして測地線を計算したマトリックスを計算する

<Distribution of Distances

*From All Vertices

starting points のすべての頂点を取る

starting points：

*From Vertices in Cluster

クラスター内の頂点のみ

>Critical Path Method(CPM)

非環ネットワークでクリティカルパスを求める。結果は、クリティカルパスを含んだ新しいネットワークとなる。2 つの Vector ファイル（及び Partition ファイル）を生成する。1 つ目のファイルは、早い時間の状態が記述され、2 つ目のファイルは、遅い時間の状態が記述される。

>Maximum Flow

Maximum Flow amang vertices

複数の頂点について最大フローを求める

<Selected Pair

2 頂点の最大フローを求める。（アルゴリズムは、直線のパスとそれらの周りのいつも選ばれる最短のパスを探す。）アルゴリズムは、テクニカルエリア（実際

のフロー[actual flow]、許容量を意味するラインの値（重み)）あるいは、分析的グラフ（すべてのラインが 1 のとき）を使う。

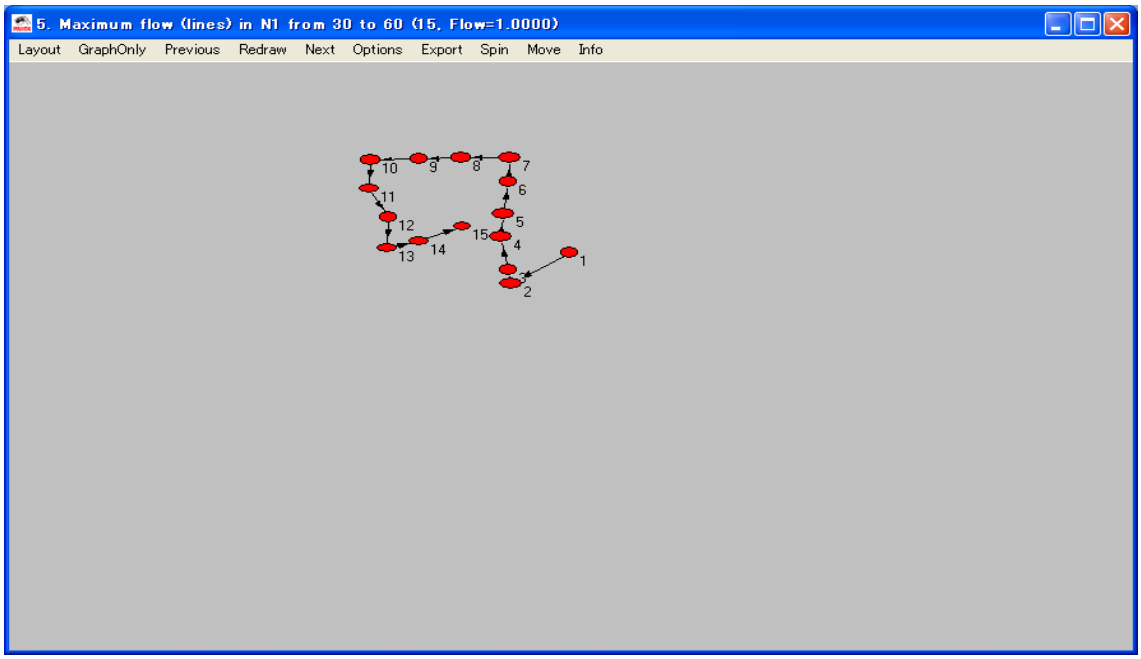
入力手順

① Find　maximum flow の from の入力

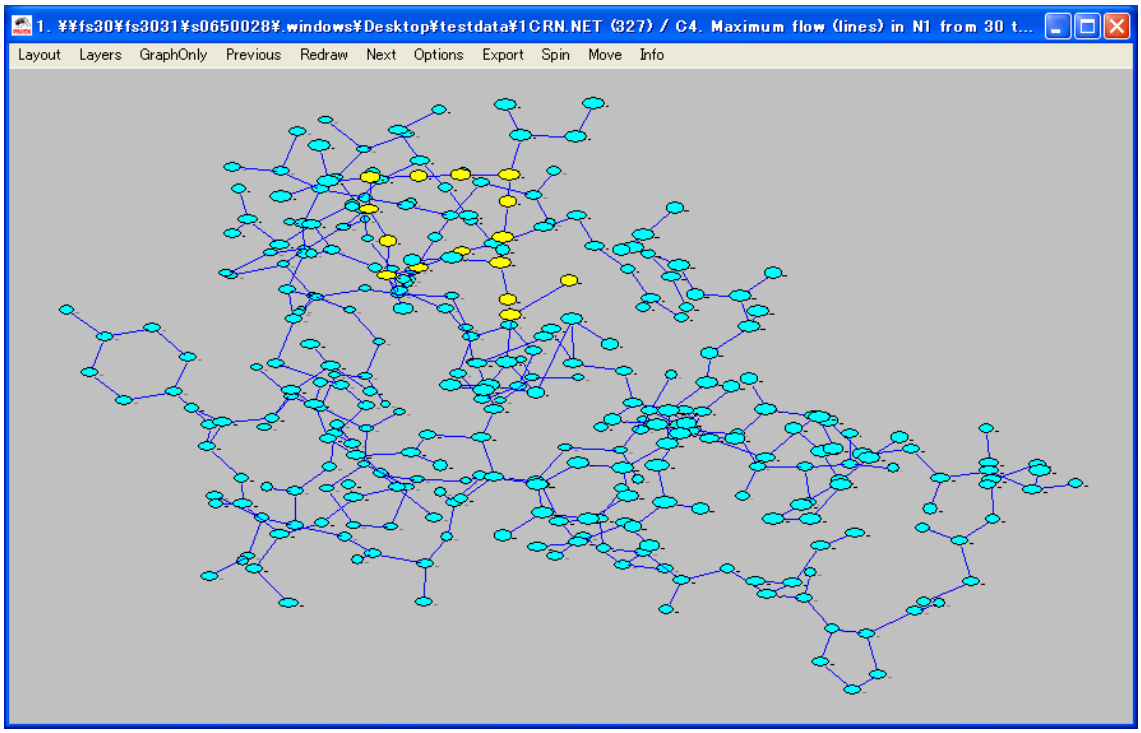
② Find　maximum flow の to の入力

③ Forget values in lines?（ラインの値（重み）を忘れるか？）

④ Identify vertices in source network?（頂点を元のネットワークと見分けるか？）




図：30 から 60 の Selected Pair の Maximum Flow の例(15 頂点、Flow=1.00000)




図：元ネットワーク内での 30 から 60 の Selected Pair の Maximum Flow

生成されるもの

ｉ）新しいネットワーク（*.net ファイル）

ii）Partition ファイル

<Pairs in Cluster

頂点の周りで定義されたクラスター間の最大フローを求める。結果で生成されるものは、ラインの値（重み）の意味が、一致した 2 頂点間の最大フローである新しいネットワークである。

（アルゴリズムが遅いため、限定的な頂点の小さなネットワークのみである。）

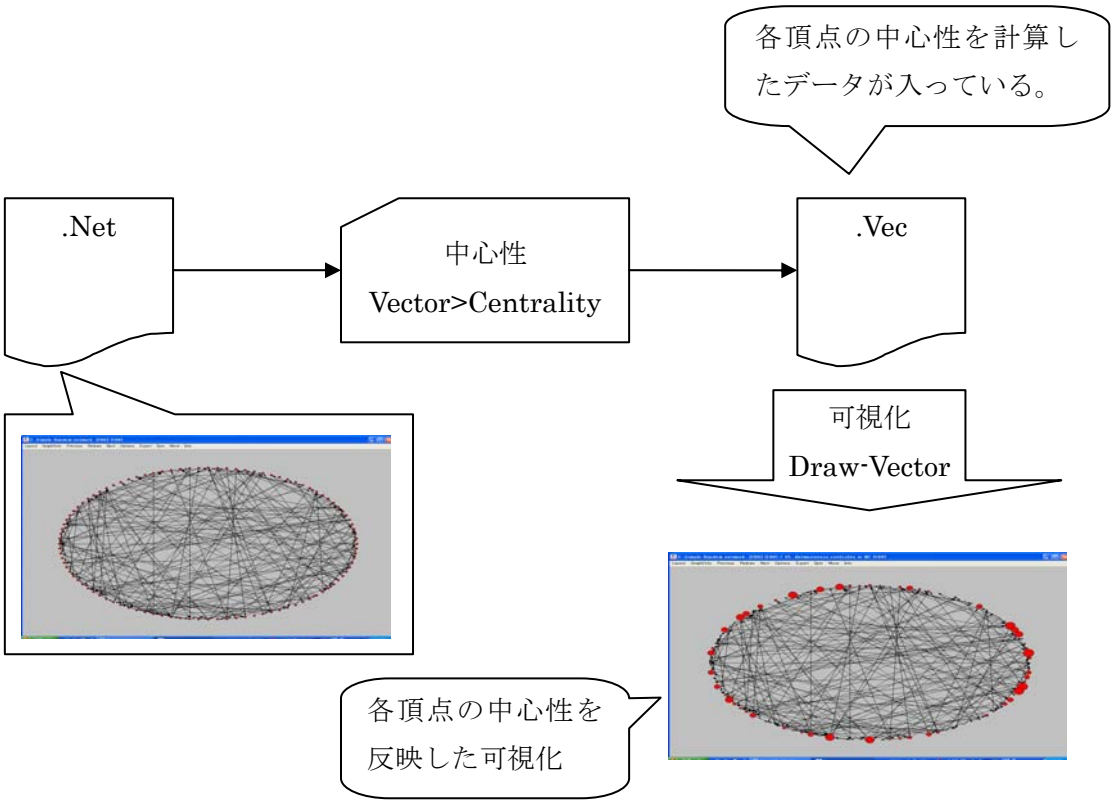
>Vector

get vector from network

*.vec ファイルを生成する操作（オペレーション）メニュー

<Centrality

中心性を求める。




図：中心性

## 中心性とは

中心性（Centrality）とは、ネットワーク分析のための概念であり、ネットワーク内の任意の頂点がどれだけ中心にあるかの性質を示す概念である。その量的な指標を中心性指標という。

中心性指標には大別して 4 つが存在する。

1、次数：ある頂点が所有している次数に着目し、次数が大きいほど中心性が高い

2、近接性：ほかの頂点とどれだけ距離が離れているかを示すもので、これが小さいほど中心性が高い

3、媒介性：その頂点を通過しないと到達できない頂点の数を表し、その数が大きいほど中心性が高い

4、固有ベクトル：どの程度ほかの頂点とつながっているかを示す。

Pajek では、「2、近接中心性（Closeeness Centrality)」と「3、媒介中心性（Betweenness Centrality)」を求めるメニューがある。この中心性を可

視化も出来る。

## １、次数中心性（Degree Centrality）

（pajek のメニューに存在する。）

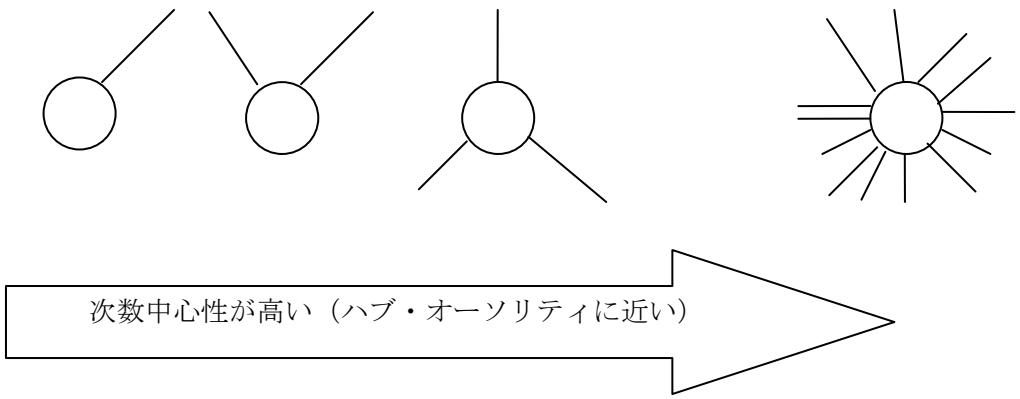
次数中心性$C_d(v_i)$、頂点$v_i$、頂点$v_i$と隣接する頂点の次数$\deg(v_i)$

ネットワーク総頂点数 N とすると、

$$C_d(v_i) = \frac{\deg(v_i)}{N-1}$$

任意の頂点の近接する頂点数（次数）と自身の頂点を除いた総頂点数の比で表される。

ある頂点の次数が高いほど、次数中心性が高い。つまり、次数中心性はハブ（あるいは、オーソリティ）に近い役割を果たす頂点を探すことである。




図：次数中心性とハブ・オーソリティ

## ２、近接中心性(Closenness Centrality)

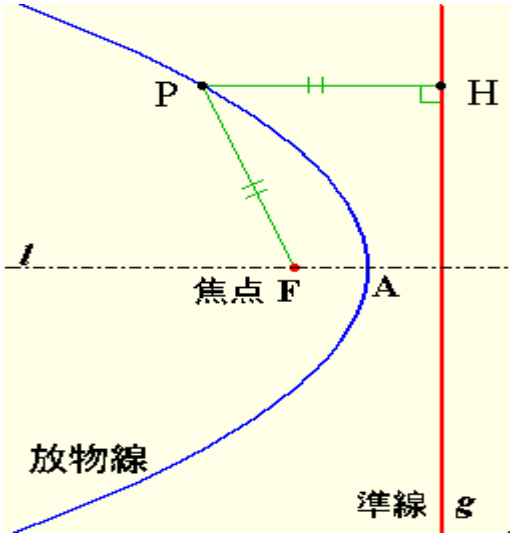
頂点$v_i$、頂点$v_i$の近接中心性$Cc(v_i)$、ほかの頂点までの距離の総和（離心率）$s(v_i)$とすると、

$$Cc(v_i) = \left(\frac{s(v_i)}{n-1}\right)^{-1} = \frac{n-1}{s(v_i)}$$

離心率が高くなるほど$Cc(v_i)$が小さくなり、中心性が高くなる。離心率eを他の頂点までの距離総和$s(v_i)$とする。

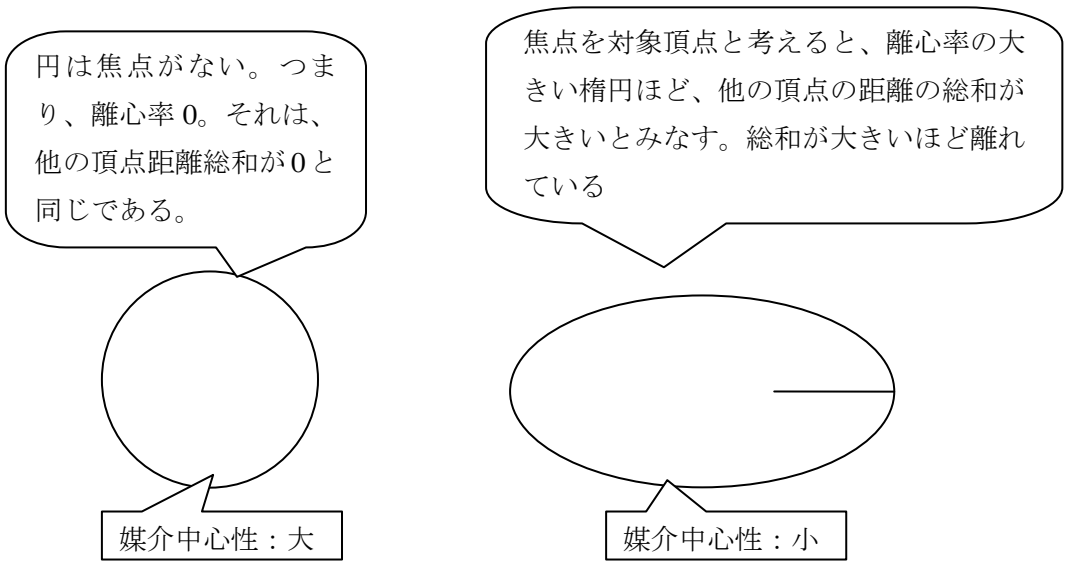
$$s(v_i) = e$$

つまり、定数$s(v_i)$という離心率の描き出す図は楕円である。




図：離心率[10]

離心率eの描く図形とは、図1.1の$\frac{PF}{PH} = e(\text{定数})$を満たすの図形のことであり、離心率=0は円で0<e<1は楕円である。つまり、円より楕円のほうが面積が大きいので、離心率が大きいとほかの頂点までの距離の総和が大きいと考えられる。



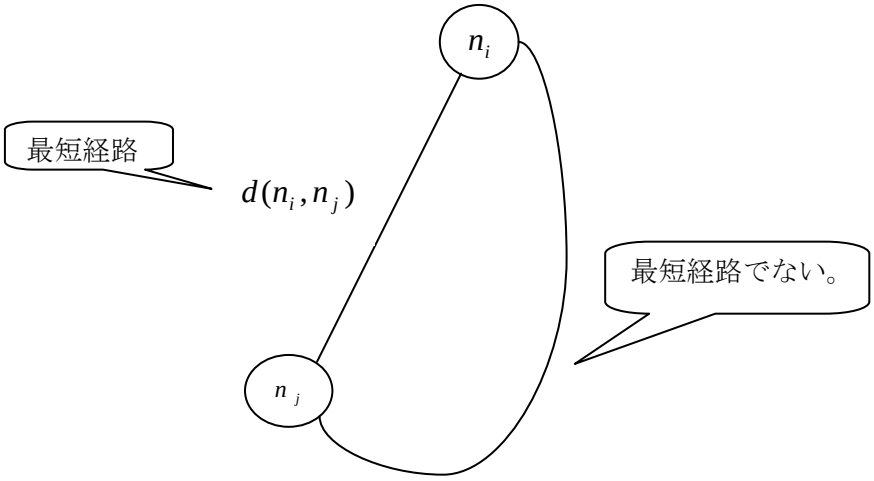

図：離心率と媒介中心性

近接中心性には、以下の式も存在する。

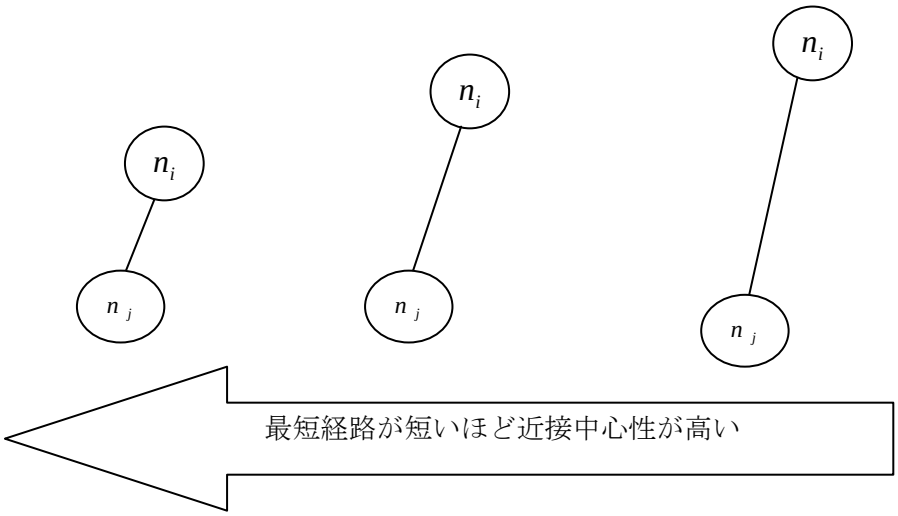
ある集団（コミュニティー）の総頂点数 g 、ネットワーク内の任意の頂点 $n_i$ 、 $n_i$ と同じコミュニティー内の任意頂点 $n_j$

$n_i$ と $n_j$ の最短距離 $d(n_i, n_j)$ 、中心性指標 $Cc(n_i)$ とすると、

$$Cc(n_i) = \left[ \sum_{j=1}^{g} d(n_i, n_j) \right]^{-1}$$




図：最短経路$d(n_i, n_j)$




図：最短経路と近接中心性

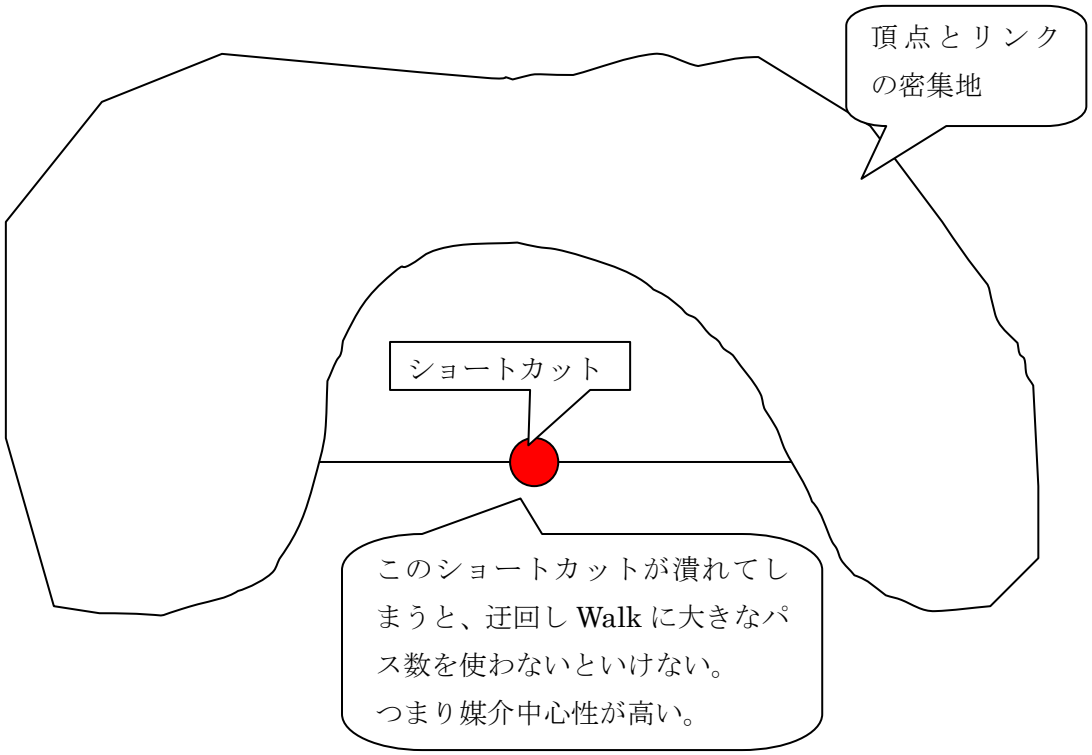
この式は、他の頂点までの距離の総和（離心率）ではなく、任意頂点の最短距離から近接中心性を求める式である。式からわかるように、任意の頂点$n_i$とその頂点のコミュニティー内の頂点$n_j$の最短距離をコミュニティー内の全頂点で計測していく。

その数が小さい(つまりほかの頂点との距離が近い)と$Cc(n_i)$が大きくなって、その頂点は中心性が大きくなる。g は、ネットワ

ークの頂点数で、$\sum^{g}$　全頂点で計算するという意味である。前の式との違いは、前の式は対象頂点と別の頂点の距離の総和に着目したが、この式は最短距離に着目するという違いである。

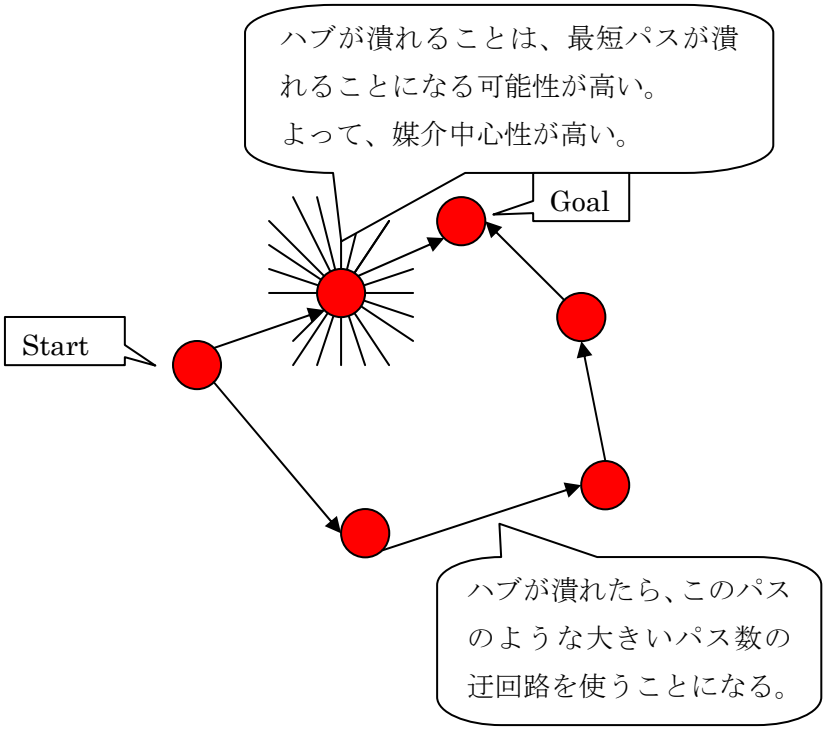
## ３、媒介中心性（Betweennes Centrality）

（pajek のメニューに存在する。）




図：媒介中心性の高い頂点の特徴(その 1)

また、ハブ・オーソリティのような頂点も、潰れてしまうと迂回路を通り Walk に大きなパス数を使う可能性が高くなるため、これらの頂点も媒介中心性が高くなる。つまり、最短パスの順路はハブ・オーソリティの数や集中度が高いほど、これらを通る確率は必然的に高くなるから、これらの媒介中心度は高くなる。




図：媒介中心性の高い頂点の特徴(その 2)

上記から「次数中心性」との違いは明確で、媒介中心性の高い頂点の特徴(その 1)より、次数中心性では考慮されなかった「ブリッジ」も中心度が高くなる。

媒介中心性$Cb_{ij}(v_k)$、頂点$v_k$が各 2 頂点$v_i$と$v_j$間の最短経路上に存在する（媒介する）頻度$b_{ij}(v_k)$、ネットワーク総頂点（ノード）n の 2 頂点間のうち$b_{ij}(v_k)$の合計値 、媒介値$BC(v_k)$とすると、

$$\mathrm{C}b_{ij}(v_k)=\frac{2BC(v_k)}{(n-2)(n-1)}$$

この式は、総頂点 n から 2 頂点$v_i$と$v_j$を除いたものと、総頂点 n から頂点$v_k$を除いたものの積と媒介値の 2 倍数$2BC(v_k)$の比によって表される。

媒介中心性を他の表現をすると、
ノードi、媒介中心性Cb(i)、iを通るj、k間の最短経路の数$g_{jk}(i)$、j、k間の最短経路の総数$g_{jk}$、ネットワークの総ノード数Nとると、

$$Cb(i) = \frac{\sum_{j<k} g_{jk}(i) / g_{jk}}{\frac{1}{2}N(N-1)}$$

（ネットワーク上に存在するノードペアの最短経路がどの程度ノードiを経過しているかの比で表現される（2004，松田））

この式は、$Cb_{ij}(v_k)$の式とは違い、媒介ノードを特定せずに、

2 ノード間の最短経路でいくつのノードを経過するかを計ることによって媒介中心性を求めている。

## 4、固有ベクトル中心性

以下の、ボナチッチ中心性（Bonacich Centrality）は、固有ベクトル中心性の代表的な手法である。

頂点$v_i$、頂点$v_i$が$v_j$を指名する結合強度$r_{ij}$、頂点$v_i$の中心性$C_i$、頂点$v_j$の中心性$C_j$、定数$\lambda$

頂点$v_i$の中心性$C_i$は、$v_i$が$r_{ij}$の強さで結合する頂点$v_j$の中心性$C_j$に依存する。

$$C_i = \frac{\sum r_{ij} C_j}{\lambda}$$

多くの頂点とつながっている中心性の高い頂点との連結が、中心性を高くするという考え方で、数学的には隣接行列$\lfloor r_{ij} \rfloor$を求めていることの他ならない．（2006，中村）

そのほかに、ポータルサイトのページランクのアルゴリズムも、固有ベクトルによって、中心性を求める。以下のものが有名である。

・HITS アルゴリズム(1998, Kleinberg)

・PageRank アルゴリズム(1998, Brin & Page)

*Closeness

クロースネス中心性を求める

1.Input

2.Output

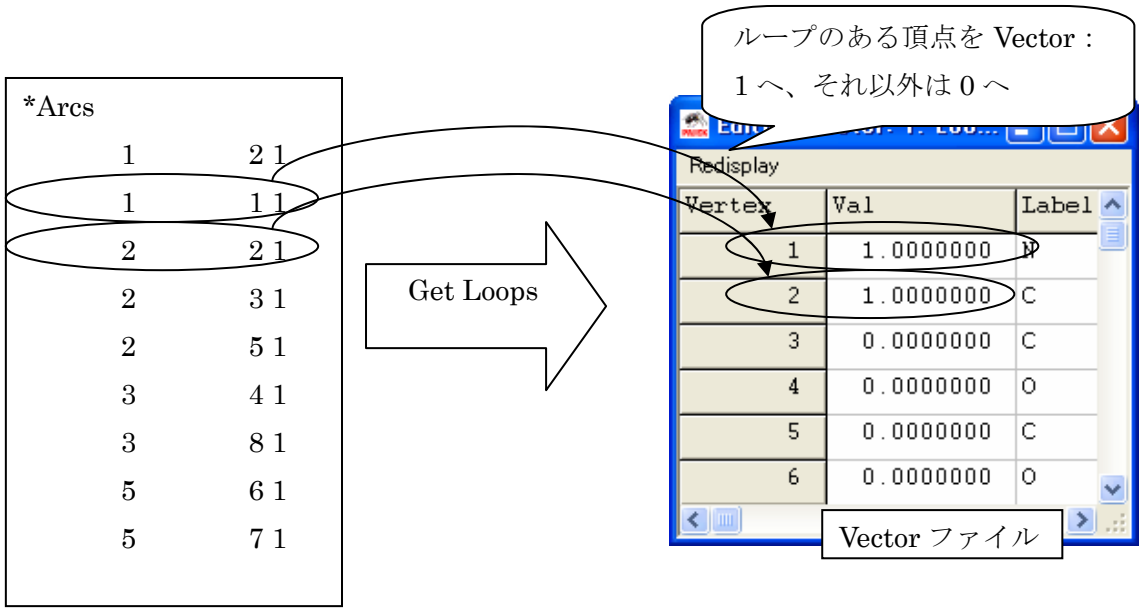
3.All

*Betweenness

媒介（Betweenness）中心性

<Get Loops

ループベクトルを保存する。




図：Get Loops

<Get Coordinate

x か y か z の Coordinate を取得する。一回ですべての Coordinate を取得することも出来る。

*x

x の Coordinate

*y

y の Coordinate

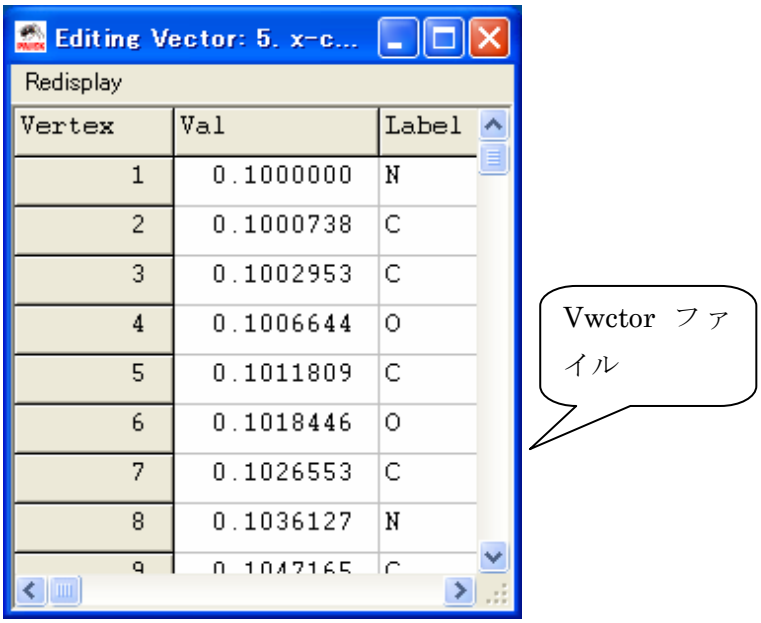
*z

z の Coordinate

*ALL

すべての Coordinate

図：x の Coordinate の例




図：x の Coordinate の例

<Important Vertices

重要な頂点を見つける。

*1-Mode:Hubs/Autorities

1-mode ネットワークのハブやオーソリティを見つける

ハブ：有向線グラフにおいて、出次数が他の頂点と比べて圧倒的に大きいもの

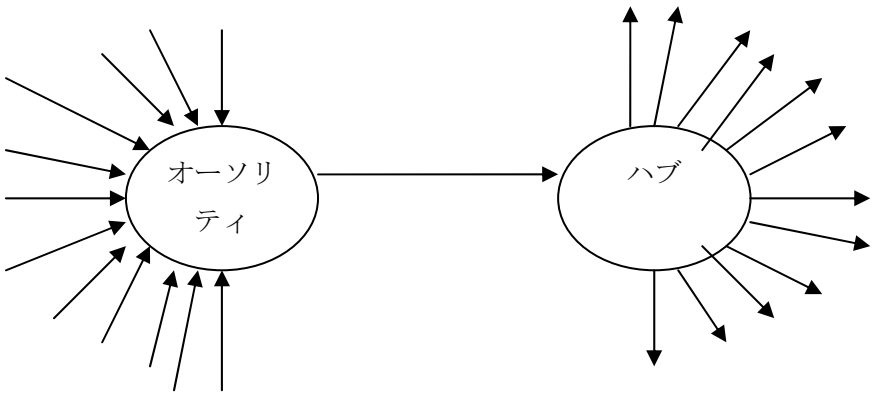
オーソリティ：有向線グラフにおいて、入次数が他の頂点と比べて圧倒的に大きいもの

オーソリティとハブのネットワーク内の意味は、

・ハブやオーソリティへの次数が高い（集中度が高い）

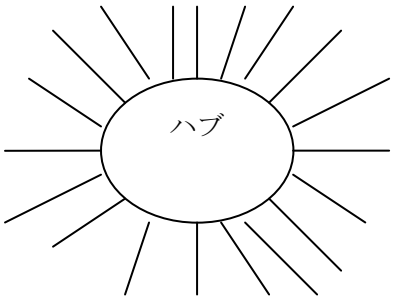
・ネットワーク内のハブやオーソリティの数が多い

ほど、ネットワークの平均パス数は大幅に減少し、伝播しやすくなり、つまり小さい世界（ネットワーク）になる。




図：オーソリティとハブ

無向線の場合は、入次数・出次数の区別なく、主にエッジが集中している頂点はハブといっている。つまり、有向線のときのみ、オーソリティとハブと区別する。




図：ハブ（無向線グラフのとき）

入力項目

①How many hubs（ハブの数）

②How many authorities(オーソリティの数)

以上の入力項目に従い、そのネットワークで定義されたハブ・オーソリティの数に従い、それらを探す。

出力は、Pertition：ハブやオーソリティとその他のノードのクラス分けと、
vector：ハブやオーソリティの度合いの計算結果の 2 つが出力される。

*2-Mode:Important Vertices

最初の部分集合（Fiest subset）と 2 つ目の部分集合(Second subset)の重要頂点を求める。

入力項目

① How many important vertices from first subset
(First subset からの重要頂点の数)

② How many important vertices from first subset
(Second subset からの重要頂点の数)

<Structural Holes

burt's measure of constraint(structural holes)

constraint=強制・制約（条件）

burt's measure of constrain=バーツ値制約条件

Structural Holes=構造の穴（複数）

Vector ファイルが生成される。

Network $P_{ij}$ ：クラス番号 i とすべての i の関係の数と比較した i と関係ある j

$a_{ij}$ ：i から j へのラインの値（重さ）

$$p_{ij} = \frac{a_{ij} + a_{ij}}{\sum_k (a_{ik} + a_{ki})}$$

ネットワークに属していてる 2 項制約条件（dyadic constraint） $C_{ij}$ ：

制約条件は、i 上の j 周りの 1 項の穴（Primary hole）が不在である。

接点（contact）j は、あなたの i の以下の企業家の意見の範囲で制約されている。

a) あなたはエネルギーが j に届き、時間の大きな投資を作られた。

b) j はいくつかの投資に対する優位なリターン（黒字）をとるための交渉が出来ることがついたいくつかの Structural Holes（構造の穴）囲まれている。

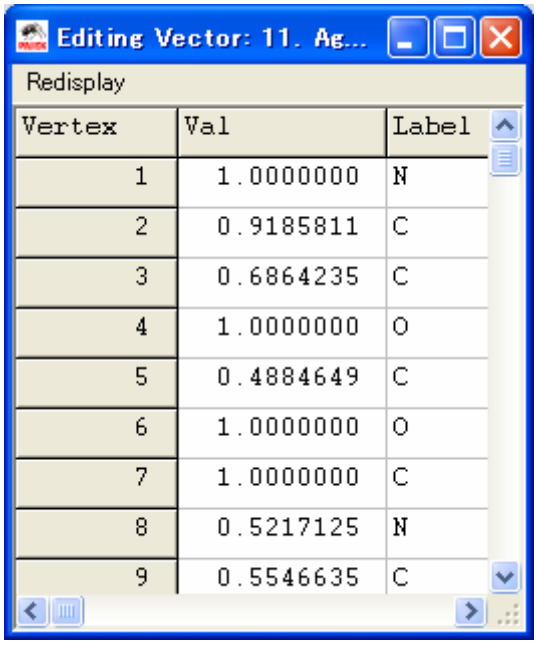
$$C_{ij} = (P_{ij} + \sum_{k,k\neq i,k\neq j} P_{if} P_{kj})^2$$

Vector は制約 $C_i$ の総量から成り立っている。

$$C_i = \sum_j c_{ij}$$

孤立頂点は、

$C_i = 1$




| Vertex | Val | Label |
|---|---|---|
| 1 | 1.0000000 | N |
| 2 | 0.9185811 | C |
| 3 | 0.6864235 | C |
| 4 | 1.0000000 | O |
| 5 | 0.4884649 | C |
| 6 | 1.0000000 | O |
| 7 | 1.0000000 | C |
| 8 | 0.5217125 | N |
| 9 | 0.5546635 | C |

図：Structural Holes

<Clustering Coefficients

Clustering Coefficients：クラスタリング係数

クラスタリング係数を計算する。（無向線ネットワークの）

無向線ネットワーク内にある違う固有の傾向の係数を計算する。

$\deg(v)$ ：頂点の v の次数

$\left|E(G_1(v)\right|$ ：頂点 v の 1-近隣（1-neighborhood）内にある頂点（vertices）の周りのラインの値（重さ）

$\left|E(G_2(v)\right|$ ：頂点 v の 1 および 2-近隣（1-neighborhood）内にある頂点（vertices）の周りのラインの値（重さ）

* $CC_1$

1-近隣（1-neighborhood）のみを考慮した係数

$$CC_1(v) = \frac{2\left|E(G_1(v))\right|}{\deg(v)\cdot(\deg(v)-1)}$$

$$CC_1(v)' = \frac{\deg(v)}{MaxDeg} CC_1(v)$$

*$CC_2$

2-近隣（1-neighborhood）を考慮した係数

$$CC_2(v) = \frac{|E(G_1(v))|}{|E(G_2(v))|}$$

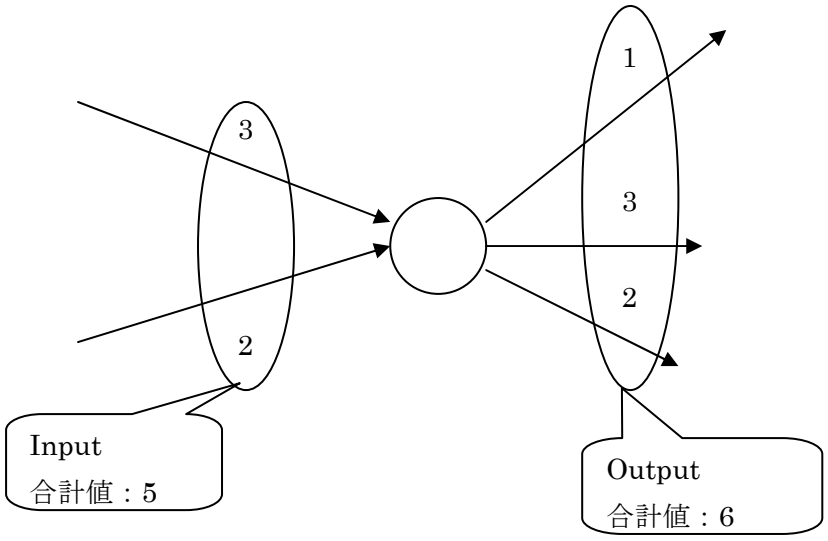
$$CC_2'(v) = \frac{\deg(v)}{MaxDeg} CC_2(v)$$

もし、$\deg(v) \leq 1$ならすべての頂点 v の係数(Coefficients)は 0 になる。

Vector と Partition ファイルが生成される。

<Summing up Values of Lines

頂点のつながったラインの入力出力をの合計値を出す。




図：Summing up Values of Lines の処理

生成されるファイルは Vector のみ。（Partition は生成されない）

*Input

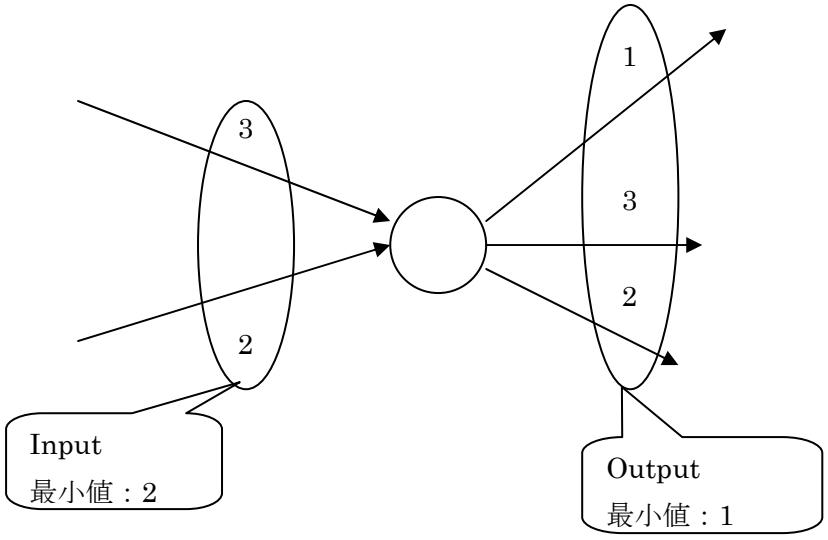
*Output

*All

Input+ Output

<Min of Valuesof Lines

頂点のつながったラインの入力出力をの最小値を出す。




図：Min of Valuesof Lines の処理

生成されるファイルは Vector のみ。（Partition は生成されない）

*Input

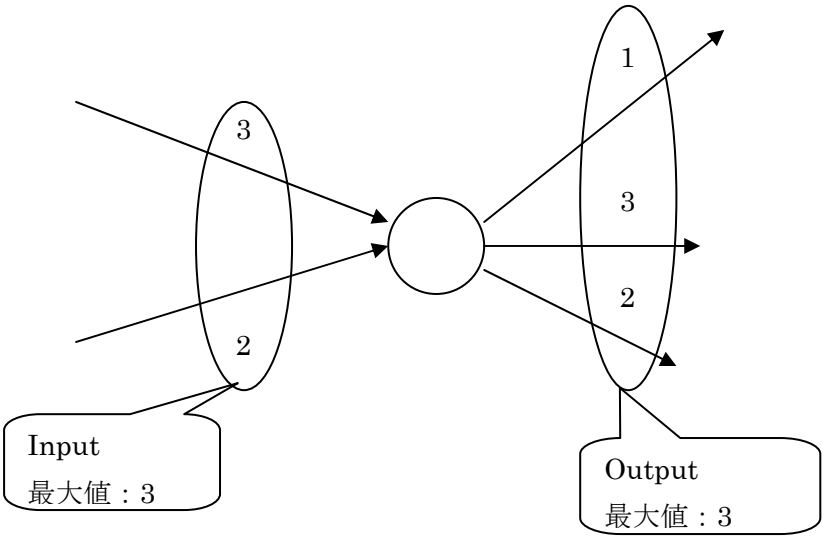
*Output

*All

Input+ Output

<Max of Values of Lines

頂点のつながったラインの入力出力をの最大値を出す。




図：Max of Valuesof Lines の処理

生成されるファイルは Vector のみ。（Partition は生成されない）

*Input

*Output

*All

Input+ Output

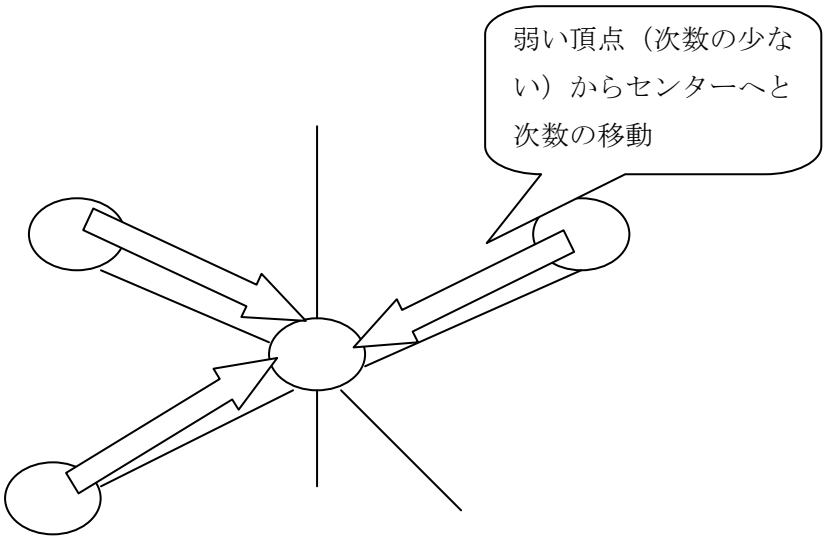
<Centers

robbery アルゴリズムに従い centers を求める。頂点は（vertices）それらから盗まれたそれらの近隣より高い次数を持つ。

Centers：

はじめのとき、初めの濃度を頂点へと与えるときは、それらの頂点か値 1 での始まりに従う。

弱い頂点を見つけたときは、近隣はそれらの濃度に従ったから盗む、あるいは、それらは、同じ合計値から盗む。

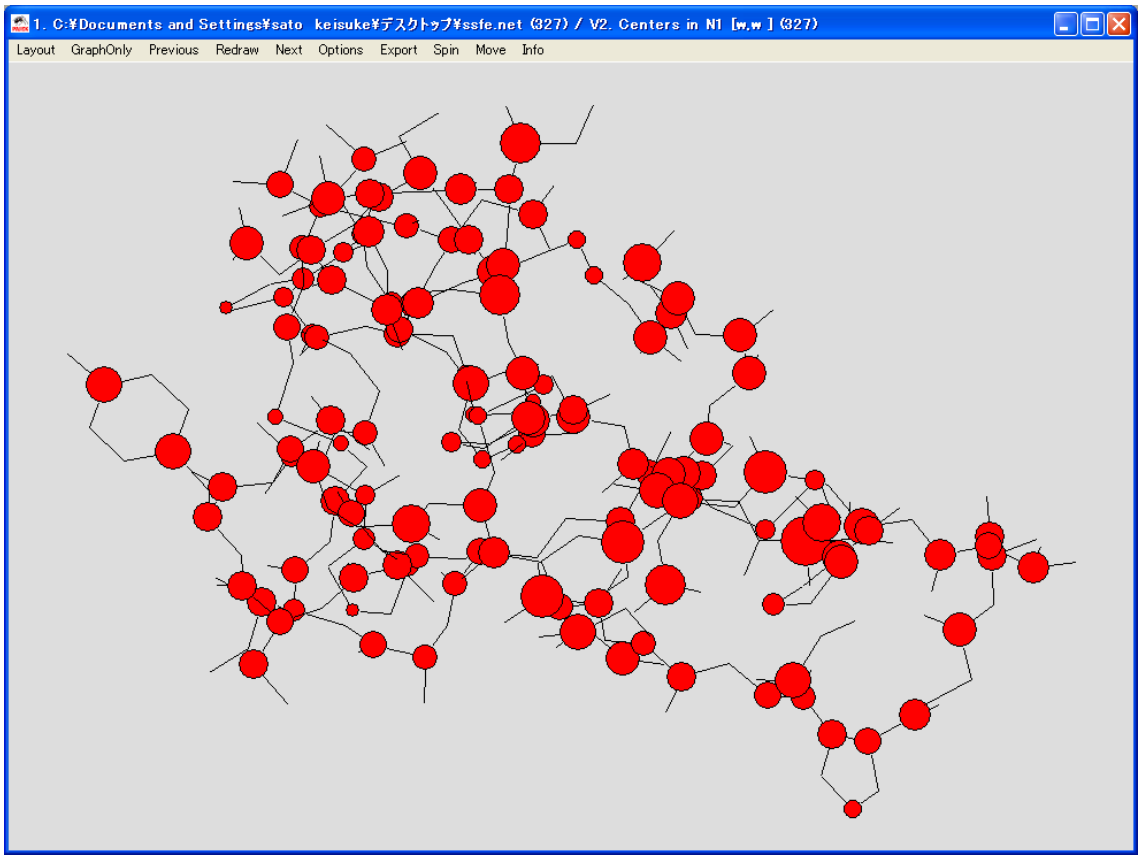



図：Center 生成の概念図

聞いてくること

① Start with weighted values(degrees of vertices)?

② Use weighted algorithm?




図：Center を求めた後の可視化

<Pcore

コアを求める。

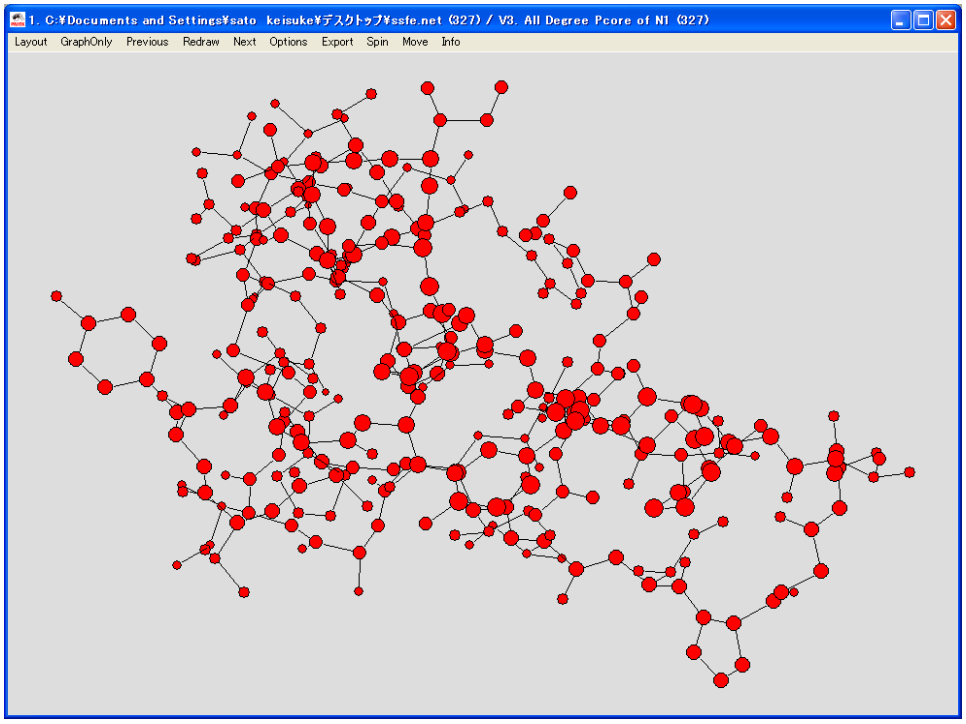
core=k-core+m-core

m-core=それぞれの線が m 以上の多重性を持つ最大サブグラフのこと

*Degree

ordinary cores

普通のコア。




図：Core All

1. Input

入次数

2. Output

出次数

3. All

無向線

4. Input+Output

入次数と出次数との合算

5. Max(input,output)

入次数と出次数の最大値

*Sum

ラインの値（重み）を考慮に入れる。（Pcore の内側にあるラインの値（重み）を合計する。）

1. Input
   入次数
2. Output
   出次数
3. All
   無向線
4. Max(input,output)
   入次数と出次数の最大値

*Max

ラインの値（重み）を考慮に入れる。（Pcore の内側にあるラインの値（重み）の最大値をとる。）

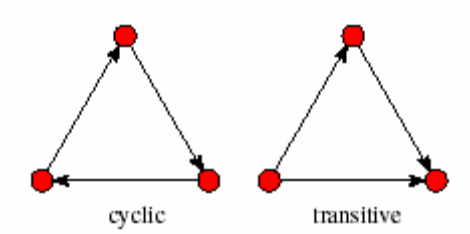
1. Input
2. Output
3. All

>Count

定義済み（pridefined)のリングに線がどれくらい属しているか

<3-Rings

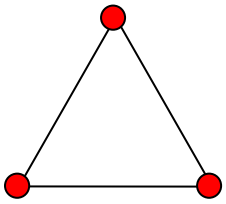
3-ring の定義済みリングにどれだけ属しているか。*.net ファイルが変更される。



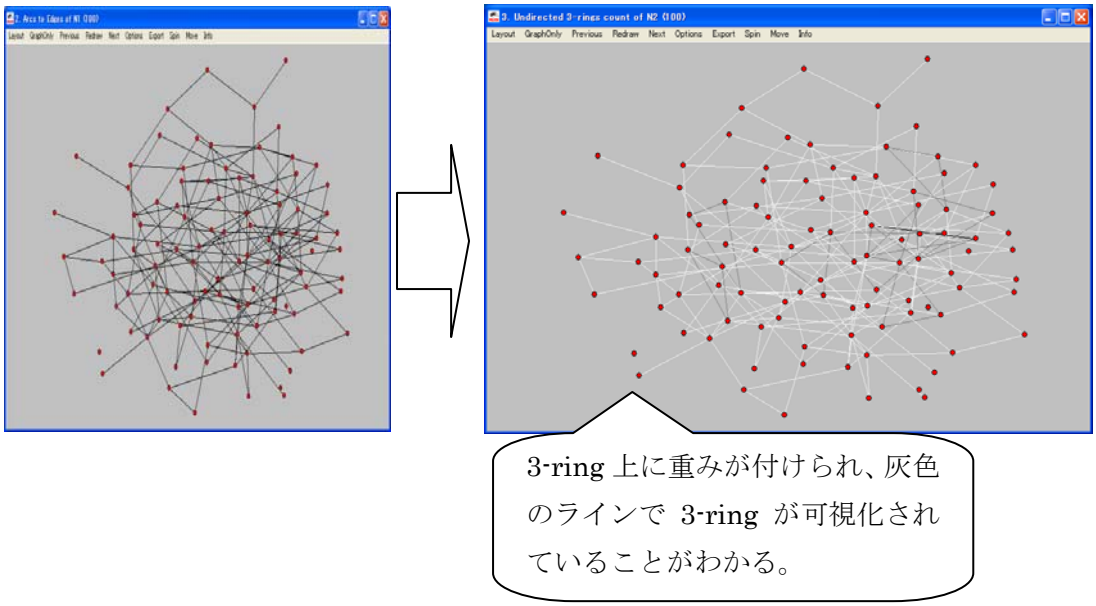

図：ラインが Cyclic か transitive（Shortcut）の 3-rings に属している[mamu]

*Undirected

無向線ネットワークの 3-ring の抽出。有向線ネットワークの 3-ring と違い、3 頂点にリング状にラインがリンクされているものが、無向線の 3-ring である。



図：無向線ネットワーク(Undirected)の 3-ring



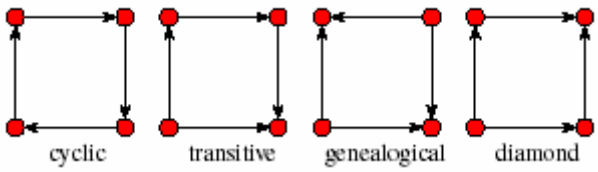

図：Undirected の 3-ring の例

*Directed

無向線ネットワークの 3-ring の抽出。

<4-Rings

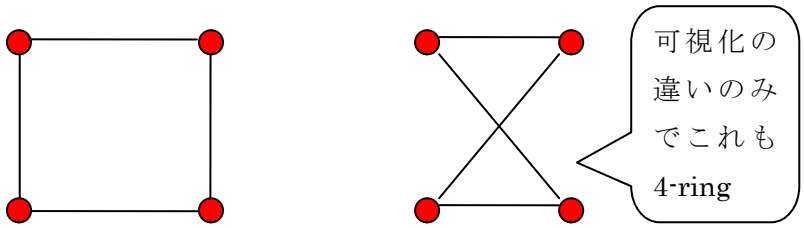
3-ring の定義済みリングにどれだけ属しているか。*.net ファイルが変更される。



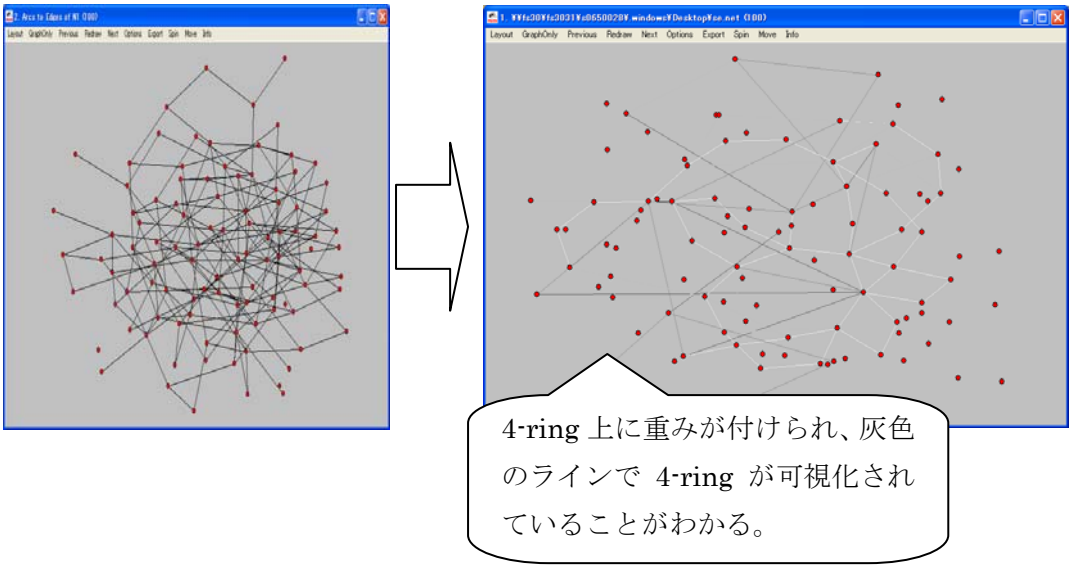

図：4-rings 上の有向線の種類[manu]

*Undirected

無向線ネットワークの 4-ring の抽出。有向線ネットワークの 4-ring と違い、4 頂点にリング状にラインがリンクされているものが、無向線の 4-ring である。




図：無向線ネットワーク(Undirected)の 4-ring

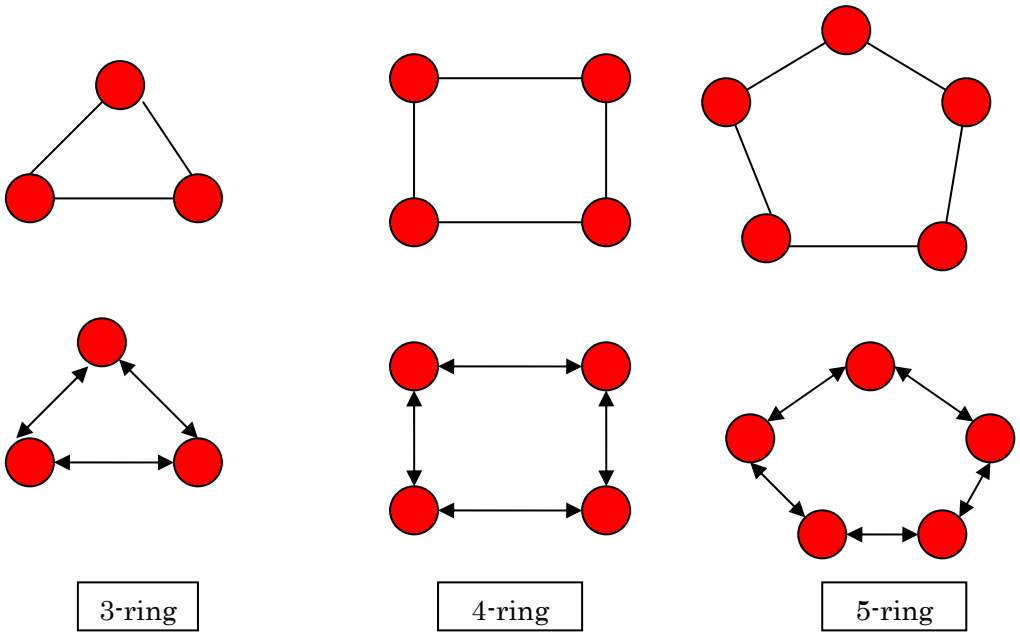



図：Undirected の 4-ring の例

*Directed

有向線ネットワークの 4-ring の抽出。

ちなみに、ring には以下のような種類がある。ただ、3-ring と 4-ring が一般的で、Pajek でもこの 2 種のみカウントできる。




図：3-ring から 5-ring の可視化

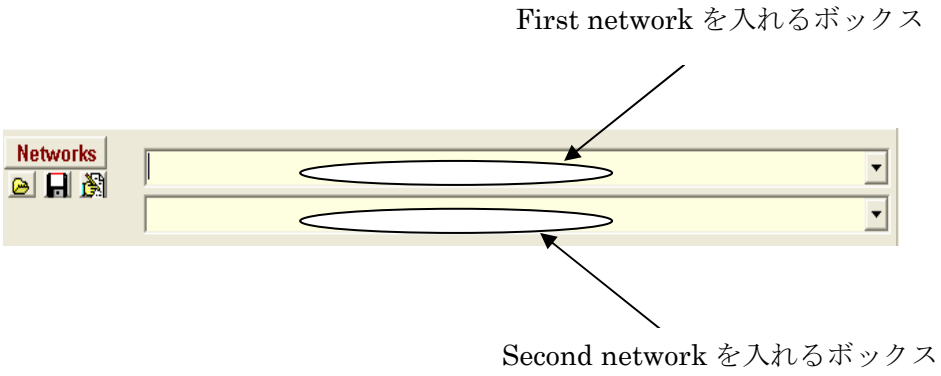
## 3.3Nets

２つのネットワークに関する処理。

1 つ目(Main)のネットワークを『First network』といい、

2 つ目のネットワークを『Second network』という。

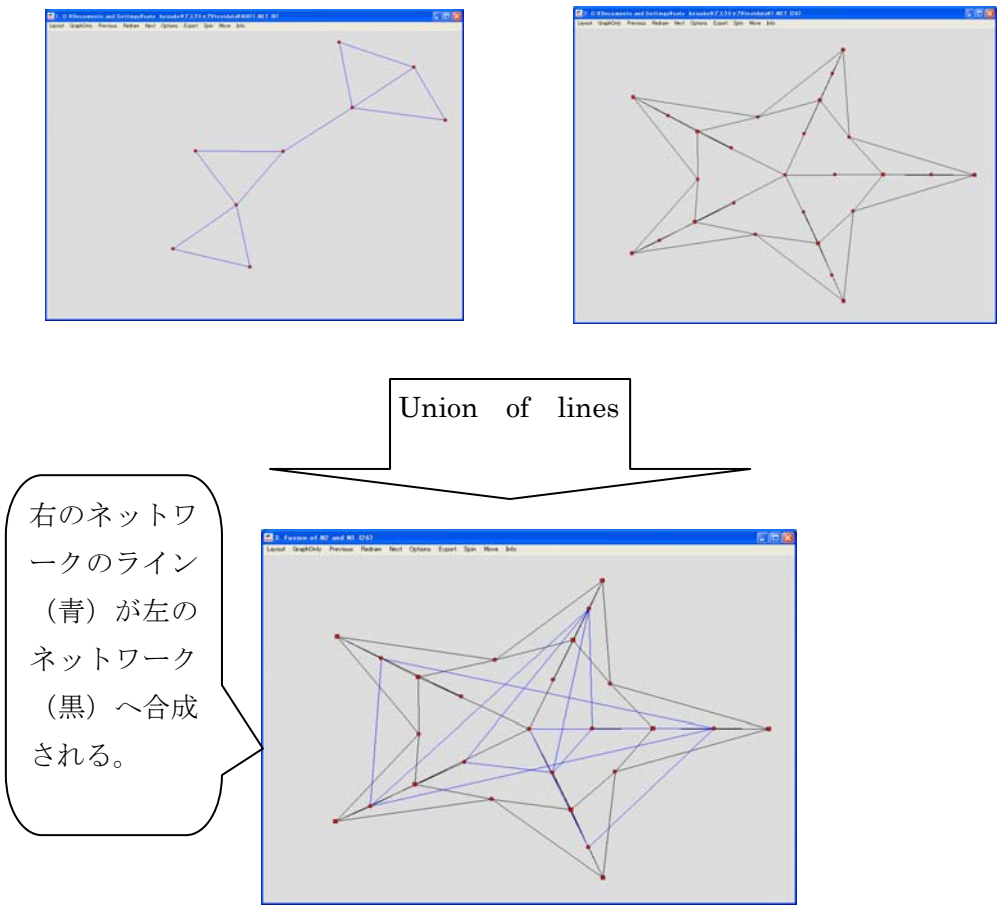
主に、Second network を First network に追加するという処理である。




図：2 つのネットワークとメインスクリーンのボックスの関係

>Union of lines

2 ネットワークのラインを統合する




図：Union of lines

>Cross-Intersection

選ばれた 2 ネットワークの交差させる

>Intersection

選ばれた 2 ネットワークを関係する数字のところで交差させる

>Cross-Differnce

選ばれた異なったネットワークを交差させる

>Defference

異なったネットワークのラインを引く。Union of lines と逆の操作。

>Union of Vertices

頂点を統合する

>Fragment(1 in 2)

2-mode ネットワークないで（1-mode ネットワークによって定義された）すべての種類の断片を見つける。

<Find

コマンドを実行する。

<Options

適当な断片モデルを選ぶ。

*Induced

*Labeled

ラベルに一番合うもの

*Check Values of lines

*Check relaton number

*Extra subnetwork

サブネットワークの頂点は特定の断片やに含まれるかあるいは一致するか？

・Retain all vertices after extracting

*Same vertices determine one fragment at most

同じ頂点の集合のどの断片として扱えるか？

*Repeating vertices in fragment allowed

>Multiply First*Second

選択された 1-mode か 2-mode のネットワークを増やす。

選択された 1-mode や 2-mode は Second network のこと

>Shrink coordinates(1to2)

<Partiotion

<Hierarchy

## 3.4Operetions

one network and something else is needed as input

>Shrink Network

Shrink Network=縮むネットワーク？

ネットワークを縮ませるオペレーション？

Shrink Network=ネットワークの種類（IN　Bolockmodel?）

<partition

<hierarchy

>Extract from Network

サブネットワークを以下の対象から引き出す処理

<Partition

サブネットワークを Partition に従い引き出す。(Partition のクラスの range を

引き出す。）

Partition ファイル(*.clu)ファイルがあらかじめないと、処理を受け付けない。

入力数字

① クラス番号

このクラス番号のネットワークをサブネットワークとして抽出する。

処理のアウトプット

①*.net に生成されたサブネットワークが入る

②*.clu には、選択されたクラス番号のみの Partition が入る

<Cluster

選択されたクラスターからサブネットワークを引き出す。

<to GEDCOM

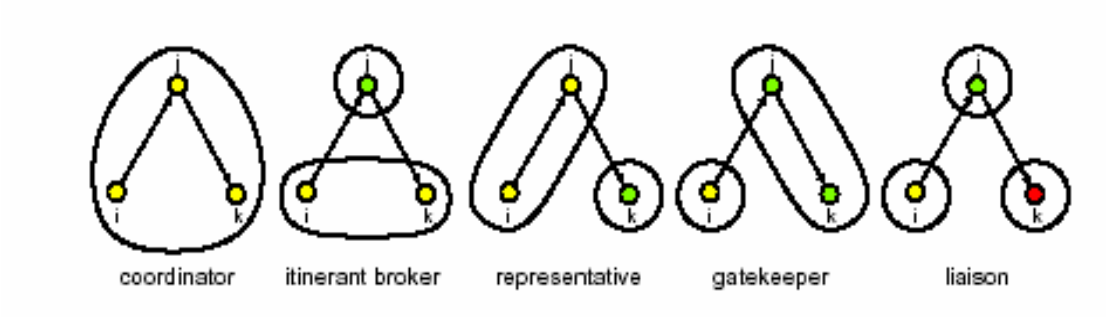
新しい GEDCOM ファイルに選択された Parition（Weakly connected component）から生成された、サブ系図(sub-genealogy)を入れる。

>Brokerage Roles

partition ファイルが必要

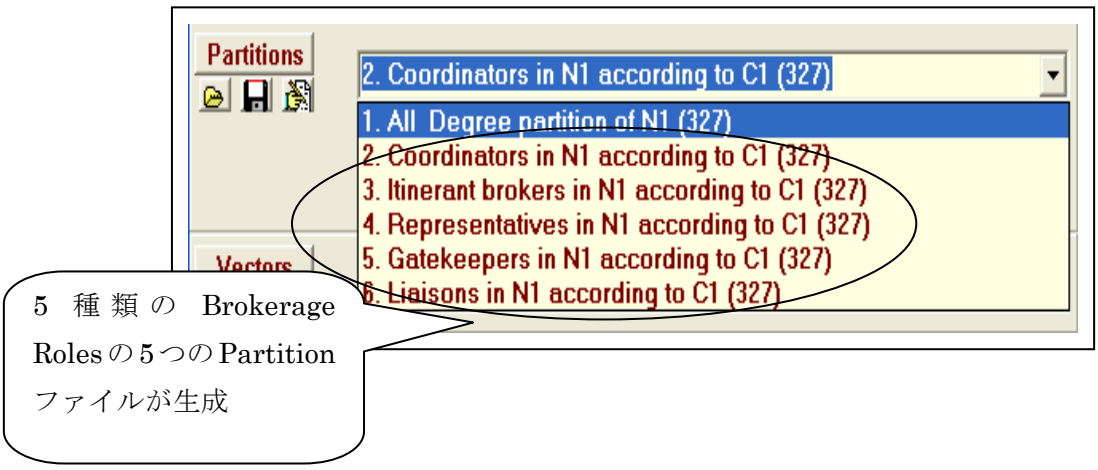
Brokerage：仲買、仲介

Brokerage Roles：




図：Brokerage Roles[Reference Manual]

処理すると、以上の 5 種類の Brokerage Roles の Partition ファイルが 5 つ生成される。




>Dissimilarity

Dissimilarity：差異性（=類似性）

共通の近隣の値にしたがったクラスター内の頂点間の選択された Dissimilarity Matrix－つまり類似性行列（$d_1, d_2, d_3, d_4$）を計算する。Corected Euclidean-like $d_5$ Manhattan-like $d_6$ の類似性も同様に計算する。

頂点が入っているクラスターファイル(*.cls)が必要。

$N_v$：入次数、出次数あるいは普通の次数（入出無関係の次数）の近隣の頂点 v の集合

＋：システム的加算

∪：和集合

\：差集合

｜：集合の濃度

1st maxdgree,2nd maxdegree：最大次数と 2 番目の最大次数

頂点 v のそれ自身の近隣を含めるか含めないか、また、（もし、頂点の値が低いなら）Upper triagle（無向線ネットワーク）か完全行列（有向線ネットワーク）をレポートウインドウに表示できる。

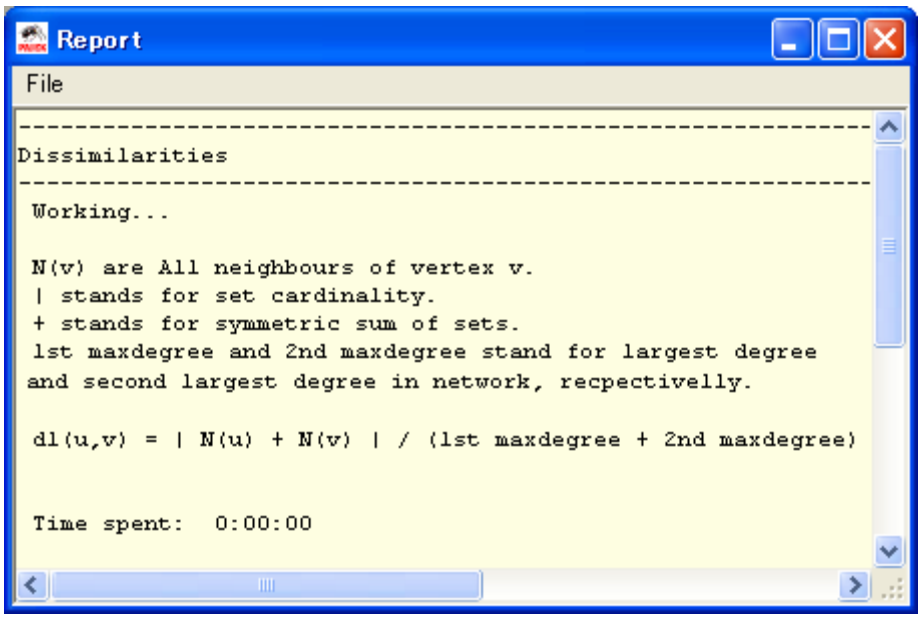
Corected Euclidean-like $d_5$ Manhattan-like $d_6$ は、いくつかの頂点上の行列 $Q=[q_{uv}]$ からなっている。たとえば、隣接行列や距離行列など。パラメーターq はたいてい 1 か 2 の値の集合である。$N_u = N_v = 0$ の時は、すべての類似性行列（$d_1, d_2, d_3, d_4$）は

1 となる。

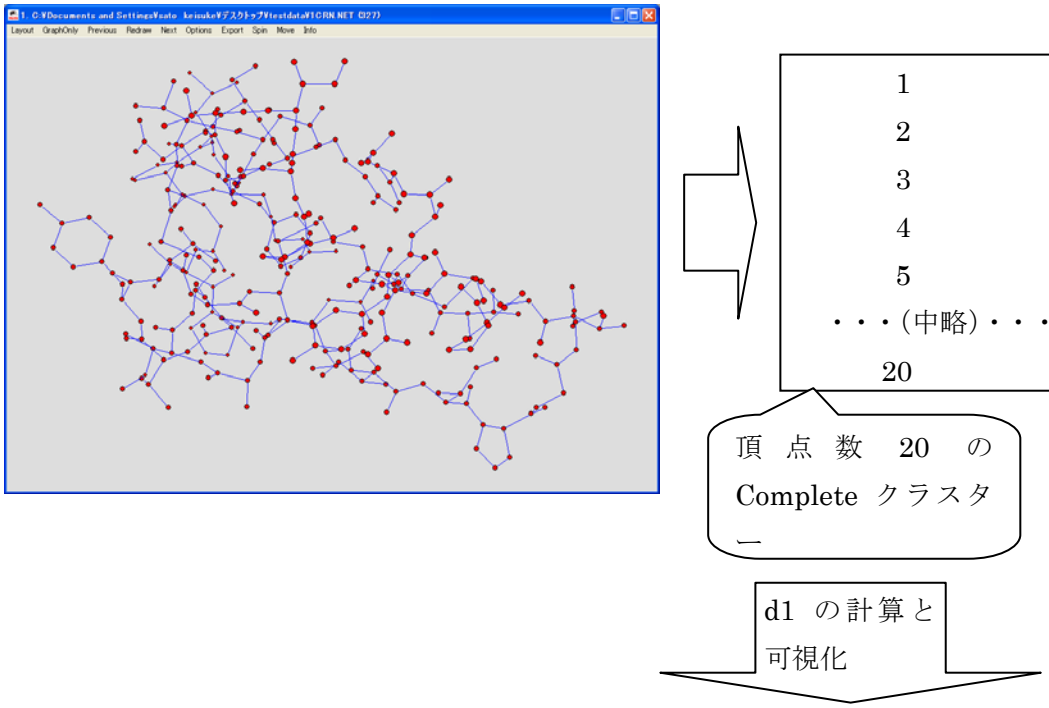
$$
\begin{aligned}
d_1(u,v) &= \frac{|N_u + N_v|}{\text{1st maxdegree} + \text{2nd maxdegree}} \\
d_2(u,v) &= \frac{|N_u + N_v|}{|N_u \cup N_v|} \\
d_3(u,v) &= \frac{|N_u + N_v|}{|N_u| + |N_v|} \\
d_4(u,v) &= \frac{\max(|N_u \setminus N_v|, |N_v \setminus N_u|)}{\max(|N_u|, |N_v|)} \\
d_5(u,v) &= \sqrt{\sum_{\substack{s=1 \\ s \neq u,v}}^{n} ((q_{us} - q_{vs})^2 + (q_{su} - q_{sv})^2) + p \cdot ((q_{uu} - q_{vv})^2 + (q_{uv} - q_{vu})^2)} \\
d_6(u,v) &= \sum_{\substack{s=1 \\ s \neq u,v}}^{n} (|q_{us} - q_{vs}| + |q_{su} - q_{sv}|) + p \cdot (|q_{uu} - q_{vv}| + |q_{uv} - q_{vu}|)
\end{aligned}
$$

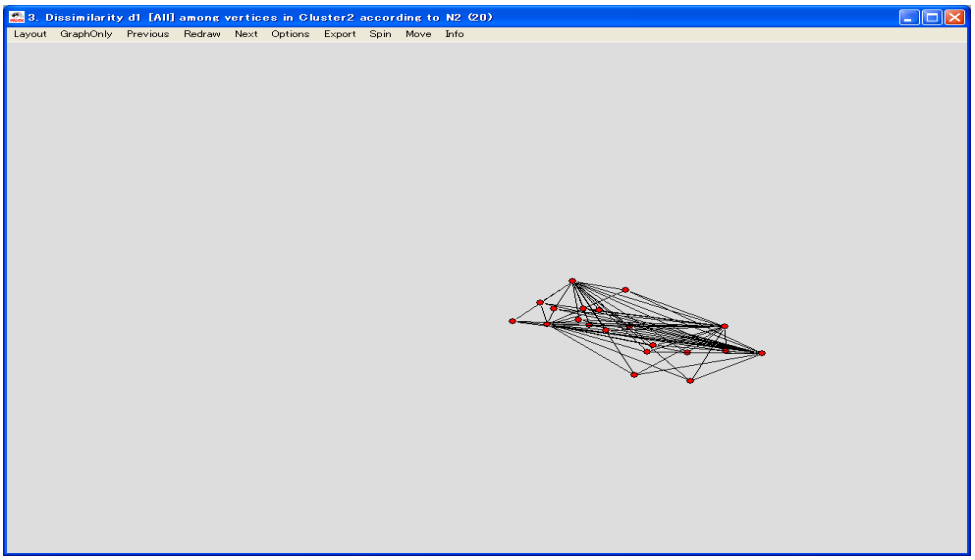
<d1

1. Input
2. Output
3. All




図：Dissimilarity d1 処理後のレポート




図：Dissimilarity の d1 処理過程

<d2

1. Input
2. Output
3. All

<d3

1. Input
2. Output
3. All

<d4

1. Input
2. Output
3. All

<d5-corrected Euclidean

<d6-corrected Manhattan

<Option

>Vector

ネットワークと Vector に関する処理。

<Network*vector

普通の Vector の行列（ネットワーク）の掛け算。結果は、新しい Vector である。

入力手順

①Number of repetition（反復回数）

<Vector#Network

結果は、新しいネットワークである。

*Input

Vector の値と一致するネットワーク内の入有向線の掛け算。

i 次（i-th）の Vector のコンポーネントと i 次(i-th)の行列の列を掛ける。

*Output

Vector の値と一致するネットワーク内の出有向線の掛け算。

i 次（i-th）の Vector のコンポーネントと i 次(i-th)の行列の行を掛ける。

<Harmonic Function

Harmonic Function：同調関数

$(G, a)$：重さ関数（Weight functioin） $a(x, y)$ を持った結合した重さグラフ

（Connected weighted graph）

S：頂点 $V(G)$ の部分集合

もし、

$$f(x) = \frac{1}{A(x)} \sum_{y} (a(x, y) f(y))$$

$\forall x \in V(G) \setminus S$

$A(x) = \sum_{y} a(x, y)$ なら

関数 $f:V(g)\to R$ は、境界(Boundary)S を持つ(G,a)上の同調(Harmonic)という。

Pajek では以下を実行する。

・関数 f は Vector によって定義づけられる。

・重さ関数$(G,a)$は、（数値化された）ネットワークによって与えられる。

・部分集合 S は、Partiiton によって定義づけられる。クラス 1 の頂点は、（頂点と合う）部分集合 S に入れられ、他の頂点は$V(G)\setminus S$ に入れられる。

・追加的に、Permutattion はコンピュータ上の頂点のオーダーによって定義づけられる。

*Input

入力近隣

*Output

出力近隣

*All

入力出力を区別しない近隣

<Summing up neighbors

*Input

*Output

*All

<Min of neighbors

*Input

*Output

*All

<Max of neighbors

*Input

*Output

*All

<Put Loops

<Put Coordinate

<Diffusion Partition

<Islands

*Vertex Weights

*Vertex Weights[simple]

>Transform

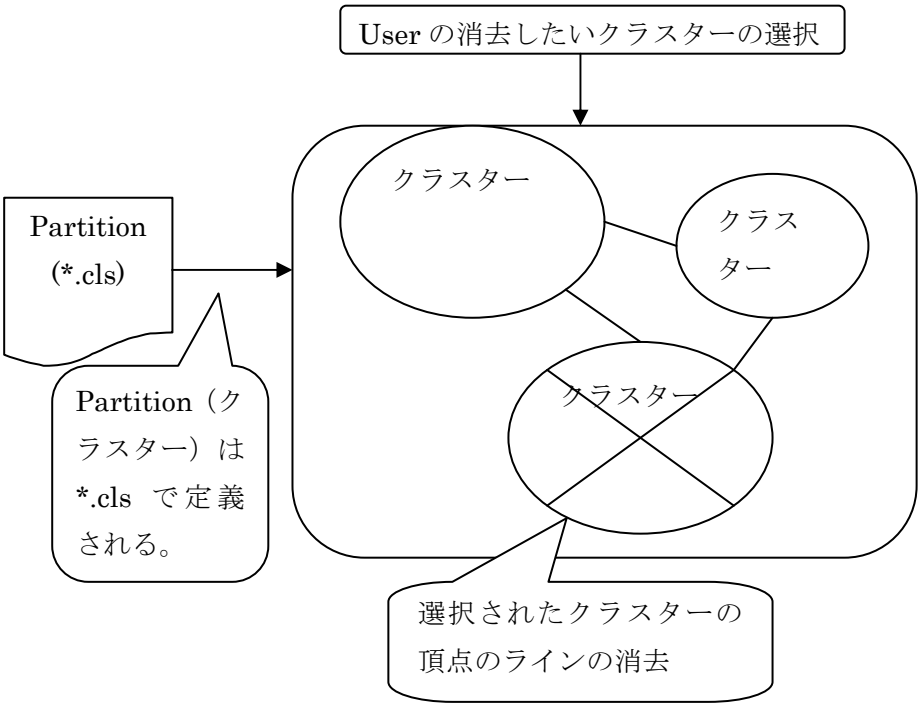
ネットワークを、Partition、Cluster、Vector に従い変形させる。以下のようなコマンドによって変形させる。

<Remove Lines

Partition（*.clu）に従い、ラインを消去する。

*Inside Clusters

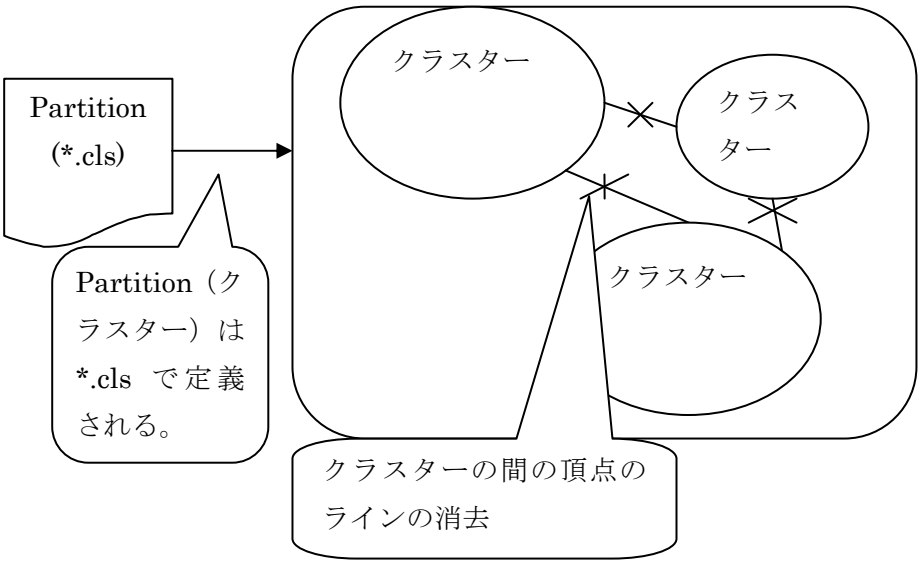
選択された同じクラスター内の頂点のラインを消去する。




図：Remove Lines *Inside Clusters

*Between Clusters

異なったクラスター間の頂点のラインを消去する。




図：Remove Lines * Between Clusters

*Inside Clusters with value

1.lower than Vector value

2.higher than vertor value

<Add

*Arcs from Vertex to Cluster

*Arcs from Cluster to Vertex

<Direction

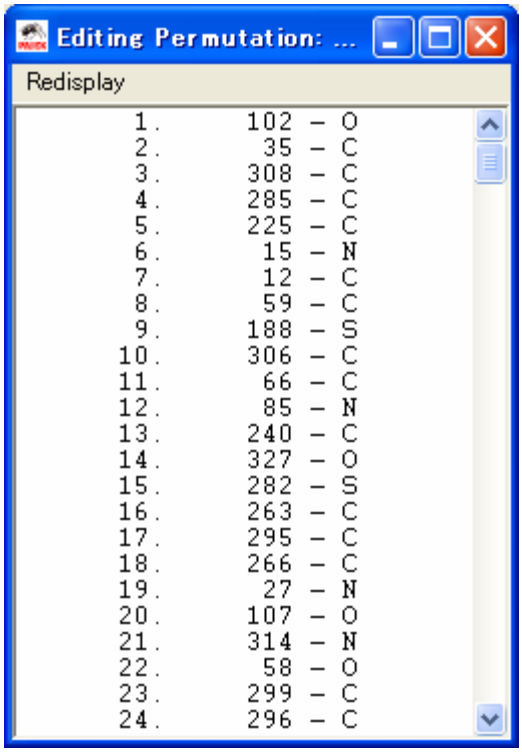
*Lower->Higher

*Higher->Lower

<Vector(s)->Line values

>Reorder

再配列(Reorder)する。再配置には、Permutation ファイル（*.per）を使用する。




図：*.per のメインスクリーンの Edit の表示の例

頂点 1 は頂点 102 へ、頂点 2 は頂点 35 へ再配置される。

<Network

Network(*.net)ファイルの頂点を、選ばれた Permutation に従い再配置する。

<Pertition

Partition(*clu)ファイルの頂点を、選ばれた Permutation に従い再配置する。

<Vector

Vector(*.vec)ファイルの頂点を、選ばれた Permutation に従い再配置する。

>Count neighbor Colors

Partition と Cluster が必要。選ばれたネットワークと Partition のための新しい Parititon を、近隣の選ばれた色を与えた頂点の確率的な互いの頂点のところに生成する。色は、定義されたクラスターファイルにより計算される。

>Coloring

<Create New

Permutation（頂点の再配置のためのファイル）によって秩序付けて定義された頂点の『経過的な』色付け。Paermutation ファイルが結果に依存する割合はかなり高い。

<Complete Old

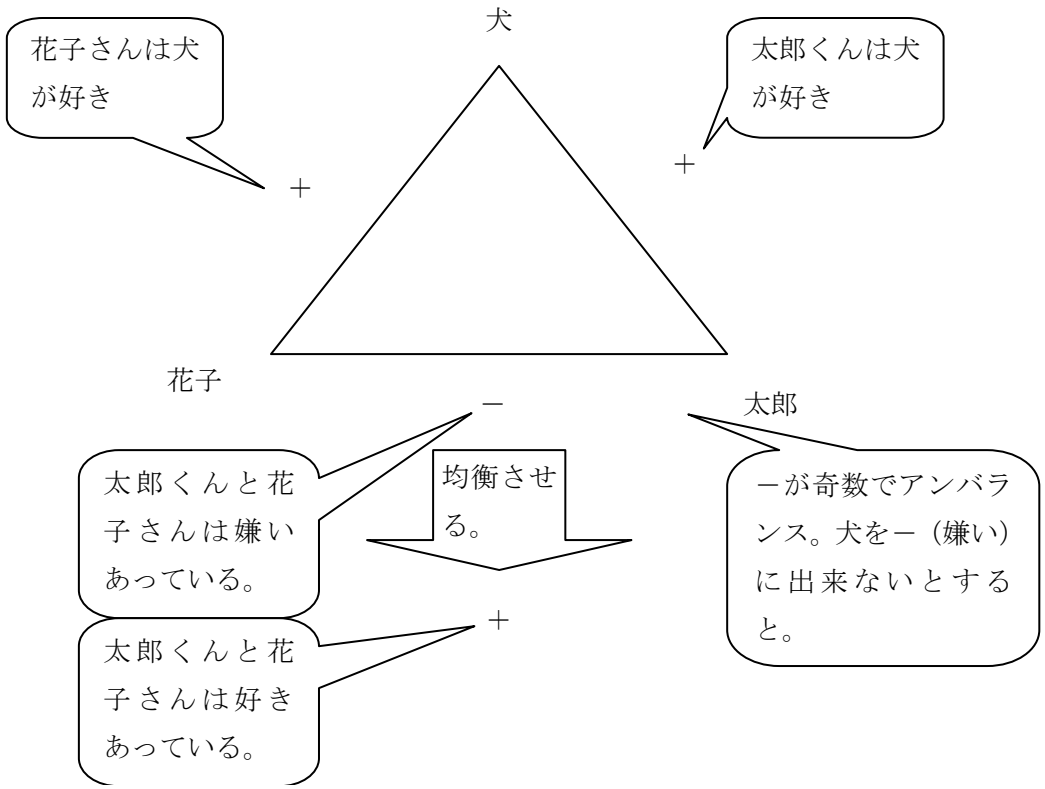
Permutation（頂点の再配置のためのファイル）によって秩序付けて定義された頂点の『完全な』色付け。例えば、手によって幾つかの頂点が色づけされ

たとしても、大半の頂点はいまだに色づけされていない頂点である。(クラス0)。この方法は、適当な色分けを支援してくれるコマンドである。

>Balance

符号付のグラフ Partition を行うためのアルゴリズム。(+、－のラインの値（重み）は、友と敵を表す。）ハイダーのバランスモデルのことである。

3 者間の関係を+（よい）－（悪い）で表した図のモデル。－の悪い関係が偶数のとき、バランス（均衡）し、奇数のときは均衡が崩れ均衡しようとするモデル。




図：ハイダーのモデル例

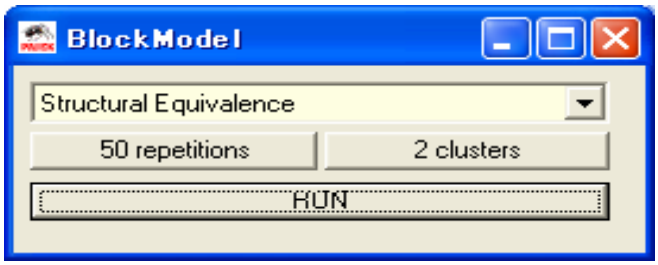
この 3 角形を基本として、大きなネットワークを形成させることも出来る。

>Blockmodeling

ブロックモデルを生成する

<Randam start

ランダム Partition による最適化を始める




<Optimize Patition

選ばれた Partition やその最適化の（判断）基準を表示する。

<Restricted Options

>Genetic Structure

与えられた Partition に従った、与えられた非環状ネットワークの発生の構造を計算する。

>Permutation

与えられた Permutation をネットワークに従い改善する。

<Travelling Salesman

*Rum

*Options

<Seriaton

*1-mode

*2-mode

<Clumping

*1-mode

*2-mode

<R-Enumeration

>Functional Composition

f ：partition か permutation

g：partition か permutation か vector

r：r=(f＊g)[v]=g[f[v]] によって生成された partition か permutation か vector

>Expand Parition

<Greedy Partition

頂点と共に不明なクラスを 0 と置く。

*Input

*Output

*All

<Influencce Partition

<Make Multiple Relations Network

>Expand Reduction

階層による削減によって削減されたネットワークと適当な階層（いつも無向ネットワーク）を再記録する。

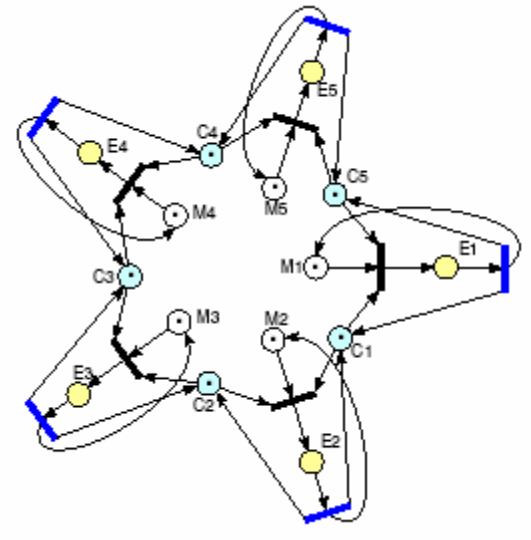
>Identify

Identify(reorder and/or join some units)

>Petri

ペトリネットを生成する。Partition ファイルが生成される。

ペトリネット：コンピュータの並列処理を表すモデル図




図：Petri net[manual]

<Random

<Complete

>Refine Partition

Refine：精製［錬］する，純化する ；洗練する、改善する

選ばれたネットワークに従い Partition を改善する。

<Strong

有向ネットワーク

<Weak

無向ネットワーク

>Leader Partition

層のかなにあるネットワークの頂点のクラスターを見つける。

層・階層（layer）：

Leader：指導者、指揮者、【コンピュータ】先行部分、【印】（*pl.*） リーダー((点線や破線)); 導管

## 3.5Partition

>Create Null Partition

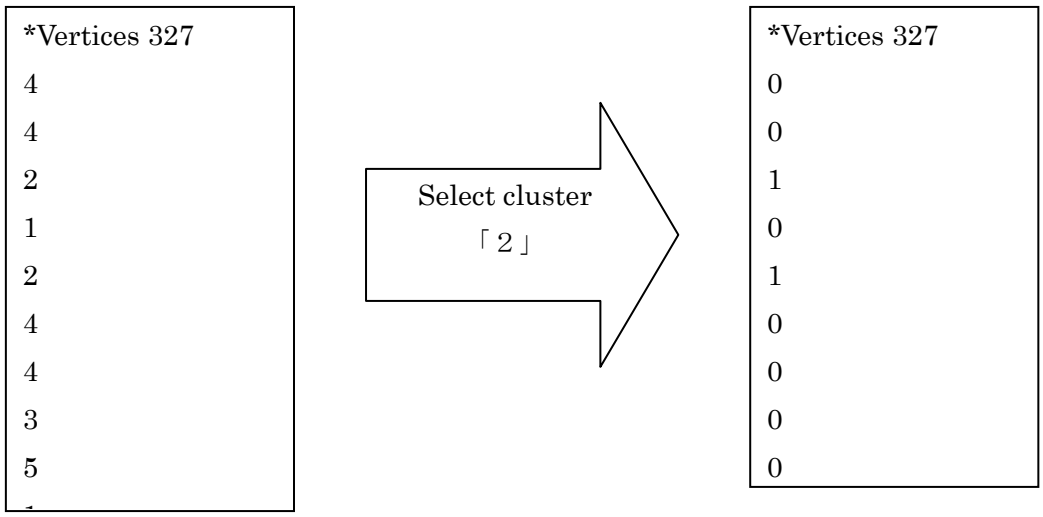
頂点数を定義（Dimension）して、Null Partition を生成する。Null Partition とは、クラス番号がすべて 0 の Partition である。

> Create Random Partition

頂点数とクラス数を定義（Dimension）して、Random Partition を生成する。Random Partition とは、クラス番号がすべて Random に割り振られた Partition である。

>Binarize

選ばれた Partition(*.clu)を Binary（0 か 1）の Partition にする。選択されたクラス番号を 1 に選択されなかったものをすべて 0 にする。よって、このメニューを実行する前に、Partition(*.clu)が Partition ボックスに入っていなければならない。

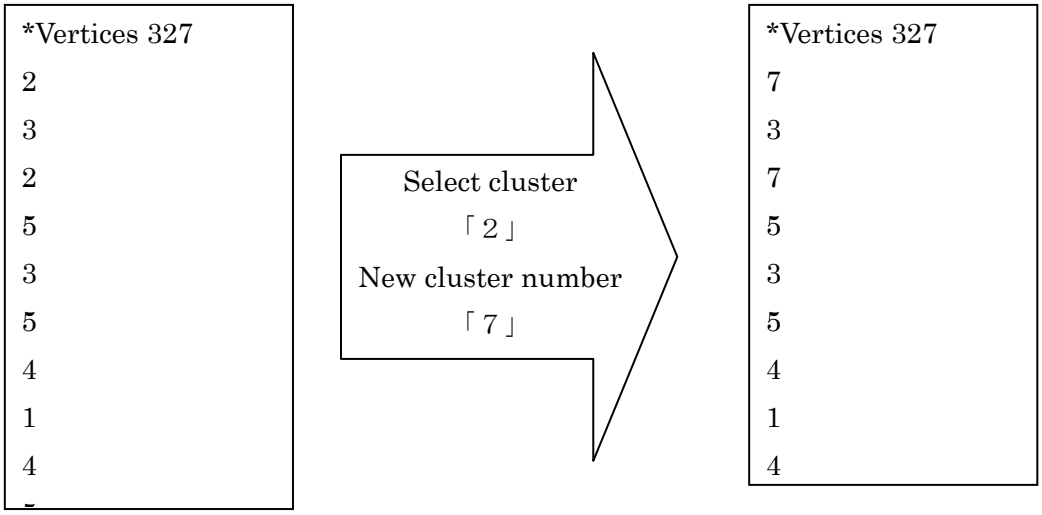



図：Partition> Binarize の処理

特定のクラス番号のクラスターを抽出することが出来る。

>Fuse Clusters

選ばれた Partition(*.clu)の選択されたのクラス番号を、別のクラス番号にする。




図：Partition> Fuse Clusters の処理

>Canonical Partition

Partition から特徴的な（Canonical）Partition を生成する。特徴的な Partition のルールは、

1)頂点 1 はクラス（番号）1

2)頂点 1 のクラスより次の頂点が小さい数ならば、頂点 1 はクラス（番号）2 へ

3)繰り返す

>Canonical Partition[Decreasing frequencies]

Partition から特徴的な（Canonical）Partition を生成する。特徴的な Partition のルールは、

1)クラス 1 は最大頻度の集合の古いクラス

2)クラス 2 は 2 番目の頻度の集合の古いクラス

3)繰り返す

>Make Network

選ばれた Partition によるネットワーク生成。

<Random Network

*Undirected

*Input

*Output

<2-mode network

>Make Permutation

選ばれた Partition による Permutation 生成。

>Make Cluster

選ばれた Partition による Cluster 生成。

>Make Hierarchy

選ばれた Partition による Hierarchy 生成。

>Make Vector

選ばれた Partition による Vector 生成。

>Coun,Min/Max vector

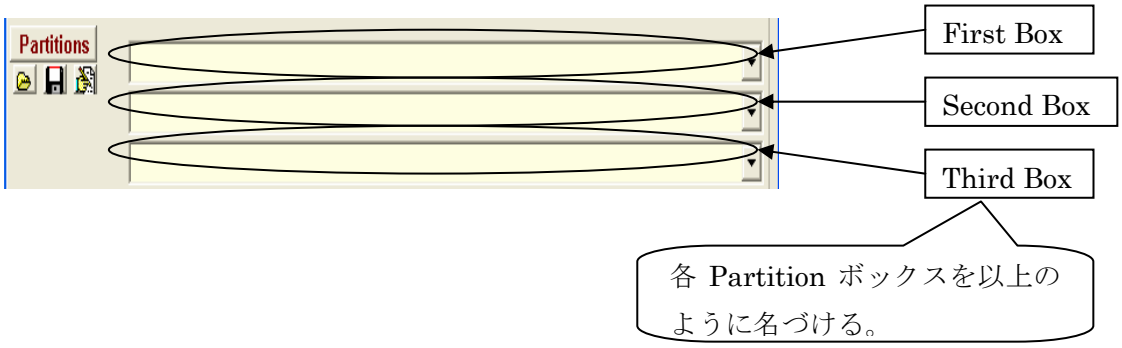
以下の情報

①クラスターの頻度

②Partition によって与えられた最大、最小の Vector の値

## 3.6Pertitions

2 つの Partition ファイルの処理。




図：Partition ボックス

>Extract secound from first

Second box で決定された基準（特定の数値間）を満たす First box の頂点を引き出す。この処理は、実際に頂点について様々な情報が保存された Partiton を持っているとき役立つ。(例：ジェンダー=性の情報を持つ Partition）幾つかの最短パスを持つとき（例えば、選ばれた頂点から 3 パス以内の距離の頂点)、ジェンダーの情報は、Partition 上の処理を実行することなしに失う。

>Add partitions

2 つの Partiotn を追加する。(例えば、非環上ネットワークの Input と Output の近隣が結合しているときに役立つ。)

>Fuse Partitions

2 つの Partition を融合させる。First box の partition の後ろに Second box の partitioin をつなげる。

>Expand

より高い（元の）定義数の Partition を上昇させる。

<First according to secound(Shrink)

First box の Partition を、Second box の Partition が縮小していくものにしたがって上昇させる。

<Insert First into Secound according to Third(Extract)

現在の Partition は、First box から Second box に定義された選択されたクラスを引き出すことによって得られる。この Sub-Patition（部分 Paritiion）は修正されたものである。この処理を使用するとき、この修正された Sub-Paritiion を Firist box の Patitiion の後ろに挿入することが出来る。

>Intersection

2 つの Partition を交差させる。

>Make Random Network

以下のランダムネットワークを生成する。

1)入次数の定義は First partition のデータに従う

2)出次数の定義は Second partition のデータに従う

>Info

2 partition の情報を表示する。

<Cramer's V,Rajski

Cramer's V,Rajski 係数を計算し、レポートする。

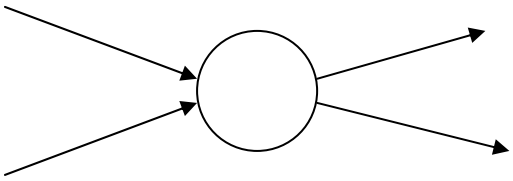
<Spearman Rank

相関係数

## 3.7Vector

Vector：ベクトル

ベクトルを使った作業を行うためのメニューである。ただし、Vector は有向線でなく、このデータ構造は、頂点半径にフィードバックされることに注意する。つまり、ベクトルの値が大きいほど頂点の半径が大きくなる。また、ベクトルの値は頂点によって属性とされている値である。これは、以下のように考えられる。頂点の持つベクトルの値は、頂点に入ってくるベクトルの量であり、ベクトルの量（入ってくるベクトルの多さ）が大きいほど入ってくるものが多いため頂点が大きくなる。

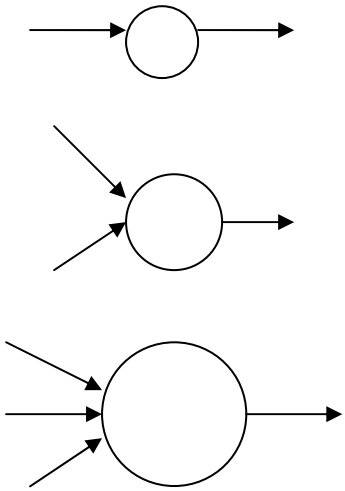


図：頂点 A ベクトルの値に対する概念図

頂点 A に入ってくるベクトル数は、2 であるため頂点 A のベクトル値は 2 である。

出て行くベクトルは無関係であることに注意

また、“無向線（edge）でもこの値は成立する。”

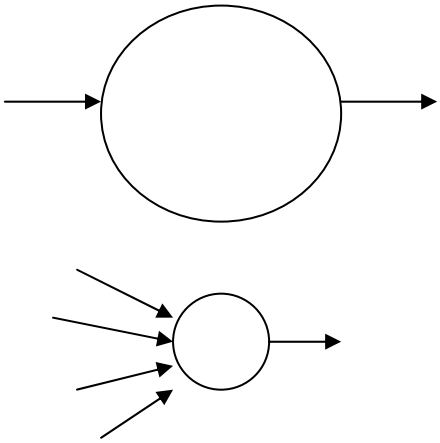


図：入ベクトルが多くなると頂点の視覚化半径が大きくなる

この大きさは各頂点が持っているベクトル量が半径にフィードバックされた結果である

入ってくるベクトルは、

注意点：Vector は上記のようにきれいに決定するものでなく、Vector が頂点に入っていく量を無視して“独立に決定することが多い”。1 つしか入っていないのに値が大きく、たくさん入っているのに値が小さいこともある。



図：入ってくるベクトルと関係なく大きくなる頂点の視覚化半径（ベクトル量）

もともと、これらの概念は抽象的であり、対象とする実ネットワークの意味からこれの解釈を考える。ある地域の交通網ネットワークを解析したのなら、そのネットワークの入ってくるベクトル（道路）が小さいのに頂点（交差点）ベクトル量が大きいなら、その頂点に入るベクトルは交通量が膨大な道路であるという解釈が出来る。

>Create Identity vector

Dimention of Vector を入力する。

値 1 の Vertex が入力した値だけベクトルファイルに生成される

>Extract Subvector

Partition ファイルが存在しないとエラー発生する。

基準は選んだパーティションの種類。

サブベクトルをベクトルから引き抜く。

>shrink vector

ベクトルの値が減少する。

>Make Partition

Partiiton に Vector をコンバートする。

<by Intervals

適当なクラス番号をとった選ばれた分かれている等しいベクトル頂点と値は、適当なクラス番号を取る。

間隔は取る（削除する）ことが出来る。

*First Threshold and Step

*Selected Thresholds

<by Truncating(Abs)

>Make Permutation

Vector から Parmutation を作成する。

>Transform

Vector ファイルを変形させる。

<Multiply by

連続的に加算する。

<Add Constant

Vector の値を連続的に加える。

<Absolute

値の絶対値をとる。

<Absolute+Sqrt

2 乗根の絶対要素

<Truncate

Vector の値の切捨て

<Exp

Vector の値の指数

<Ln

Vector の値の自然対数。

<Power

Vetor の値のべき乗

<Normalize

Vector の値の標準化

*Sum

合計値を 1 とする標準化

*Max

最大値を 1 とする標準化

## 3.8Vectors

2 つのネットワーク（2 つの*.net ファイル）の vector 処理に関するメニュー

>Add Vectors

2 つの Vector ファイルを一緒にする

>Subtract Second from First

First から Second を引く

Difference of selected vectors

>Multiply Vectors

Vectors を増やす。

product of selected vectors

>Divide First by Second

2 つの vector ファイルを分配（分類）する。

>Linear Regression

2 つの Vector ファイルを線形回帰にフィットさせる。

>Min(V1,V2)

選ばれた Vector の少ない要素。構成は、Min(V1,V2)。

>Max(V1,V2)

選ばれた Vector の大きい要素。構成は、Max(V1,V2)。

>Fuse Vectors

2 つの Vector ファイルを融合する

>Transform

2 つの Vector を他の 2 つの Vector へ変形する。

<Cartesian->polar

直交座標から極座標

<polar->Cartesian

極座標から直交座標

>Info

## 3.9Permutation

Permutation：順序の変更，交換；【数】順列;

ネットワークにおける Permutation とは、そのネットワークの頂点を入れ替える（renumbering）することである。

頂点を入れ替えるとは、頂点番号（ノード番号）を入れ替える。いままである頂点のあったところに別の頂点との入れ替えが起こる。

メインスクリーンの Permutation メニューに*.per ファイルが生成される。

*per ファイルのデータ構造は、

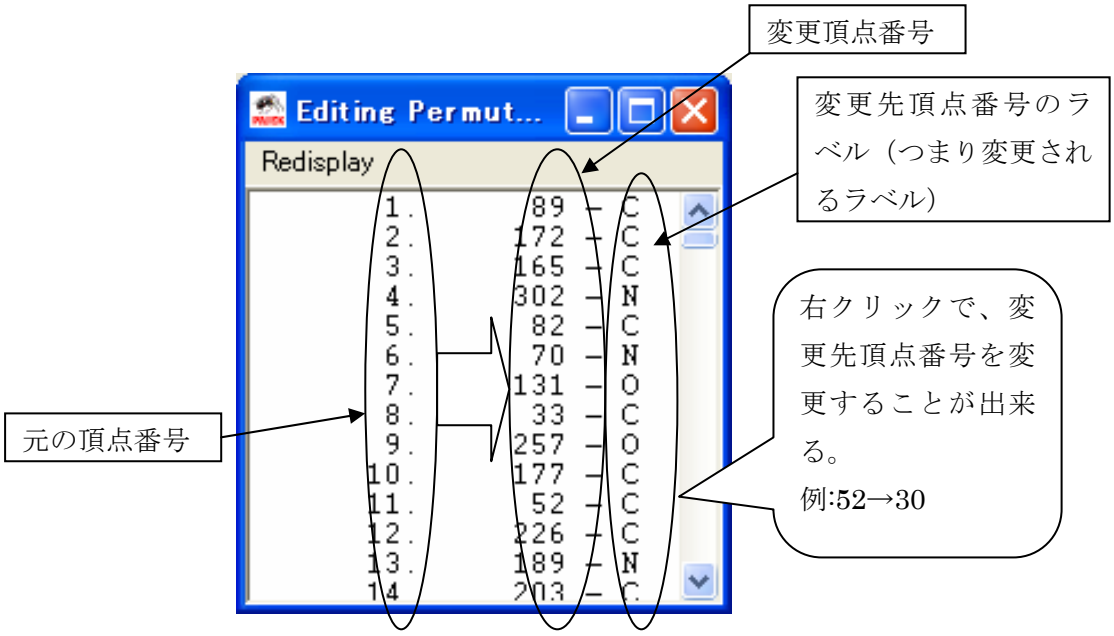
*Vertices ノード数

1 ~n のノード番号（n 行 1 列）

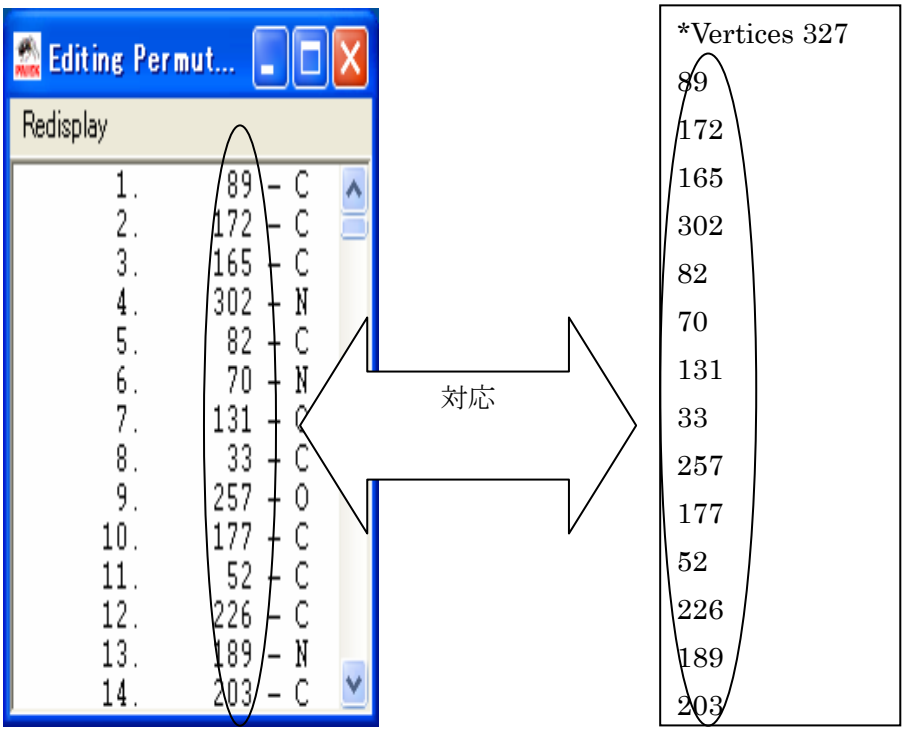
となっている。

*.net ファイルの Vertices データ部分のノード番号を変更するためのものである。

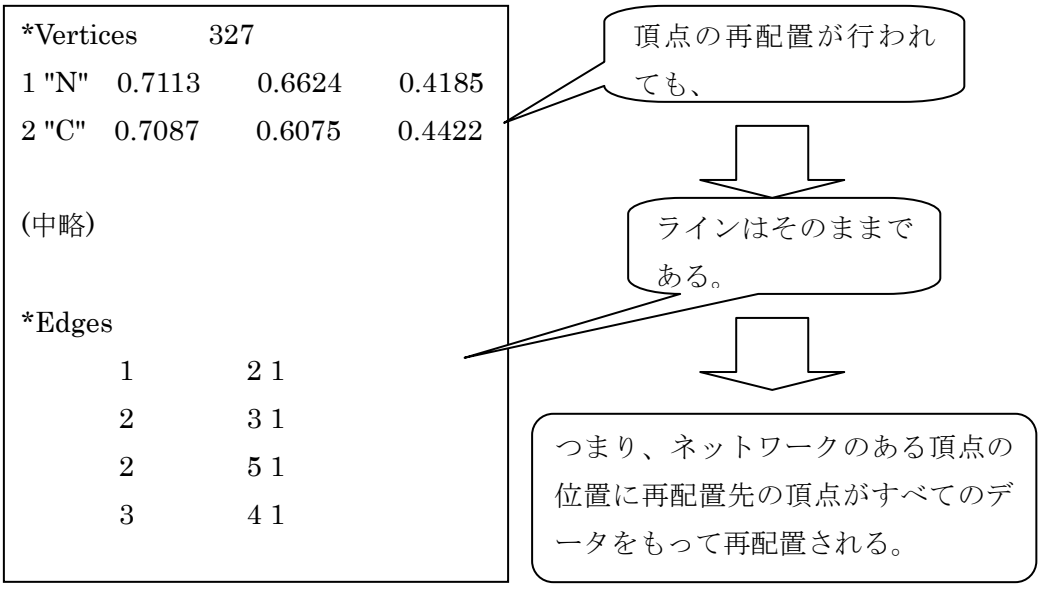
メインスクリーン上の Edit ボタンを押すと、Edit サブテーブルが
『ノード番号　新ノード番号-新ノード番号』
という形で表示される。
ダブルクリックでノード番号が変更可能。



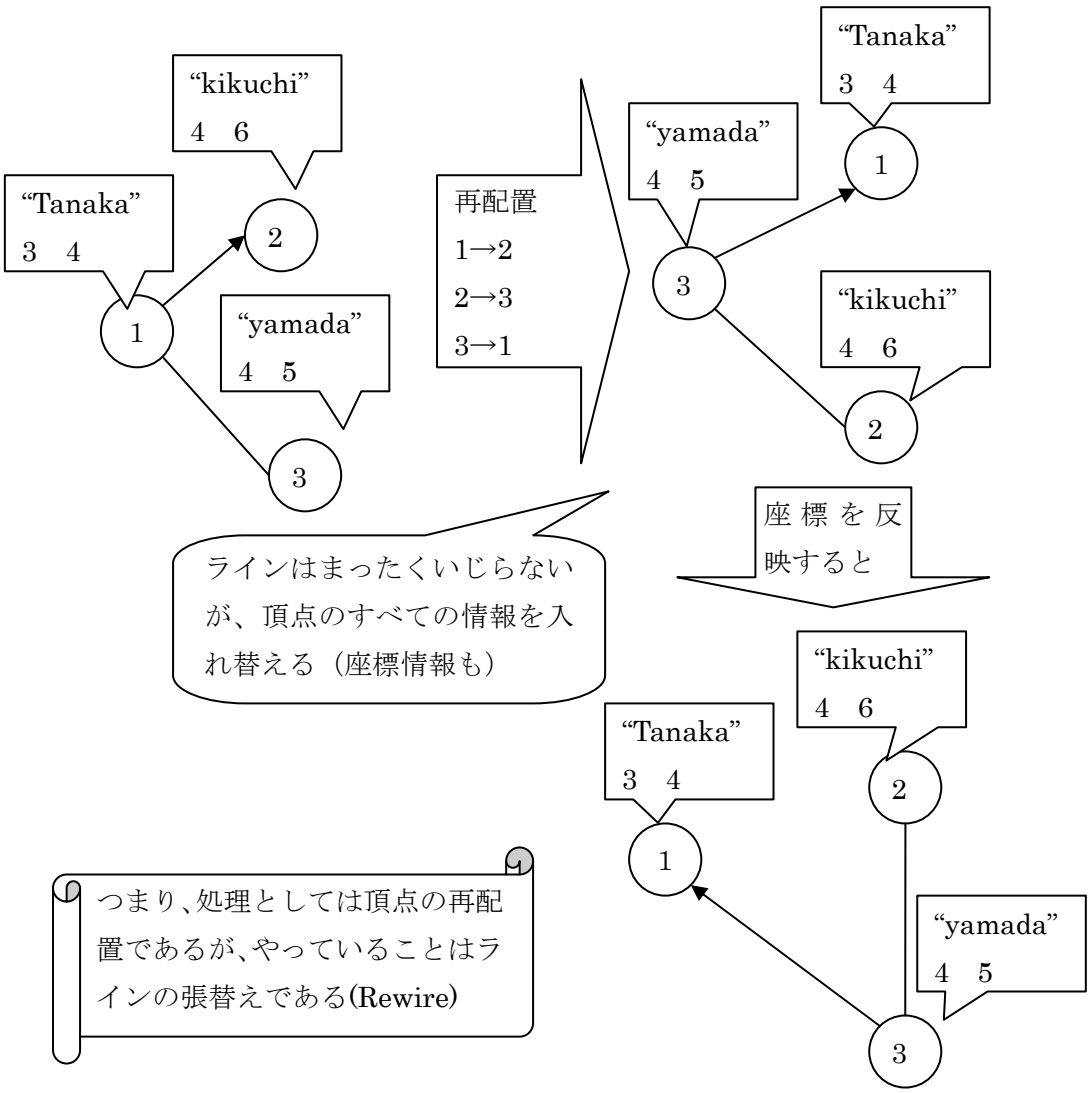

図：Permutation の Edit メニューの解説



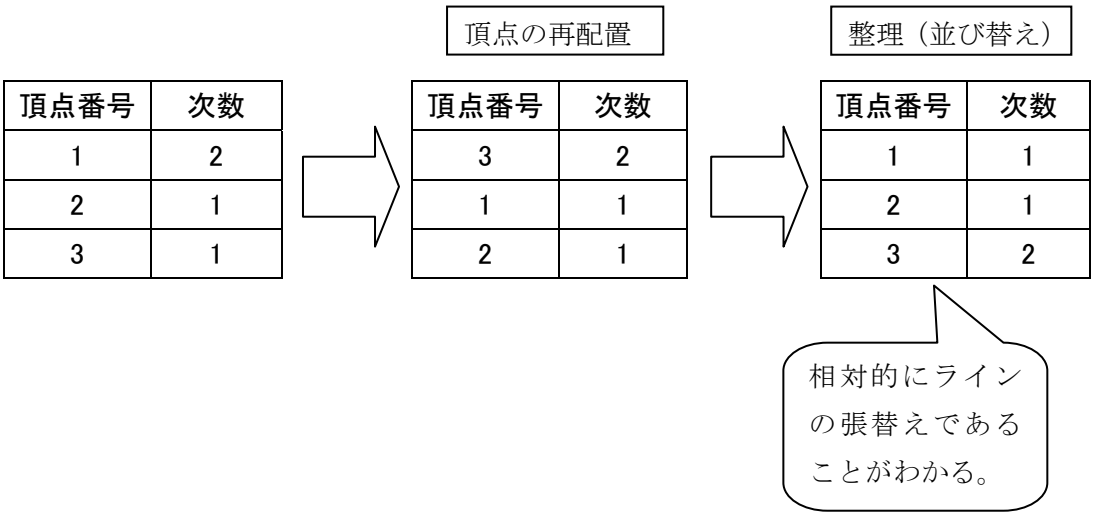

図：上図 per ファイルのテキストエディタでの表示と Edit との比較




図：*.net ファイルと再配置




図：Permutainon で行っていること




図：上図の次数に注目したもの

*.per ファイルを生成する際に、Demention（*.per ファイルに写したい頂点番号の範囲）を指定する。対象*.net ファイルの頂点数より少ない値を指定すると、指定した頂点番号までを、*.Per ファイルへ書き込み、それ以外の頂点は無視される。
このときの*.per ファイルの構成
『ノード番号.　新ノード番号』
となる。

>Identity

*.net の*Verics の頂点番号と同一の頂点番号の*.per ファイルが生成される。

>Random

*.net ファイルの頂点番号をランダムに並び替えた頂点番号が*.per ファイルに書かれる。

>Random 2-mode

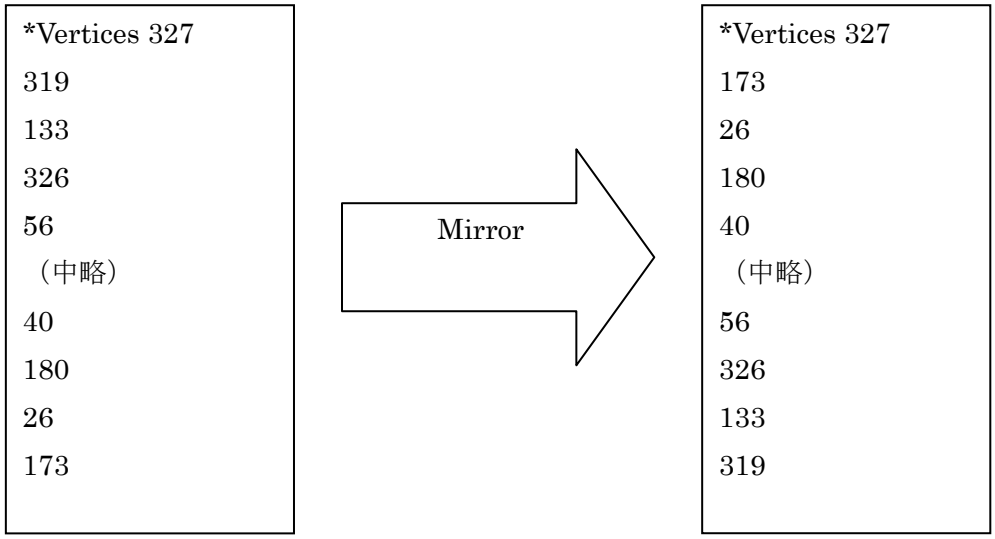
2-Mode（２部グラフ）ネットワークの頂点番号を頂点数を Deimenton（定義）されただけ、ランダムに並び替えた*.per ファイルが生成される。

>Inverse

逆順=Inverse にする
それは下記>Mirror で実現できる
Inverse：
データ構造に変化なし
どのような意味があるのか

>Mirror

鏡に映したように、逆順に並び替えた(sort)*.per のデータ構造となる。




図：Mirror

>Make Partition

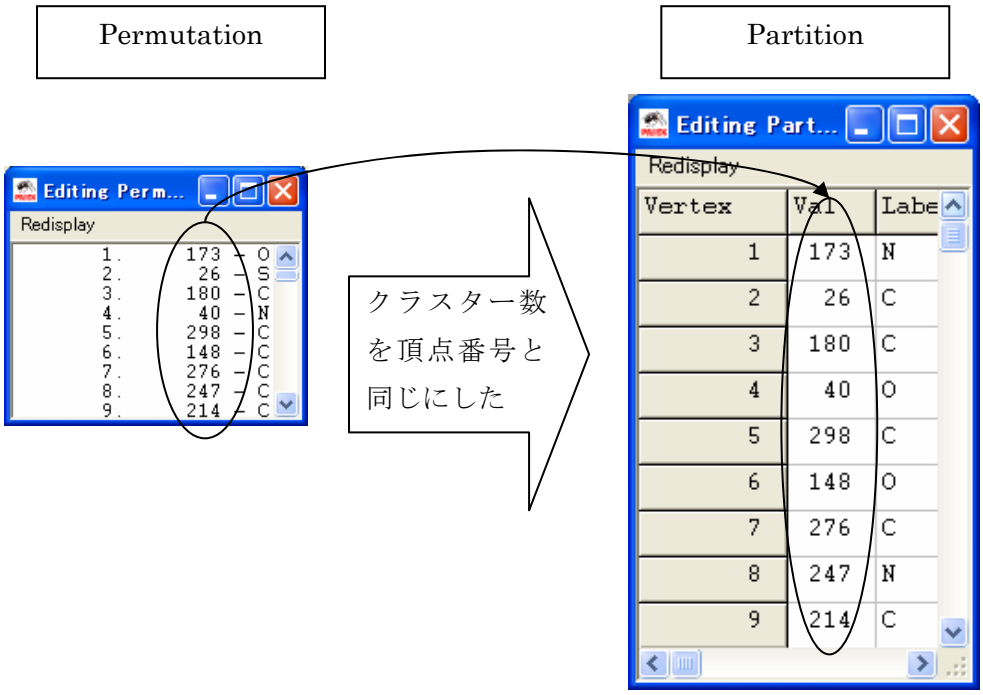
入力項目

①Number of Cluster と聞いてくる

（Between 頂点番号 1 and 頂点番号最大値と入力値が制限される）

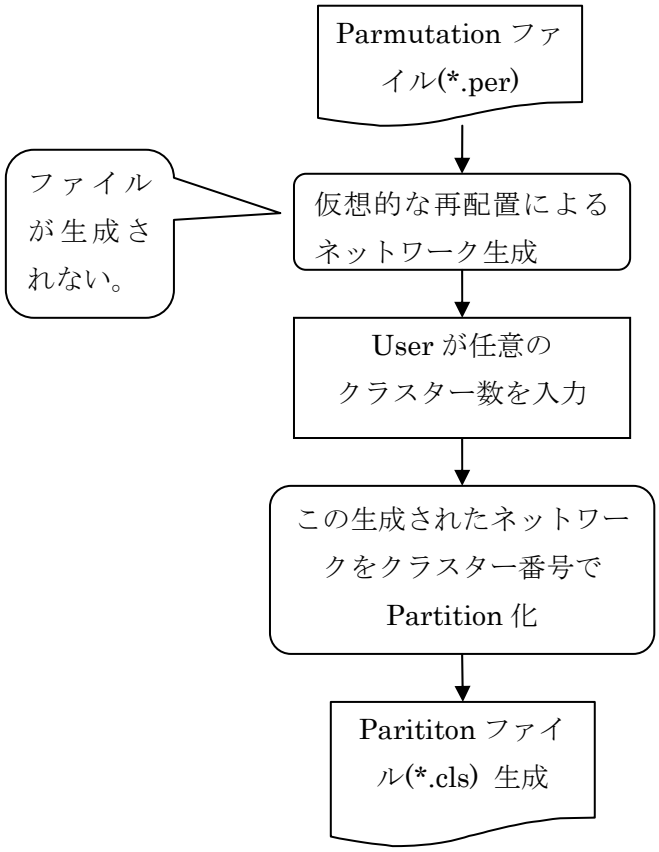
どの数だけクラスターを作るかということ

Permutation のデータ構造に変化はない。だが、Partition メニューに*.cls ファイルの生成される。

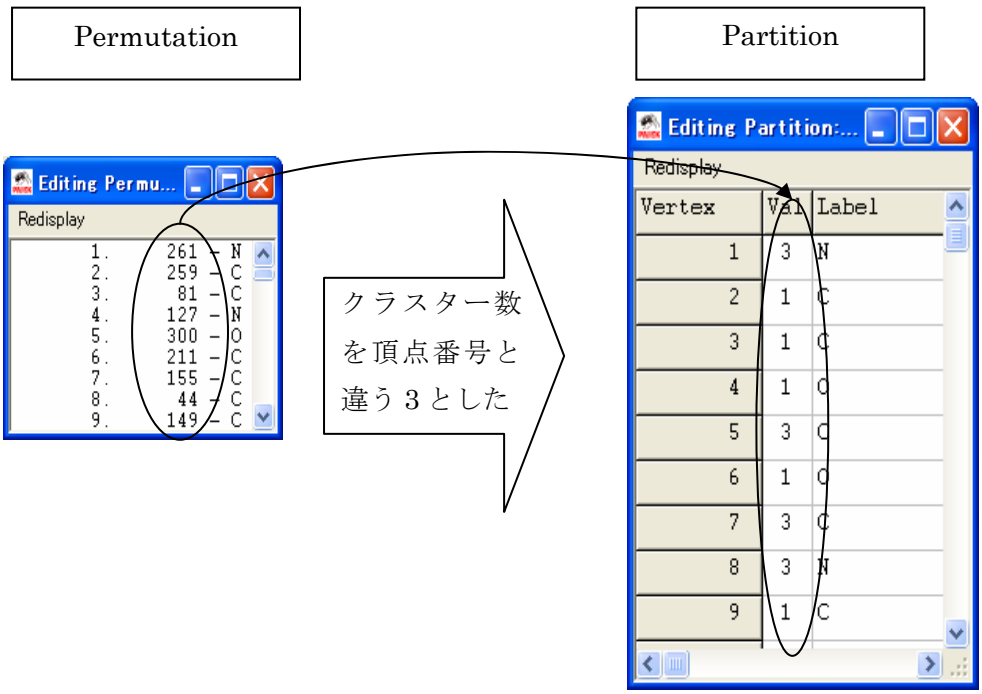



図：Make Partition

つまり、Permutation（*per）のデータ番号（頂点再配置）から Partition(*cls)のクラス番号を生成する。



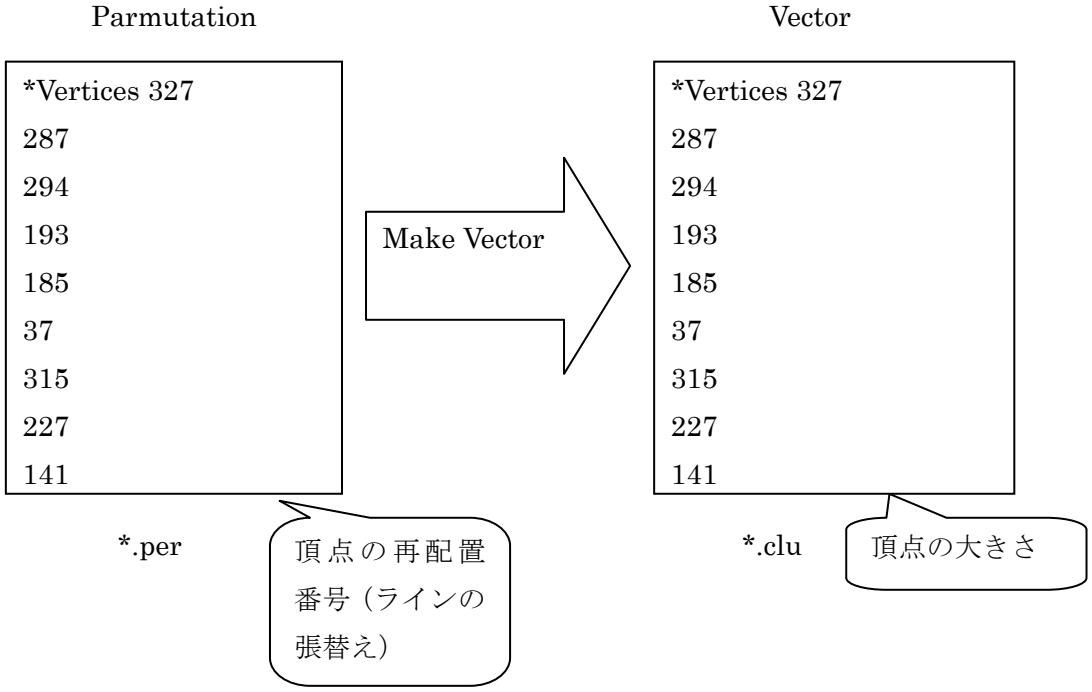

図：Make partition の処理手順




図：Make Partition

>Make Vector

*.per のデータがそのまま*.vec となる。データ構造は.per と同様なので、拡張子が変わるだけとも取れる。



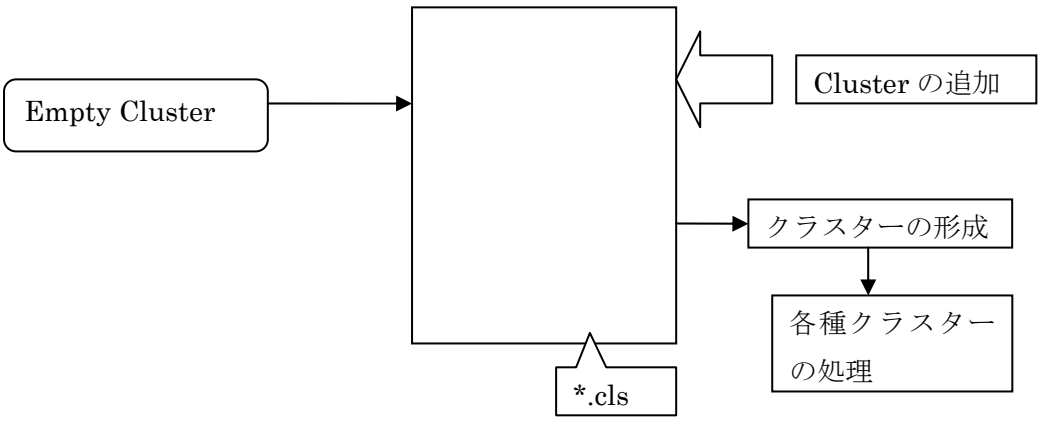

図：Make Vector

## 3.10Cluster

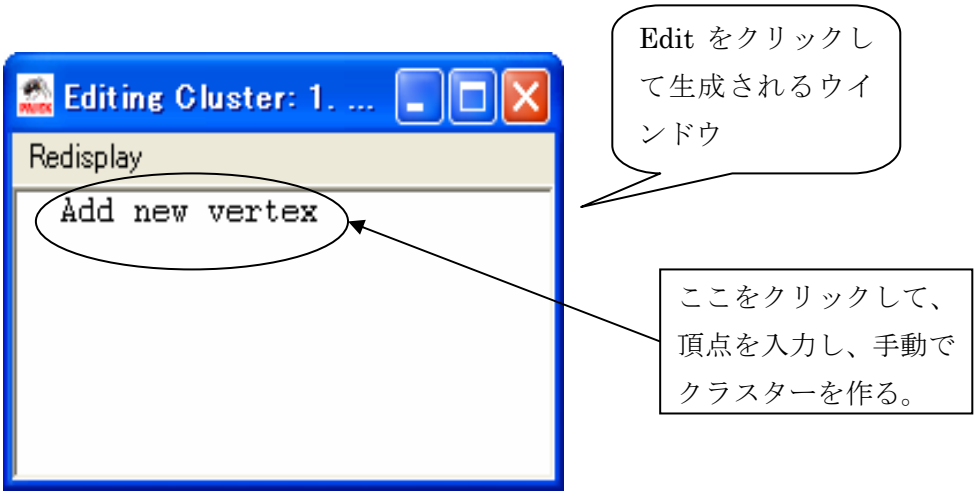
メインメニューの Cluster ボックスに*.cls を生成する。

>Create Empty Cluster

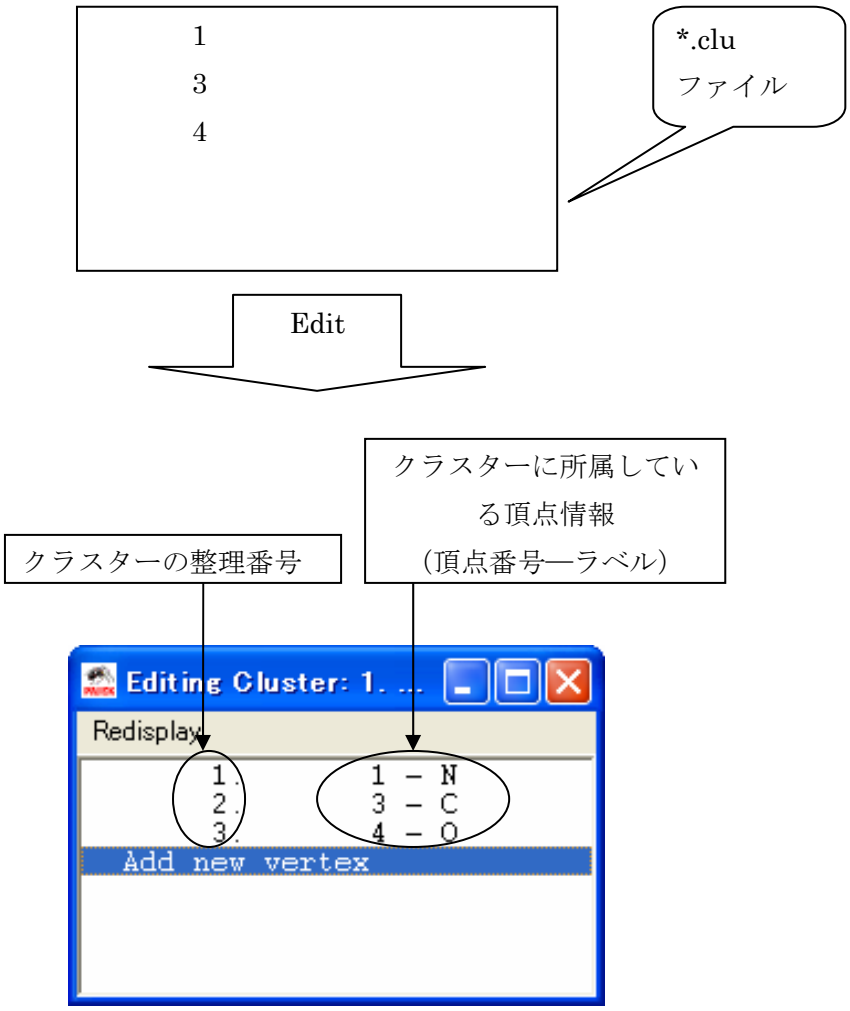
何の情報もないクラスターを生成する。Edit で手動かつ自由にクラスターを形成するためのファイル。




図：Empty Cluster

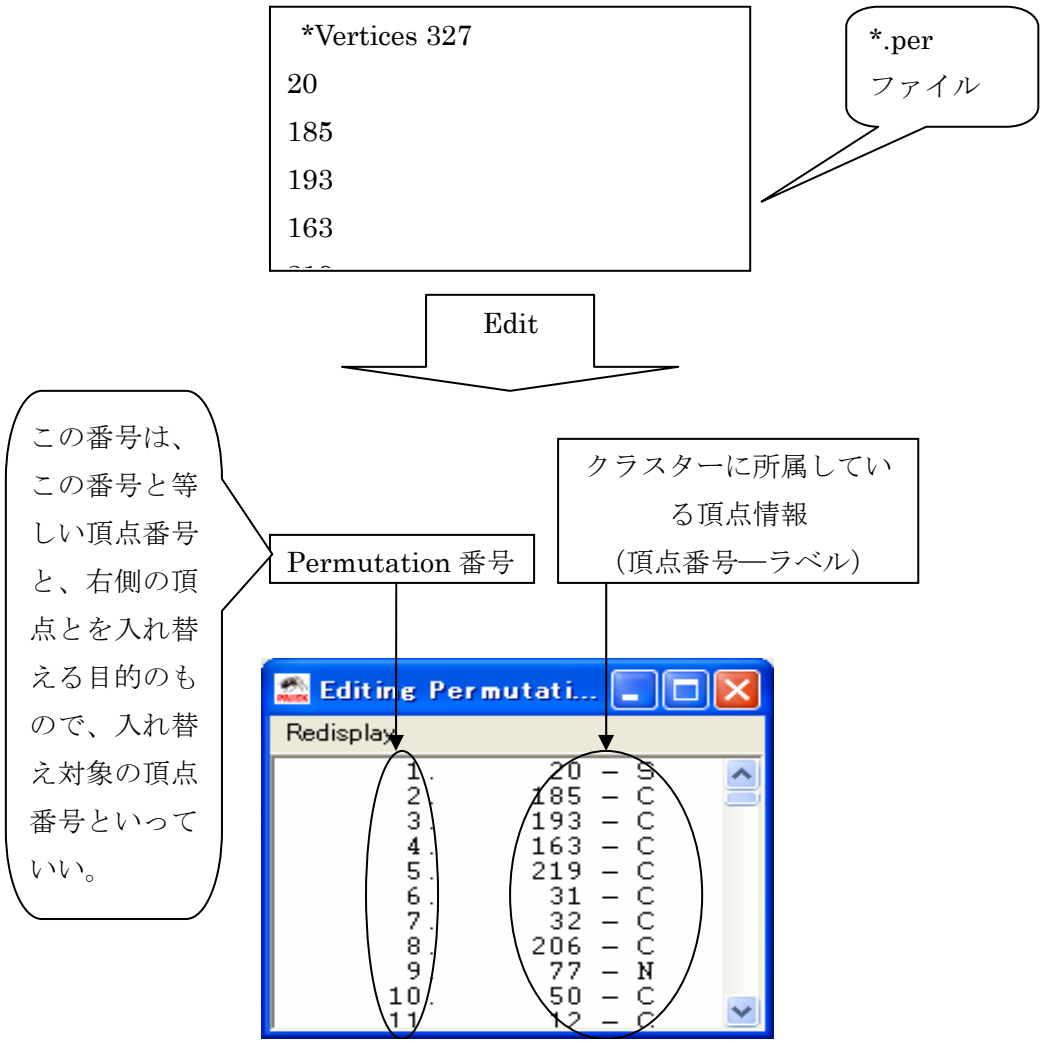



図：Empty Cluster ファイルの頂点追加




図：cluster ボックス Edit によるウインドウの説明

Pertition ファイルと Edit ウインドウが同じ構成だが、意味がまったく違うことに注意する。



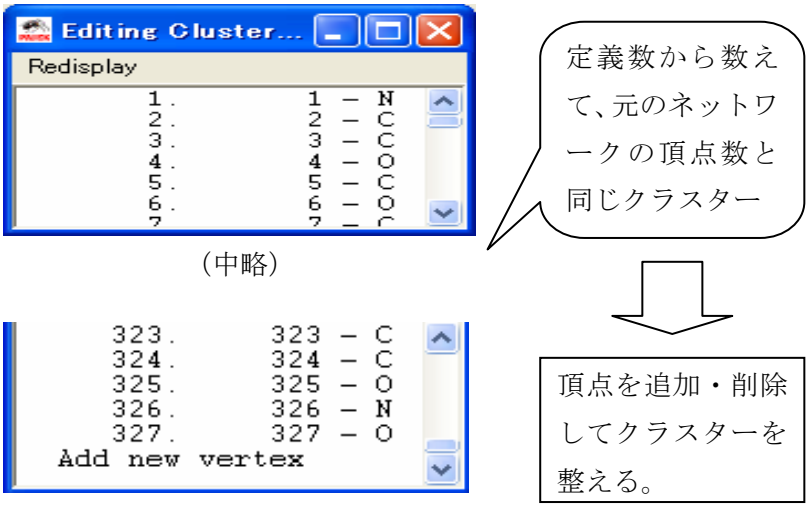

図：*.per と Permutation ボックス Edit によるウインドウの説明

>Create Complete Cluster

入力された定義数に従い、.net ファイルから定義数だけの頂点番号の頂点を、クラスター化する。

入力項目

①定義数（どれだけの量の頂点をクラスター化するか）




図：Create Complete Cluster

>Make Partition

クラスターをパーティションに変形する。

Partition の定義数を入力する際に、対象の,net ファイルの頂点数と同じ数でないと、可視化できないことに注意。

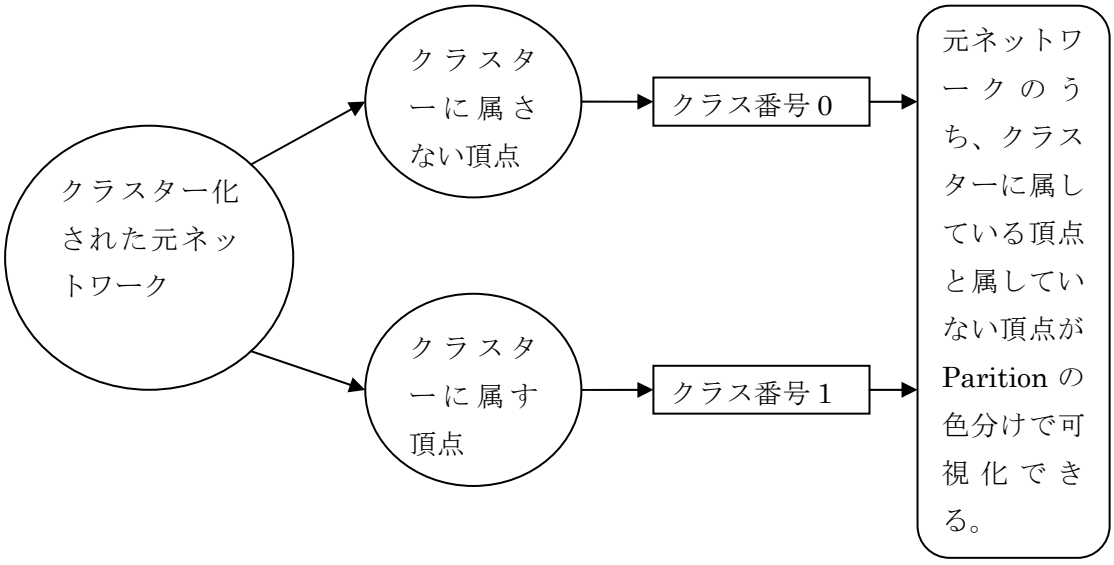
手順

①*.net の頂点数と同じ数の Partition 定義数を入力。

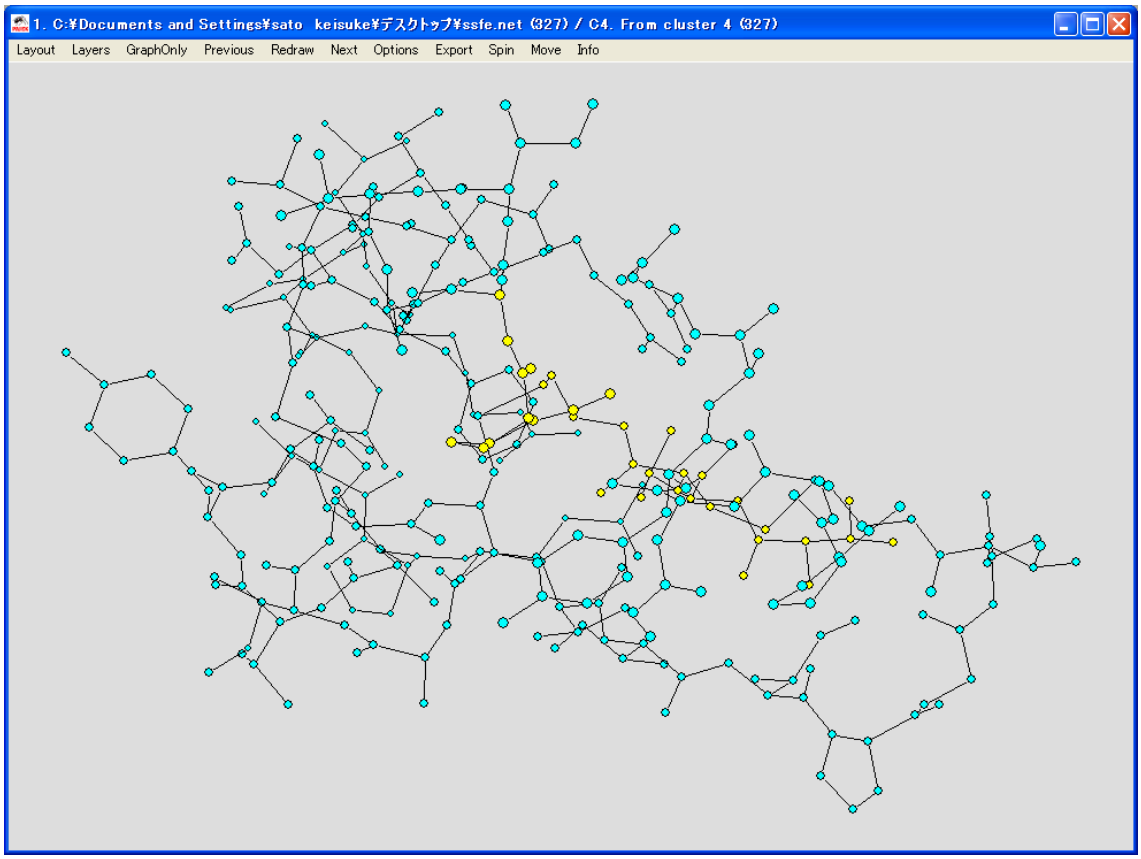
②クラスターに属している頂点をクラス番号 1 になる。

③クラスターに属していない頂点をクラス番号 0 になる。

④可視化したときに、色別でクラスターとクラスターでないものがわかる。




図：Make Partition




図：35 の Complete cluster の Partition 化による可視化

頂点番号 1~35（Complete cluster）がそれに属していないも頂点と別の色（つまり別のクラス番号）となっていることがわかる。

>Binarize Partition

上記と同様に、クラス番号を 0 か 1 にした Cluster の Partition。Cluster と Partition が必要。

## 3.11Hierarchy

このメニューは、すべて Hierarchy ファイル（*.hie）を使用し処理する。つまり、Hierarchy ファイル（*.hie）がないと処理できない。

>Extract Cluster

Hierarchy から cluster を引き出す。

>Make Network

Hierarchy から Network を作成する。

>Make Partition

Hierarhy から Partition を作成する。

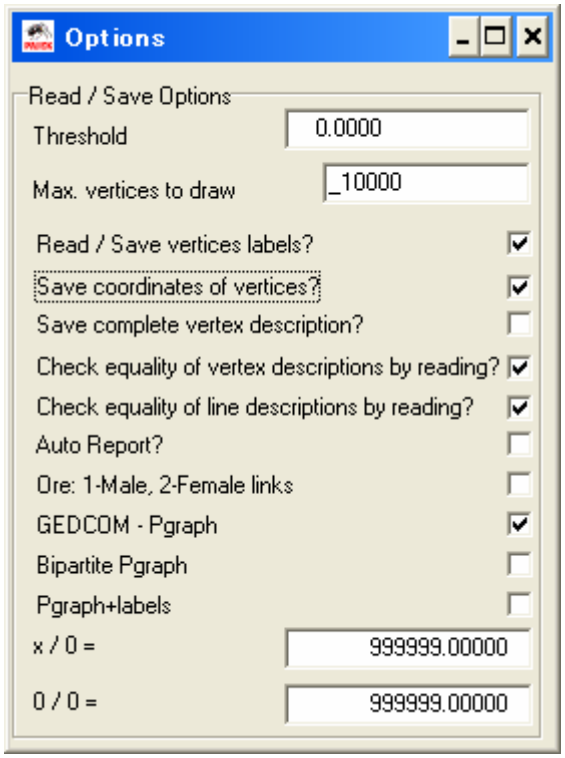
>Make Permutation

Hierarchy から Permutation を作成する。

## 3.12Options

>Read/Write

以下の設定を行うためのコマンド。




図：Options>Read/Write の設定画面

<Threshold

Threshold：しきい値

与えられた 2 頂点間の Generate line よりも高い（絶対に）ラインの値

<Max.vrtices to draw

ネットワークで最大値をもつ頂点に従った視覚化。

<Read/Save vertices labels?

頂点のラベル、座標、その他の描写に関わることを読み込むか読み込まないかの指定。大規模なネットワーク・ラベルの長さが長い場合、これらが読み込もうとすると、これらの読み込み処理が非常に遅くなってしまう。読み込み処理の速度の遅延を防ぐためにこの設定をはずすことが可能である。

また、ある.Net ファイルの頂点のラベルを、別の.Net ファイルの頂点のラベルから上書きすることも可能である。

Net>Transform>Add>Vertex Labels form File

<Save coordinates of vertices?

頂点座標を、読み込みするかしないかの設定。

<Save complete vertex description?

完全な頂点の描写（例：形、時間、間隔など）を読み込むかしないかの設定。

<Check equality of vertex descriptions by reading?

大規模ネットワークの可視化の処理速度向上のために、頂点の描写を変える。

チェックする：空間を同じ頂点の描写ときも、しばしば繰り返して読み込む。（例、頂点の形）

チェックしない：時間を読み込むときそれらの描写が違うとき、時間を読み込む。（例、現ネットワークのタイムスタンプ）

<Check equality of line descriptions by reading?

大規模ネットワークの可視化の処理速度向上のために、ラインの描写を変える。

チェックする：空間を同じラインの描写ときも、しばしば繰り返して読み込む。（例、点線か実線かのパターン）

チェックしない：時間を読み込むときそれらの描写が違うとき、時間を読み込む。（例、現ネットワークのタイムスタンプ）

<Auto Report?

テキスト形式のレポートを rep1.rep ファイルに自動で書き込む

<ore:1-Male,2-Female links

Ore グラフという系図を読み込んだとき、2 種類の異なった有向グラフが生成される。

1 の値を持つ有向線は、父から子へという関係を持つ。

2 の値を持つ有向線は、母から子へという関係を持つ。

<GEDCOM-Pgraph

系図を読み込むとき、pgraph のフォーマットを使う（ノードはカップルか個人的か）

<Bipartite Pgraph

個人的なマリッジ、トライアングル、サークルというスクエアを持つ、2 部 p グラフを生成する。

<Pgraph+labels

GEDCOM ファイルを読み込んだとき、pgraph のラインのラベルもまた取り付ける。

<x/0

0 と 0 でないものを分離したとき結果を指定する。

<0/0

0 と 0 を分離したとき結果を指定する。

>Blockmodel

縮小されたブロックモデルの型を選ぶ。

>Ini file

*.ini：pajek の構成が保存されているファイル。

Option で設定された変更点を保存する。

<Load

*.ini ファイルを読み込む。

<Save

*.ini ファイルを書き込む。

>Fontsize

フォントサイズを変更する。

## 3.13Info

読み込まれた各種ファイルの情報（Information）を表示する。

Edit で情報を表示することとの違いは？

>Network

ネットワークに関する情報（*.net ファイルのデータ）の表示。

<General

*number of vertices

頂点数を表示する。

*number of arcs,edges and loops

有向線、無向線、ループの数を表示する。

*density of lines

ライン（Edge、Arce）の濃度を表示する。

*sort lines according to their values (ascending or decending)to find the most/least important lines

<Line Values

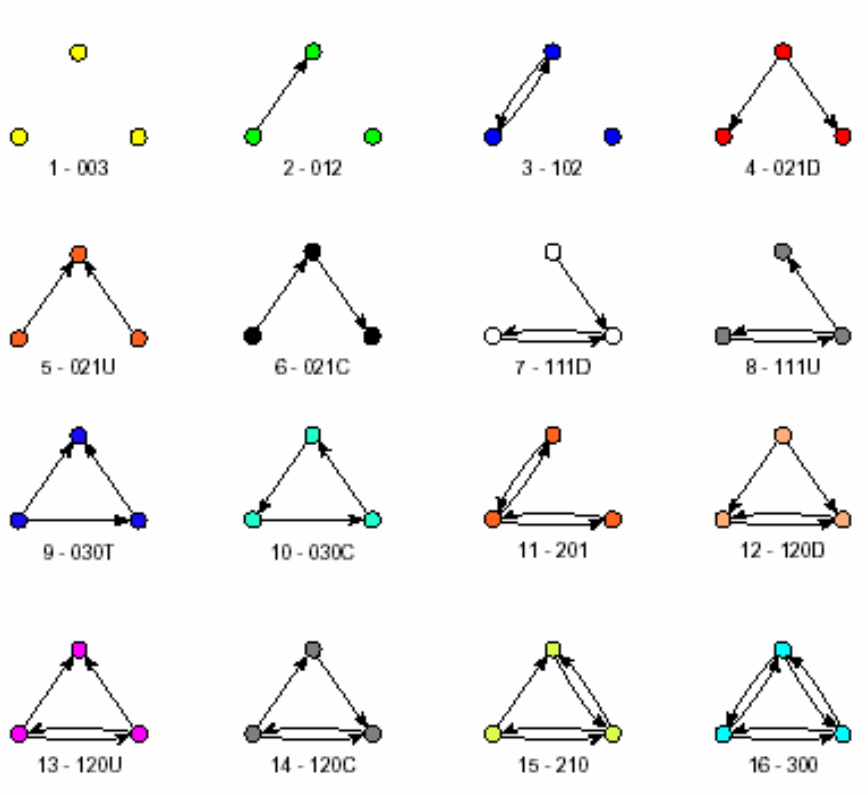
<Indices

Indices=?

<Triadic Census

Triad：3 ノード間のラインの状態

Triads：Triad の種類をすべて（16 種類）集めたもの




図：Trads[manu]

Triads の記号

『タイプ番号－3 つの番号』

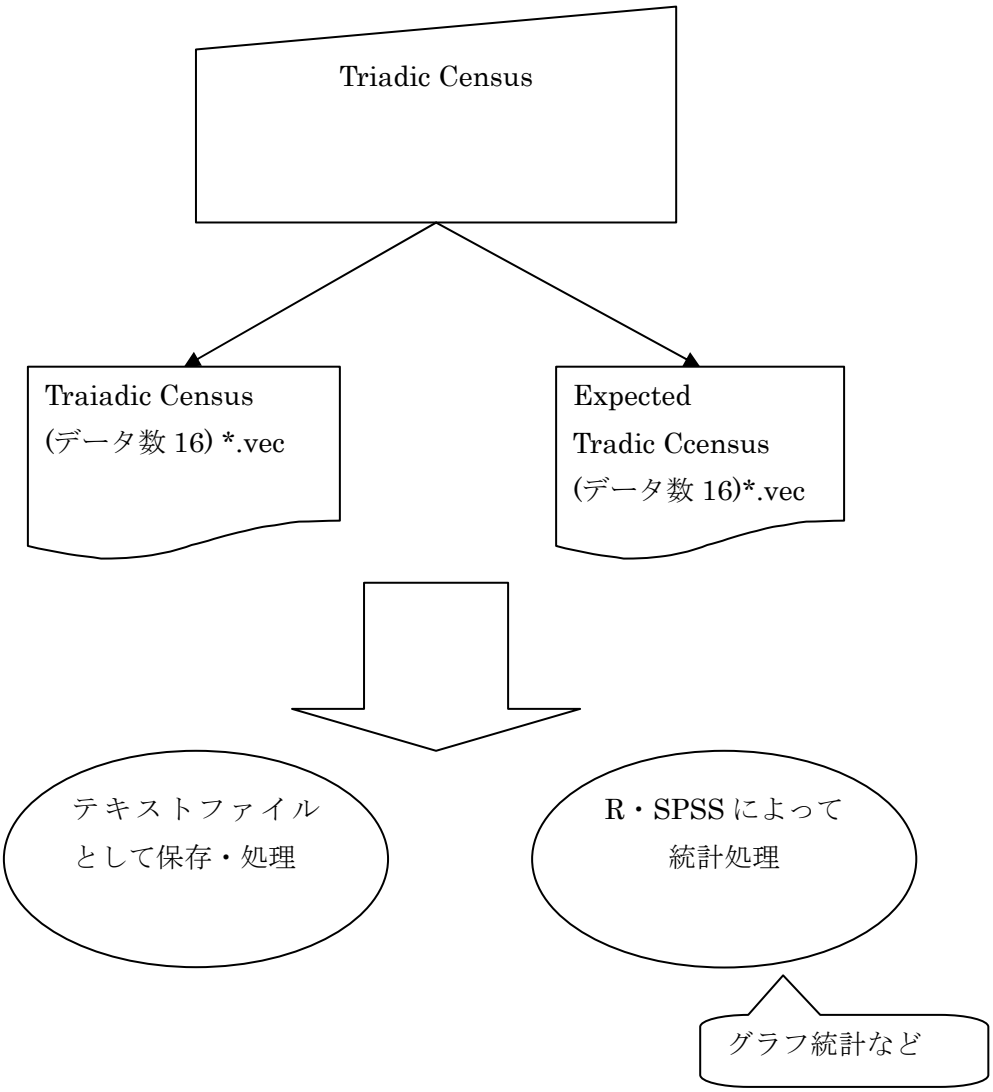
3 つの番号の意味は、

『(有向線が張られていない間隔数)（有向線の数）(両向有向線の数)』

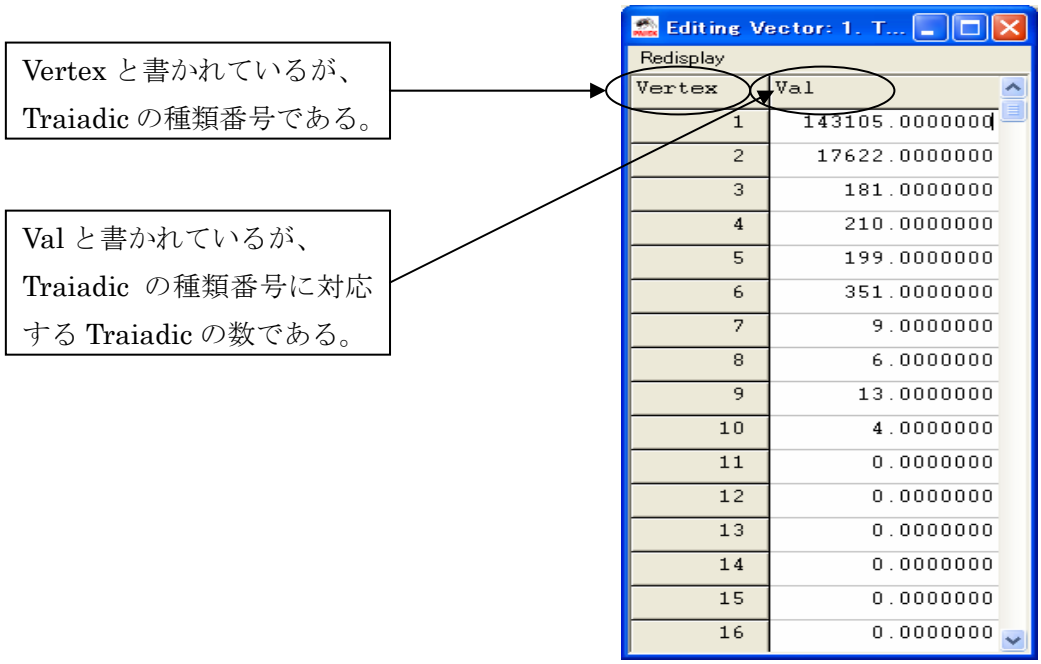
同じ 3 つの番号で別の種類の Tiads は､最後にアルファベットの記号を付ける。

処理すれば、Vectors メニューにファイルに以下の 2 つの*.vec が生成される。

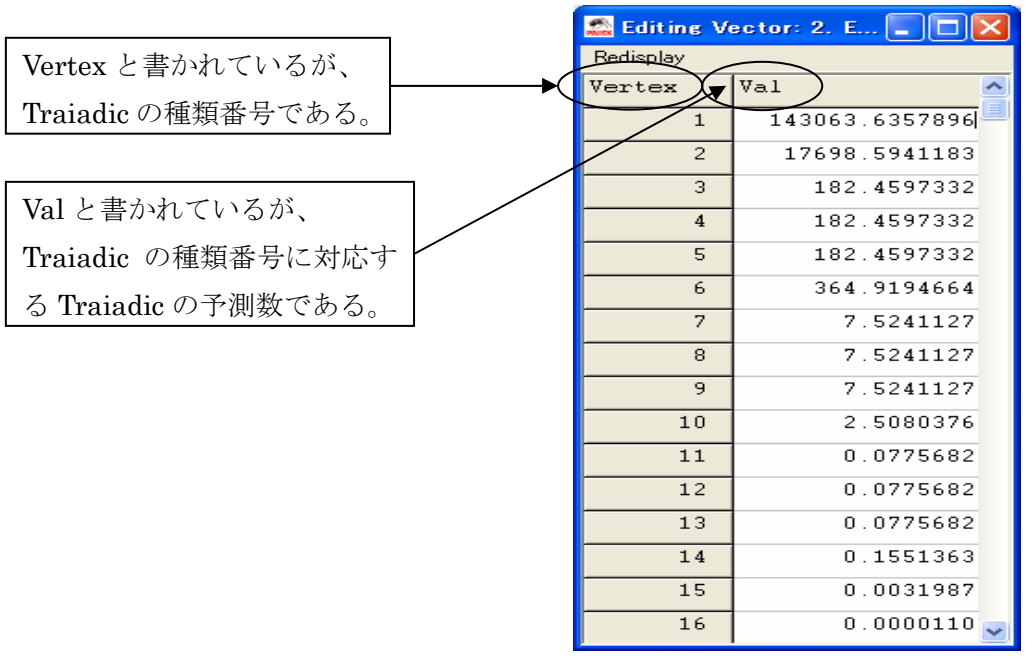
この vector ファイルで可視化することは出来ない。なぜなら、データ数は必ず 16 であり、ネットワークサイズと合わないし、可視化を前提にしていない。



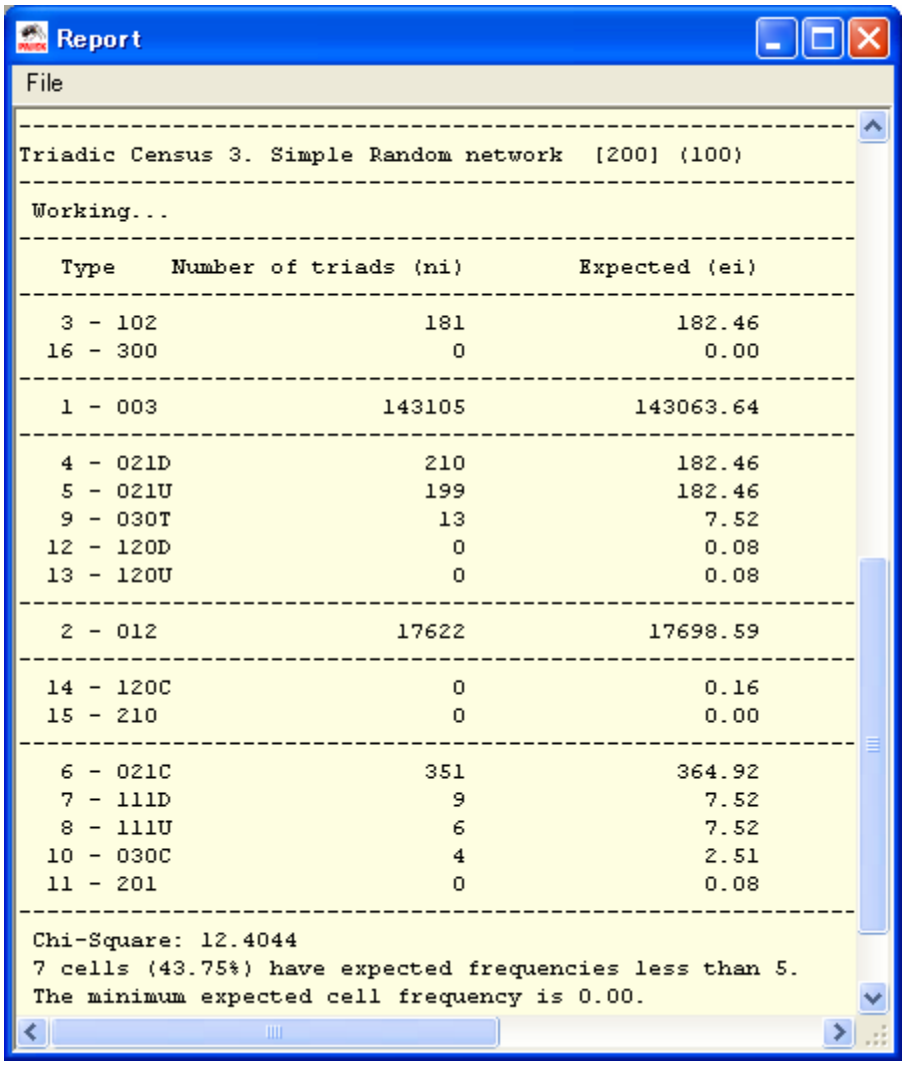

図：Triadic Census の処理過程




図：Traiadic Census



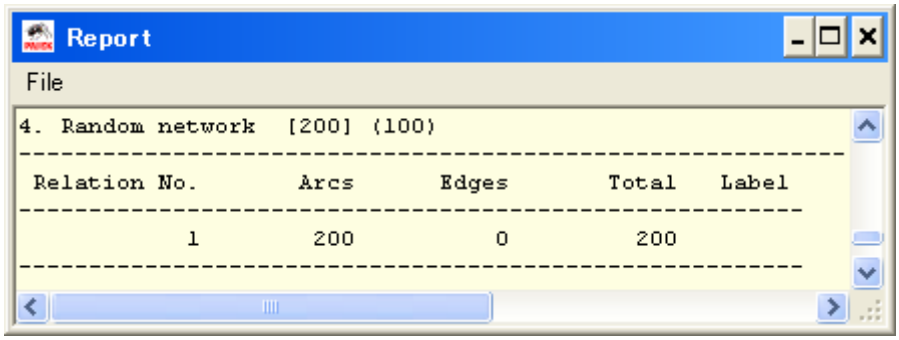

図：Expected Traiadic Census




図：Report

<Multiple Relations

以下のネットワークの要素に関する Report を出す。



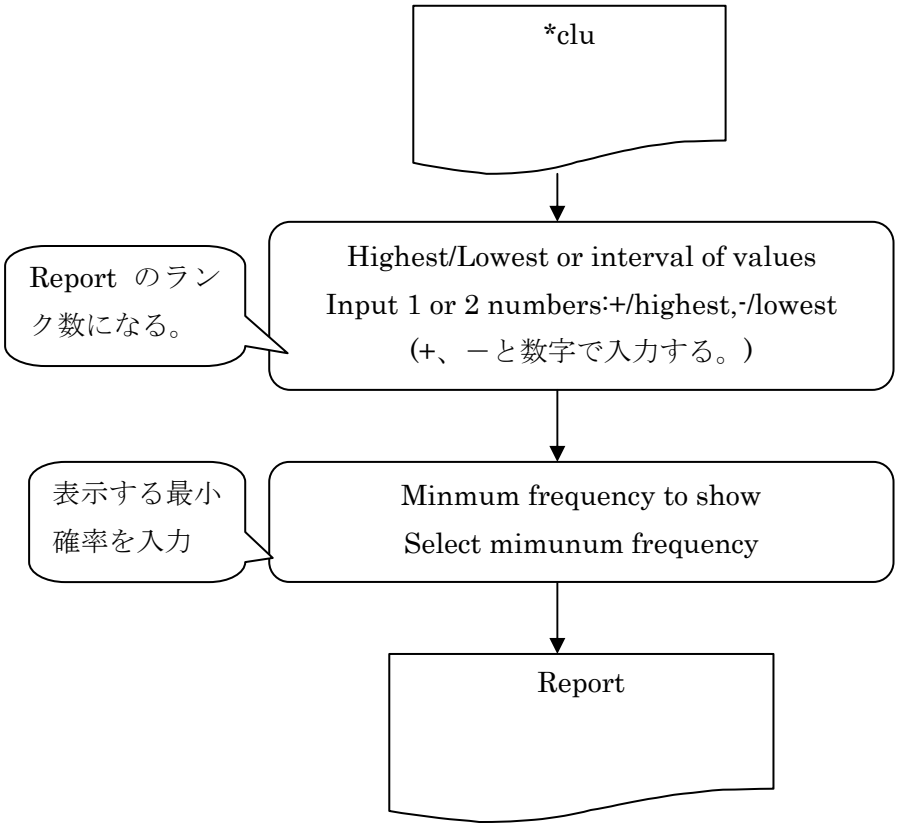

図：Multiple Relations の Report

<Vertex Label->Vertex Number

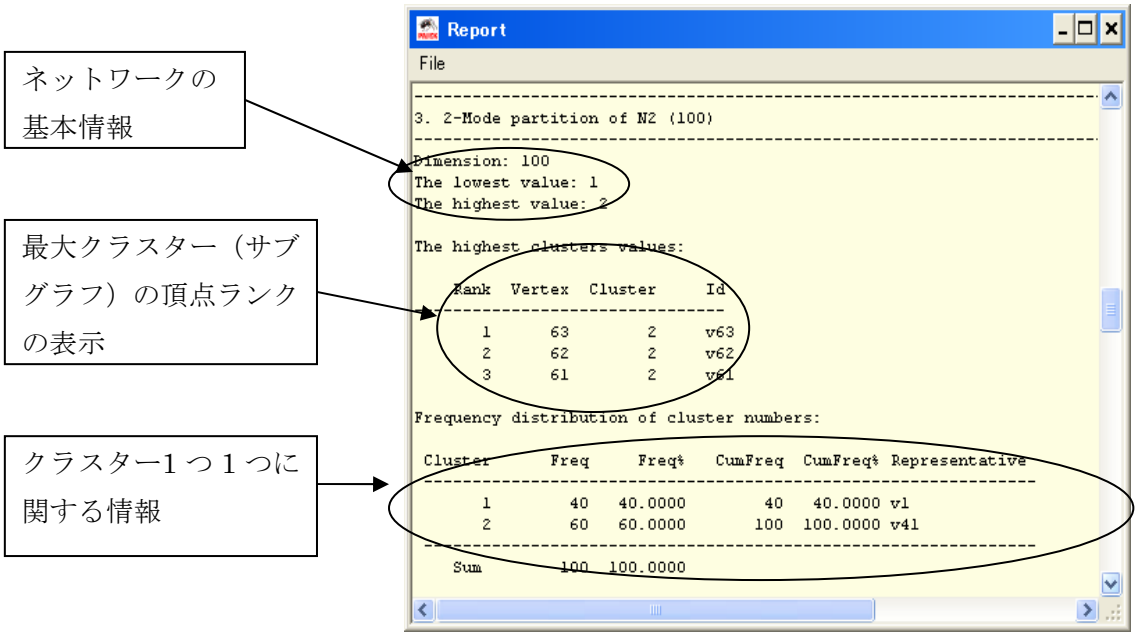
頂点を指定、あるいはラベルを指定する。

入力された値に従い、頂点とラベルの情報がサブウインドウ上に表示される。

>Perrition




図：info Perrition の処理過程




図：info partition の Report

>Hierarchy

Hierarchy に関する情報を生成する。

>Vector

Vector に関する情報を生成する。

サブウインドウでの表示

>Memory

使用マシンのメモリに関する情報。

>About

Pajek の Copyright や Version に関する情報。

## 3.14Tools

>R

<Send to R

<Locate R

>SPSS

<Send to SPSS

<Locate SPSS

Pakek のフォルダに、SPSS のシンタックスファイルとして書き込まれる

>Web Browser

Vertex をシフトか右クリックをすると web ブラウザーが開かれる

>Add Program

ツールメニューに特別なプログラムを実行する項目を追加する

>Edit Parameters

追加された項目のプログラムをエディトする

>Remove Program

追加された項目を消す

## 3.15Draw

・*.net

・*.cls

・*.vec（2 つを使用する場合もあり）

の 3 つのファイルを適宜利用して可視化する

>Draw

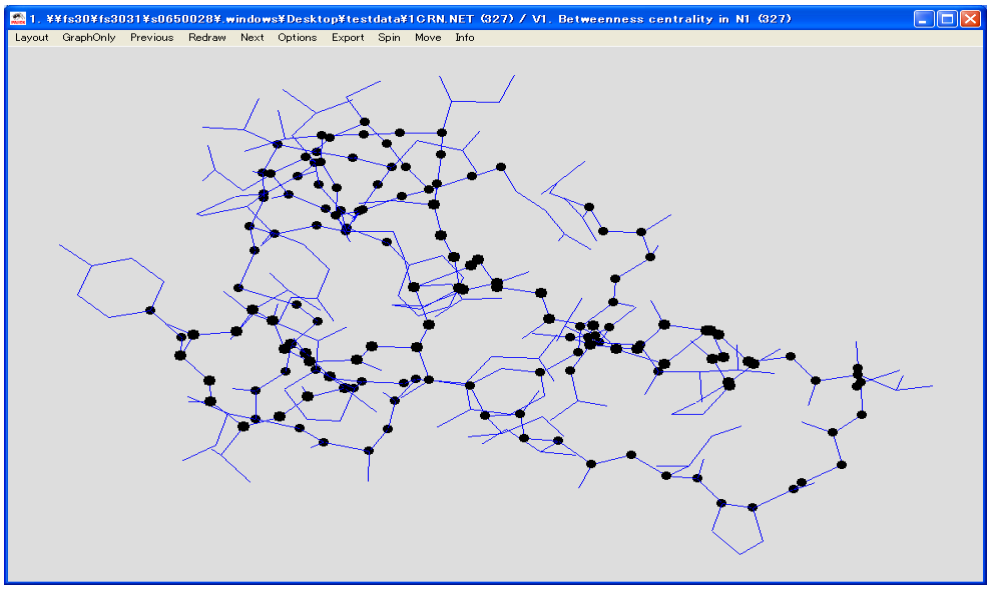
*.net を可視化することが出来る。

>Draw-Partition

Partition(*.cls)ファイルに従った、頂点がどのクラスに属しているかを、クラスごとに色分けして可視化できる。

>Draw-Vector

Vector(*.vec)ファイルに従った、頂点の大きさを*.vec の値に反映させて可視化できる。ネットワーク(*.net)の頂点数と*.vec の定義数が等しくなければならない。




図：Bitweenness 中心性を計算した Vector ファイルを反映した可視化の例

中心性の高い値ほど頂点が大きくなっていることがわかる。

>Draw-2Vectors

2 つの Vector(*.vec)ファイルに従った、頂点の大きさを*.vec の値に反映させて可視化できる。

メインスクリーンの Vector ボックスの 1 番目に入っている*.vec は頂点の幅になり、2 番目に入っている*.vec は頂点の高さになる。


上図の場合は、2.All closeness centrality in N1(327)のデータが、頂点の幅、

1 .Betweenness centrality in N1(327)が頂点の高さとして頂点のサイズが変更される。

>Draw-Partition-Vector

Partition のクラスごとの色分けと、Vectoe の頂点サイズの変更を同時に行う。

>Draw-Partition-2Vectors

Partition のクラスごとの色分けと、2 つファイルの Vectoe の頂点サイズの変更を同時に行う。

>Draw-SelectAll

Null-Partition を使ったネットワークを可視化する。

# 4.Drawウインドウツールバー上の各種メニュー

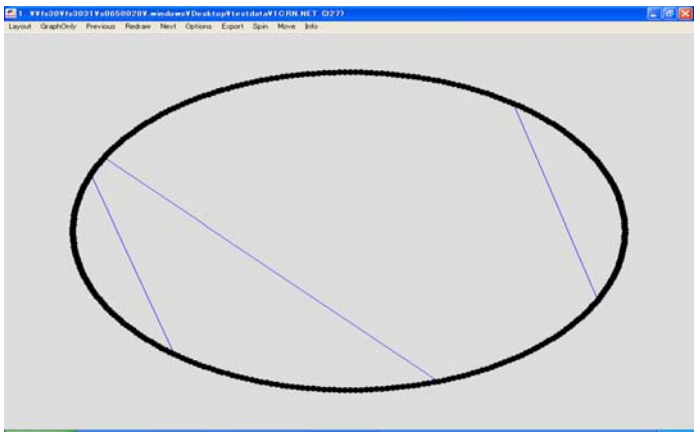
## 4.1Layout

>Circular

円状に頂点を配置する。

1.Original

普通の円状の配置



2.Using Permutation

Permutation を反映した円状の配置

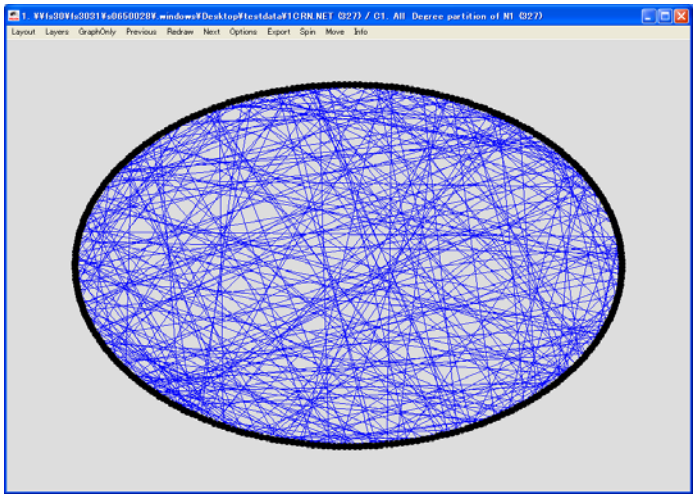
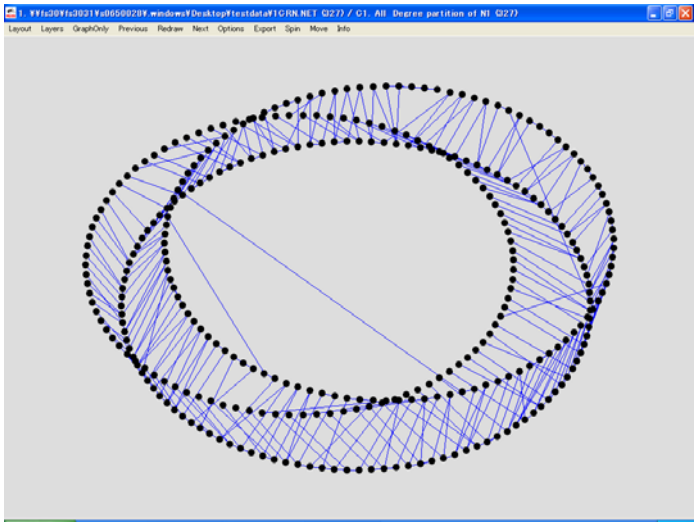


図 Randam permutation による可視化

3.Using Partition

Partition によって、クラスター毎に分離する円状の配置



図：次数による Partition の可視化

4.Randam

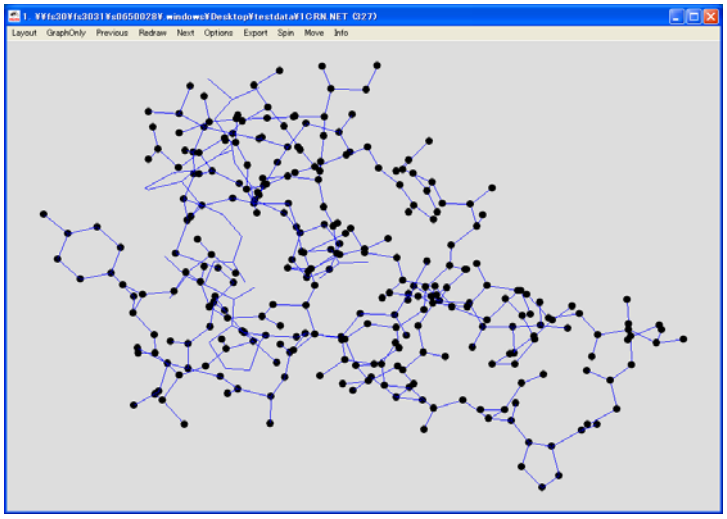
頂点を円状ランダムの配置する

>Energy

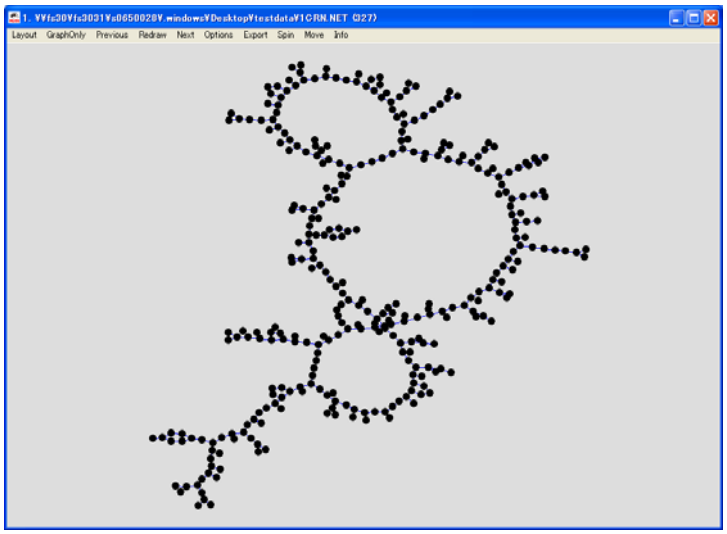
自動的にネットワークのレイアウトを生成する。

1.Kamada-kawai

頂点の配置の自動レイアウトのためのアルゴリズム




図：座標を反映した配置




図：同じネットワークで、Kamada-kawai アルゴリズムで最適化した可視化

(a)Free

配置可能なすべての位置

(b)Fix first and last

最初から最後までの頂点が画面の角に一致するように

(c)Fix one Vertex in the middle

選んだ頂点が中心（Middle)に来るように

(d) Selected group only

Partition から選んだ頂点のみ

(e) Fix Selected Vertices

選んだ頂点に合うように

partition(.clu)を読み込んだ視覚化のみメニューに出てくる

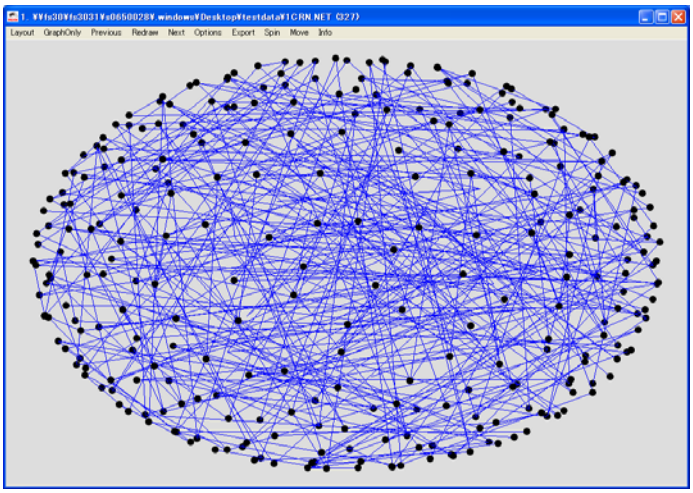
2.Fruchterman Reingold

Kamada-kawai 以外の自動レイアウトのためのアルゴリズム

(a)2D

Optimization：

2 次元空間（Plene）に置く




球状に配置されている。Spin してみるとわかる。

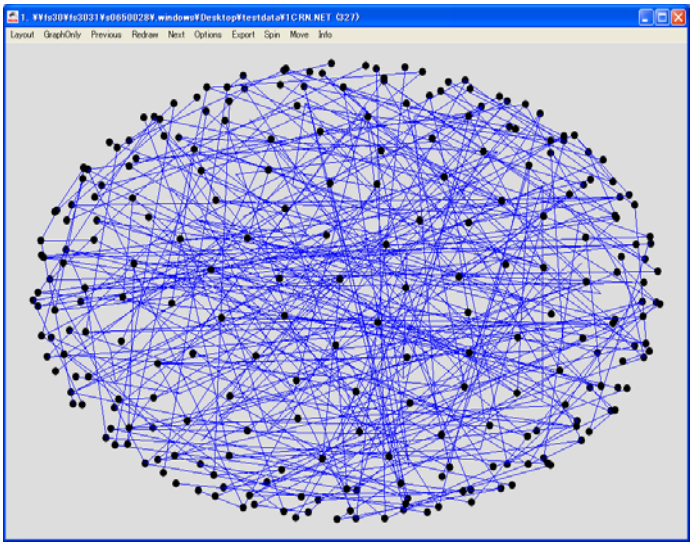
Spin>Spin around>スピンしたい角度

あるいは、ScrollBar を表示して回転させる。

Option>ScrollBar On/Off

(b)3D

3 次元空間（Spece）に置く




(c)Factor

Fruchterman Reingold 最適化(optimization)を使っているとき、頂点間の最適な距離のための要素の入力

3.Srarting Positions

エネルギー描写（energy drawing）のための初期配置

>Eigen values

Lanczos アルゴリズム

Eigen-=固有の

固有値、固有ベクトルを用いた描写（Lanczos アルゴリズム）

ラインの値を計算するかしないかもできる？

1.111

2つか3つの固有値を選ぶとアルゴリズムは符合している固有ベクトルと計算される

(a)111

はじめに符合した固有値の 3 つの固有ベクトルを計算する

(b)112

はじめに符合した固有値と 2 番目に符合された 1 の 2 つの固有ベクトルを計算

(c)122

はじめに符合した固有値と 2 番目に一致する 2 の 1 つの固有ベクトルを計算

(d)123

はじめに一致した固有ベクトル､2 番目に一致した 1 と 3 番目の固有値に一致した 1 を計算する。

(e)11

2 つのはじめの固有値に一致した固有ベクトルを計算する。(2 次元描写)

## 4.2Layers

pertition(.clu)を読み込んだ視覚化のみメニューに出てくる。

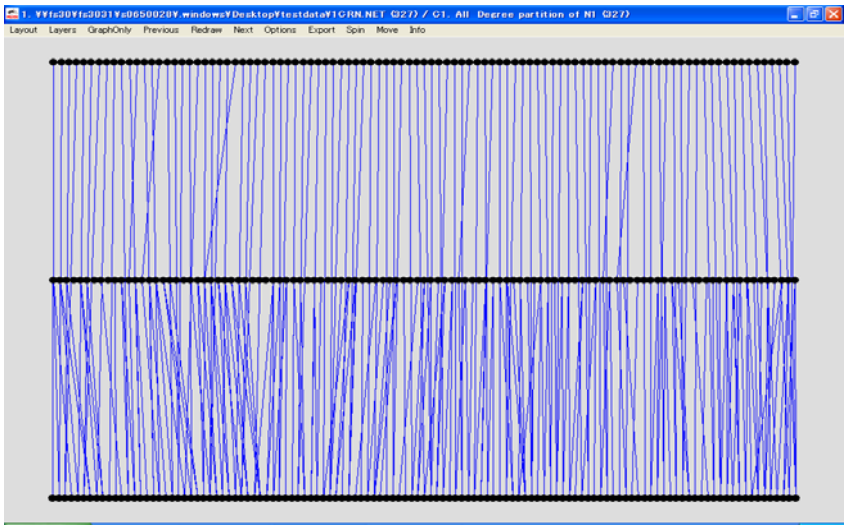
>Type of Layout

2D か 3D のいずれの表示にするかを決定する。

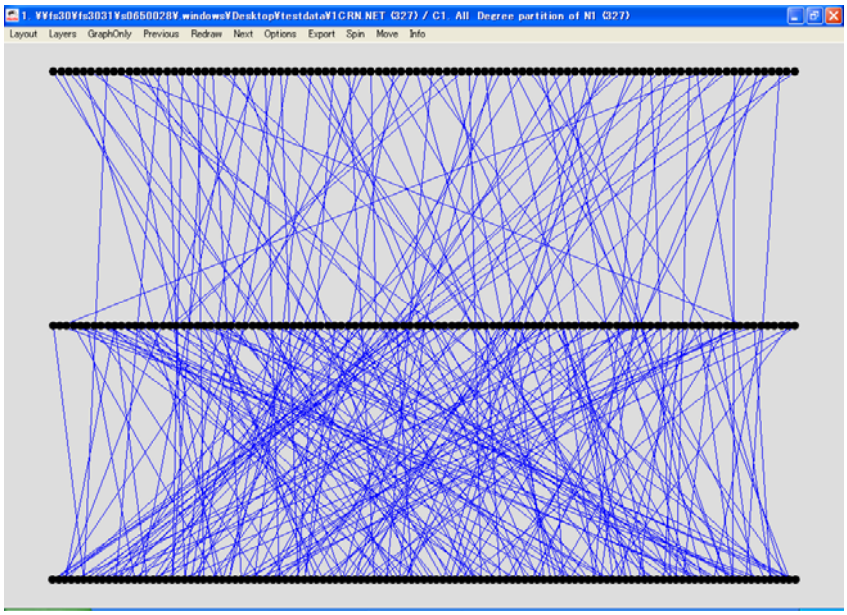
2D－x,y

3D－ z

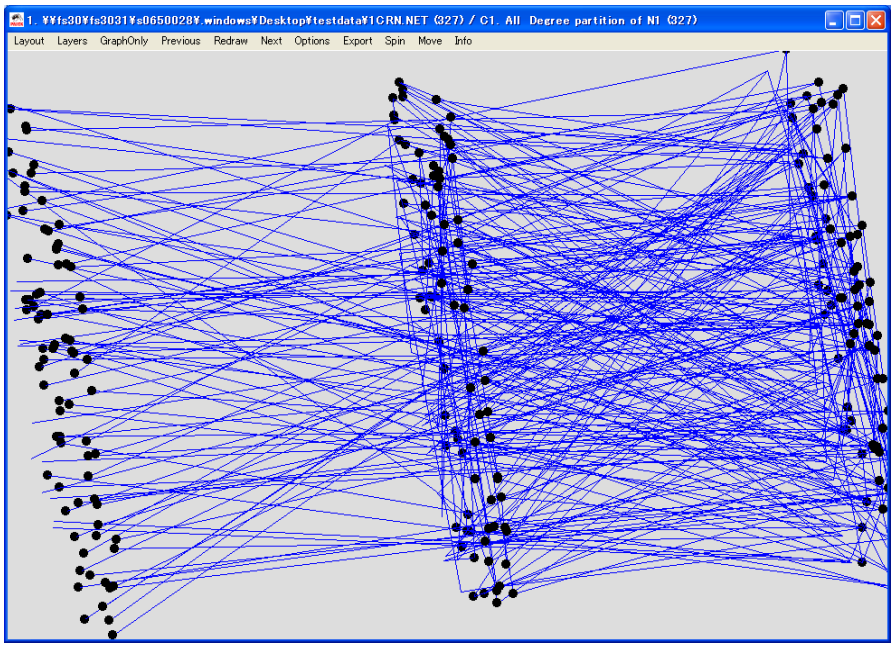
>In y direction




>In y direction+random in X



>In Z direction




立体状に配置されている。Spin してみるとわかる。

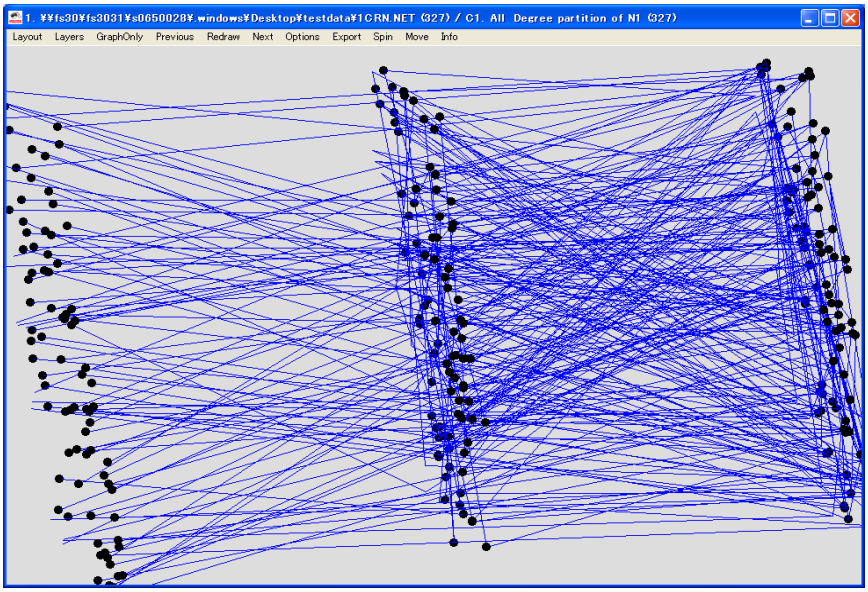
Spin>Spin around>スピンしたい角度

あるいは、ScrollBar を表示して回転させる。

Option>ScrollBar On/Off

>In Z direction+randam in xy

Z 方向へ、x と y をランダムにして表示する。



>Averaging x coordinate

>Averaging x and y coordinates

>Tile in x direction

>Tile in xy plane

>Optimize layers in x direction

>Optimize layers in xy plane

>Resoluttion

## 4.3GraphOnly

ラベルやアローを取り除いた可視化ネットワークを表示する。

## 4.4Previous

前に表示した可視化ネットワークを表示する。

## 4.5Redraw

可視化ネットワークを再描写する。

## 4.6Next

前に表示した可視化ネットワークを表示した後、次の前に表示した可視化ネットワークに戻り表示したいとき。

## 4.7Options

>Transform

可視化を変化させる。

1.Fit area

(a)max(x),max(y),max(z)

可視化の大きさを x,y,z でフィットさせる。

(b)max(x,y,z)

実際の形を整えつつ、可視化エリアをフィットさせるように再調整する。

2.Resize

選ばれた要素（x,y,z）の 3 方向すべてへ可視化を再調整する。

入力項目

①Resize factor in x direction

②Resize factor in y direction

③Resize factor in z direction

3.Translate

可視化を空間上で変形（移動）させる。

①Translate factor in x direction

②Translate factor in y direction

③Translate factor in z direction

4.Reflect y axis

y 軸で可視化を反射させる。

5.Rotate 2D

x,y 平面上で、指定された角度で回転させる。

6.Fisheye

可視化を、フィッシュアイ上に変形させる。

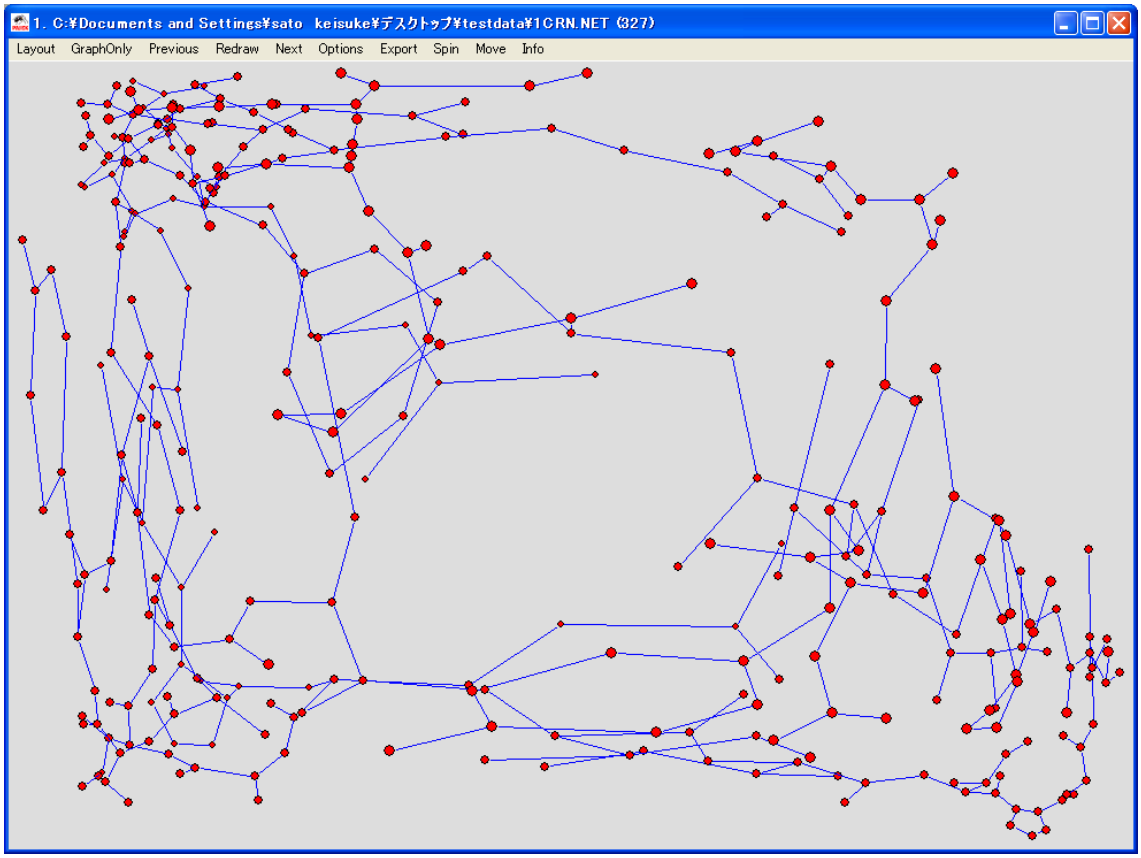
*Cartesian

直交（デカルト）座標上のフィッシュアイ

入力項目

①amount of movement

（「どれだけ動かすか？」で、値が大きいほど端に大きく寄る）




図：amount of movement：3 の Fisheye *Cartesian の例

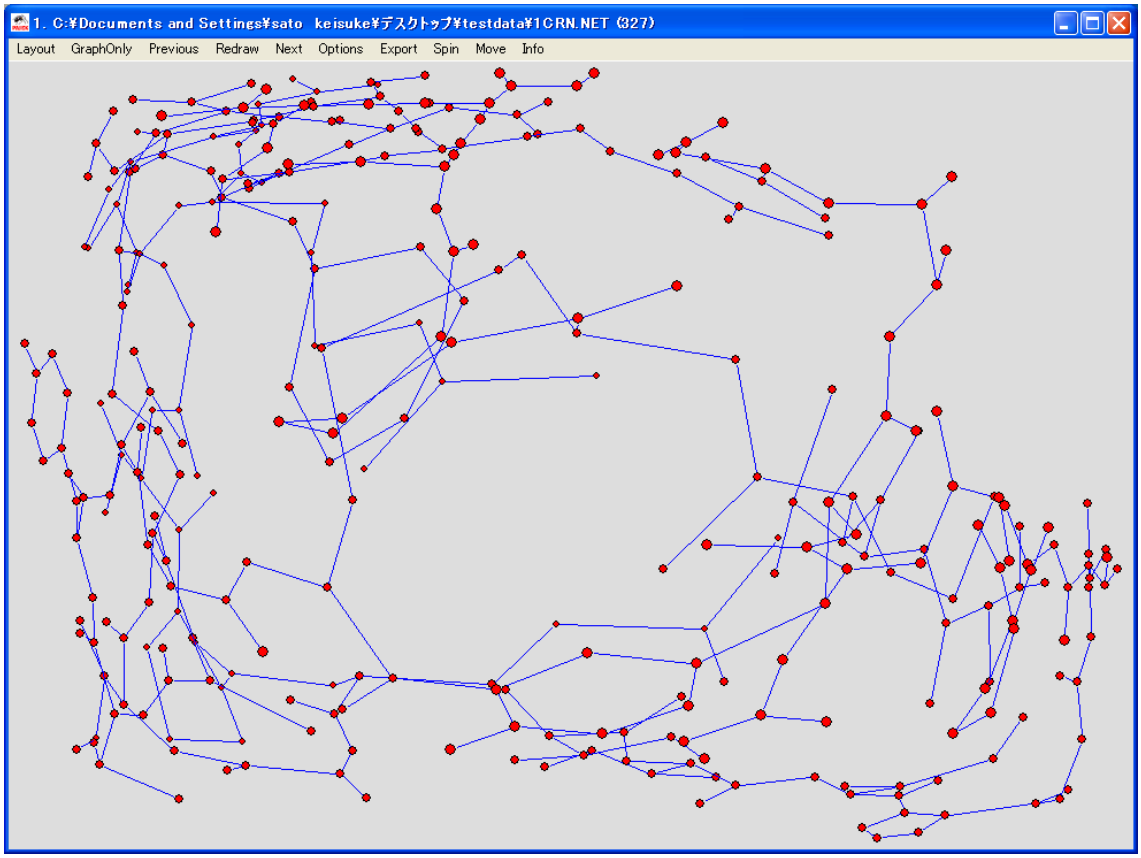
*Polar

極座標上のフィッシュアイ

入力項目

①amount of movement

(「どれだけ動かすか？」で、値が大きいほど端に大きく寄る)




図：amount of movement：3 の*Polar Fisheye の例

>Valies of lines

>Mark vertices using

ラベルやクラスの番号などを可視化したり、可視化を消去したりする。

1、Labels(ctrl+L)

ラベルを可視化する。

2、Numbers(ctrs+N)

頂点番号を可視化する。

3、Partition Cluster

Partition（クラス番号）を可視化する。

4、Vector Values

Vector の値（頂点の大きさ）を可視化する。

5、No Labels(ctrl+D)

ラベルを表示しない。または、消す。

6、No Labels No Arrows

有向線の向きの矢印とラベルを消去する。

7、Mark Cluster Only

現在クラスターに属している頂点のみ、ラベルを可視化する。

>Lines

ラインの可視化についてのメニュー

1.Draw Lines

(a)Edges

無向線（Edges）を表示するかしないかのチェック。

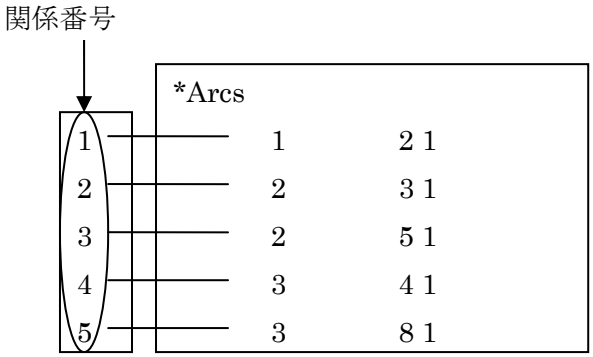
(b)Arcs

有向線（Arcs）を表示するかしないかのチェック。

(c)Relations

選んだ関係番号のラインのみ表す。

関係番号：*.net ファイルの*Arcs、*Edges のライン定義の部分の上から、関係番号 1,2,3･･･と自動的に付けられていく番号。*.net では、ユーザーは入力したり、指定することはない。




図：*.net ファイルと関係番号

補足：関係番号 1 の行の意味は、頂点 1 から 2 へラインの値（重さ）1 でつなげるという意味。

2.Mark Lines

ラインのラベルや値の可視化。

(a)No

(b)with Labels

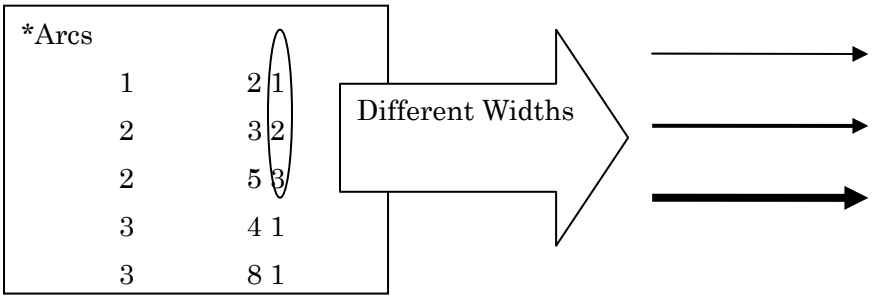
ラインのラベルを可視化する。

(c)with Values

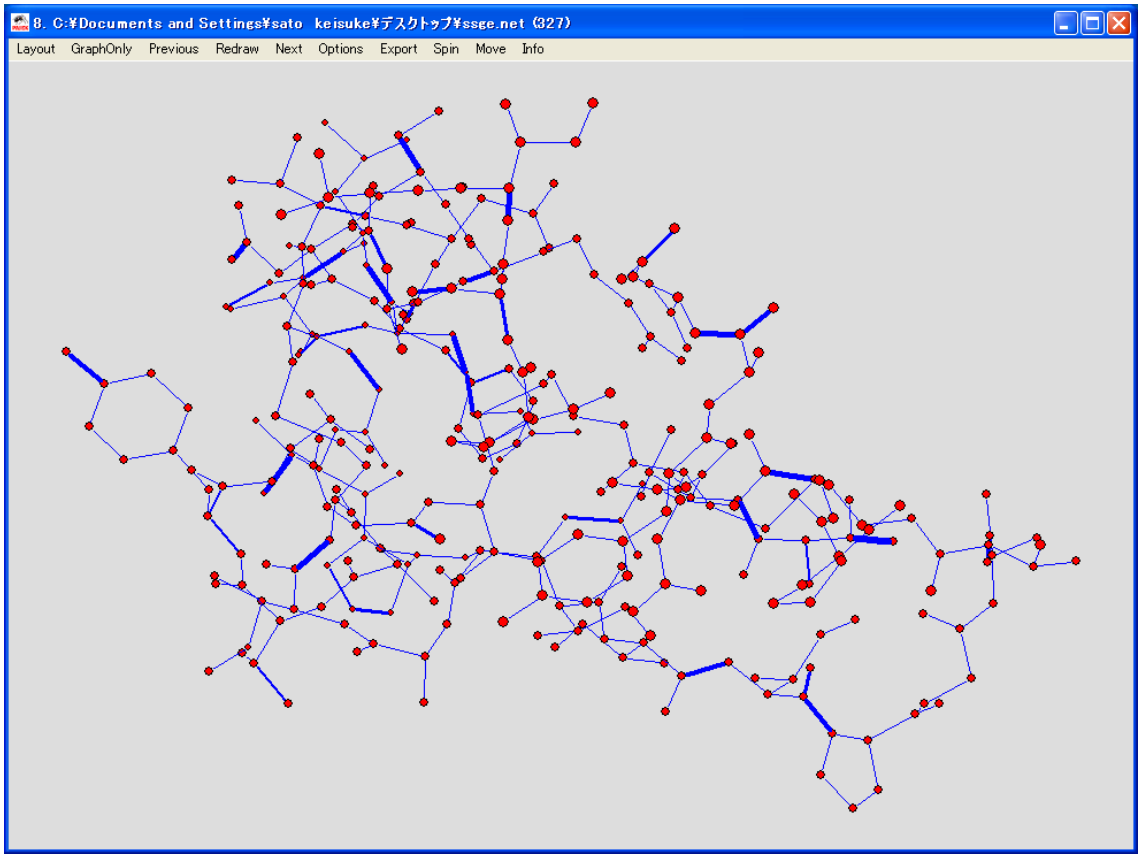
ラインの値（重さ）を可視化する。

3.Different Widths

これをチェックすると、ラインの横幅がそのラインの値（重さ）に応じた幅になる。



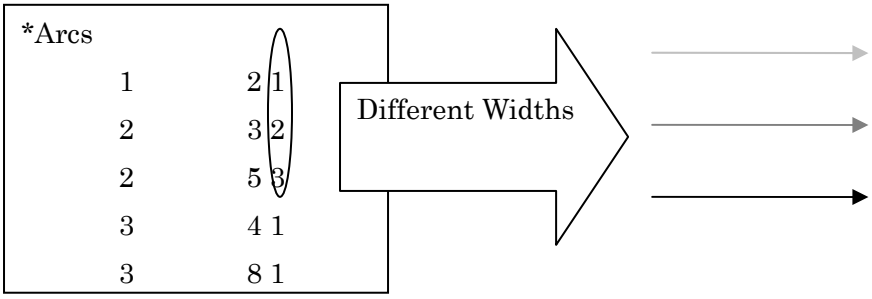

図：ラインの値（重み）とそのラインの幅での可視化



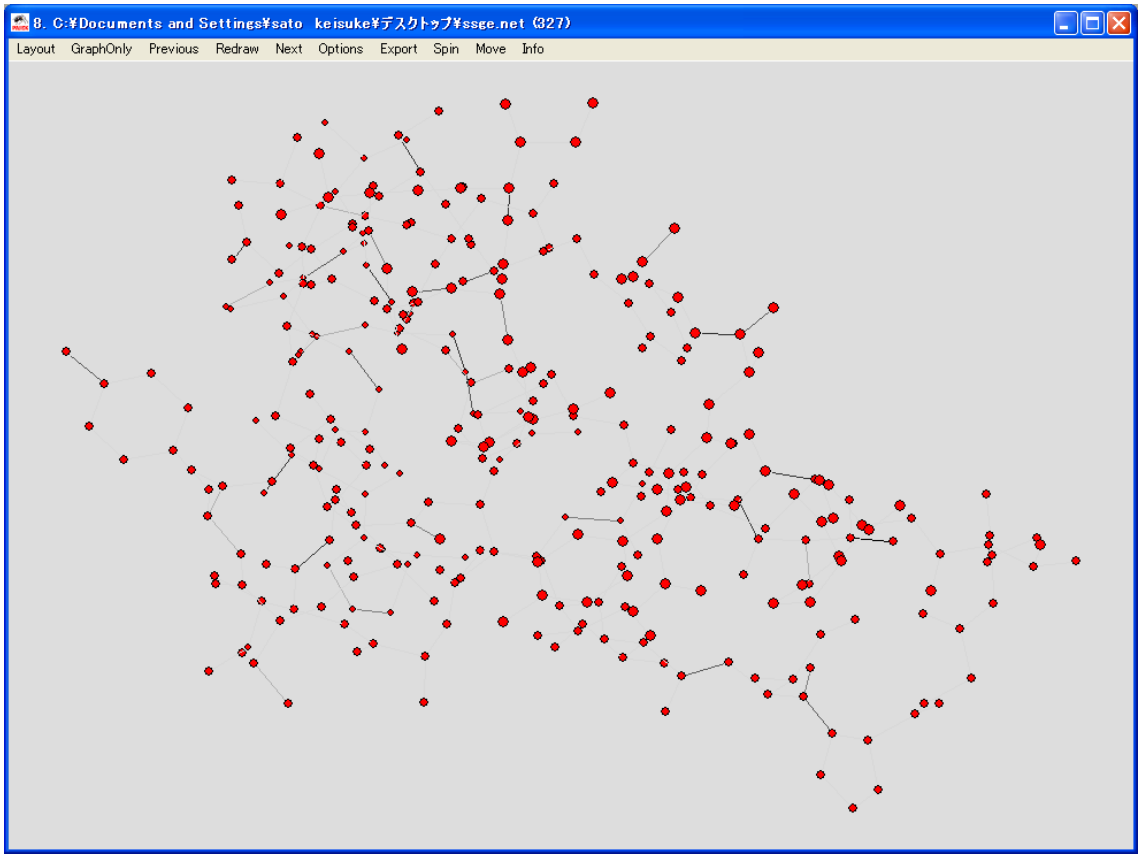

図：ラインの値（重み）をラインの幅で反映した可視化

4.GreyScale

これをチェックすると、ラインがラインの値（重み）にしたがって、黒くなる。重みが少ない順から白→灰色→黒と灰色のスケール（GreyScale）になって可視化される。



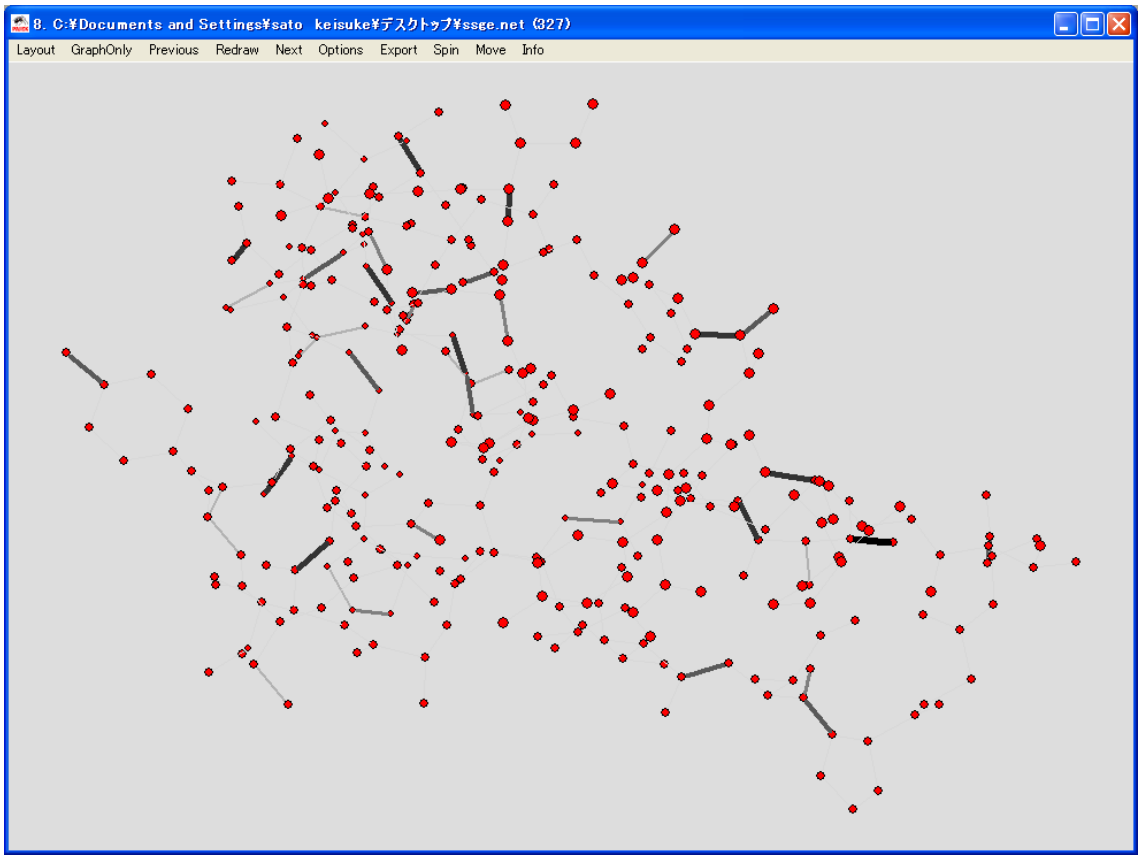

図：ラインの値（重み）とその灰色のスケールでの可視化




図：ラインの値（重み）を灰色のスケールで反映した可視化

補足：ラインの値（重み）が最小のものは、白で可視化され見えていない。

Different Widths と GreyScale は両立でき、両方チェックすると以下のように、ライン幅とグレースケールでラインの重みを可視化する。




図：ラインの値（重み）を灰色のスケールで反映した可視化

>Size

以下のものの可視化サイズを変更する。

<of vertices

頂点の大きさのサイズ変更。

入力項目

①頂点の大きさ（0 は auto）

<of vertices defined in input file

*.net ファイルの頂点の大きさの設定に従い頂点の大きさを変える。

| | | | |
|---|---|---|---|
| 大きさの設定 | x_fact | x 軸方向に何倍拡大するか | x_fact 3：x 方向に 3 倍拡大 |
| | y_fact | y 軸方向に何倍拡大するか | x_fact 5：y 方向に 5 倍拡大 |

表：*.net の頂点の大きさ設定

ただし、Vector の大きさが入るとわかりにくくなる。

<of vertitces Border

頂点の枠の横幅のサイズ変更。

入力項目

①頂点の枠の横幅

<of Lines

ラインの横幅のサイズ変更。

<of Arrows

矢印のサイズ変更。

<of Font

フォントサイズの変更。

<Default

デフォルト設定へ戻す。

>Colors

<Back Ground

可視化ウインドウのバックグラウンドの色の変更。

<Vertices

頂点の色の変更。

<Vertices Border

頂点の枠の色の変更。

<Edges

無向線の色の変更。

<Arcs

有向線の色の変更。

<Font

フォントの色の変更。

<Partition Colors

Partition の色設定を変更する。クラス番号に従い、可視化のときの色分けがなされるが、そのクラス番号と色の対応を設定できる。

<Relation Colors

Relation の色設定を変更する。

<Default

デフォルト設定に戻す。

>Layout

1.Redraw during moving

2.Real xy proportions

3.Arrows in the Middle

4.Size of vertex 0

(a)Hide vertex

(b)Hide attached lines

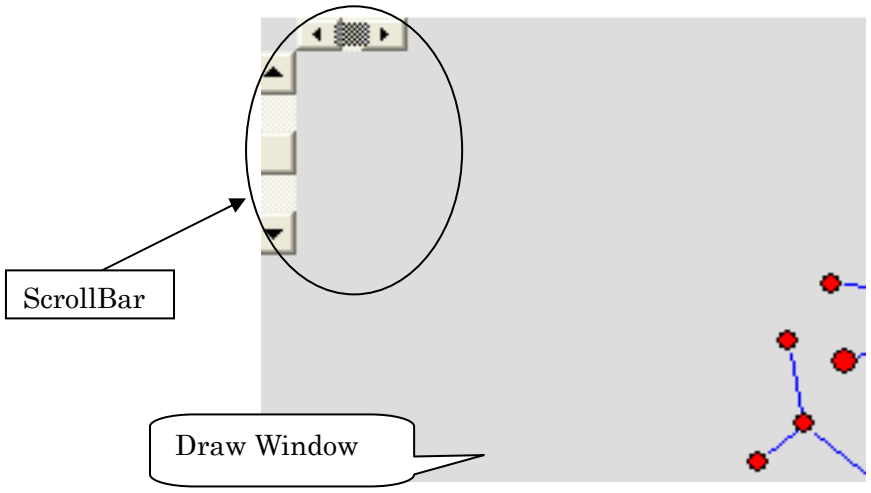
5.Decimal Places

6.Show SubLabel

>ScrollBer On/Off

ScrollBar を ON/OFF（出したり閉まったり）する。

ScrollBar：可視化されたネットワーク 3 次元的を回転させるためのバー




図：ScrollBer

>Interrupt

Interrupt：さえぎる; 妨げる; 中断する;割り込む(む), 中断（する）;インタラプト

>Previous/Next

1.Max number

2.Seconds to wait

3.Optimaize Layouts

(a)kamada-kawai

(b)2DFrucht.Rein

(c)3DFrucht.Rein

(d)No

4.Apply to

(a)Network

(b)Partition

(c)Vector

>Select all

## 4.8Export

可視化されたネットワークを各種画像ファイルに送る。

>EPS/PS

EPS フォーマットに送る。

WYSIWYG：What You See Is What You Get(見たものは取得したもの)

クリップか WYSIWYG を入れるか入れないか

Partition によって定義づけられた、色あるいは白黒の可視化ネットワークが EPS ファイルデータとなる。

PS フォーマットに送る。

PS：Program stream

MPEG-2 システムにおけるプログラムストリーム。

>SVG

SVG：Scalable Vector Graphics

XML によって記述されたベクターグラフィック言語のこと、或いは、SVG で記述された画像フォーマットのこと。

SVG（XML によって記述されたベクターグラフィック言語）や HTML 内で使用することが出来る。

1.General

2.Labels/Arcs/Edges

3.Partition

(a)Classes

(b)Classes with semi-lines

(c)Nested Classes

4.Line Values

(a)Classes

(b)Nested Classes

(c-1)Options

(c-2)Different Colors

(c-3)Using GreyScale

(c-4)Different Widths

5.Multiple Relations Network

6.Current and all Subsequent

Subsequent：

>VRML

VRML：Virtual Reality

3 次元の物体に関する情報を記述するためのファイルフォーマット。WWW 上

で利用されることを前提に設計された。

>MDL MOL file

MDL MOL file：MOD Molfile format

http://www.mdli.com （Chemscape Chime).

>Kinemages

Kinemages format：

1.Current Network Only

2,Current and all Subsequent

3,Multiple Relations Network

>Bitmap

Bitmap：windows の画像ファイル形式

>Options

EPS、SVG、VRML の初期設定のオプション

## 4.9Spin

>Spin around

角度を指定して可視化ネットワークを回転させる。

>Perspective

Perspective：遠近［透視］画法; 透視画; 遠景; 見通し, 展望; 正しい見方, 観点; 釣合い.

頂点間の距離を小さく（あるいは、なく）可視化する。

>Normal

標準頂点（Nomal vector）を旋回させる（Spin around)。

>Step in degrees

回転描写したときのステップ次数

Step in degrees：

## 4.10Move

>Fix

1. X
2. Y
3. Radius

Radius：半径

>Grid

格子の位置を座標で指定すると、可視化画面に青い点が入る。この青い点を参考に、頂点の座標を見ることができる。

>Circle

同心円の中心点の位置を座標で指定すると、可視化画面に青い点が同心円上に入る。

この青い点を参考に、中心点からの頂点の座標を見ることができる。

>Grasp

追加された頂点を動かす。

1.Closest Class Only

2.Closest Class and Higher

3.Closest Class and Lower

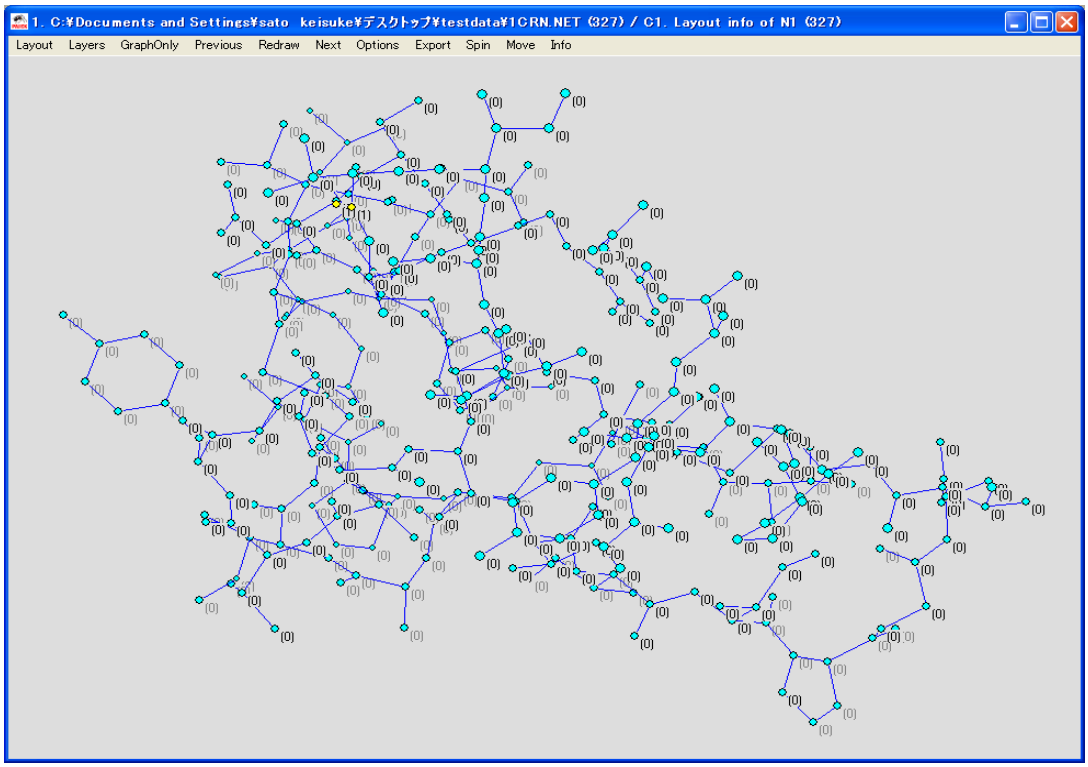
## 4.11Info

以下の美的な特徴をの可視化を行う。

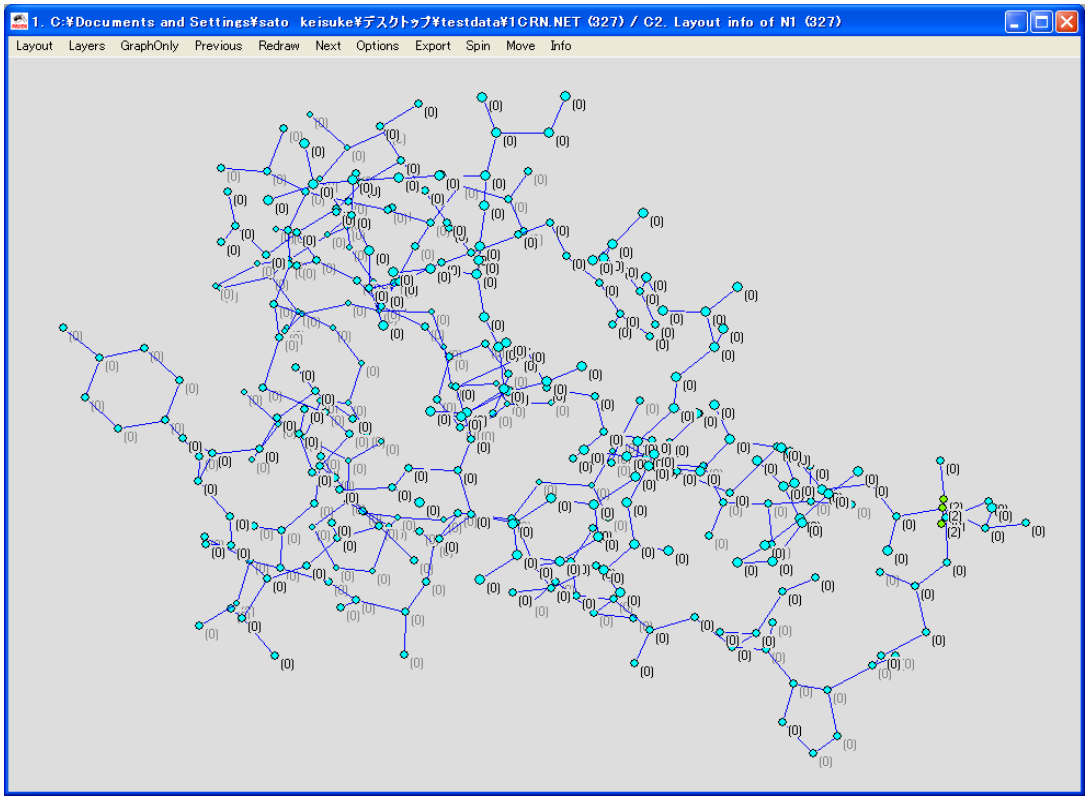
このときの頂点は、0 と 1 の間の値でないと処理できない。

レポートに Partition の方法が表示され、Partition ボックスに、*.clu ファイルが生成される。

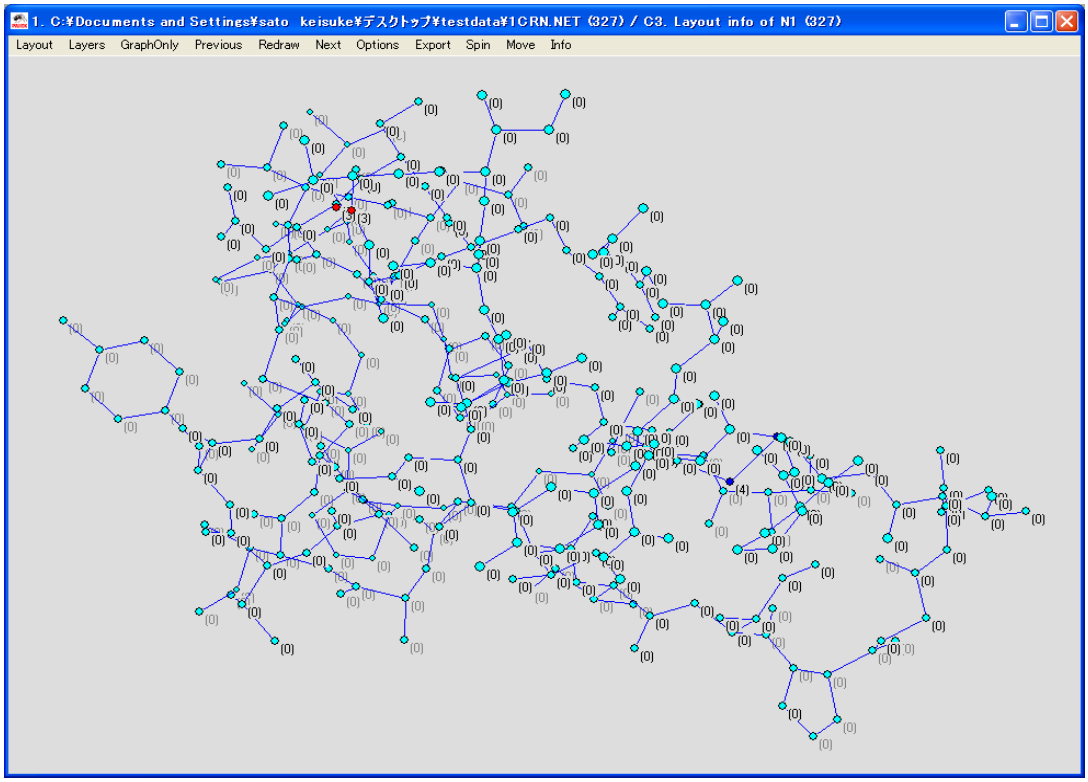
>Closest Vertices



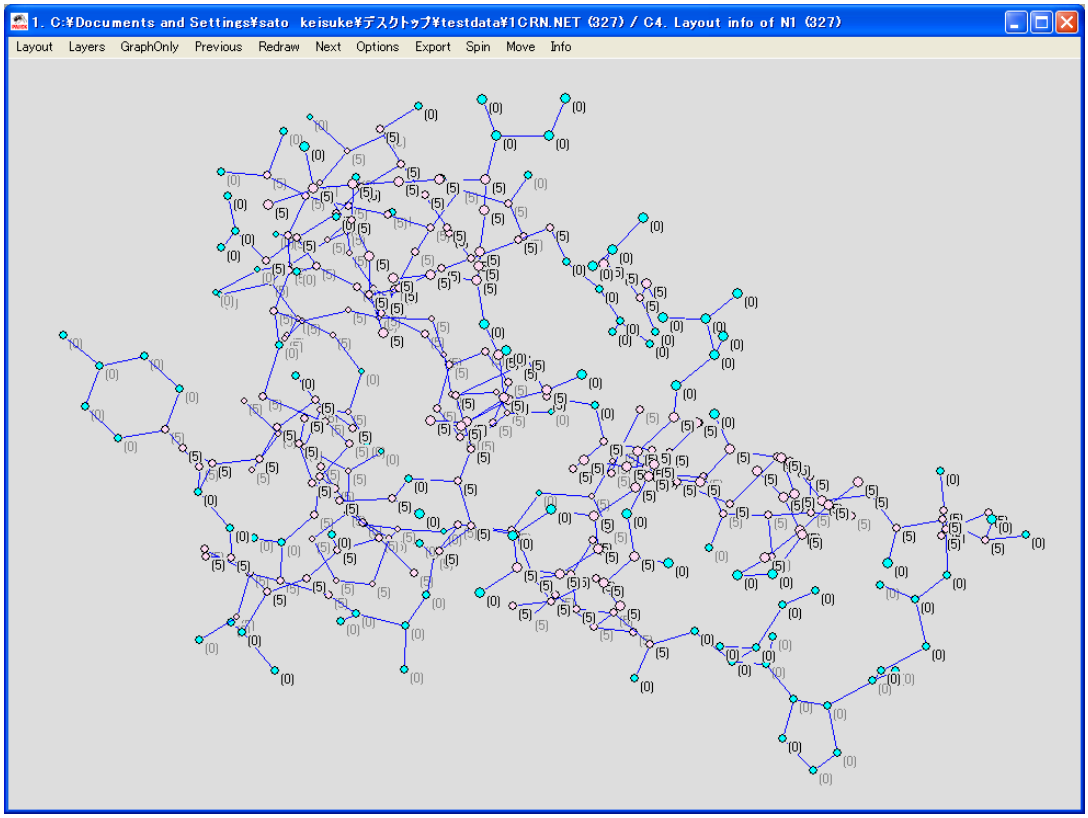
>Smallest Angle



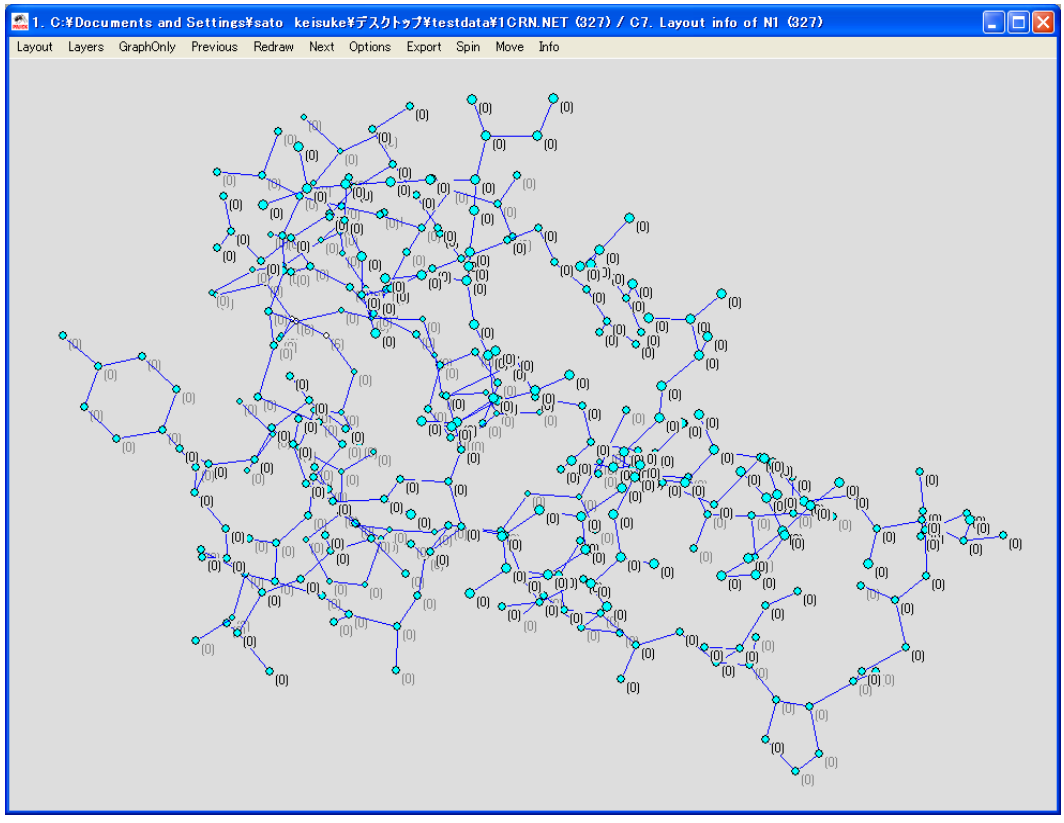

>Shortest/Longest Line



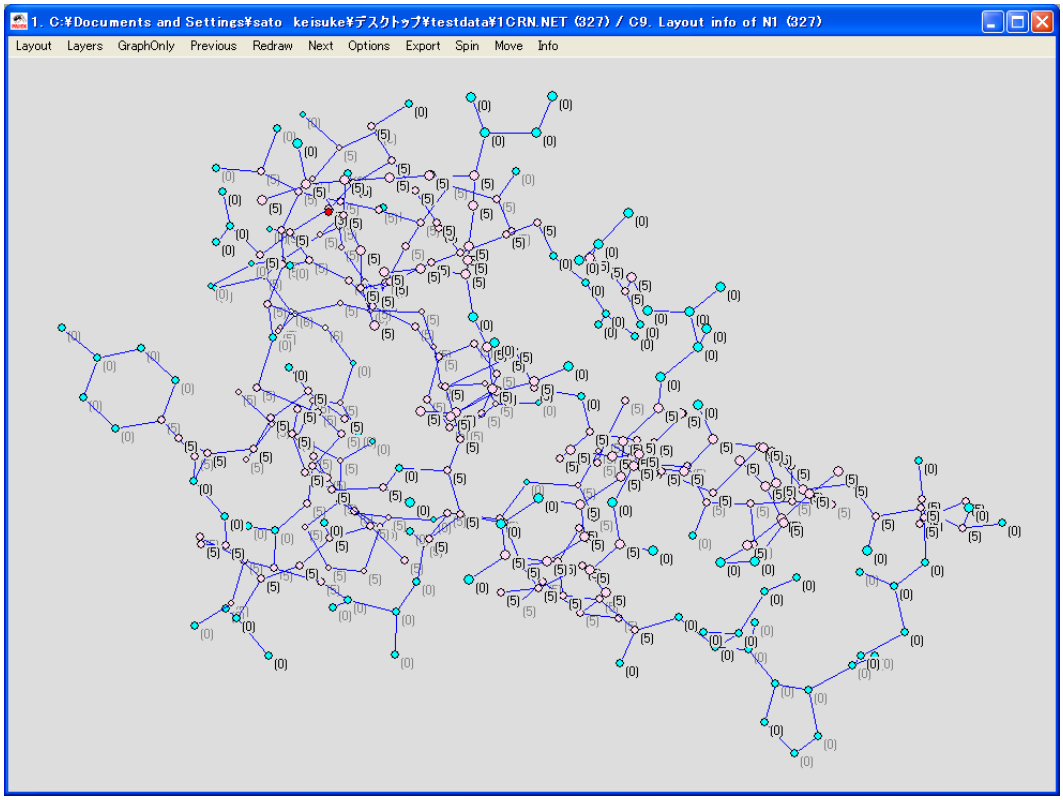
>Number of crossings if lines

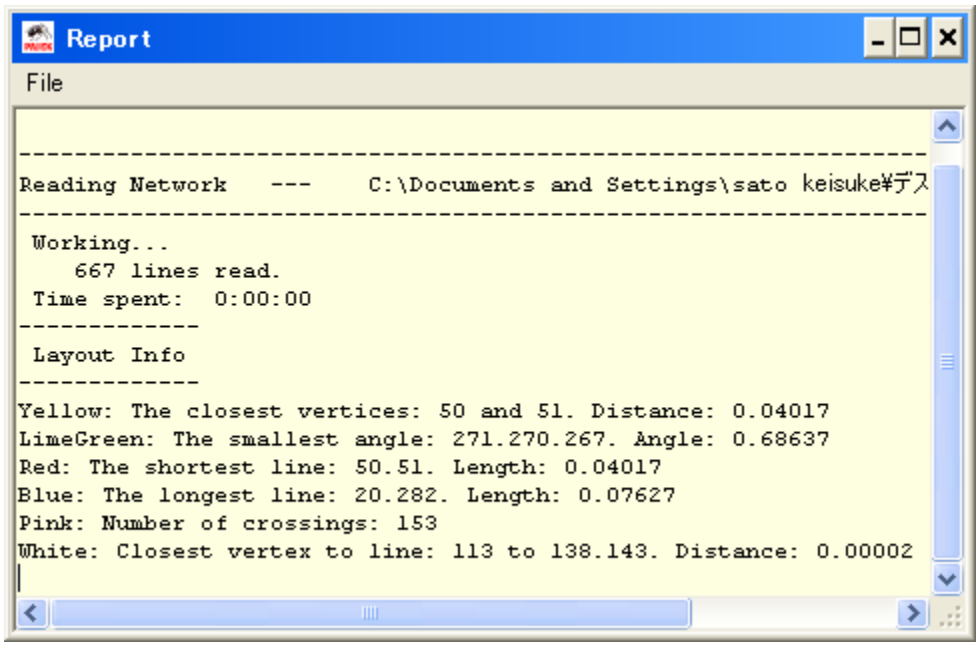



>Vertex Closest to Line



>All properties



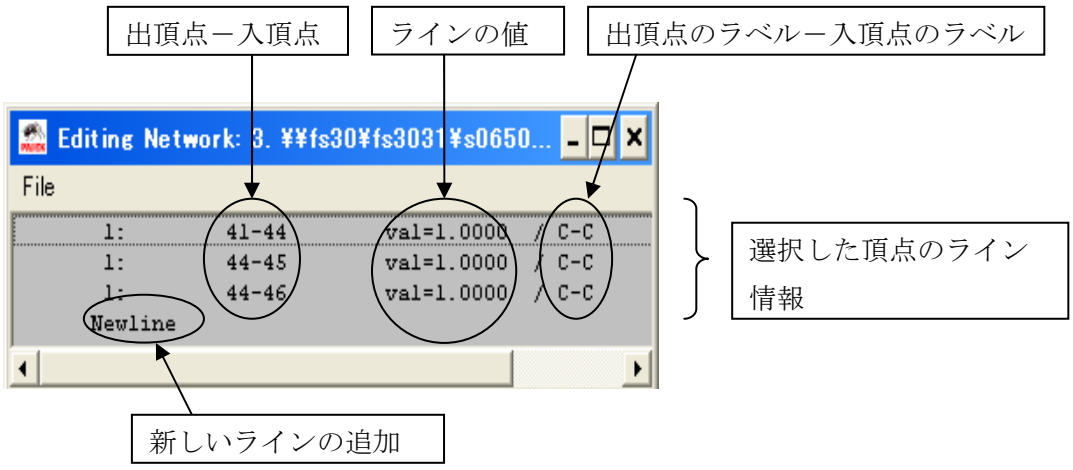

図：All Propaties のレポート例

## 4.12 可視化画面自体処理

頂点のクリック

1、頂点の右クリック

その頂点のリンク情報の一覧ウインドウが表示される。そのウインドウで、新しいラインの追加を行うことが出来る。




図：可視化画面での頂点情報

2、頂点の左クリック

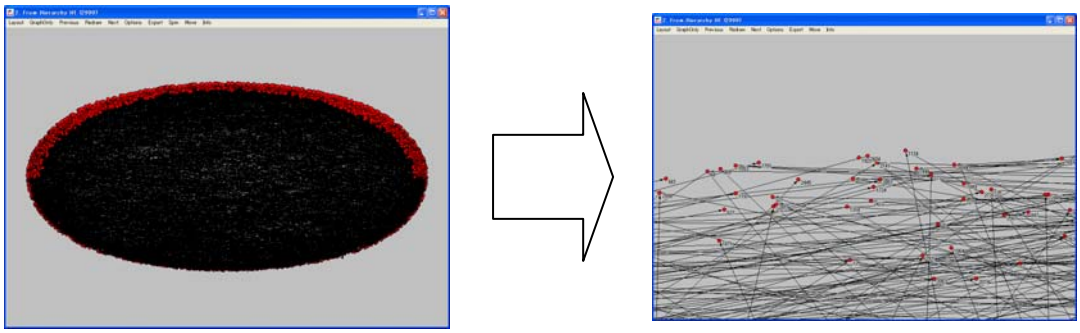
頂点を左クリックしたまま、ドラッグするとその頂点を移動させることが出来る。

Parririon毎に可視化したときのParitionの移動

Parititon に左クリックをして移動させることが出来る。

可視化の拡大

頂点のない箇所で右クリックして、拡大したいエリアを指定するとその可視化ネットワークを拡大することが出来る。「Redarw」コマンドで拡大を解除できる。



図：右クリックによる範囲指定での可視化の拡大

# 5.参考文献

[1] NetScience UCINET6 https://www.netscience.ne.jp/software/ucinet/

[2]ISP のネットワーク性と生産性，峰滝和典，2006，pp.6
http://www.stanford-jc.or.jp/research/publication/DP/pdf/DP2006_006_J.pdf

[3]著者の役割を考慮した共著ネットワークの比較分析：HITS アルゴリズムに基づく手法の改善，芳鐘冬樹・野澤孝之，pp3,4
http://svrrd2.niad.ac.jp/faculty/fuyuki/shihyo_yoshikane.pdf

[4]Complex Networks 複雑ネットワークの科学 他の頂点への平均距離（頂点ごとの L）の小ささ
http://www.riec.tohoku.ac.jp/~uep/pukiwiki.php?Complex%20Netwoks-%CA%A3%BB%A8%A5%CD%A5%C3%A5%C8%A5%EF%A1%BC%A5%AF%A4%CE%B2%CA%B3%D8

[5]代謝ネットワーク解析に基づく形成モデルの提案，松田大典，2004，pp.10
http://www.es.dis.titech.ac.jp/thesis/2003/02m35616.pdf

[6]ネットワーク分析用語集
http://www5.ocn.ne.jp/~yasuda/networds.htm

[7]立教大学大学院社会学研究科 応用社会学専攻守ロゼミ 第 5 章社会ネットワークの構造分析＜中心性＞
http://www.soi.wide.ad.jp/class/99001/slides/11/14.html

[8]人的つながりと組織に関する研究―ソーシャル・キャピタルのネットワーク分析－, 2006, 中村克久，pp16,17
http://ds9.jaist.ac.jp:8080/ResearchData/main/2005/knakamu/修論.pdf

[9]Vladimir Batagelj and Andrej Mrvar, Pajek Raferrence Manual List of commands with short explanation version 1.15,Ljubljana,July 25 2006

[10]離心率
http://www.osaka-kyoiku.ac.jp/~tomodak/quadratic/reference/rishin.html

[11]Network workshop NICTA,Sydney,June 2005「Newworks」Vladimir Batagelj University of Ljubojana