社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2006－FI－85（4）
2006／11／16

大規模クラスタリング結果のグラフによるインタラクティブな可視化手法

井上　悦子[†]　　吉廣　卓哉[††]　　中川　優[††]

[†] 和歌山大学大学院システム工学研究科
[††] 和歌山大学システム工学部

本研究では，大規模なクラスタリング結果のグラフを用いた可視化手法を提案する．一般にグラフによる要素間の関係の可視化においては，データが大規模になるとノード数・リンク数ともに増加するため，グラフが過密になり要素間の関係把握が困難になる．提案手法では画面上の表示ノード数・リンク数を一定範囲に保ちながら，全体の概観から詳細な要素間の関係までの表示粒度を連続的に変更することができ，いわばクラスタリング結果空間を自由に探索することができるようなインタラクティブな閲覧環境を実現している．このような閲覧環境の実現にはユーザの操作にリアルタイムに追従できる高速なグラフ変形が必須であるが，本稿ではこれを実現するようなデータ構造とアルゴリズムを提案する．また，Java を用いて試作を行い，操作感について評価を行った．

A New Interactive Graph-based Visualization Method for Large-scale Clustering Analyses

Etsuko INOUE,[†] Takuya YOSHIHIRO[††] and Masaru NAKAGAWA[††]

[†] Graduate School of Systems Engineering, Wakayama University
[††] Faculty of Systems Engineering, Wakayama University

This paper presents a new interactive graph-based visualization method for large-scale clustering analyses. Generally in graph-based visualization, the number of nodes and links seriously increases as the size of data becomes large, which results in overcrowded graphs with lots of nodes and links. To avoid the problem, we propose a new method which keeps the number of nodes and links while changing view levels between general view to see highly clustered nodes and the detailed view to see each element of data. The novel feature of our method is that we can explore the whole clustering result with our real-time operations in our viewer. To realize such interactive viewer, we present data structure and algorithms to make real-time changes of graph shape within a short time not to have users feel any stress. We also implement such viewer and make evaluation about user operations.

1. はじめに

階層的クラスタリングとは，多次元のデータセットに対して，要素間の類似度（ユークリッド距離や相関係数等が頻繁に用いられる）に基づいて比較的「近い」要素群をクラスタとして発見する分析手法の一つである．階層的クラスタリングの結果を可視化する代表的な手法として，デンドログラム（樹状図）がある．デンドログラムは，全要素の中で最も類似度の高いものを一つのクラスタ（複数の要素が結合してできた要素を特にこのように呼ぶ）に結合することを繰り返し，最終的に一つのクラスタになるまでの過程を木構造で表現したものである．デンドログラムを用いることで，各要素がどのような分布でどの程度類似しているかを概観することができる．しかし，デンドログラムは木構造の表現であるため，例えばある要素に注目した時に，比較的近い要素であっても，要素間の類似度を直接的に知ることができないという欠点がある．

この問題を解決し，多数の要素間の類似関係を一望する可視化手法として，グラフによる可視化が注目されている．グラフを用いた可視化手法では，要素をノードで表し，要素間の類似度をリンクで表すため，全ての要素間の類似度を正確に表現することができる．しかし一方で，データが大規模である場合にはノード数，リンク数共に膨大になるため，グラフが過密になり要素間の関係を把握することが困難になる．グラフによる可視化においては，いかにしてユーザの見たい部分を適度な密度で見せるか，を工夫することが重要である．

グラフによる可視化例の一つに，Fisheye View[1]がある．Fisheye View は，大規模なグラフを 2 次元平面上に描画し，この中で注目したい部分を拡大表示し，

それ以外の部分は全体が概観できるように縮小表示することで，全体の概観と注目部分の詳細化を両立している．しかし，2 次元平面上に描画したグラフの拡大率を変更するだけでは，注目部分に応じて適切にリンク密度を変更することができない．このため，部分によりリンク密度に疎密差のあるグラフでは，過密部分を適切に表示することができない点で限界がある．

階層化グラフ[2]もグラフによる可視化例として挙げられる．この手法は，まず全ての要素を 1 画面に表示できる数になるようにクラスタ化し，これらのクラスタをノードとしてグラフ形式で表示する．ユーザがクラスタを選択することでそのクラスタに含まれる要素を展開し，同様により詳細なグラフを表示する．この操作を繰り返すことにより，ユーザは全体の概観から詳細の把握までを行うことが可能である．この手法は画面内のノード数とリンク数を見やすい範囲に保つことができるため，グラフ密度による問題は発生しない. しかし, この手法ではクラスタを詳細化する時に，そのクラスタ内の要素しか表示できない点が欠点となる．つまり，ある階層では別のクラスタに含まれているにもかかわらず，実はそれなりに近い要素がある場合に，これを発見できない．（このような例は，要素が比較的均一に分布している場合などに良く見られる．）

そこで本研究では，階層化グラフの手法をさらに発展させ，要素数が多い場合にも見たい範囲を任意の階層，適切な密度で表示することができる可視化手法を提案する．提案手法では，表示グラフ中に注目したい「中心ノード」を設定し，このノードを中心として表示画面の階層を連続的に変化させることで，リアルタイムに画面上のグラフを変化させる．階層を連続的に変化させるため，中心ノードに近いノードが常に表示範囲に残るため，階層化グラフで見られたような，表示ノードが特定のクラスタ内に限られる問題は解決される．また，本方式は，中心ノードの変更と階層変化の操作により，いわばクラスタリング結果空間を自由に探索することができるような，既存手法とは一線を画した新しいインタラクティブな閲覧環境を実現している．このような閲覧環境を実現するためには，ユーザの操作をリアルタイムに追従する，画面上の高速なグラフ変形を実現する必要がある．具体的には，現在表示しているグラフから，ユーザの操作に応じた次のグラフへの高速な状態遷移を実現するためのデータ構造とアルゴリズムが必要となる．本稿では，中心ノード変更と階層変化の操作を，データが大規模であってもユーザにストレスのない速度で行えるようなアルゴリズムを提案する．また，このアルゴリズムを，Java を用いて実装し，大規模なデータであってもストレスなくクラスタリング結果を閲覧できることを確認する．

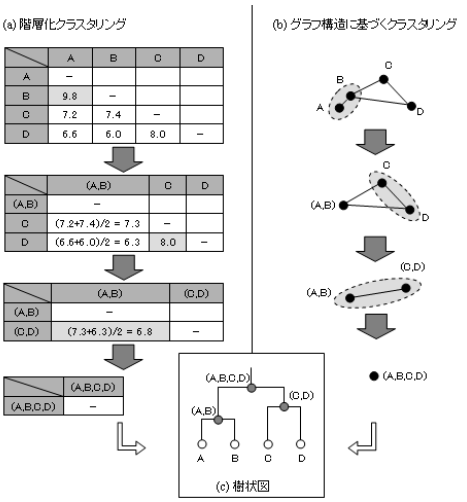

図 1　2 つのクラスタリングと樹状図の可視化

2. 準備

2.1 対象とするクラスタリングアルゴリズム

本研究では，樹状図（デンドログラム）で表現可能なクラスタリング結果を対象とする．樹状図は，階層的クラスタリング結果の可視化に用いられる二分木である．階層的クラスタリングは，要素間の関係を表す類似度と呼ばれる数値を求め，その類似度に基づいて要素の分類などを行う統計分析手法のひとつである[3]. 類似度に用いられる数値の例として，相関係数やユークリッド距離などが挙げられる．階層的クラスタリングでは，分析対象であるすべての 2 要素間の類似度を計算し，類似度が最大であるデータの組を 1 つのクラスタと呼ばれるグループへと結合し，全体がひとつのクラスタに結合されるまで結合処理を繰り返す（図 1(a)）．この結合処理の過程を可視化したものが，樹状図である（図 1(c)）．図 1(a)の例では，クラスタと他の要素との類似度として，全要素間の類似度の平均値を採用する群平均法を用いているが，この他に最短距離法，最長距離法，ウォード法などがある[3].

また，階層的クラスタリング以外の手法によるクラスタリング結果であっても，樹状図で表現できる場合は同様に本研究の対象として扱うことができる．一例として，グラフ構造に基づくクラスタリング[4]がある. これは，元々グラフとして表現される大規模ネットワーク（WWW サイトのリンク接続構造やソーシャルネットワークの知人関係など）に対して，比較的リンク

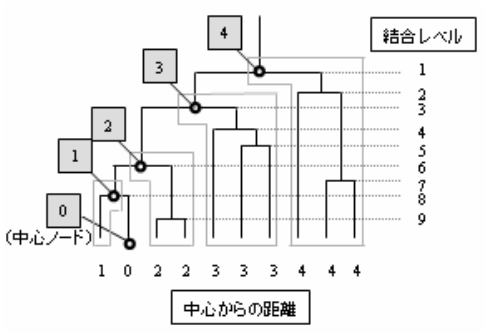

図 2　中心からの距離の設定例

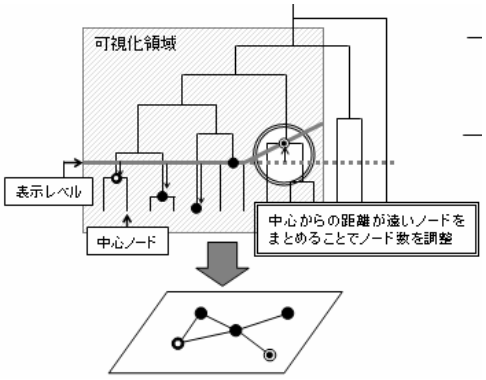

図 3　初期グラフの生成

が密な部分をクラスタとみなし，グラフ構造を階層化するクラスタリング手法である（図 1(b)）．グラフ構造に基づくクラスタリングでは，リンクが存在するノード間には関係が定義されるが，リンクが存在しないノード間には関係が定義されない．この点が，先ほどの階層的クラスタリングとは異なるが，クラスタリング結果は同様に樹状図で表現できる（図 1(c)）．なお，孤立した部分グラフが存在する場合，樹状図が複数に分断される可能性があるが，本稿では説明の簡単化のため，全体を分断されないひとつの樹状図で表せるものとする．

2.2　クラスタリング結果の可視化

本研究では，大規模なクラスタリング結果を効果的に可視化するために，注目する要素を中心に樹状図中の可視化領域をグラフ化し，そこから可視化領域を自由に移動させるように，グラフ状態の連続的な変化の実現が目的である．本節では，初めに生成する特定要素を中心としたグラフに必要となるパラメータを定義し，以後の可視化領域の基準となるこの初期グラフの生成の流れを説明する．また，初期グラフに対してどのように可視化領域を移動するかについても説明する．ただし，グラフ描画の際のレイアウトは本研究の対象外とし，既存のばねモデルなどを利用するものとする．

クラスタリング結果は，樹状図の葉または節をノード，要素またはクラスタ間の類似度を対応するノード間のリンクとすることで，グラフとして可視化できる．データ数 N の樹状図には，各データに対応する N 個の**葉ノード**と，N-1 個の**節ノード**が存在する．これらの全ノードの集合を V (|V| = 2N-1)，V の全ノード対に対するリンク集合を E (|E|≦(2N-1)^2/2)で表す．各節ノードには，根に近い方から 1,2,…, N-1 の値を割り当て，この値を各節ノードの**結合レベル**と呼び（図 2），ノード v の結合レベルを l(v)で表す．結合レベルが k である位置で，樹状図を水平に切断して得られる k 個のノードは，N 個の全データを k 個に集約した結果である（図 3）．このように，表示する要素の集約度合いを決定する値を**表示レベル** display_level (1≦display_level≦N)とし，display_level=k で得られる k 個のノードを**表示レベルノード**と呼ぶ．表示レベルノードは必ず，葉ノードであるか，結合レベルが k 以上の節ノードとなる．

ただし，表示レベルだけでは，display_level の値と同数のノードを抽出することになり，ノード数を制御することができない．そこで本研究では，表示ノード数を制御するために，可視化領域の中心となるノード（**中心ノード**）と**中心からの距離**の概念を導入する．これを用いて，Fisheye View[1]などの歪みを利用した可視化と同様に，中心からの距離が遠いノードを集約して親ノードに置き換えることで，全体として表示するノード数を制御する．さらに，表示条件として中心からの距離の最大値を設定し，この値よりも中心からの距離が遠いノードを表示しないことで，可視化対象とする領域の制御が可能となる．

中心ノード v_{center} が与えられた時，ノード v∈V の中心からの距離を $d(v_{center}, v)$で表す．中心からの距離は，図 2 に示すように，中心ノード v_{center} の中心からの距離 $d(v_{center}, v_{center})$を 0 とし，中心ノードの親ノード $p(v_{center})$ の中心からの距離 $d(v_{center}, p(v_{center}))$ を 1，さらにその親ノード $p(p(v_{center}))$の中心からの距離 $d(v_{center}, p(p(v_{center})))$ を 2，というように，中心ノードから順に 1 つずつ距離を増加させながら親ノードを辿って設定していく．根にあたる節ノード v_{root} の中心からの距離 $d(v_{center}, v_{root})$ に設定される値が，中心からの距離の最大値となり，これを d_{max} で表す．ノード v の中心からの距離は，v と v_{center} の最も近い共通の親ノードの中心からの距離と同じ値を設定する．表示レベル display_level に対して得られるノードのうち，中心からの距離が最小であ

るノードを**表示レベルに対する中心ノード** v'_{center} とする．また，可視化領域を制限する中心からの表示最大距離 d_{max_disp} は，v'_{center} の中心からの距離 $d(v_{center}, v'_{center})$ を基準に，これよりどれだけ大きい値の距離のノードまでを画面に表示するかを示している．つまり，表示されるノード v は，必ず $d(v_{center}, v'_{center}) \leqq d(v_{center}, v) \leqq d(v_{center}, v'_{center}) + d_{max\_disp}$ を満たす．

図 3 を用いて，クラスタリング結果の樹状図からの初期グラフの生成例を示す．初期グラフは，次の手順で生成する．まず，表示レベル display_level と中心ノード v_{center} を決める．これらから，中心レベルに対する中心ノード v'_{center} が求まる．中心からの表示最大距離 d_{max_disp} から，可視化対象とする領域を決定する．もし，これらの条件を満たすノード数が，**画面あたりの最大ノード数** max_display_nodes を超過する場合，ノード数が max_display_nodes 以下になるまで，中心からの距離が遠く，同じ距離であれば結合レベルの大きいノードから順に，親ノードに置換する処理を繰り返す．このようにして決定したノードを**表示ノード**と呼び，表示ノードの集合を**表示ノードリスト** V_{disp} ($V_{disp} \subseteq V$, $|V_{disp}| \leqq$ max_display_nodes)と呼ぶ．次に，表示ノードリスト V_{disp} の全ノード対に対し，類似度に基づくリンクを定義する．このリンクの集合を**表示候補リンクリスト** E_{cand} ($E_{cand} \subseteq E$, $|E_{cand}| \leqq 2*(|V_{disp}|-1)^2/2$)と呼ぶ．$|E_{cand}|$が大きい場合に，リンクが過密なグラフとなることを防ぐために，**画面あたりの最大リンク数** max_display_links で表示するリンク数を制御する．表示候補リンクリスト E_{cand} のうち，類似度が大きい順に max_display_links 個のリンクを抽出した集合を**表示リンクリスト** E_{disp} ($E_{disp} \subseteq E_{cand} \subseteq E$, $|E_{disp}| \leqq$ max_display_links)と呼ぶ．最終的に，V_{disp} および E_{disp} より，初期グラフを表示する．以上より，初期グラフを決定するためのパラメータは，表示レベル display_level，中心ノード v_{center}，中心からの表示最大距離 d_{max_disp}，画面あたりの最大ノード数 max_display_nodes，画面あたりの最大リンク数 max_display_links の 5 つである．

初期グラフを定義する 5 つのパラメータはすべて変更可能であるが，特に重要なのが，可視化領域を制御する表示レベル display_level と中心ノード v_{center} である．表示レベル display_level は，表示するデータの粒度を決定するため，display_level を増加させると粒度が細かくなってノードは分割され，逆に display_level を減少させると粒度が粗くなってノードは結合される．つまり，表示レベルを変更することで，樹状図内の可視化領域を垂直方向に移動できる．また，中心ノード v_{center} は，樹状図内の可視化領域を決定する基準となるため，これを変更することにより，樹状図内を水平方向に移動できる．したがって，表示レベルと中心ノードを変更により，樹状図内の任意領域を可視化できる．

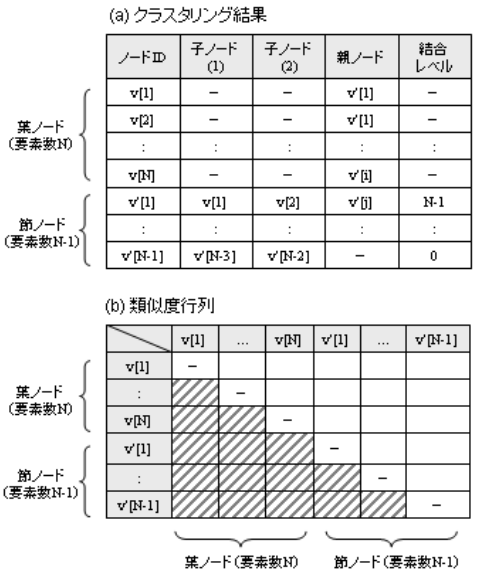

図 4　クラスタリング結果と類似度行列の例

3. グラフの状態変化と計算量

3.1 データ構造

本研究では扱うデータは以下のように，(A)初期グラフや可視化領域を制御するパラメータに依存し，グラフの状態によって更新されるデータと，(B)グラフの状態やパラメータからは独立したデータとに分類できる．本方式を実現するソフトウェアは，その両方をメモリ上に保持する必要がある．

(A)に属するデータには，表示グラフを決定する 5 つのパラメータ（表示レベル display_level，中心ノード v_{center}，中心からの表示最大距離 d_{max_disp}，画面あたりの最大ノード数 max_display_nodes，画面あたりの最大リンク数 max_display_links），表示ノードリスト V_{disp}，表示リンクリスト E_{disp} および表示候補リンクリスト E_{cand}，全ノードの中心からの距離（$v \in V$ であるすべての v に対する中心からの距離 $d(v_{center}, v)$の集合で，要素数は|V|となる）がある．それぞれのデータは 2.2 節に述べた通りである．

(B)に属するデータには，図 4 に示すクラスタリング結果と類似度行列がある．クラスタリング結果は，N 個の葉ノードと N-1 個の節ノードに関して，それぞれの親ノード，2 つの子ノード，結合レベルをまとめたデータである．類似度行列は，データ数 N に対して，

表 1　ノードの基本操作

function union_nodes(v_1, v_2)

1. **if** ($d(v_{center}, v_1)$!= $d(v_{center}, v_2)$ exit
2. **if** ($p(v_1)$!= $p(v_2)$) exit
3. V_{disp} ← V_{disp} - {v_1, v_2} + {v_p (= $p(v_1) = p(v_2)$)}
4. E_{cand} から，v_1 または v_2 を端点に持つリンクを全て削除
5. v_p に対して，$e = (v_p, v) \in E, v \in V_{disp}$ を満たす e 全てから成るリンクリスト E'を生成し，類似度で降順にソート
6. E_{cand} と E'をマージし，E_{cand} を生成

function divide_node(v)

1. **if** (v が葉ノード) exit
2. V_{disp} ← V_{disp} - {v (= $p(v_1) = p(v_2)$)} + {v_1, v_2}
3. E_{cand} から，v を端点に持つリンク全てを削除
4. v_1, v_2 に対して，$e = (v_1$（または v_2）, v')$\in E$, $v' \in V_{disp}$ を満たす e 全てから成るリンクリスト E' を生成し，類似度で降順にソート
5. E_{cand} と E'をマージし，E_{cand} を生成

function add_node_by_dist(d)

1. **if** ($v \in V_{disp}$ に対して，$d(v_{center}, v)$の最大値≧d) exit
2. $v \in V_{disp}$ かつ $d(v_{cemter}, v) = d$ である v のうち，l(v) が最小である v_{add} を V_{disp} に加える
3. v_{add} に対して，$e = (v_{add}, v) \in E, v \in V_{disp}$ を満たす e 全てから成るリンクリスト E' を生成し，類似度で降順にソート
4. E_{cand} と E'をマージし，E_{cand} を生成

function delete_nodes_by_dist(d)

1. if ($v \in V_{disp}$ に対して，$d(v_{center}, v)$の最大値 != d) exit
2. V_{disp} から，d(vcenter, v) = d である v 全てを削除
3. E_{cand} から，2 で削除されたノードを端点とするリンク全てを削除

葉ノードと節ノードを合わせた[2N-1]x[2N-1]の行列となる．ただし，本研究では関係の方向性を考慮しないため，(u, v)と(v, u)は等しくなり，実際に必要となるデータ量は$(2N-1)^2/2$ となる．

3.2　グラフに対する基本操作

グラフの状態変化の説明をスムーズに行うために，基本操作となるノードの結合・分割，中心からの距離によるノードの追加・削除の４つの関数を定義する．それぞれの関数について，簡単に説明する．なお，本稿のアルゴリズムでは，画面上のグラフの表示変更に関する記述を省略しているが，表示ノードリスト V_{disp} および表示候補リンク E_{cand} が変更された場合には，表示リンクリスト E_{disp} を更新し，削除対象ノードについてはそのノードに連結された全リンクを削除してからノードを削除し，追加対象ノードについてはノードを追加してからそのノードに連結された全リンクを追加する処理を自動的に行うことで，画面上のグラフに変更を反映するものとする．また，これらの処理は画面上のノードやリンクの追加・削除を伴うため，グラフレイアウトの問題を含んでいる．本研究では，グラフ変化時に毎回再描画を行うのではなく，グラフの変化後に（ノード追加時には適切な初期位置に配置してから）時間とともに少しずつノードの位置を最適化するようなグラフ描画プログラムを想定している．

ノード結合（union_nodes）は，表示ノードリスト中の 2 つのノードを，親ノードに置換する関数である．結合対象の 2 ノード v_1 および v_2 は，中心からの距離と親ノード v_p が同一であるものとし，これを 1-2 行でチェックする．第 3 行は，表示ノードリスト V_{disp} に関する処理で，v_1 と v_2 を削除し，v_p を追加する．4-6 行目は，表示候補リンクリスト E_{cand} に関する処理で，削除対象である v_1，v_2 の一方を端点とする全リンクを削除し，追加対象である v_p を端点とするリンクリスト E' を生成し，E'を類似度で降順にソートした後，全リンクが類似度の降順になるように E_{cand} と E'を統合し，E_{cand} を更新する．リンクリストに関する処理は，他の 3 つの関数でも共通である．ノード分割（divide_node）は，表示ノードリスト中の 1 つの節ノードを 2 つの子ノードに置換する関数である．分割対象のノード v は節ノードであるものとし，これを第 1 行でチェックする．2-5 行は，表示ノードリストおよび表示候補リンクリストに対する変更で，ノード結合と同様である．

中心からの距離によるノード追加（add_node_by_dist）は，中心からの距離 d を指定し，中心からの距離が d であるノードを表示ノードリストに追加する関数である．中心からの距離が d であるノードのうち，結合レベルが最小のノードを追加することで，グラフ状態の変化を最小限にできる．第 1 行では，表示ノードリスト V_{disp} に中心からの距離が d であるノードが存在しないことをチェックする．第 2 行では，中心からの距離が d であるノードのうち，結合レベルが最小のノード v_{add} を V_{disp} に追加する．3-4 行では，ノード結合と同様にリンクリストを更新する．中心からの距離によるノード削除（delete_nodes_by_dist）では，中心からの距離 d を指定し，表示ノードリストから，中心からの距離が d であるノード群を削除する関数である．第 1 行では，表示ノードリスト V_{disp} に中心からの距離が d

表 2　ノード数の制御

function unroll_nodes()

1. **while** ($|V_{disp}|$ < max_display_nodes)
2. $v \in V_{disp}$かつ $d(v_{center}, v) = d_{rollup}$であるvのうち, l(v)が最小の節ノード v_{divide} を選ぶ
3. **if** (v_{divide} != null)
4. divide_node(v_{divide})
5. **if** ($v \in V_{disp}$かつ $d(v_{center}, v)=d_{rollup}$であるvの数 = 0)
6. 　　d_{rollup} += 1
7. **else** return

function rollup_nodes

1. **while** ($|V_{disp}|$ > max_display_nodes)
2. 　　$v \in V_{disp}$かつ $d(v_{center}, v) \geqq d_{rollup}$のうち, $d(v_{center}, v_1) = d(v_{center}, v_2)$ かつ $p(v_1) = p(v_2)$を満たす v_1, v_2 を選ぶ
3. **if** (v_u != null)
4. 　　union_nodes(v_1, v_2)
5. **else** return

であるノードが存在することをチェックする．第2行では，中心からの距離が d である全ノードを，表示ノードリスト V_{disp} から削除する．第3行では，表示候補リンクリストから，削除対象のノードを端点とする全リンクを削除する．

3.3　表示レベルの変更

表示レベルの変更は，表示レベルを減らす処理（=樹状図のより浅いレベルを見る）と，表示レベルを増やす処理（=樹状図のより深いレベルを見る）の 2 つの場合に分けて説明する．これらの処理では，表示ノード数を制御するために，中心からの距離によりノードを結合する関数（rollup_nodes）およびノードを分割する関数（unroll_nodes）を先に定義する（表2）．ここで，$v \in V_{disp}$かつ表示レベルノードでないvのうち，中心からの距離の最小値を d_{rollup} とする．表示レベルノードでないノードが存在しない場合は，V_{disp} のうち，中心からの距離の最大値に 1 加算した値を d_{rollup} とする．

中心からの距離によるノード分割（unroll_nodes）は，表示ノードリストの要素数$|V_{disp}|$が，画面あたりの最大ノード数 max_display_nodes を下回る場合に，表示レベルノードでないノードを分割することで，ノード数を制御する関数である．この時，中心からの距離が近い非表示レベルノードから分割を行う．中心からの距離によるノード結合（rollup_nodes）は，表示ノードリストの要素数$|V_{disp}|$が，画面あたりの最大ノード数 max_display_nodes を超過する場合に，中心からの距離が遠いノードを結合し，親ノードに置換する．

表 3　表示レベルの増減

algorithm decrease_display_level

1. $v \in V_{disp}$かつ $d(v_{center}, v) < d_{rollup}$ であるvのうち, l(p(v))が最大となる v_x とその兄弟 v'_x を選ぶ（ただし，$d(v_{center}, v_x) \leqq d(v_{center}, v'_x)$）
2. **if** (v_x = null and v'_x = null) exit
3. union_nodes(v_x, v'_x)
4. display_level ← l(p(v_x)) - 1
5. **if** ($v_x = v'_{center}$)
6. 　　**if** ($d(v_{center}, v) + d_{max\_disp} < d_{max}$)
7. 　　add_node_by_dist($d(v_{center}, v) + d_{max\_disp}$)
8. unroll_nodes()

algorithm increase_display_level

1. **if** ($|V_{disp}|$ = max_display_nodes)
2. 　　d ← d_{rollup}
3. 　　**while** ($v \in V_{disp}$かつ $d(v_{center}, v)$ = d である v の数 = 1)
4. 　　d -= 1
5. 　　v_{divide} ← $v \in V_{disp}$かつ $d(v_{center}, v) < d$ である v のうち，l(v)が最大の節ノード
6. **else**
7. 　　v_{divide} ← $v \in V_{disp}$かつ $d(v_{center}, v) < d_{rollup}$ であるvのうち，l(v)が最大の節ノード
8. divide_node(v_{divide})
9. display_level ← l(v_{divide})
10. **if** ($v_{divide} = v'_{center}$)
11. 　　**if** ($d(v_{center}, v_{divide}) + d_{max\_disp} \leqq d_{max}$)
12. 　　delete_nodes_by_dist($d(v_{center}, v_{divide}) + d_{max\_disp}$)
13. 　　unroll_nodes()
14. rollup_nodes()

表示レベルを減らす処理（decrease_display_level）では，表示ノードのうち，親ノードの結合レベルが最大のノード 2 つの結合を行う（表 3）．1-4 行では，結合対象となる 2 つの表示レベルノードを決定し，union_nodes 関数により結合する．第 5 行では，新たな表示レベル display_level を設定する．6-8 行は，表示レベルに対する中心ノード v'_{center} を結合した場合に，表示対象とする距離の範囲を調整する処理である．第 9 行では，unroll_nodes を実行し，表示ノード数の不足を調整する．

表示レベルを増やす処理（increase_display_level）では，表ノードのうち，結合レベルが最大である節ノードの分割を行う（表 3）．ただし，表示ノード数が画面あたりの最大ノード数に等しい場合，ノード分割に続

表 4　中心ノードの変更

```
algorithm change_center_node
1.  中心ノードを v_center とし, v∈V に対して d(v_center, v) を再計算する
2.  d ← v∈V_disp に対して，d(v_center, v)の最大値
3.  if (d > d(v_center, v_center) + d_max_disp)
4.      while (d > d(v_center, v_center) + d_max_disp)
5.          delete_nodes_by_dist(d)
6.          d -= 1
7.      unroll_nodes()
8.  else if (d < d(v_center, v_center) + d_max_disp)
9.      while (d > d(v_center, v_center) + d_max_disp)
10.         add_node_by_dist(d+1)
11.     rollup_nodes()
```

いて，中心からの距離によるノードの結合によりノード数を調整する必要がある．この時，次に結合される距離の節ノードを分割すると，直後に再結合してしまうことになるため, 結合対象から除外する必要がある．1-8 行では，この点に留意し，ノードの分割を行う．表示ノード数が最大である場合は，ノード数を 1 つ減らすための結合処理の対象となるノードを除外して分割対象を決定する（1-5 行）．表示ノード数が最大値に満たない（$|V_{disp}|$ < max_display_nodes）場合で，ノード数調整のための分割を考慮する必要がなく，表示ノードリスト中の結合レベルが最大の節ノードが分割対象となる（6-7 行）．第 9 行では，新たな表示レベル display_level を設定する．10-13 行は，表示レベルに対する中心ノード v'_{center} を分割した場合に，表示対象とする距離の範囲を調整する処理である．第 14 行では，rollup_nodes を実行し, 表示ノード数が超過している場合の調整処理を行う．

3.4　中心ノードの変更

中心ノードの変更には，新たな中心ノードに対する全ノードの中心からの距離を再設定し，これに合わせて表示対象とする距離の範囲を調整し，グラフ状態の変更が必要となる．新たな中心ノードとして，表示ノードリストのうち, 中心からの距離が d_{rollup} 未満の表示レベルノードを指定する場合は，グラフ状態を若干変更するだけで実現できるのに対し，それ以外のノードを指定する場合は，可視化対象の領域が大きく変化するので初期グラフの再生成が必要となる．そのため，ここでは前者の場合のみを対象とする．

中心ノードの変更（change_center_node）を説明する．第 1 行で, 新たな中心ノード v_{center} からの距離を再計算し，2-5 行で画面に表示する距離の最大値を，現在の表示ノードリストと表示最大距離から設定する．以降の処理は，表示範囲の調整である．7-16 行は表示範囲が超過している場合で，超過している距離のノードを削除し（8-11 行），削除処理によって減少したノード数を調整する（12-16 行）．残る 17-27 行，は表示範囲が不足している場合で，不足している距離のノードを追加し（18-21 行），追加処理によって増加したノード数を調整する（22-26 行）．

3.5　アルゴリズムの計算量

本方式の実現にあたって最も重要な計算量は，表示レベルの増減と中心ノードの変更を行うアルゴリズムの計算量である. これらの計算量が十分に小さければ，大規模なクラスタリング結果であっても，ストレスなく閲覧操作を行うことができると考えられる．

まず，基本操作のノードの追加・削除にかかる計算量であるが, union_nodes, divide_node, add_node_by_dist, delete_nodes_by_dist の各関数ともにリンクの追加・削除に伴い E_{cand} の操作（ソート及びマージ）を行うため，$O(|V_{disp}|^2)$の計算量が必要である．これを踏まえて表示レベルを変更する 2 つの関数 increse_display_level, decrease_display_level を見ると，ノードの追加・削除以上に時間のかかる処理はない．しかし，rollup_nodes, unroll_nodes の各関数により複数回のノード結合・分割が行われるため，表示レベルの変更 1 回について計算量は $O(|V_{disp}|^2)$×分割／結合の回数となる．

ここで，表示レベルが変更され複数回のノード結合（あるいは分割）が行われた場合，画面上ではノードの結合（あるいは分割）が一つずつ順番に行われても問題がないことが重要である．ユーザは複数ノードが結合（あるいは分割）される間待つことになるが，こ

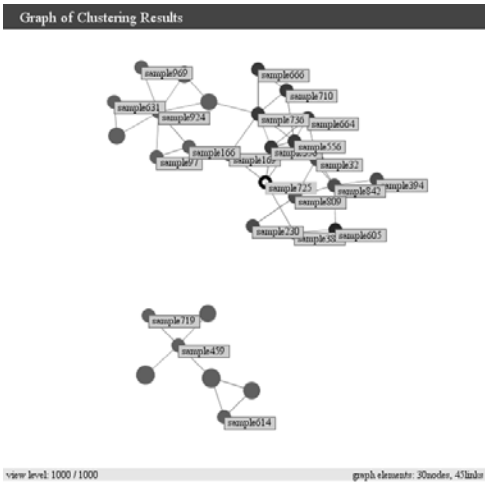

図 5　Java アプレット上での表示例

の間は画面上のグラフは変化し続けており，ノード数がよほど多くない限り，ユーザがストレスを感じることはないと考えられる．これより，現実的には計算量$O(|V_{disp}|^2)$が十分小さければ良く，実際にこれはデータの規模にかかわらず表示ノード数のみに依存するため，十分小さいと結論できる．

一方，中心ノードの変更（change_center_node）では，1 行目で中心からの距離を再計算する部分は O(|V|)の計算量がかかるが，それ以外は上記の議論と同じである．中心からの距離を計算するだけであるので，|V|がよほど大きくない限りはユーザのストレスにはならないと考えられるが，結論としてこの部分の計算量は$O(|V|) + O(|V_{disp}|^2)$×結合（または分割）の回数となる．

4. 試作と評価

4.1 Java アプレットの試作

提案手法を大規模なクラスタリング結果の可視化に適用し評価するために，本手法を実装した Java アプレットを試作した．グラフ描画ライブラリとしては，Processing[5]および TRAER.PHYSICS[6]を用いた．本プログラムでは，クラスタリング結果および類似度行列のデータを，あらかじめ生成してファイルに保存したものを，起動時に読み込んで可視化を行う．データの読み出しにかかる処理時間を省くために，起動中はこれらのデータはすべてメモリ上に保持するものとした．

本プログラムでは，パラメータの変更による可視化領域の変更に対し，グラフ状態を連続的に変化させる．例えば，表示レベルを減らす処理の場合，「親の結合レベルが最大である 2 ノードと関連するリンクを削除」→「削除した 2 ノードの親ノードと関連するリンクを追加」→「中心からの距離が最小の非表示レベルノードと関連するリンクの削除」→「削除したノードの子ノード 2 つと関連するリンクの追加」というように，グラフ状態を段階的に変化させる．これにより，変化するノードとその周辺ノードの関係を確認しながら，可視化領域をインタラクティブに操作できる．

アプレット画面の例を図 5 に示す．画面は，グラフの描画領域と最下部のグラフ情報の表示領域に分かれる．グラフ情報としては，表示レベル（現在の表示レベル／表示レベルの最大値）と，グラフに描画しているノード数とリンク数を表示する．

グラフについて説明する．各ノードは，中心からの距離を 0 から最大値までを黒色から白色に対応させ，中心ノード（または中心からの距離が最小のノード）を白抜きした環状の円で表示する．また，節ノードの場合は，その節ノード中に含まれる葉ノード数が一目でわかるよう，表示する円の大きさに変化をもたせることで，中心からの距離ごとのデータの分布傾向なども，簡単に確認できる．同様に，各リンクは，類似度の大小を，線色の濃淡（白色～黒色）と線の太さに対応させて表示する．

4.2 評価と考察

本手法の評価するために，データ数 1 万のクラスタリング結果データを試作アプレットに適用し，グラフによる可視化を行った．これらのデータは，1 データにつき 10 個のランダムな値を生成し，これらのランダム値の相関係数の絶対値をデータ間の類似度とした．その結果，数百程度のデータを扱うのと同程度のストレスを感じない処理速度で，グラフを操作できた．しかし，データ規模が大きくなるほど，限定した領域をグラフによって可視化しただけでは，全体像が掴み難いという問題がある．これを解決するためには，クラスタリング結果である樹状図の全貌と，現在グラフで可視化している領域を矩形で強調するなどの工夫が考えられる．

5. おわりに

本研究では，大規模なクラスタリング結果を，グラフを用いて効果的に可視化するための手法を提案した．本手法では，中心ノードや表示レベル，中心からの距離などに基づいて可視化領域を制御することで表示するノード数を制御し，大規模なデータであっても一定の見易さを保ったグラフとして可視化した．また，可視化領域を自由に移動させるようなインタラクティブな動作を実現するために，扱うデータをグローバルデータと画面依存データの 2 つに切り分けることで，クラスタリング結果データの規模に関わらず，一定の計算量でグラフの状態遷移が可能となった．本手法を Java アプレットとして実装し，データ数 1 万のクラスタリング結果に適用したところ，滑らかなグラフの状態遷移が可能であることが確認できた．

参考文献

[1] G. W. Furnas. “Generalized fisheye views,” Proc. ACM SIGCHI '86 Conference on Human Factors in Computing Systems, pp.16-32, 1986

[2] David Auber, Yves Chiricota, Fabien Jourdan, Guy Melancon, “Multiscale Visualization of Small World Networks,” infovis, p.10, IEEE Symposium on Information Visualization, 2003.

[3] 神嶌敏弘: “データマイニング分野のクラスタリング手法 (1),” 人工知能学会誌, vol.18, no.1, pp.59-65, 2003.

[4] M. E. J. Newman: "Fast algorithm for detecting community structure in networks," Phys. Rev. E vol.69, 066133, 2004.

[5] Processing. http://processing.org/

[6] TRAER.PHYSICS. http://www.cs.princeton.edu/~traer/physics/