データ宝石箱：大規模階層型データの グラフィックスショーケース

伊藤 貴之　梶永 泰正　池端 裕子

日本アイ・ビー・エム(株) 東京基礎研究所　E-mail: itot@computer.org

例えば、計算機のファイルシステム。例えば、大会社の人事組織。例えば、Yahoo に代表されるカテゴリ分類型のウェブサイト。身のまわりには、階層構造で整理されたデータは非常に多く存在する。これらの階層型データを対話的に表現する GUI の大半は、まず上位階層を表示し、ユーザーの選択操作によって徐々に局所的に下位階層を表示させるように構築されている。これらの GUI とは逆に、最初から階層型データ全体を一画面に展開して表示することで、データの分布を一目で理解するような視覚化手法はないものか、ちょうど宝石店のショーケースのようにデータ全体を見渡せる視覚化手法はないものか？ という発想が本報告の動機である。

本報告で提案する視覚化手法は、まず最下位階層を構成するデータのアイコンを長方形で囲み、その長方形の集合を囲む長方形を作成して上位階層を表現し … という処理を最上位階層まで反復することで、データを画面空間に配置する。画面空間を有効活用するために、本手法では長方形をできるだけ隙間なく配置して占有空間の最小化を図る。画面空間中の隙間を高速に探し出すために、本手法では長方形の中心点群を連結する三角メッシュを生成し、面積の大きい三角形要素の内部に長方形を配置し、その長方形の中心点を追加することで三角メッシュを更新する … という処理を反復している。

Data Jewel-Box: A Graphics Showcase for Large-scale Hierarchical Data Visualization

Takayuki ITOH　Yasumasa KAJINAGA　Yuko IKEHATA

IBM Research, Tokyo Research Laboratory

File systems of computers, company organizations, and category-based search websites such as Yahoo. We see many kinds of large-scale hierarchical data in our daily life. Most of GUI for these data first represent the higher-level of the data, and provide the interface so that users can select their interested portion of the data and locally explore the lower-levels. On the other hand, there are not so many visualization techniques that give the overview of the data by locating whole lower-level data onto a display space. How we can realize such technique that distributes the data like a showcase of jewel shops? … That is the motivation of this paper.

The paper presents a visualization technique that locates whole lower-level portion of hierarchical data on a display space. It first generates rectangles that enclose icons of the lowest-level data. It then repeats the process of generating rectangles that enclose the lowest-level rectangles, until the highest-level rectangles are enclosed by the largest rectangle. To minimize the occupation of display spaces, our technique efficiently searches for gaps that rectangles can be located without overlaps with adjacent rectangles. We use Delaunay triangular meshes that connect centers of rectangles to quickly search the gaps.

1. はじめに

本報告は、大規模階層型データの新しい視覚化手法を提案する。本手法では、階層型データを構成する下位階層データをすべて画面空間に配置することで、データの全貌を一画面に表示する。

図 1　本手法によるデータ表示例。

図 1 に、本手法によるデータ表示例を示す。本手法では、階層型データの下位階層に分類されたデータを、色のついたアイコンで示す。また、そのアイコンの集合を囲む長方形で 1 個の階層を示す。長方形を入れ子構造にすることで、階層の深さを表現する。宝石店のショーケースが、宝石やアクセサリーを種類ごとに分類して、商品を一望できるように陳列しているのと同じように、本手法は階層で分類されたデータを画面上で一望できるような視覚化を実現している。

本報告が対象としている階層型データは、身の周りに非常に多く存在する。例えば計算機のファイルシステム、会社の人事構造、金融データや商品データ、Yahoo に代表されるカテゴリ分類型のウェブサイト、並列計算機のプロセス管理 … 本手法は、これら多種類の大規模階層型データに対する傾向分析、検索、監視、などの目的で有用であると考えられる。

本報告が対象としているデータは、一般的には座標情報を持たない。よって図 1 のようなデータ視覚化を実現するためには、データに画面空間上の座標値を与え、データを画面空間に配置する技術が必要である。本手法においては、階層を表現する長方形を画面空間に配置する技術、それも長方形をできるだけ隙間なく配置することで、画面空間上の占有面積を小さくするような配置手法が必要である。

占有面積の最小化の問題は古くから、VLSI 回路の基板配置、板金や服飾型紙への部品配置などの用途で知られている [Mur96]。これらの用途では、遺伝子アルゴリズムなどの最適化手法を用いて部品の配置を実現している例が多い。しかし、数分～数時間の計算時間を要することが多く、対話的操作を要する視覚化の分野には向かない。本報告においては、最適解でなくてもいいから、ある程度良好な配置結果を、短時間に算出する手法が必要となる。

本報告では、まず長方形群を短時間に隙間なく配置する新しいアルゴリズムを提案する。本手法では、長方形の中心点を連結する三角メッシュを生成する。その三角メッシュを構成する三角形要素のうち、大きい要素の周囲には隙間がある可能性が高い、という直感に基づいて、大きい要素の周辺に隙間を探して他の長方形を配置する。

続いて本報告では、上記のアルゴリズムを用いたデータ配置結果を示し、従来手法と比較する。

2. Delaunay 三角メッシュを用いた長方形群の配置アルゴリズムの提案

2.1 概要

n 個の長方形群を与えられたとき、これをできるだけ小さい占有面積で xy 平面上に配置することを考える。ただし、長方形の辺はすべて x 軸または y 軸に平行であるとする。このとき本手法では、以下の手順で長方形を配置する。また、図 2 に長方形群の配置例を示す。

1．　長方形を面積順にソートする。
2．　大きい長方形から順に、
(2a) 次に配置する長方形を、すでに配置されている隣接長方形と干渉せず、しかも占有領域を拡大せずに配置できるような隙間を探す。
(2b) 隙間が見つかったら、その位置に長方形を配置する
(2c) 隙間が見つからなかったら、占有面積を拡大してもいいから、隣接長方形と干渉しない位置に長方形を配置する。

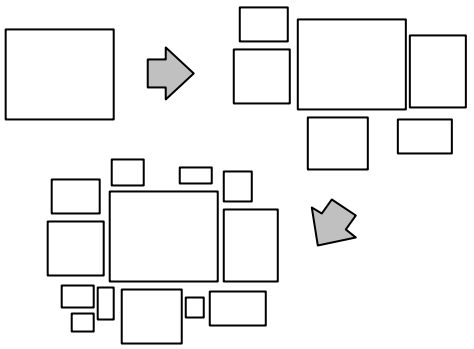

図 2　本手法における長方形群の配置例。まず面積が最大である長方形を配置し、続いて隙間を探しながら、大きい順に長方形を互いに隣接するように配置する。

本報告は対話的なデータ視覚化を目的としているため、最適な配置結果を求めるというより、短い計算時間（例えば、100 個の長方形を PC で 1 秒以内）で配置できる手法を用いることを重視する。そこで本報告では、長方形を配置

するための隙間を高速検出するために、長方形群の中心点を連結する三角メッシュを生成し、三角メッシュを参照しながら隙間を検出する手法を提案する。

ここで、本手法で生成する三角メッシュについて説明する。本手法ではまず、面積が最大である長方形を画面空間の原点に配置する。続いて、その長方形よりやや大きく、長方形を完全に包括する長方形領域（以下、占有領域と呼ぶ）を生成し、その 4 頂点と長方形の中心点を連結する三角メッシュを生成する（図 3(左)参照)。以後、長方形の配置にあわせて、三角メッシュも更新する。大きい順に k 番目までの長方形を配置したときに、三角メッシュは k 個の長方形の中心点、およびそれを囲む長方形領域の 4 頂点、の合計 $(k+4)$ 個の頂点によって構築される(図 3(右)参照)。

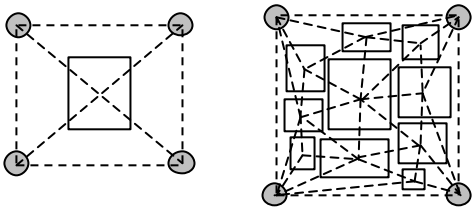

図 3（左）三角メッシュ(点線)の初期状態。面積が最大である長方形だけが配置された状態である。(右）すでに配置された長方形の中心点群、占有領域の 4 頂点の中心点、を連結する三角メッシュ。

2.2 長方形を配置する隙間の検出方法

本手法では、長方形を適切に配置できる隙間を検出するために、三角メッシュを構成する三角形要素の中から、以下の手順に従って三角形要素を抽出する。

[手順 1] まず大きい三角形要素を抽出し、その内部に長方形を配置することを試みる。なぜなら、大きい三角形要素ほど、その内部に隙間がある可能性が高いからである。
[手順 2] まず占有領域の内側にある三角形を抽出し、その内部に長方形を配置することを試みる。なぜなら、占有領域の内側にある三角形要素ほど、その内部に配置した長方形が占有領域からはみ出して、占有領域を拡大する可能性が低いからである。

上記の手順にそって、1 個の長方形に対して、それを内部に配置できる三角形要素を高速に検出する方法について述べる。本手法ではまず、各々の三角形要素の 3 頂点について、占有領域の頂点と重複する頂点の数 n_B を特定する。続いて、三角形要素を n_B 値で 0～3 の 4 種類に分類する。図 4(左)に n_B 値の例を示す。この例からもわかる通り、n_B 値が小さい三角形要素ほど占有領域の内側にあることがわかる。

続いて、$n_B = 0$ である三角形要素を大きい順に探索する。筆者らは大きさの基準に「内接円の半径」を用いた。もし探索した三角形要素が以下の 2 条件を満たす場合には、その三角形要素の内部に長方形を配置する。

[条件 1] 長方形を、隣接長方形と干渉せずに配置できる。
[条件 2] 長方形を、占有領域をはみ出すことなく配置できる。

三角形要素が上記の 2 条件の両方を満たさない場合は、他の三角形要素に対して同様な判定処理を行う。また、[条件 1]は満たすものの[条件 2]は満たさない、という三角形要素が初めて見つかったときに、その三角形要素を記録しておく。

もし $n_B = 0$ である三角形要素の中に、2 条件の両方を満たす三角形要素が存在しないようであれば、$n_B = 1$、$n_B = 2$、$n_B = 3$ の順に三角形要素を探索し、同様な判定処理を行う。

もし 2 条件の両方を満たす三角形要素がまったく存在しないなら、前述の処理で記録された三角形要素内部に長方形を配置する。このとき、配置した長方形は占有領域からはみ出しているので、占有領域の 4 頂点を移動して、すべての長方形を内包する占有領域を再構成する(図 4(右)参照)。

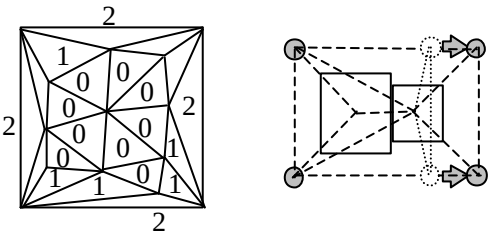

図 4（左）三角形要素の占有領域 4 頂点との隣接数。（右）長方形の 1 個が占有領域をはみ出したため、占有領域 4 頂点のうち 2 頂点を矢印の向きに従って移動した状態。

2.3 長方形の配置座標値の算出

三角形要素内部に長方形を配置するに際して、本報告では以下の 2 つの仮定にしたがって配置座標値を算出する。

[仮定 1] 三角形要素の辺の近くよりも重心近くに配置するほうが、配置する長方形が隣接長方形と干渉する可能性が低い。
[仮定 2] 三角形要素の 3 頂点のいずれかが、すでに配置された長方形の中心点である場合には、その長方形に接するように長方形を配置することで、

不必要な隙間を生じることを避けられる。

以上の仮定にしたがって本手法では、まず三角形要素の 3 頂点と重心を結ぶ線分を引き、その線分上で 3 頂点上のいずれかの長方形に隣接する位置に長方形を配置する（図 5 参照）。配置した時点で、この長方形と隣接長方形との干渉判定を行う。

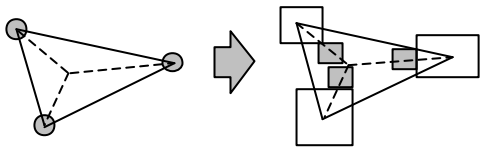

図 5　三角形要素内部への長方形の配置。本手法では、三角形要素の 3 頂点と重心を結ぶ線分上で、かつ既に配置されている長方形に隣接するような位置に、新しい長方形を配置する。この例では、塗りつぶした 3 個の長方形のいずれかの位置に配置する。

2.4 長方形の配置にともなう三角メッシュの局所修正

本手法では、2.3 節に示した方法で *(k+1)* 番目の長方形の配置座標値を算出した後に、その長方形の中心点を三角メッシュに追加して、三角メッシュを更新する。この処理ではまず、長方形の中心点を内包する三角形要素の 3 頂点と、いま配置した長方形の中心点を連結する（図 6(左)参照）。続いて、三角メッシュが Delaunay の条件を満たし続けるように、三角形要素の辺を局所修正する（図 6(右)参照）。なお三角メッシュにおける Delaunay の条件とは、「任意の三角形要素に対して生成された外接円の内部に、その三角形要素の 3 頂点以外の頂点を包括しない」という条件を、全ての三角形要素が満たすことである。

以上の処理は、Incremental Delaunay 三角メッシュ生成法 [Slo87] という手法に類似している。

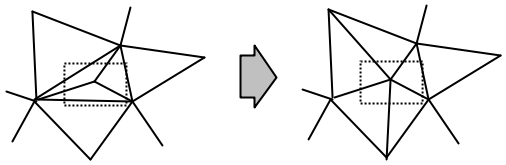

図 6(左)長方形の中心点を三角形要素の 3 頂点に連結した状態。(右) 三角形要素の辺を局所修正して、三角メッシュを更新した状態。

2.5 アルゴリズム

本章のまとめとして、Delaunay 三角メッシュを用いた長方形群の配置アルゴリズムの擬似コードを図 11 に示す。

3. 長方形群の配置手法を用いた大規模階層型データの視覚化の提案

前章で示した長方形群の配置手法を用いて、大規模階層型データ全体を画面空間に配置するアルゴリズムを示す。

図 7 に、本手法による階層型データの画面配置順の例を示す。本手法では、まず最下位階層を構成するデータをアイコン（図 7 の場合は正方形）で表現したものを隙間無く配置する。続いて、これを包括する長方形を生成する。続いて、上位階層を構成する長方形群を隙間無く配置し、同様にこれを包括する長方形を生成する。以上の処理を、最下位階層から最上位階層に向けて反復することで、データ全体の配置を決定する。

本手法における階層の配置順を、木構造で図示したものが図 8 である。配置順を決定するために本手法では、最上位階層から幅優先アルゴリズムで下位階層を探索し、探索順を記録する。すべての階層を探索し終えたら、探索順と逆の順序で長方形の配置処理を行う。

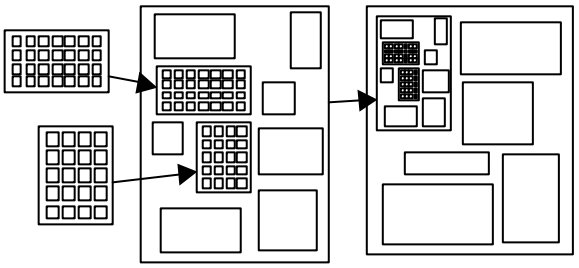

図 7　階層型データの画面配置順。まず最下位階層のデータを配置し、続いて上位階層に向かって配置処理を反復する。

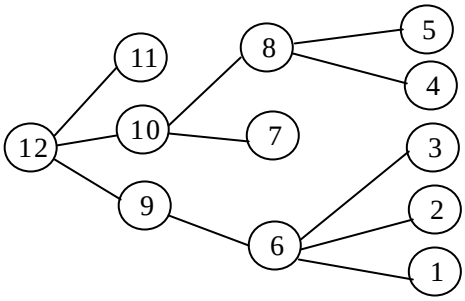

図 8　階層の幅優先探索を用いたデータの配置順。この木表現のノード 1 個が、表示画面上の長方形 1 個に相当する。

4. 実行例

本手法を実装した例を示す。筆者らの実装は、データ配置部分に Visual C++ を用い、GUI に Java Swing を用いている。筆者らの実行環境は Windows 2000 および IBM IntelliStation Z-Pro (Pentium III 933MHz)である。

図 9 は、あるウェブサイトを構成するウェブページ群、およびそこから直接リンクされているウェブページ群を視覚化したものである。この例では、ディレクトリ階層を用いてウェブページを分類している。また、正方形のアイコンが個々のウェブページを表しており、その色がウェブページの属性値（この例では更新日時）を表している。データの大きさはアイコン 835 個、長方形 160 個であり、160 回の配置処理の合計計算時間は 0.3 秒であった。

筆者らの実装では、データを表すアイコンに高さを与えることができる。図 10 は、図 9 を回転することで、アイコンの高さを示したものである。アイコンの高さが別の属性値（この例ではウェブページのアクセス数）を表している。

図 9,10 に示した例とは別に、さらに大規模な階層型データで本手法を実行した例があるが、大きな紙面を要するので本報告では省略し、代わりに以下のウェブサイトで公開する。
http://www.trl.ibm.com/projects/webvis/

図 9 本手法を用いたデータ視覚化の例(1)。

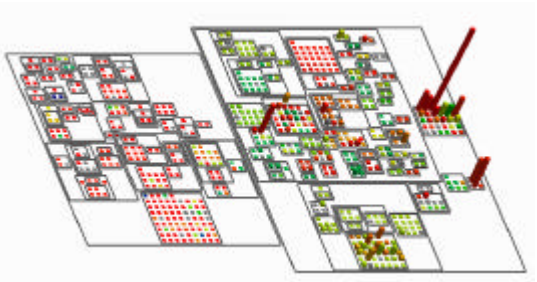

図 10 本手法を用いたデータ視覚化の例(2)。図 9 を回転したものである。

以上の例からわかるように本手法は、下位階層のデータの全貌を一望する目的、および下位階層のデータの属性値を多次元的に表現する目的に向いている。ウェブページ以外の題材では、例として以下のようなデータの視覚化を想定できる。

- 人事組織データの、組織ではなく人に注目した視覚化。
- 金融データの、支店単位や業界単位ではなく個々の顧客に注目した視覚化。
- 並列計算機プロセスデータの、CPU 単位ではなく個々のプロセスに注目した視覚化。

5. 従来手法との比較

前章までに示したアルゴリズムおよび実行例から、本手法の特徴を以下の通り列挙する。

- 階層型データの下位階層データの全貌を一望する目的において有利である。（逆にいえば、まず先に上位階層を見たい、というような目的には必ずしも向いているとは限らない。）
- データは平面上に配置されているので、3 次元 CG に慣れ親しんでない人にも操作が容易である。また統計結果の印刷配布などの目的にも向いている。
- アイコンの面積、高さ、色などから、下位階層データのもつ統計値を多次元的に表示する目的に向いている。
- 下位階層データどうしがリンク構造をもつときに、グラフ視覚化手法 [Doi01] との連携によってそれを表現することが可能である。
- 筆者らの実行例では数秒程度の計算時間がかかっているが、この計算時間はデータ読み込みの時間より短いくらいなので、本質的な問題と考える必要は無い。

これらの観点を前提にして、従来のいくつかの階層型データ視覚化手法と本手法を比較する。

[木構造表示型の視覚化手法] Windows Explorer などの普及ソフトに代表されるように、階層型データの視覚化手法で最もポピュラーな手法は、木構造を表示するものである。大規模階層型データの対話的な探索に向いた手法として、Hyperbolic Tree [Lam96], Cone Tree [Car95], Fractal Views [Koi95] などの各手法が提案されている。これらの手法は、上位階層から順に下位階層に辿っていくように表示する目的には向いているが、下位階層のデータ量が大きいときにその全データを表示するような目的には向いていないので、本手法とは性格が異なるものである。

[空間分割型の視覚化手法] 帯グラフを入れ子状にして、階層型データを構成するシェアを表示する TreeMaps [Joh91] が有名である。この手法では、下位階層データの形状表現に自由度がないので、アイコンなどを使って下位階層を表現することができない。

[3 次元の入れ子構造を用いた表現] 半透明な直方体を入れ子状に配置することで 3 次元的に

階層構造を表現する Information Cube [Rek93] が知られている。半透明表示のできるグラフィックス環境が必要であること、ユーザー側に 3 次元 CG の操作スキルを要すること、階層構造が深いときに透明度の調節が難しいなどの制限がある。また、データ配置の計算時間の測定結果などが論文中に見当たらないので、対話性に関して不明な点が残る。

6. むすび

本報告では、大規模階層型データを下位階層にいたるまで一画面で表示する視覚化手法を提案した。本手法では、一階層を構成するデータ群を隙間なく配置して、それを長方形で囲む、という処理を下位階層から上位階層に向けて反復することにより、階層型データ全体を画面空間に配置する。長方形群を干渉なく、かつ隙間なく配置できる位置を高速に検出するために、本報告では長方形の中心点群を連結する Delaunay 三角メッシュを用いるアルゴリズムを提案した。

今後の課題として、以下のようなことを検討中である。

- さらに大きなデータに対する視覚化の実験と検討。
- 時間変化を伴う階層型データのシームレスな視覚化手法の開発。
- 階層構造の構築方法の検討。
- 階層型グラフデータの視覚化手法 [Doi01] との連携。
- 本手法で用いた長方形群の配置手法に対する、計算時間や配置結果の評価と分析。
- 本手法を用いた GUI やシステムの構築。

謝辞

多くの助言と協力を下さった、日本アイ・ビー・エム(株)東京基礎研究所梶谷浩一氏、青野雅樹氏、井上恵介氏、山田敦氏、田島玲氏、土井淳氏、山口裕美氏に、感謝の意を表する。

参考文献

[Car95] Carriere J., et al., Research Report: Interacting with Huge Hierarchies Beyond Cone Trees, *IEEE Information Visualization '95*, pp. 74-81, 1995.

[Doi01] 土井，伊藤，梶永，池端，階層型グラフデータのための可視化手法，Visual Computing / 情報処理学会グラフィクスとCAD合同シンポジウム，pp. 47-50，2001.

[Joh91] Johnson B., et al., Tree-Maps: A Space Filling Approach to the Visualization of Hierarchical Information Space, *IEEE Visualization '91*, pp. 275-282, 1991.

[Koi95] Koike H., Fractal Views: A Fractal-Based Method for Controlling Information Display, *ACM Trans. on Information Systems*, 13, 3, pp. 305-323, 1995.

[Lam96] Lamping J., Rao R., The Hyperbolic Browser: A Focus+context Technique for Visualizing Large Hierarchies, *J. Visual Languages and Computing*, 7, 1, 33-55, 1996.

[Mur96] Murata H., Fujiyoshi K., Shigetoshi N., Kajitani Y., VLSI Module Placement Based on Rectangle-Packing by the Sequence-Pair, *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, 15, 12, 1518-1524, 1996.

[Rek93] The Information Cube: Using Transparency in 3D Information Visualization, Third Annual Workshop on Information Technologies & Systems, pp. 125-132, 1993.

[Slo87] Sloan S.W., A Fast Algorithm for Constructing Delaunay Triangulation in the Plane, *Advances in Engineering Software*, 9, 34-55, 1987.

```
長方形群の配置 () {
  /* 初期化 */
  長方形群を面積でソートする;
  仮の長方形占有領域をつくる;
  三角メッシュを初期化する;

  /* 長方形ごとのループ */
  面積の大きな長方形から順に {
    三角形要素を n_B = 0,1,2,3 の順に {
      未処理三角形要素のうち、内接円が
      最大であるものを選択する;
      選択した三角形要素の内部で、
       長方形をおく位置を決定する;
      隣接長方形と干渉するなら {
         戻って他の三角形要素を選択する;
      }
      占有領域の拡大が必要なら {
        この三角形要素を記録しておく;
        戻って他の三角形要素を選択する;
      }
      長方形を配置し、メッシュを更新する;
      次の長方形へ;
    }
    もし占有領域を拡大せずに長方形を配置
    できないなら {
      記録してあった三角形要素の内部に
      長方形を配置し、メッシュを更新する;
      次の長方形へ;
    }
  }
}
```

図 11 本報告で提案する長方形配置アルゴリズムの擬似コード。