平安京ビュー ～ 階層型データを碁盤状に配置する視覚化手法

伊藤 貴之○　小山田 耕二

（京都大学学術情報メディアセンター）

HeiankyoView: Orthogonal Representation of Large-scale Hierarchical Data

Takayuki ITOH, Koji KOYAMADA

ABSTRACT Visualization is very useful for various large-scale computation areas. One of the authors has reported a hierarchical data visualization technique, and applied it to various large-scale computing data including large-scale Web sites, but also for data mining results, Web search queries, network intrusion detection results, and distributed processes. This paper presents the new hierarchical data visualization technique, HeiankyoView, which can be also applied to various large-scale computing data. It places data items of input data while it packs the items as many as possible in small display areas, and aligns the items along X-axis and Y-axis of the display area. The technique represents nodes of hierarchical data as a set of rectangles, which are parallel to X-axis and Y-axis. Here the technique orthogonally divides the display area by extension lines of edges of previously placed rectangles. By referring the grid-like subspaces of the display area, the technique quickly finds the adequate positions to place remaining rectangles.

Keywords: Visualization, Hierarchical Data, Rectangle Packing, Heiankyo.

1. はじめに

階層型データの視覚化手法は、情報視覚化の研究分野の中でも、特に活発に研究されている分野のひとつである。代表的なものとして、以下のような手法がある。

- 木構造を表現した手法。Hyperbolic Tree [Lam96], Cone Tree [Car95], Fractal Views [Koi95] など。
- 画面空間の再帰分割による手法。Treemap [Joh91]など。
- 3 次元空間で入れ子状に階層構造を構築し、半透明表示する手法。Information Cube [Rek93], H-BLOB [Spr00] など。
- 2 次元空間で入れ子状に階層構造を構築する手法。「データ宝石箱」[Yam03]など。

本報告は、「できるだけ小さな画面空間に多くの情報を、碁盤状に整理された形式で表現する」という方針による階層型データの画面配置手法「平安京ビュー」を提案する。「平安京ビュー」は、著者自身による従来の視覚化手法「データ宝石箱」と類似した条件を前提にして構成されている。しかし、アルゴリズムが大きく異なるため、視覚化結果も大きく異なる。

2. 長方形の入れ子構造による階層型データの表現

「平安京ビュー」では、階層型データを構成する葉ノードをアイコンで表現し、枝ノードを入れ子状の長方形の枠で表現している。

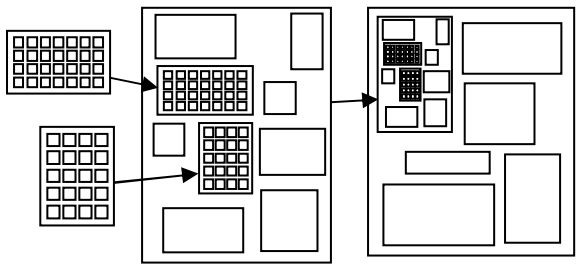

図 1　階層型データの画面配置順。まず最下位階層の葉ノードを配置し、続いて下位階層から上位階層に向かって配置処理を反復する。

図 1 に、「平安京ビュー」による階層型データの画面配置アルゴリズムを示す。本手法では、まず最下位階層に属する葉ノードに対応するアイコン（図 1 の場合は正方形）を隙間無く配置する。続いて、この上位階層に属する枝ノードを表現するために、アイコンを包括する長方形を生成する。さらに、上位階層の枝ノードを表現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。以上の処理手順は、著者の一人が過去に提案している「データ宝石箱」と同等の手順である。

3 章では、各階層を構成する長方形群を画面配置するための新しいアルゴリズムを紹介する。

3. 画面空間の格子分割による長方形の画面配置

3.1 概要

「平安京ビュー」では、まず階層型データ中の葉ノード群を正方形のアイコンで表現し、これを格子状に配列する。続いてこれらの葉ノードの親にあたる枝ノードを、葉ノード群を囲む長方形で表現する。続いて、長方形で表現された枝ノード群を画面上に効率よく配置することで、限られた画面空間上に大量の情報を表現する。

「平安京ビュー」では、枝ノードを表現する長方形群を、画面空間の横軸および縦軸に平行に、1 個ずつ順に配置するものとする。このとき提案手法は、以下の条件を満たすように長方形群を画面空間に配置する。

[条件 1] すでに配置されている長方形と干渉しない位置に長方形を配置する。
[条件 2] [条件 1]を満たす位置のうち、配置面積の拡大量が最小である位置に長方形を配置する。

3.2 節以下にて、長方形群の画面配置アルゴリズムについて述べる。また、アルゴリズムを要約した擬似コードを、図 6 に示す。

3.2 画面空間の格子分割

「平安京ビュー」では、長方形群の画面配置状況を管理するために、すでに配置された長方形の辺の延長線を用いて、画面領域を格子状に分割する（図 2 参照）。このとき、この分割処理によって作成された個々の格子領域について、すでに長方形が配置されている領域か、まだ配置されていない領域か、を判定し記録する。図 2(右)の場合、灰色に塗られた格子領域は長方形が配置された領域であり、塗られていない格子領域は長方形が配置されていない領域である。

以上の分割手法により、画面空間が x 軸方向に p 個、y 軸方向に q 個の領域に分割されたとする。このとき、画面空間を分割する線分の x 座標値を小さい順にソートしたものを $x_0 \sim x_p$、y 座標値を小さい順にソートしたものを $y_0 \sim y_q$ と表す。このとき「平安京ビュー」は、各々の長方形を配置する際に、$p \times q$ 個に分割された格子領域を順に探索し、その内部に長方形を配置できるかどうか確認する。

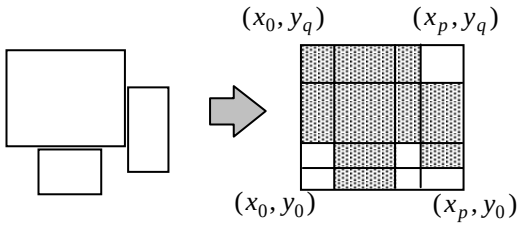

図 2. (左)すでに配置されている長方形群の例。(右)すでに配置されている長方形の辺を延長した線分で画面空間を格子状に分割した例。

3.3 画面空間の格子分割

「平安京ビュー」では、長方形の位置を決定するために、格子領域を探索しながら複数の候補位置を設定し、その中から最適な位置を選択する。

格子領域の探索順は以下の通りである。まず画面空間の中心点を含む格子領域を特定する。この格子領域が左から s 番目、上から t 番目にあるとき、本章ではこれを[s,t]と記述する。提案手法では、まず格子領域[s,t]を探索し、続いてそれを中心にして渦巻状に格子領域の探索順を定義する（図 3(左)参照）。例えば[s-1,t-1]～[s-1,t+1]を探索し、続いて[s-1,t+1]～[s+1,t+1]を探索し…という順に格子を探索する。

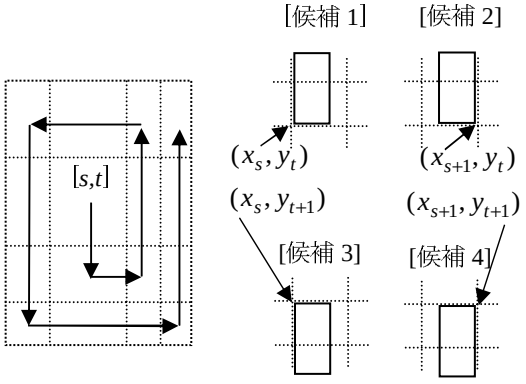

図 3. (左)格子領域の探索順。(右)格子領域内部への長方形配置の候補位置。

このような手順を用いる理由は、長方形を画面空間の外側に配置するよりも、内側に配置するほうが、配置面積の拡大量が小さくなる可能性が高いからである。つまり、画面空間の内側の格子領域から順に探索することで、[条件 1][条件 2]の両方を満たす位置を早期発見し、処理

の高速化に寄与できるからである。

続いて「平安京ビュー」では、探索された各々の格子領域について、長方形の配置を試みる。このとき、すでに配置されている長方形との干渉判定によって、[条件1]を満たすかどうか確認する。また、長方形の配置による配置面積の拡大量の算出によって、[条件 2]を満たすかどうか確認する。

「平安京ビュー」では、1 個の格子領域について、以下に示す 4 点にて長方形の配置を試みる。まず長方形の縦横の長さを(*w*,*h*)とする。また、[*s*,*t*]番目の格子領域の 4 頂点の x 座標値は x_s, x_{s+1} であり、y 座標値は y_t, y_{t+1} である。また本報告では、4 頂点の x 座標値が x_a, x_b で y 座標値が y_c, y_d である長方形の位置を、$[x_a, x_b, y_c, y_d]$ と記述する。このとき「平安京ビュー」は、長方形の 1 頂点と格子領域の 1 頂点が重なるように、以下の 4 つの候補位置に長方形の配置を試みる（図 3(右)参照）。

候補 1: $[x_s, (x_s + w), y_t, (y_t + h)]$

候補 2: $[(x_{s+1} - w), x_{s+1}, y_t, (y_t + h)]$

候補 3: $[x_s, (x_s + w), (y_{t+1} - h), y_{t+1}]$

候補 4: $[(x_{s+1} - w), x_{s+1}, (y_{t+1} - h), y_{t+1}]$

以上のように候補位置を設定することで、長方形の 1 辺が隣接長方形の辺と同一線上に位置する可能性が高くなる。これが長方形の配置結果を「碁盤状に」見せる要因である。

3.4 画面空間の格子分割

3.3 節に示した手法で導き出した候補位置の 1 つを、$[x_a, x_b, y_c, y_d]$ と表現する。このとき「平安京ビュー」では、x_a, x_b, y_c, y_d の 4 値と格子辺の座標値について、以下のような関係

$$x_i \le x_a \le x_{i+1}, x_j \le x_b \le x_{j+1}$$
$$y_k \le y_c \le y_{k+1}, y_l \le y_d \le y_{l+1}$$

が成立する整数値 i, j, k, l を特定する。続いて、格子領域 [*i*,*k*]～[*j*,*l*] の合計 (*k*-*i*+1)×(*l*-*j*+1) 個に対して、すでに配置されている長方形との干渉を判定する。まったく干渉が見られなければ、その位置は[条件 1]を満たしていると判定される（図 4(左)参照）。

この位置に配置した長方形が、画面空間をはみ出さない場合には、その位置は[条件 2]を満たしていると判定し、提案手法は格子領域の探索を中断し、その位置に長方形を配置することに決定する。画面空間をはみ出す場合には、画面空間の拡大量を算出する。このとき、

- [条件 1]のみを満たす位置が他にない場合
- 他の[条件 1]のみを満たす位置と比較して画面空間の拡大量が最小である場合

には、この位置を「仮の位置」とし、画面空間の拡大量とともに記録する。格子領域の探索処理を終了するまで「画面空間をはみ出さない位置」が全く見つからなかった場合には、この「仮の位置」に長方形を配置する。

以上の処理によって長方形の位置を決定したら、長方形の辺を延長した線分で格子領域を分割する（図 4(右)参照）。

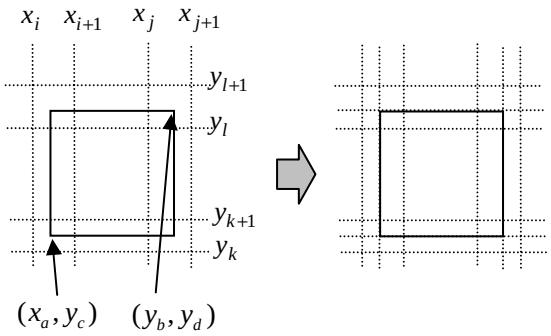

図 4. (左)干渉判定の対象となる格子領域群の特定。(右)格子領域群の分割。

4. 実行例

図 5 に、「平安京ビュー」による階層型データの表示例を示す。この表示に用いたデータは、葉ノード 1067 個、枝ノード 67 個を含む。画面配置計算に要した計算時間は約 0.1 秒であった。

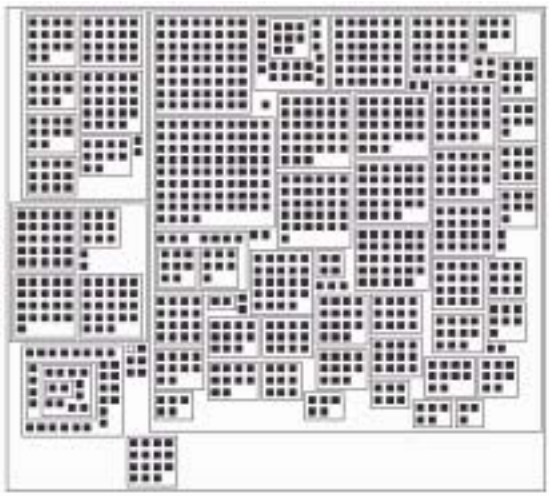

図 5. 表示例。

5. まとめ

本報告では、階層型データを構成する情報を、碁盤状に画面に配置する新しい手法「平安京ビュー」を提案し、その実行例を示した。現在著者らは、「平安京ビュー」について、表示結果の定量評価と、応用事例の開発を進めている。次報ではこれらの結果について報告したい。

謝辞

日頃貴重な議論を頂く京都大学学術情報メディアセンター関係諸氏、および日本アイ・ビー・エム（株）東京基礎研究所関係諸氏に感謝します。

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.

[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.

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

[Spr00] Sprenger T. C., et al, H-BLOB: A Hierarchical Visual Clustering Method Using Implicit Surfaces, *IEEE Visualization 2000*, pp. 61-68, 2000.

[Yam03] 山口, 伊藤, 池端, 梶永,階層型データ視覚化手法「データ宝石箱」とウェブサイトの視覚化,画像電子学会論文誌 , Vol. 32, No. 4, pp. 407-417, 2003.

0) 長方形を 1 個ずつ選択し、以下の処理を適用する。

1) 画面中心に位置する格子領域から順番に、渦巻状に 1 個ずつ格子領域を探索する。

2) 格子領域内部に 4 通りの候補位置を算出し、各々の候補位置について、(2-1)～(2-4)の処理を適用する。すべての候補位置について処理が終わったら 1) に戻って他の格子領域を探索する。

(2-1) 長方形を試しに候補位置に配置し、すでに配置されている長方形との干渉を判定する。
(2-2) 長方形の干渉があるならば、2)に戻って他の候補位置を算出する。
(2-3) 長方形の干渉がなく、しかもその配置によって配置面積の拡大を生じないならば、格子領域の探索を中断して、配置を決定し、3)に進む。
(2-4) 長方形の干渉がなく、配置によって生じる配置面積の拡大が過去最小ならば、その位置を「仮の位置」として、拡大量とともに記録する。

3) (2-3)から進んできたときはその位置に、さもなければ「仮の位置」に、長方形を配置し、格子分割を更新する。0) に戻って、次の長方形について処理を進める。

図 6. 提案手法の擬似コード。