長方形の入れ子構造による階層型データ視覚化手法の計算時間および画面占有面積の改善

伊藤貴之，山口裕美，小山田耕二

An Improvement of Nested-Rectangle-Based Hierarchical Data Visualization Technique

Takayuki ITOH, Yumi YAMAGUCHI, and Koji KOYAMADA

ABSTRACT

Hierarchical data visualization is very useful for understanding, monitoring, searching, and analyzing various large-scale data. Authors are focusing on a hierarchical data visualization technique, which represents leaf-nodes as icons, and non-leaf-nodes as nested rectangles. Key technology of the visualization technique is the algorithm for packing rectangles onto limited display spaces.

This paper proposes an improved method for packing rectangles for hierarchical data visualization. The method 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. This paper provides numerical evaluations of results of the method, and proofs the advantages of the method against existing hierarchical data visualization techniques.

Keywords: Visualization, Hierarchical data, Rectangle packing

1. はじめに

階層型データを理解するための視覚化手法は、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。階層型データ視覚化手法の代表的なものとして、以下のような手法が知られている。

- 画面空間の再帰分割による手法 1)2)4)8)13)。
- 木構造を表現する手法 3)9)10)。
- 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 12)14)。
- 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」7)15)16)17)など。

本論文の提案手法は、上記に紹介した手法のうち「データ宝石箱」の改良手法に相当する。提案手法による視覚化の例を、Fig.1 に示す。

「データ宝石箱」は、階層型データの葉ノードを長方形のアイコンで、枝ノードを長方形の枠で表現し、階層構造を 2 次元の長方形群の入れ子構造で表現する手法である。この手法は、階層型データ中の葉ノードと枝ノードの親子関係よりも、階層型データ全体に分布する葉ノード群を全て一画面に表現することに主眼をおいた視覚化手法である。この手法はすでに、ウェブサイトを構成する多数のウェブページのアクセス分布 15)や、分散計算環境に散在する多数のジョブの負荷分布 17)の視覚化に適用されており、階層型データの全体像を概観するだけでなく、例えば数千ものウェブページ群やジョブ群の中からの局所的特徴発見や局所的詳細表示などの目的に有用であることを実証している。

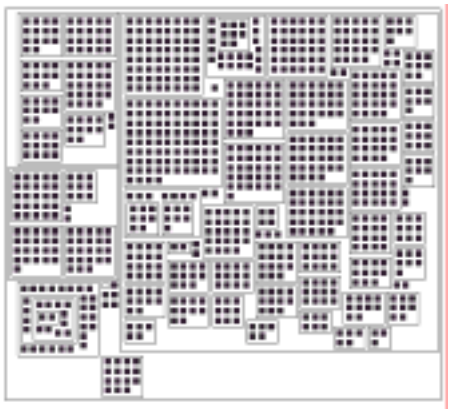

Figure 1. Example of hierarchical data visualization by the proposed technique.

「データ宝石箱」で技術的に重要な点は、枝ノード群を表現する任意の大きさ・形状の長方形群を画面空間に有効に配置できるという点である。前述したウェブサイトや分散計算環境などの実用目的において、「データ宝石箱」は以下のような前提のもとで階層型データを視覚化する手法である。

[前提 1] 葉ノードや枝ノードを干渉させないように配置する。この要求は、階層型データを構成するデータ要素（例えばウェブページや計算機）を、画面上でクリック可能な状態で提示したいときに重要である。

[前提 2] 葉ノードを同一形状で統一的に表現できる。あるいは各々の葉ノードを、あらかじめ指定された任意形状で表現できる。この要求は、階層型データを構成するデータ要素をすべて平等に表示したいとき、あるいは意図的に特定のデータ要素を拡大（または縮小）表示したいとき、に重要である。

[前提 3] 不均一な幅や深さを有する階層型データにも適用できる。この要求は、例えばウェブサイトを構成するウェブページをディレクトリ階層で木構造化したデータなど、幅や深さに不均一性を有するデータへの適用に重要である。

以上の前提のもとで「データ宝石箱」は、以下の要求をできるだけ満たすように長方形領域を画面配置する。

[要求 1: 長方形領域の正方形化] 枝ノードを表現する長方形領域を、できるだけ正方形に近い形状で表現する。この要求は、ある枝ノードの配下にあるノード群を、視覚的に一群として認識させたいときに重要である。

[要求 2: 長方形領域の占有面積最小化] 画面空間上の占有面積を小さくするように配置する。この要求は、限られた画面空間（例えばウィンドウ空間）にできるだけ多くの情報を提示したいときに重要である。

[要求 3: 長方形領域の配置結果安定化] 葉ノードや枝ノードの画面上の理想的な位置が与えられている場合には、他の要求を満たし、かつ理想的な位置に十分近い位置に配置する。この要求は、階層型データを構成するデータ要素の追加や削除に対して変化の小さい画面配置結果を得たいとき、また意図的に特定のデータ要素を特定の位置に配置したいとき、に重要である。

[要求 4: 計算時間の短縮] この要求は、リアルタイム性を要求される用途において特に重要である。

文献 15)では、**要求 2** 以外を満たす手法として、すでに画面配置されている長方形群の中心点を連結する三角メッシュを生成し、それを参照しながら残りの長方形の候補位置をいくつか選定し、その中から最も適切な位置に残りの長方形を配置するアルゴリズムを提案している。また文献 7)16)では、その拡張手法として、各長方形の画面上の理想位置を記述した「テンプレート」を導入することで、**要求 2** も満たすアルゴリズムを提案している。

本論文は、「データ宝石箱」における長方形群の画面配置結果を改善する手法を提案する。本論文の提案手法は、「データ宝石箱」と同様に上記の要求を満たし、かつ「データ宝石箱」や、関連手法である Quantum Treemap 1)と比較して、良好な画面配置結果を得ることができる。「データ宝石箱」が採用した長方形配置手法は、すでに画面配置されている長方形の中心点を連結する三角メッシュを参照したのに対して、本論文で提案する長方形配置手法では、すでに画面配置されている長方形の辺を延長する線分群を生成し、この線分群を用いて画面空間を格子状に分割したものを参照する。提案手法では、この格子構造を参照しながら、残りの長方形の候補位置をいくつか選定し、その中から最適な位置に残りの長方形を配置する。この新しい長方形配置手法により、本論文の 5 章では、提案手法は「データ宝石箱」と比較して、いくつかの要求において改良を実現できていることを示している。

2. 関連研究

本章では、階層型データの視覚化に関する従来手法、およびその他の画面配置手法について述べる。

2.1 空間分割型の階層型データ視覚化手法

与えられた画面領域を分割して階層型データを表示する手法として、入れ子状の帯グラフで階層型データを表示する TreeMaps 8)や、階層構造にしたがって円グラフを内側から外側に積み上げる手法 4)などが挙げられる。以下、TreeMaps に関する多くの改善手法について要約する。

TreeMaps は帯グラフ状の空間分割手法であるがゆえに、下位階層に属するデータ要素が非常に細長い長方形となり、視覚的に認識できなくなるケースが多いことが問題であった。これを改善する手法として、以下のような 2 次元的な空間分割手法が提案されている。

- Squarified Treemap 2) : 各データ要素をできるだけ正方形に近い形状の長方形領域で、画面領域を分割する手法。以下 SQT と略する。
- Strip Treemap 1) : 複数の帯グラフを並べるようにして、画面領域を分割する手法。以下 STT と略する。
- Ordered Treemap 13) : 階層型データの各階層に属するデータ要素に順序が割り当てられている場合に、この順序にしたがって配置の隣接性を保持し、かつ各データ要素をできるだけ正方形に近い形状の長方形領域で 2 次元的に分割する手法。以下 OT と略する。

SQT,STT,OT はいずれも、階層型データ中の一階層配下にあるノード群を表現する空間分割の新しいアルゴリズムである。階層構造全体を視覚化するためには、SQT,STT,OT のいずれかを、階層構造の上位から下位に向かって再帰的に呼び出すようにして、各階層に割り当てられた長方形領域の分割を進めればよい。このようにして、SQT,STT,OT を再帰的に呼び出す階層型データ視覚化手法は、Clustered Treemap と総称されている。

一方、階層型データを構成する葉ノードを同一形状で表現しながら、TreeMaps と同様な空間分割型の視覚化を実現する、Quantum Treemap 1)（以下 QT と略する）という手法が提案されている。この手法は、SQT,STT,OT のいずれかを用いて画面空間を長方形分割した後に、同

一形状の葉ノードを配置しやすいように各空間の大きさを調整して、最後に各領域に葉ノードを配置する、というものである。本論文では以下、SQT,STT,OT のいずれかを用いて画面空間を長方形分割する QT を、それぞれ SQQT,STQT,OQT と称する。

QT の問題点として、不均一な深さを有する階層型データにおいて、すべての葉ノードを同一形状で表現する方法が自明ではなく、しかもその有効な手段が提示されていない、という点があげられる。つまり QT は、1 章であげた前提 3 を満たすことを保証していない、と考えられる。

また TreeMap 全体に言える課題として、時系列データに適用したときに、いくつかのノードが画面上で非常に大きく移動するために、データの微小変化に対して表示結果が非常に大きく変化してしまう、という問題点が指摘されている。

2.2 その他の階層型データ視覚化手法

階層型データの視覚化手法で最も一般的な表示方法は、木構造を表示する方法である。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 10), Cone Tree 3), Fractal Views 9)などの各手法が提案されている。

筆者らの提案手法は、長方形の入れ子によって 2 次元的に階層構造を表現しているが、これに類似した発想で、半透明な直方体の入れ子によって 3 次元的に階層構造を表現する Information Cube 12) や、半透明な等値面の入れ子によって 3 次元的に階層構造を表現する H-BLOB 14) が知られている。

2.3 階層型データ以外のデータに対する画面配置手法

階層型データ以外のデータ視覚化においても、データを構成するノード群を有効に画面配置する問題は、重要な問題として議論されている。

画面領域を有効に使うという観点で有名な手法として、Design Gallery 11) が知られている。Design Gallery では、多次元変数を与えられたノード群に対して、画面上に最も分散して配置されるように、2 次元の座標軸を自動算出する。しかしこの手法でも、画面上でのノードの重なりを避けることは難しい。

また、画面領域を有効に使うために、力学モデルを用いてノード間の距離を適正に保つ手法が、グラフデータ視覚化 5)6)などの分野で知られている。しかしこの手法は、計算時間がかかることが多い。

3. 長方形の入れ子構造による階層型データの視覚化

本章では「データ宝石箱」7)15)について述べる。「データ宝石箱」は、データを構成する葉ノードをアイコンで、枝ノードを長方形の枠で表現し、階層型データ全体を一画面に配置する。配置方法の概要を Fig.2 に示す。「データ宝石箱」ではまず、最下位階層を構成する葉ノードを格子状に配置する。これを長方形で囲むことで、一つの枝ノードを表現する。続いて、その上位階層を構成するノードを敷きつめて、それを長方形で囲んで一つの枝ノードを表現する。同様な処理を、下位階層から上位階層に向かって繰り返すことにより、階層型データ全体を画面配置する。

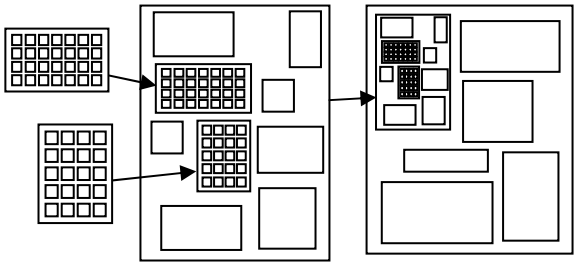

Figure 2. Processing flow of Data Jewelry Box.

ここで、葉ノードを表現するアイコンも長方形であると仮定すると、1 個の枝ノードの下位階層にある葉ノードおよび枝ノードは、長方形の集合であると考えることができる。言い換えれば「データ宝石箱」は、

各々の枝ノードについて、
その下位階層にある葉ノードおよび枝ノードを、
1 章で挙げた要求を満たすように画面配置する

という処理を最下位階層から最上位階層に向かって反復する手法であると言える。

「データ宝石箱」では、一階層を表現する長方形群を、以下の配置順により 1 個ずつ画面配置する。長方形に理想的な位置が与えられていない場合には、長方形を面積の大きい順に 1 個ずつ配置する。この配置順は、まず大きい長方形を配置し、続いてその隙間に小さい長方形を配置したほうが、占有面積が小さくなる可能性が高い、という経験則に基づいている。

長方形に理想的な位置が与えられている場合には、まず面積が最大である長方形を配置し、続いてその長方形に対して理想位置間の距離が小さい順に長方形を 1 個ずつ配置する。この配置順は、面積が最大である長方形を中心として放射状な順番で長方形を配置することで、不必要な隙間を生じる可能性を抑えられる、という経験則に基づいている。

ここで、*i-1* 個の長方形がすでに画面配置されているとする。このとき「データ宝石箱」では、1 章で挙げた要求を満たす位置を高速に算出するために、すでに配置されている *i-1* 個の長方形の中心点を連結する Delaunay 三角メッシュを生成し、これを参照しながら *i* 番目の長方形の候補位置を求める。

それに対して本論文では、すでに配置されている長方形の中心点を連結する三角メッシュを生成する代わりに、すでに画面配置されている長方形の辺を延長する線分群を生成し、この線分群を用いて画面空間を格子状に分割

したものを参照しながら残りの長方形の候補位置を求める手法を提案する。4 章では、この新しい手法の詳細について述べる。5 章では、提案手法が「データ宝石箱」に対して、計算時間、画面占有面積などの点で改良を実現したことを示す。

4. 格子分割を用いた長方形群の画面配置

本章で提案する階層型データの画面配置手法は、以下の点で「データ宝石箱」と同じ考え方を踏襲する。

- 階層型データを構成する葉ノードをアイコンで、枝ノードを長方形群で表現する。
- 最下位階層から最上位階層に向かって処理を進めることで、階層型データ全体を一画面に表現する。
- 1 章で示した要求を満たす候補位置を高速に算出し、その中から最適な位置に長方形を画面配置する。

提案手法では、各々の枝ノードの直下にあるノード群を表現する長方形群を画面空間に配置するために、画面空間を格子状に分割したデータを参照する。4.2 節に論述するとおり、提案手法は「データ宝石箱」と比較して、計算時間および画面占有面積の点で改善を図るものである。

提案手法の処理手順のうち、個々の長方形を画面配置する部分の処理手順は、以下の通りである。

1. 所定の配置順にしたがって長方形 R を 1 個選ぶ。
2. 格子分割データを参照しながら、R の候補位置を複数算出する。
3. 各々の候補位置について、以下の処理を反復する。

(a) 候補位置に R を配置して、**前提 1** を満たすか否かを判定する。

(b) **前提 1** を満たす場合には、**要求 1,2,3** の満足度の評価値を算出し、その評価値が他のどの候補位置よりも良好であれば、その候補位置を記録する。

4. 記録された候補位置に R を配置する。
5. 格子分割データを更新する。

以下、4.1 節で長方形に関する前提、4.2 節で格子構造データの定義について述べる。続いて 4.3 節で候補位置の算出方法、4.4 節で候補位置の評価値算出方法、4.5 節で長方形の配置後の処理について述べる。

なお提案手法の中で、以下の点は「データ宝石箱」と同じ方法を採用している。

- 入力された長方形の配置順。
- 入力された長方形がすべて同一形状の葉ノードである場合には、その葉ノード群を囲む長方形領域ができるだけ正方形に近くなるように、葉ノード群を格子状に配置すること。

4.1 長方形に関する前提

まず、本章で扱う長方形の定義と、提案手法の入出力仕様を規定する。提案手法では、各々の枝ノードの直下にあるノード群を表現する長方形群を配置する空間を、x 軸および y 軸を座標軸とする 2 次元直交座標系空間とする。また、与えられる長方形群の辺は、すべて x 軸または y 軸に平行であるとする。また本章では、すでに画面配置されている長方形群を包括する長方形領域を「画面領域」と呼ぶ。

ある枝ノードの直下にあるノード群を表現する各々の長方形について、

w: 水平方向の辺の長さ
h: 垂直方向の辺の長さ
x: 長方形中心点の画面上の x 座標値
y: 長方形中心点の画面上の y 座標値
u: 長方形中心点の画面上の理想位置の x 座標値
v: 長方形中心点の画面上の理想位置の y 座標値

とすると、提案手法の入出力は以下のように記述できる。

入力: 各々の長方形について、*w* および *h* が与えられる。それとは別に、*u* および *v* が長方形に与えられている場合と与えられていない場合がある。

出力: 入力した各々の長方形について、*x* および *y* が算出される。

なお、葉ノードを表現する長方形の *w* および h は、提案手法の実行前に利用者が与えるものとする。本論文の Fig1.8,9,10 に示す実行例では、全ての葉ノードについて *w* および *h* の値を *w=h=1* としており、その結果として全ての葉ノード（黒く塗られたアイコン）は同一面積の正方形として表現されている。一方、この値を葉ノードごとに変化させることで、各々の葉ノードが異なる形状・大きさの長方形として表現されるように意図的に設定することも可能である。

枝ノードを表現する長方形の *w* および *h* は、利用者によって与えられるのではなく、その直下にあるノード群の画面配置処理を終了した時点で算出される。

4.2 格子分割データの概要と定義

「データ宝石箱」は Fig.3(左)に示すように、長方形の中心点を連結する三角メッシュ辺上に、長方形の候補位置を算出する。筆者らは「データ宝石箱」による長方形配置結果を観察し、以下のような問題点を発見した。

- データによっては配置結果に乱雑さが生じる場合があること。
- 不必要な隙間を生じる場合があること。

Fig.10(上)に示す長方形配置結果にも、同様な問題点が観察される。

提案手法では、この問題点を解消する考え方として、「複数の長方形の辺が同一直線上に整列するように長方形を配置したほうが、長方形配置結果に不必要な隙間を生じる可能性を低減できる」という直感(Fig.3(右)参照)に基づいたアルゴリズムを提案する。

提案手法では、「複数の長方形の辺が同一直線上に整列するように」という目標を実現するために、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し、この分割に用いた「長方形の辺の延長線」を参照しながら長方形を配置する。例えば、Fig.4(左)のように長方形群が配置されている場合には、画面領域はFig.4(右)に示すように格子分割されているとする。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否かを判定し記録する。Fig.4(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。

このような格子構造の適用は、長方形の画面配置結果を改善するだけでなく、計算時間の改善にも寄与できると考えられる。「データ宝石箱」では、既に配置されている長方形と、これから配置する長方形の干渉判定処理が、計算時間の大きな部分を占めていた。提案手法では、この干渉判定処理を、以下のような単純な処理に置き換えられることから、計算時間を改善できると考えられる。

- 4.3 節にて説明する候補位置のうちの 1 箇所に、長方形 R を仮配置する。
- R と重なる格子領域群を検出する。
- 検出された格子領域のうち最低 1 個が、既に配置されている別の長方形に占められていれば、R は別の長方形と干渉していると判定される。

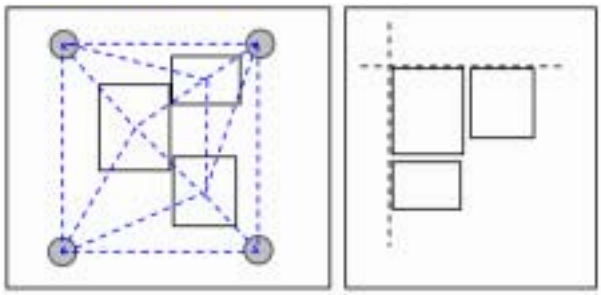

Figure 3. (Left) Rectangle layout by Data Jewelry Box. (Right) Rectangle layout by the proposed technique.

以上の格子分割により、画面領域が 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。このとき本論文では、格子分割結果を以下のように表す。

- 画面領域を分割する y 軸に平行な$(p+1)$本の線分について、x 座標値を小さい順に x_0 ～ x_p と表す。
- 画面領域を分割する x 軸に平行な$(q+1)$本の線分について、y 座標値を小さい順に y_0 ～ y_q と表す。

y 軸に平行な線分 $x=x_s$ および $x=x_{s+1}$, および x 軸に平行な線分 $y=y_t$ および $y=y_{t+1}$ の、合計 4 線分に囲まれる格子領域を$[s,t]$と表す。各々の格子領域は、どの長方形に占有されているか、あるいはいずれの長方形にも占有されていない空領域であるか、を表す数値を記録しているものとする。

4.3 長方形配置の候補位置の算出

i-1 個の長方形がすでに画面領域に配置されており、4.2 節に示した手法によって画面領域が $p \times q$ 個の格子領域に分割されているとする。このとき提案手法では、各々の格子領域内部に、i 番目の長方形の候補位置を算出する。そして各々の候補位置に長方形の配置を試み、最適と判断される候補位置に i 番目の長方形を配置する。

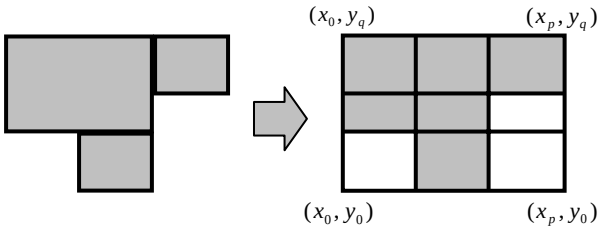

Figure 4. (Left) Already placed rectangles. (Right) Grid-like subdivision of display space by extension lines of edges of the placed rectangles.

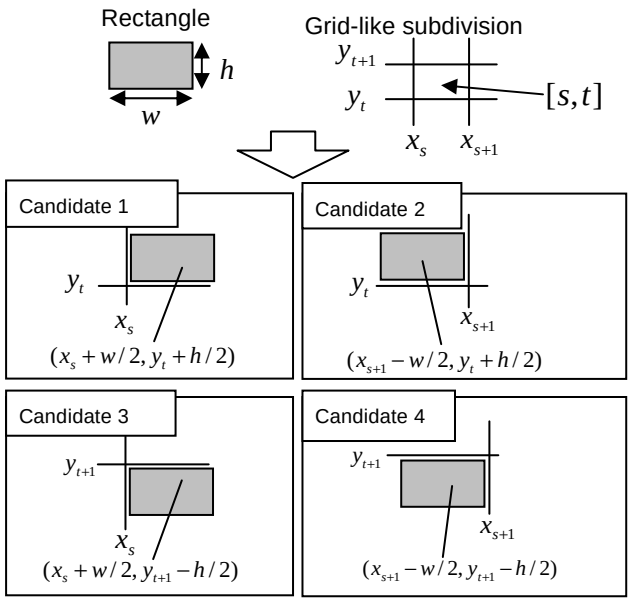

Figure 5. Four candidate positions for placing a rectangle.

ここで 4. 2 節に示した定義により、$[s,t]$番目の格子領域の 4 頂点の x 座標値を x_s および x_{s+1} と表し、y 座標値を y_t および y_{t+1} と表す。このとき提案手法は、まず $[s,t]$番目の格子領域が、すでに別の長方形に占有されているかどうかを確認する。どの長方形にも占有されていない場合には、これから配置する i 番目の長方形の 1 頂点と、格子領域 $[s,t]$ 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中心点の配置を試みる（Fig.5 参照）。

候補 1: $((x_s + w/2), (y_t + h/2))$
候補 2: $((x_{s+1} - w/2), (y_t + h/2))$
候補 3: $((x_s + w/2), (y_{t+1} - h/2))$

候補 4: $((x_{s+1} - w/2), (y_{t+1} - h/2))$

4.4 長方形配置の候補位置の評価方法

提案手法は候補位置を算出した後に、各々の候補位置を評価する。まず、すでに配置されている長方形との干渉判定によって、各々の候補位置が**前提 1** を満たすかを評価する。続いて、各々の候補位置に対する**要求 1,2,3** の満足度の評価値を算出し、評価値が最も良好であった位置に長方形を配置する。

ここで長方形の中心点の候補位置の 1 つを(x, y)と表現する。このとき提案手法では、以下のような関係

$$x_j \leqq x-w/2 \leqq x_{j+1}$$
$$x_k \leqq x+w/2 \leqq x_{k+1}$$
$$y_l \leqq y-h/2 \leqq y_{l+1}$$
$$y_m \leqq y+h/2 \leqq y_{m+1}$$

が成立する整数値 *j,k,l,m* を特定する（Fig.6(左)参照）。続いて、格子領域 $[j,l]$～$[k,m]$ の合計$(k-j+1)\times(m-l+1)$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。これらの格子領域のうち、他の長方形に占有されているものが 1 個もなければ、長方形は他の長方形と干渉しない、つまりその候補位置は**前提 1** を満たしていると判定される。さもなければ提案手法は、その候補位置への長方形の配意を断念し、他の候補位置について同様な処理を反復する。

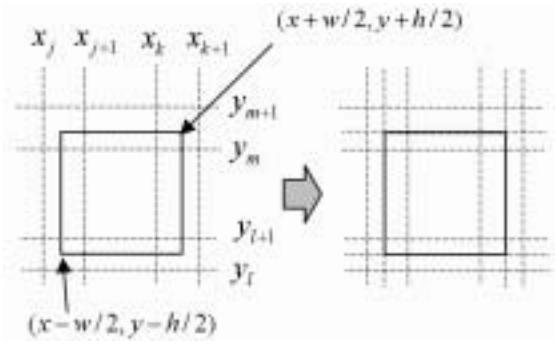

Figure 6. (Left) Integer values j,k,l,m and coordinates of vertices of a rectangle. (Right) Subdivision of grid.

続いて提案手法では、**前提 1** を満たす候補位置に対して、以下の評価式を用いて、**要求 1,2,3** の満足度を評価する。

$$aA + bS + cD + dT \ldots(1)$$

ただし、各変数は以下の意味をもつものである。

a,b,c,d: ユーザーが指定する正定数

A: 長方形の配置後に形成される画面領域のアスペクト比

S: 長方形の配置後に形成される画面領域の面積拡大量

D: 理想座標値*(u,v)*と候補位置*(x,y)*のユークリッド距離

T: 後述する候補位置の優先度から導かれる定数

提案手法では、式(1)によって算出される評価値が最小である候補位置に、長方形を配置するものとする。なお *D* 値は、理想座標値*(u,v)*が与えられていないときは、常にゼロであるとする。

T の導出方法は以下の通りである。配置した長方形の 1 頂点に重なる格子領域の 1 頂点を (x_s, y_t) とする。このとき提案手法では、候補位置の優先度を、以下の通り規定する。

優先度 1: (x_s, y_t)を共有する最大 3 個の格子領域が、既に全て他の長方形に共有されている候補位置(Fig.7(左上)参照)。

優先度 2: (x_s, y_t)を共有する最大 3 個の格子領域のうち 1 個が、既に他の長方形に共有されている候補位置(Fig.7(右上)参照)。

優先度 3: (x_s, y_t)を共有する最大 3 個の格子領域のうち 2 個が、既に他の長方形に共有されている候補位置(Fig.7(左中)参照)。

優先度 4: (x_s, y_t)を共有する最大 3 個の格子領域のうち 0 個が、既に他の長方形に共有されている候補位置(Fig.7(右中)参照)。

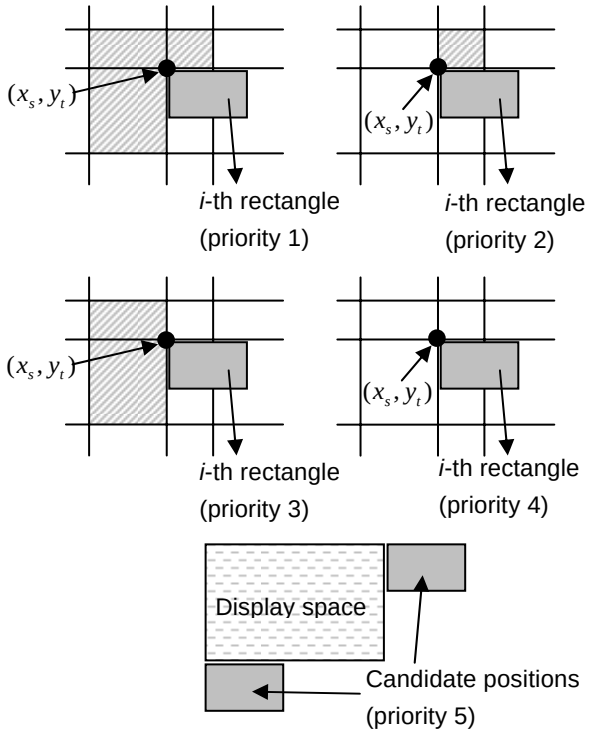

Figure 7. Five stages of priority of candidate positions.

優先度 1 の候補位置に長方形を配置することで、長方形が十字やＴ字を形成するように配置されるので、不必要な隙間を生成する可能性がさらに低くなると考えられる。優先度 2 の候補位置に長方形を配置することで、長方形が一直線上に並ぶように配置されるので、不必要な隙間を生成する可能性がさらに低くなると考えられる。

さらに提案手法では、画面領域内部に長方形を配置できる場所がまったく見つからなかった場合のために、以

下の候補位置を算出する。

優先度 5: 画面領域の外側に、画面領域に接するように長方形を配置できる位置。例えば Fig.7(下)に示す 2 箇所のような位置に候補位置を設定できる。この位置に長方形を配置した場合には、画面領域は拡大せざるを得ないので、優先度を低く設定する。

これらの 5 段階の優先度に対して提案手法では、優先度 *k (k=1..5)*に対し

$$t_1 \le t_2 \le t_3 \le t_4 \le t_5 \quad \ldots(2)$$

となる定数 t_k を、式(1)において $T=t_k$ として用いる。

4.5 長方形配置の決定とそれに伴う格子領域の更新

提案手法では前節で説明した手法により、**前提 1** を満たす候補位置について、式(1)を用いて評価値を算出し、その評価値が最小である候補位置を記録する。すべての候補位置について以上の処理を反復し、最終的に記録された候補位置に長方形を配置する。

以上の処理によって *i* 番目の長方形の位置を決定したら、*i* 番目の長方形の辺を延長した線分で格子領域を分割する（Fig.6(右)参照）。以上をもって *i* 番目の長方形の配置処理を終了し、*(i+1)*番目以降の長方形について同様の処理を反復する。

5. 実行例

提案手法を実装して実行した例を示す。筆者らは提案手法を Java JDK 1.4.2 を用いて実装し、IBM ThinkPad T42 (CPU 1.8GHz, RAM 512MB) および WindowsXP 上で実行した。(※Java およびすべての Java 関連の商標は、Sun Microsystems, Inc.の米国およびその他の国における商標。IBM および ThinkPad は、IBM Corporation の米国およびその他の国における商標。Windows は、Microsoft Corporation の米国およびその他の国における商標。) なお本実験では、式(1)(2)中の定数を、以下のように設定した。

$$a=0.4,\ b=0.1,\ c=0.4,\ d=0.1$$

$$t_1 = t_2 = 0.0,\ t_3 = t_4 = 0.5,\ t_5 = 1.0$$

本論文では提案手法の実行結果を、従来手法である QT, データ宝石箱と比較した結果を示す。なお本実験では、枝ノードおよび葉ノードに順列を仮定していないことから、OQT は採用せず、SQQT,STQT の 2 種類を実験に採用した。

QT を比較対象に選んだ理由は以下の通りである。2.1 節にて紹介した空間分割型の視覚化手法のうち TreeMaps は、長方形領域の集合で階層型データ全体を表現する、という観点で提案手法と共通している。その中でも QT は、葉ノードを同一形状で表現できるという点で、最も提案手法に近い特徴を持つ手法である。そこで本論文では、データ宝石箱だけでなく QT も比較対象に選んでいる。

本実行結果では、従来手法との比較のために、以下の(a)～(d)の数値評価基準を設けた。この数値評価基準は、文献 1)7)で用いられているものと同等である。

(a)長方形領域のアスペクト比: 要求 1 に対応する評価基準である。枝ノードを表現する長方形領域の、短辺の長さに対する長辺の長さの比を、アスペクト比として測定する。枝ノードを構成する下位階層ノードを視覚的に認識しやすくするためには、一般的にアスペクト比が小さいほどよいとされており、その理想値は 1 である。

(b)葉ノード面積総計に対する画面占有面積の比: 要求 2 に対応する評価基準である。葉ノード面積の総計に対する、最上位階層を示す長方形面積の比を算出する。この値が小さいほど、画面領域を有効活用していると評価することができる。

(c)配置結果の安定性: 要求 3 に対応する評価基準である。非常に類似する 2 個の階層型データを用意し、両者の配置結果の距離の平均値を測定する。ただし本測定では、一階層を構成する長方形領域の四隅の座標値を(-1,-1)から(1,1)の間に正規化した座標値を用いて距離を算出している。この値が小さいほど、配置結果が安定していると評価することができる。なおこの測定において、データ宝石箱および提案手法では、以下の実験方法を用いている。

- 1 個目の階層型データを画面配置する。
- 1 個目の階層型データの配置結果を理想位置として、それを入力情報として 2 個目の階層型データを画面配置する。
- 両者の配置結果の距離の平均値を測定する。

(d)計算時間: 要求 4 に対応する評価基準である。階層型データを構成する全ての葉ノードおよび枝ノードの画面空間上の位置を算出する処理時間を、計算時間として測定する。

5.1 均一な深さを有する階層型データを用いた実験

まず QT も適用可能なデータとして、均一な深さを有する階層型データを入力データとした実験結果を示す。

最初に本実験では、階層型データの最上位階層の直下に 100 個の枝ノードを自動生成し、各々の枝ノードの下に 10 から 1000 の間の乱数で決定された個数の葉ノードを自動生成する、という手順で機械的に生成した階層型データを用いた。この方法で生成した 100 個の階層型データについて、「長方形領域のアスペクト比」「葉ノード面積総計に対する画面占有面積の比」の各数値を算出した。以上の 2 種類の算出結果を度数分布表にしたものを、Tab.1,2 に示す。また、そのうちの 1 個の階層型データに関する視覚化結果のうち、SQQT および STQT を用いた視覚化結果を Fig.8 に、データ宝石箱および提案手法を用いた視覚化結果を Fig.9 に示す。なお、これらの測定実験において、理想座標値(u,v)は入力していない。

続いて上記の方法で生成した 100 個の階層型データの各々について、非常に類似する階層型データを生成した。

生成手順は以下の通りである。階層型データを構成する各階層のうちランダムに数個を選択し、そのいくつかの階層では葉ノードを 1 個追加し、残りの階層では葉ノードを 1 個削除し、という方法で微量だけ葉ノードを増減したデータを生成し、これを「非常に類似する階層型データ」とした。

続いて 100 組の階層型データの各々に対して、以下の処理を行った。まず元の階層型データを画面配置し、その葉ノード・枝ノードの画面上の位置を記録した。続いてそれらの位置を理想座標値*(u,v)*として、非常に類似する階層型データを画面配置し、両データ中の対応する葉ノード・枝ノードの画面上の距離を算出し、この平均値を「配置結果の安定性」として算出した。以上の算出結果を度数分布表にしたものを、Tab.3 に示す。

Table 1. Aspect ratois of uniformly nested hierarchical data.

アスペクト比	SQQT	STQT	データ宝石箱	提案手法
1.0〜1.1	0	0	1	1
1.1〜1.2	78	0	96	98
1.2〜1.3	22	0	3	1
1.3〜1.4	0	0	0	0
1.4〜1.5	0	1	0	0
1.5〜1.6	0	5	0	0
1.6〜1.7	0	4	0	0
1.7〜1.8	0	44	0	0
1.8〜1.9	0	24	0	0
1.9〜2.0	0	12	0	0
2.0〜	0	10	0	0

Table 2. Area ratois of uniformly nested hierarchical data.

面積比	SQQT	STQT	データ宝石箱	提案手法
1.0〜1.5	0	0	0	0
1.5〜2.0	0	0	0	0
2.0〜2.5	0	0	0	0
2.5〜3.0	39	47	0	3
3.0〜3.5	42	50	15	97
3.5〜4.0	13	2	80	0
4.0〜4.5	6	0	5	0
4.5〜5.0	0	0	0	0
5.0〜	0	1	0	0

続いて Tab.1,2 に示す実験と同様な階層型データ生成手法で、最上位階層の直下の枝ノードの個数を 100 個から 1000 個まで変化させたときの､処理時間の変化を測定した。この実験結果を Tab.4 に示す。なお、この測定実験において、理想座標値(u,v)は入力していない。

以上の結果を、1 章に示した 4 つの要求とあわせて考察する。

要求 1 については、Tab.1 に実験結果を示した「長方形領域のアスペクト比」からもわかるように、もともと「データ宝石箱」は QT よりも優位性があったが、提案手法においてもその優位性は保持されている。

要求 2 については、Tab.2 に実験結果を示した「葉ノード面積総計に対する画面占有面積総計の比」からもわかるように、提案手法が「データ宝石箱」に対して改善を実現し、QT と互角に近い結果が得られている。ただし互角に近いといっても、画面空間の空白部分の形成には大きな差がある。Fig.8,9 からも観察できるように、QT では最下位階層の下にある葉ノード間に間隔が空きやすく、提案手法では最上位階層の直下にある枝ノード間に空白が形成されやすい傾向にある。両傾向は一長一短の関係にあり、決定的な優劣はないものと考えられる。

要求 3 については、Tab.3 に実験結果を示した「配置結果の安定性」からもわかるように、もともと「データ宝石箱」は QT よりも優位性があったが、提案手法においてもその優位性は保持されている。

要求 4 については、Tab.4 からもわかるように、提案手法は「データ宝石箱」の計算時間を大幅に改善し、QT と比較しても良好な計算時間を実現していることがわかる。

Table 3. Average distances between corresponding rectangles of uniformly nested hierarchical data.

対応長方形間距離	SQQT	STQT	データ宝石箱	提案手法
0.0〜0.1	0	0	0	1
0.1〜0.2	0	0	88	98
0.2〜0.3	0	0	12	1
0.3〜0.4	0	0	0	0
0.4〜0.5	0	0	0	0
0.5〜0.6	0	1	0	0
0.6〜0.7	1	11	0	0
0.7〜0.8	90	88	0	0
0.8〜0.9	9	0	0	0
0.9〜1.0	0	0	0	0
1.0〜	0	0	0	0

Table 4. Computation time (msec)

枝ノード数	SQQT	STQT	データ宝石箱	提案手法
100	180	151	81	31
200	420	350	220	60
300	1072	610	481	80
400	1693	911	1142	130
500	2784	1262	2584	301
600	4006	1562	4667	440
700	5658	2012	7961	540
800	7319	2422	8972	711
900	9834	2944	12798	981
1000	11064	3415	18551	1131

5.2 不均一な深さを有する階層型データを用いた実験

続いて、不均一な深さを有する 4 種類の階層型データを入力データとした実行結果を示す。2.2 節でも述べたとおり、文献 1)では、不均一な深さをもつ階層型データに対して、QT を用いて統一的な大きさで葉ノードをアイコン表現する方法について論述していない。そこで本実験では、「データ宝石箱」と提案手法を実行した結果を比

較する。

本実験で用いた階層型データは、文献 7)で用いられているデータと同一のデータである。具体的には、以下の 4 種類のフリーソフトウェア

データ 1: Java Development Kit 1.3.1.04
(http://java.sun.com/j2se/)

データ 2: Apache Tomcat 4.1.12
(http://jakarta.apache.org/tomcat/)

データ 3: Apache AXIS 1.0
(http://ws.apache.org/axis/)

データ 4: IBM Web Services Toolkit 3.3
(http://www.alphaworks.ibm.com/tech/webservicestoolkit/)

について、インストール直後のディレクトリ階層を枝ノード､ファイルを葉ノードとして作成したものである。これらのデータの葉ノード数、枝ノード数、葉ノードの深さの最小値および最大値を、Tab.5 に列挙する。

Table 5. Sizes of non-uniformly nested hierarchical data.

	データ 1	データ 2	データ 3	データ 4
葉ノード	694	2742	4080	10431
枝ノード	116	395	537	1534
葉ノードの深さの最小値	1	1	1	1
葉ノードの深さの最大値	7	12	9	12

4 種類のデータのうち、「データ 1」に関する視覚化結果を、Fig. 10 に示す。4.3 節にて、「データ宝石箱」では画面配置結果に乱雑さが生じる場合があると記述したが、Fig.10(上)においても一部の葉ノードの配列に乱雑さが生じている。Fig.10(下)に示す提案手法の画面配置結果では、このような葉ノードの乱雑な配置を低減しているのが観察できる。

また 4 種類のデータに対して、実験結果から得られた(a)～(d)の数値評価結果を、Tab.6,7,8,9 に示す。この数値結果においても 5.1 節の実験結果と同様に、「データ宝石箱」と比較して提案手法が、特に画面占有面積比と計算時間を大きく改善していることがわかる。なお本実験では､Tab.8 の実験結果に限り､5.1 節と同様な方法にて「非常に類似する階層型データ」を生成したものを用いている。

画面配置アルゴリズムの改善および拡張のための課題として、以下の 2 点を列挙する。1 点目は、提案手法の実行のために与えられる画面領域（例えばウィンドウシステム中の 1 個のウィンドウ領域）が正方形でない場合の対応である。本論文の**要求 1** では、長方形による占有領域を正方形に近い領域にする、という要求を掲げてきたが、与えられる画面領域が正方形でない場合には、階層型データの最上位階層に限り、与えられる画面領域のアスペクト比を尊重した長方形配置結果が要求されると考えられる。2 点目は、全ての葉ノードが同一形状でない場合に対する実験である。これにより例えば、原則として全ての葉ノードを同一形状で表現しながら、同時に特定の葉ノードだけを拡大強調する、という視覚化が実現できると考えられる。これにより、QT に対する別の優位性が主張できると考えられる。

Table 6. Aspect ratios of non-uniformly nested hierarchical data.

アスペクト比	データ 1	データ 2	データ 3	データ 4
データ宝石箱	1.245	1.237	1.209	1.242
提案手法	1.252	1.217	1.173	1.182

Table 7. Area ratios of non-uniformly nested hierarchical data.

面積比	データ 1	データ 2	データ 3	データ 4
データ宝石箱	8.874	9.439	9.601	18.826
提案手法	5.781	7.911	6.938	8.643

Table 8. Average distance between corresponding rectangles of non-uniformly nested hierarchical data.

対応長方形間距離	データ 1	データ 2	データ 3	データ 4
データ宝石箱	0.202	0.141	0.192	0.134
提案手法	0.186	0.150	0.180	0.145

Table 9. Computation times of non-uniformly nested hierarchical data. (msec)

	データ 1	データ 2	データ 3	データ 4
データ宝石箱	160	350	290	430
提案手法	20	30	60	240

6. むすび

本論文では、長方形の入れ子構造を用いた階層型データ視覚化手法のための、長方形の画面配置に関する改善手法を提案した。この手法では、すでに画面配置されている長方形の辺を延長する線分で画面空間を格子状に分割したデータを作成する。そして、その格子分割データを参照しながら、まだ画面配置されていない長方形の候補位置を高速に算出し、その候補位置の中から長方形の配置を決定する。本論文の 5 章では、提案手法による長方形の画面配置結果を数値評価し、従来手法と比較して優位性があることを実証した。

筆者らは､提案手法をさまざまな階層型データの理解、分析、検索、監視などに実用するための研究を進めている。いまのところ、

- ネットワーク不正アクセスの監視・発見

- 原子力システムの監視・発見
- 遺伝子発現率データの分析
- 科学技術計算結果データベースの探索
- 大規模有機化合物データのマイニング
- 大規模文書データの検索
- 社会問題などを扱った統計データの分析

などへの適用を進めている。

参 考 文 献

1) Bederson B., Schneiderman B.,Ordered and Quantum Treemaps: Making Effective Use of 2D Space to Display Hierarchies, ACM Transactions on Graphics, Vol. 21, No. 4, pp. 833-854 (2002).
2) Bruls D.M., Huizing K., Wijk J. J., Squarified Treemaps, Proceedings of Data Visualization 2000, pp. 33-42 (2000).
3) Carriere J., et al., Research Paper: Interacting with Huge Hierarchies beyond Cone Trees, IEEE Information Visualization 95, pp.74-81 (1995).
4) Chuah M., Dynamic Aggregation with Circular Visual Designs, IEEE Information Visualization '98, pp.35-43 (1998).
5) 土井, 伊藤, 力学モデルを用いた階層型グラフデータ画面配置手法の改良手法とウェブサイト視覚化への応用, 芸術科学会論文誌, Vol. 3, No. 4, pp. 250-263 (2004).
6) Herman I., Melancon G., Marshall M. S., Graph Visualization and Navigation in Information Visualization: A Survey, IEEE Transactions on Visualization and Computer Graphics, Vol. 6, No. 1, pp.24-43 (2000).
7) Itoh T., Yamaguchi Y., Ikehata Y., and Kajinaga Y., Hierarchical Data Visualization Using a Fast Rectangle-Packing Algorithm, IEEE Transactions on Visualization and Computer Graphics, Vol. 10, No. 3, pp. 302-313 (2004).
8) Johnson B., et al., Tree-Maps: A Space Filling Approach to the Visualization of Hierarchical Information Space, IEEE Visualization '91, pp. 275-282 (1991).
9) Koike H., Fractal Views: A Fractal-Based Method for Controlling Information Display, ACM Transactions on Information Systems, Vol. 13, No. 3, pp.305-323 (1995).
10) Lamping J., Rao R., The Hyperbolic Browser: A Focus+context Technique for Visualizing Large Hierarchies, Journal of Visual Languages and Computing, Vol. 7, No. 1, pp. 33-55 (1996).
11) Marks J., et al., Design Galleries: A General Approach to Setting Parameters for Computer Graphics and Animation, ACM SIGGRAPH '97, pp. 389-400 (1997).
12) Rekimoto J., The Information Cube: Using Transparency in 3D Information Visualization, Third Annual Workshop on Information Technologies & Systems, pp.125-132 (1993).
13) Shneiderman B., et al., Ordered Treemap Layouts, IEEE Information Visualization Symposium 2001, pp. 73-78 (2001).
14) Sprenger T. C., et al, H-BLOB: A Hierarchical Visual Clustering Method Using Implicit Surfaces, IEEE Visualization 2000, pp. 61-68 (2000).
15) 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 Visual Computing 特集号, Vol. 32, No. 4, pp. 407-417 (2003).
16) 山口, 伊藤, 長方形の入れ子構造を用いた階層型データ視覚化手法の拡張, 情報処理学会論文誌, Vol. 44, No. 10, pp. 2469-2477 (2003).
17) Yamaguchi Y., Itoh T., Visualization of Distributed Processes Using "Data Jewelry Box" Algorithm, CG International 2003, pp. 162-169 (2003).

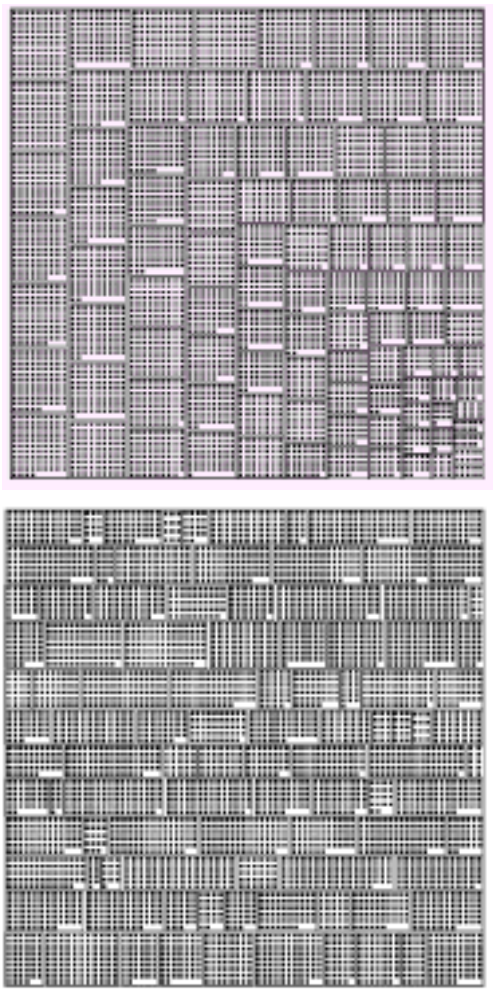

Figure 8. Example of uniformly nested hierarchical data. (Upper) Visualized by SQQT. (Lower) Visualized by STQT.

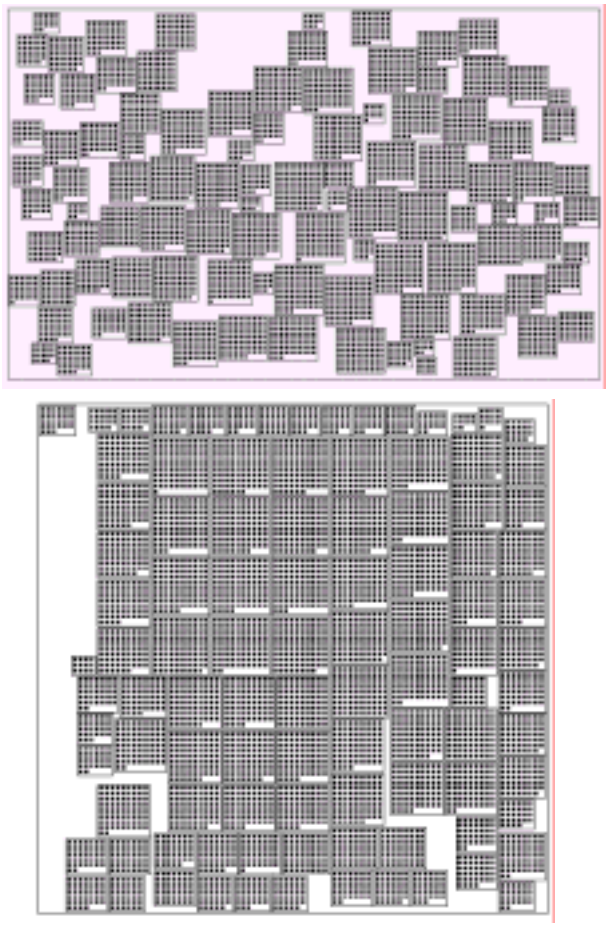

Figure 9. Example of uniformly nested hierarchical data. (Upper) Visualized by Data Jewelry Box. (Lower) visualized by the proposed technique.

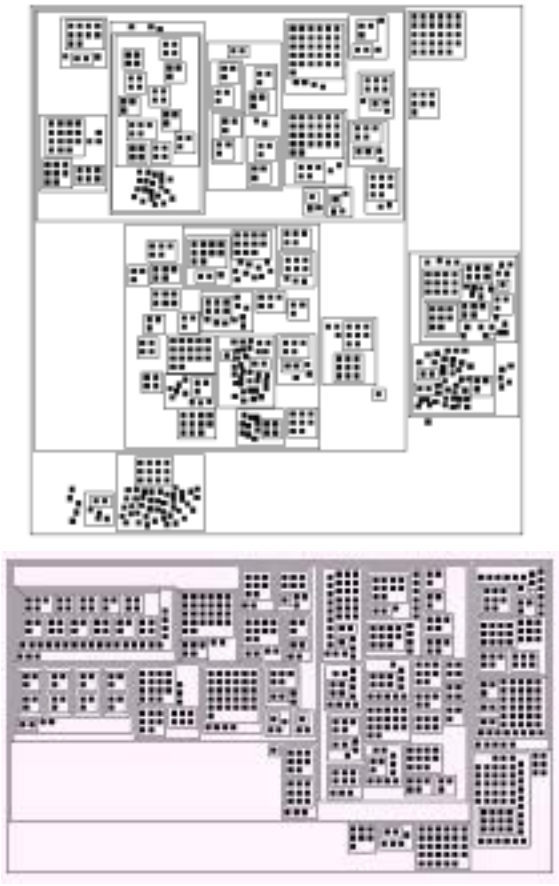

Figure 10. Example of non-uniformly nested hierarchical data. (Upper) Visualized by Data Jewelry Box. (Lower) visualized by the proposed technique.