筑波大学 情報学群 情報メディア創成学類

卒業研究論文

リーフの矩形幅が均一な
空間充填型の階層構造表現手法

小林 愛実

指導教員　三末 和男，志築 文太郎，田中 二郎

2012 年 2 月

概要

階層構造における表現手法の 1 つとして Treemap がある．これは，ノードを矩形として表し，親子関係は入れ子状に配置することで表す手法である．Treemap はリーフのみが重みを持つ階層構造を対象とし，この重みを矩形の大きさで表現する．これに加えて，色の変化や 3 次元化などの工夫を加えることにより，複数のデータを表現することもできる．

本研究では，複数のデータを表現するための工夫の中で，Treemap の矩形内に新たなチャートを描画する表現に注目した．従来の Treemap の矩形内にチャートを描こうとした際，矩形の形状が様々であることを問題視した．なぜなら，チャートの各軸の目盛間隔を統一することが難しいと考えたからである．

そこで，この表現に適した新たな表現手法として，リーフの矩形幅が均一な表現を提案した．また，アルゴリズムを開発し，既存のアルゴリズムと比較することで評価を行った．そして，開発したアルゴリズムを実在するデータへ適用させ，その結果を既存の表現と比較した．

目 次

図 目 次

第1章　はじめに

1.1　階層構造

データ構造の１つに “階層構造” と呼ばれるものがある．これは，グラフ理論の木に基づく構造である．例えば，ファイルシステムにおけるディレクトリ構造，会社などの組織の構造がこれに該当する．

木はノードからなる．ノードは単一の親を持ち，複数の子を持つ．親を持たないノードをルートと呼ぶ．子を持たないノードをリーフと呼ぶ．リーフ以外のノードを節ノードと呼ぶ．

1.2　Treemap

階層構造の可視化手法の１つとして “Treemap”[1] がある．Treemap は，リーフのみが重みをもつ階層構造を対象データとして扱う．この例を図 1.1 に示す．

Treemap は，各ノードを矩形として表現する手法である．ノードの矩形は，面積が重みに対応するように設定される．ノード間の親子関係は矩形を入れ子上に配置することにより表現される．親子関係を持たないノード同士は，重ならないように配置される．

ルートの矩形形状は予め設定しておく．その他のノードは，同じ親を持つノード同士の重みの比を元に，親ノードの矩形を一定方向に分割する．分割する方向は，階層毎に変化する．図 1.1 に示す階層構造を Treemap にしたものを図 1.2 に示す．

矩形の大きさは 1 次元データを表すことができる．加えて，色相や明度，彩度といった色に関するパラメータを変化させることにより，より多くのデータを表現することもできる．Treemap は拡張の余地がある表現であるといえる．

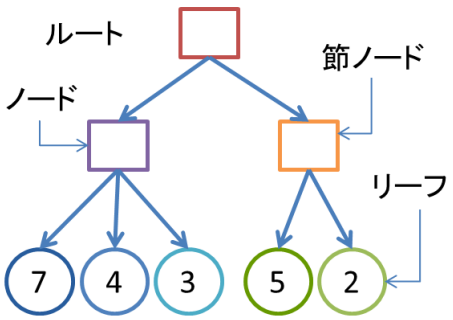

図 1.1: Treemap が対象とする階層構造の例

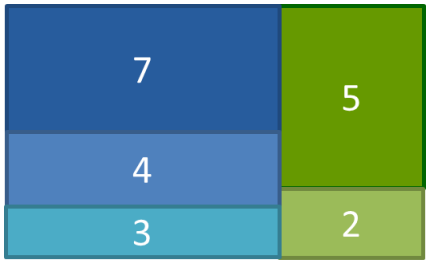

図 1.2: Treemap の描画例

1.3 対象とするデータ

本研究では，各リーフが 2 次元データを持つ階層構造である．この 2 次元データは一方の次元に絶対量を表すデータを有する．例えば，月別降水量データにおいて，各月の降水量がこれに当たる．また，絶対量の総和をリーフの重みとして扱う．

1.4 矩形にチャートを埋め込む表現

先に述べた Treemap において，リーフ矩形内に 2 次元データを表すチャートを描く表現が存在する [2]．チャートとはデータを表すための表現のことであり，例えば棒グラフや折れ線グラフがこれにあたる．しかし，従来の Treemap では，その矩形内に新たなチャートを描く表現に適していないと考えた．なぜなら，従来の Treemap は，1 つの Treemap 内に描かれる相異なる矩形やチャートの大きさが異なる．そのため，各矩形内にチャートを描画した場合，各軸に同じ目盛間隔を適用させることは難しい．もし，目盛間隔を統一した場合，矩形の形状を生かせない，潰れたようなチャートが描画される可能性が考えられる．

1.5 研究の目的

本研究では，チャート同士の比較の容易さを得ることを目的とする．Treemap ベースでリーフの矩形の幅が一定な表現手法を開発する．これにより，チャート同士の目盛間隔を統一することができるため，比較が容易になると考えられる．

なお，本研究における “チャート同士の比較の容易さ” は，Treemap のリーフ矩形内に描かれたチャート同士を目で見て，各リーフの持つデータを比較し，その特徴を容易に見いだせるか，である．

第2章　関連研究

本研究では，階層構造の表現手法を開発することを目的としているが，階層構造の表現手法は既に様々存在する．関連研究として，既存の階層構造表現手法について以下に述べる．

また，リーフ矩形内にチャートを描画した際に想定される状況も検討する．その表現がチャートを埋め込む表現に適しているかどうかを考察する．

2.1　領域分割によって階層を表現する手法

2.1.1　Treemap

階層構造の表現手法の中で，領域分割によって階層を表現する手法として Treemap[1] が挙げられる．Treemap は，各ノードの重みに応じ，予め設定された矩形を一定方向に分割していく手法である．分割する向きは階層ごとに縦横交互になされる．例えば，1 階層目を縦方向に分割すると，2 階層目は横方向に，3 階層目は縦方向，と繰り返される．また，ノード同士の親子関係は矩形を入れ子状に表現することにより示される．

Treemap のアルゴリズムでレイアウトされた各リーフ矩形は，親となるノードが同じリーフ同士に関しては幅あるいは高さが統一される．そのため，親ノードが同じリーフの矩形内チャート同士は 1 軸の目盛間隔を統一することもでき，比較が容易かもしれない．しかし，それ以外のリーフ同士を比較しようとした場合，その比較は難しいと考えられる．また，階層毎に同じ方向で分割しているため，極端に細長い形状になりやすい．そのため，リーフの矩形内に新たなチャートを描こうとした際，極端な矩形であるとチャートが潰れてしまう可能性が高い．

2.1.2　形状を意識した Treemap

矩形の形状を意識した Treemap の変形として，Squarified Treemap[3] や Circular Treemap[4] が挙げられる．Squarified Treemap は，各矩形の形状が可能な限り正方形に近づくように形成される表現である．Circular Treemap は，各ノードを矩形ではなく円形で表す表現である．

しかし，何れの表現においても各ノードの各辺，各直径の長さは大きく異なる．そのため，チャートを埋め込もうとした際には，チャートの軸の目盛間隔を統一することが難しい．例え統一したとしても，極端な表現となってしまう可能性が高い．例えば縦軸の最大値は 100 であるが，データの最大値は 10，というような表現であり，これは適切とはいえない．

2.1.3 位置を意識した Treemap

矩形の位置を意識した Treemap の変形として，Ordered Treemap[5] や Spatially ordered treemap[6] が挙げられる．Ordered Treemap は，順序付きの階層構造データに対して使用される表現であり，その順序関係を維持しながら Treemap を形成する．Spatially ordered treemap は，例えば都道府県のように各ノードが地理的な位置関係を有する階層構造データに対して使用される表現であり，その位置関係を維持しながら Treemap を形成する．また，Cluster Treemap[7] と呼ばれるものもある．これは，データ同士の距離を 2 次元チャート内の配置に反映させたものである．

これらの表現はあくまで位置を意識した表現であるため，その形状については上述の形状を意識した表現と比べて優先度が低い．そのため，リーフ矩形の形状に関しては，形状を意識した表現と同等，あるいはそれ以上にバラつきが生じるものと考えられる．よって，チャートを埋め込もうとした際には，チャートの軸の目盛間隔を統一することが難しいと考えられる．

2.2 矩形を詰め込む手法

階層構造の表現手法の中で，矩形を詰め込んでいく手法の 1 つとして，平安京ビュー [8] が挙げられる．平安京ビューでは，リーフ矩形は全て同じ大きさの正方形として表現される．節ノードは，自身の子となるノードを囲う矩形の枠として表現される．各矩形は重ならず，節ノードがより小さくなるように配置される．また，各ノードのレイアウトは，格子分割を意識して行われている．

平安京ビューのリーフ矩形内に新たなチャートを描くとする．例えば，各リーフの持つチャートとして描くためのデータを比較した時に，ある特定のリーフの持つデータのみ，その値が大きい場合を考える．全てのリーフ矩形が同じ大きさであるため，各チャートの軸を統一しようとすると，殆どのチャートが潰れたような形となる．これにより，各チャート内での有意差や各チャートの特徴となる変化の発見を妨げる可能性があると考えられる．

第3章　Treemapにチャートを埋め込む表現における問題

Treemapとして描かれる矩形の中にチャートを埋め込む表現がある．しかし，従来のTreemapを用いた場合，チャートどうしの対比，という観点においては適切でないと考えた．なぜなら，従来のTreemapは，各リーフの持つ重みが面積に対応しており，その矩形は様々な形状，大きさだからである．

3.1　複数チャートの比較しやすさ

本研究で扱う，矩形内に描くチャートは，2次元データを描画したものである．2次元平面上に描画でき，垂直に交わる2軸を有する．例えば，2次元の棒グラフや折れ線グラフ，散布図などがこれに当たる．

従来のTreemapやそれを基にした表現のリーフ矩形内にチャートを描こうとした場合，矩形の形状が様々であるため軸の目盛間隔を統一することが難しい．これを無理に統一しようとした場合，チャートが偏ってしまったり，データ特徴が見いだせない様な軸のとり方をしてしまう可能性がある．

一般に，複数のチャートを比較したい場合，図3.1に示すようにチャート毎の軸の目盛間隔を統一し，並べて比較する．特に同じ期間の変化量などを示す時系列データや各パラメータの値を示すデータなどにおいては，チャートの幅を統一し，横軸に同一データ（時間やパラメータの種類など）を割り当てることが多い．

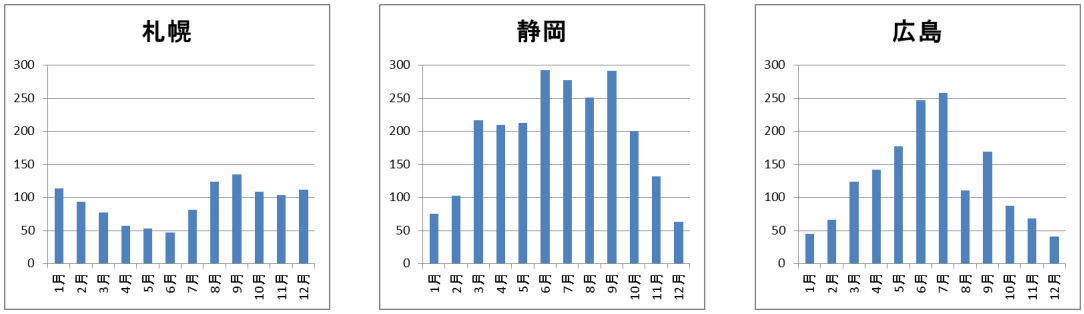

図 3.1: 複数チャートを比較する例

3.2　Squarified Treemap にチャートを入れた例

ここで，Squalified Treemap を用いてレイアウトを行い，リーフ矩形内にチャートを描いた例を図 3.2 に示す．これは，全国の降水量を都道府県毎に描いたものである．各矩形の面積は，年間降水量を表す．各チャートは，月別降水量を表し，横軸が月，縦軸が降水量である．なお，チャートの横軸は矩形の横幅をデータ数で等分することにより棒グラフの棒の幅を設定した．縦軸は全てのチャート間で目盛の間隔が統一されるように設定した．

図 3.2 を見ると，各リーフ矩形の横幅は様々である．棒グラフの棒の幅にもばらつきがある．そのため，複数の棒グラフを比較する際，棒の高さだけでなく，面積にも注目しやすく，正しい比較が難しい．

また，所々に縦長の矩形が存在する．これらの矩形では，その中に描かれた棒グラフが矩形内の下方に偏っている．そのため，1 つの棒グラフ内のデータ同士の差を発見しづらい．

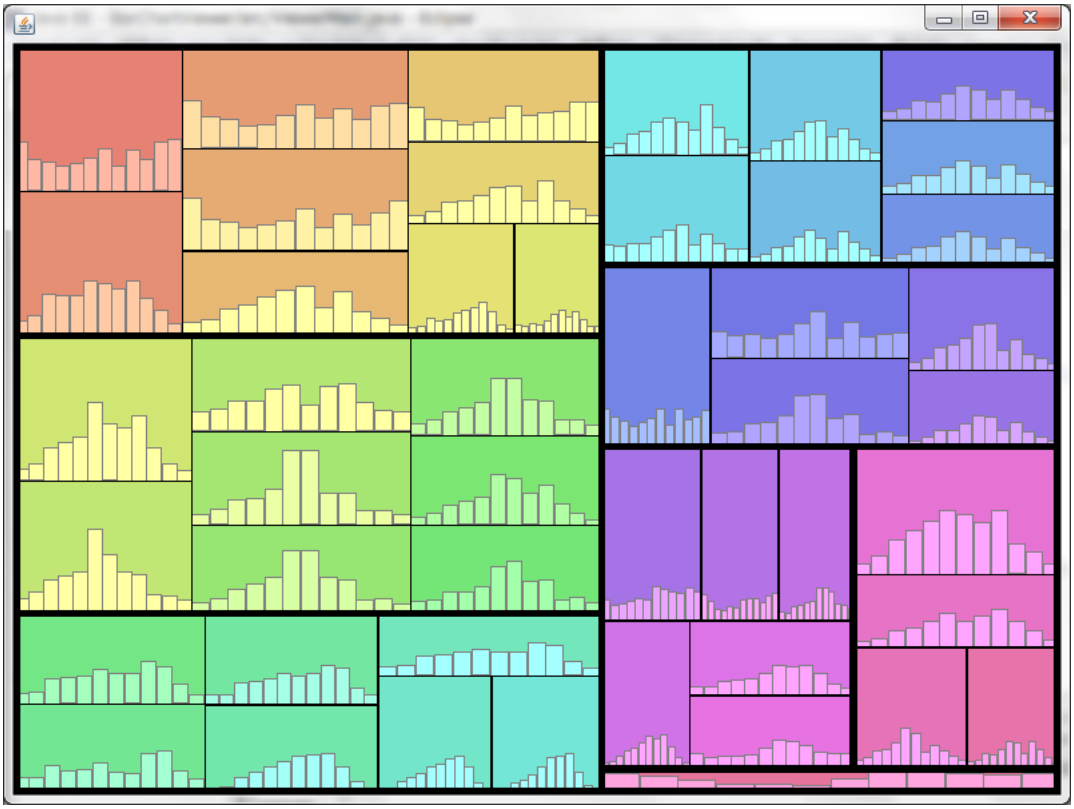

図 3.2: 都道府県別降水量を表す例

3.3 横幅のばらつき

Squarified Treemap によるレイアウトでは，リーフ矩形は正方形に近づくように形成される．そのため，図 3.2 においても各リーフ矩形の横幅はそれぞれ異なる．棒グラフの棒の幅は，リーフ矩形の横幅をデータ数で等分するため，これもそれぞれ異なる．

一般に，棒グラフは棒の高さを比較するチャートである．しかし，棒の幅がそれぞれ異なるチャート同士を比較しようとすると，高さだけでなく，棒の面積にも注目しやすい．そのため，例え高さが同じデータ同士であっても，面積を比較してしまい，同じ値ではないという判断をする可能性がある．

3.4 矩形上部の空白

本研究では，リーフ矩形内のチャート同士の比較を容易にすることを目的とした．そのため，図 3.2 では，棒グラフの縦軸の目盛間隔を全体で統一した．これは，縦軸の目盛間隔を統一しない場合，異なるリーフ矩形内の棒グラフ同士を比較する際，見たままのスケールで比較することは出来ないからである．棒グラフをそれぞれ適切な大きさに拡大，あるいは縮小して比較する必要がある．

一般に，描画が偏っているチャートは，描画領域を適切に使用できていない．図 3.2 では，一部の矩形において矩形上部には何も描かれず，空白となっている．リーフ矩形の形状を有効に使用できていない．また，棒グラフが縦方向に潰れたような形状になっている．そのため，棒グラフ内のデータ毎の有意差が見いだせない可能性もある．

3.5 本研究で扱う問題点

既存の Treemap へチャートを埋め込む際の問題点は以下の通りであるとわかった．

- 矩形の幅が様々であるため，棒グラフの幅もばらついてしまう点
- 縦軸の目盛間隔を統一するにあたり，描画領域の上部に空白ができてしまう点

本研究では，これらの問題に着目し，解決することを目指す．

第4章　リーフの矩形幅が均一な表現

4.1　リーフの矩形幅の統一

本研究では，リーフの矩形幅を均一にすることにより，リーフ矩形内のチャート同士の比較を容易にすることができると考えた．まず，横軸の目盛は統一することができるため，チャート同士の 1 軸を同一視することができる．各データの差は高さとしてのみ現れるため，比較も容易である．また，リーフ矩形の高さは重みにより変化し，重みはリーフが持つデータに依存するものとすれば，大きなデータを持つリーフ程，高さの大きい矩形を形成し，その矩形内に描かれるチャートの縦軸の最大値も大きく取ることができる．そのため，縦軸の目盛間隔を統一した場合においても，データ同士の有意差を認識できるチャートを描くことができると考えられる．

“リーフの矩形幅が均一”とは，具体的に図 4.1 のようなチャートを想定する．親となるノードに関係なく，全てのリーフの矩形幅が均一な表現である．なお，本論文ではリーフ矩形の横幅を統一することを目標にしたレイアウトを行うが，縦と横を逆にして，縦幅を統一するレイアウトとすることも可能である．

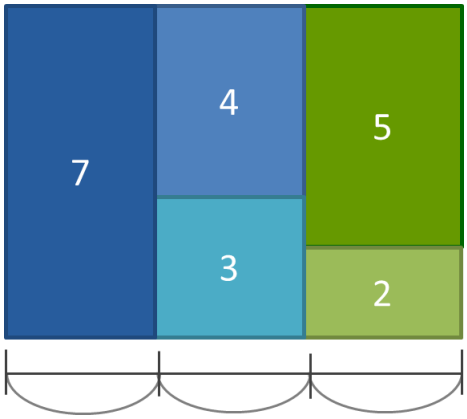

図 4.1: 提案手法の描画例

4.2　美的基準の検討

本研究では，今までの美的基準でも考慮されてきたアスペクト比に加え，充填率というものも考える．すなわち，余白を設けることも規制しない．

はじめに，本研究における充填率は全リーフ矩形の面積を足して，それをチャート全体の面積で割ったものとする．充填率の取り得る値の範囲は，1 以下の正の数である．また，アスペクト比は，矩形の長辺の長さを短辺の長さで割ったものとする．アスペクト比の取り得る値の範囲は，1 以下の正の数である．

充填率の低いチャートは空間効率が悪く，あまり好ましくない．しかし，従来の Treemap と同様に充填率を 1 に保とうとすると，やはり多くの場合，同じ方向に矩形を積み上げていくしかない．充填率の問題は解決するが，各リーフ矩形のアスペクト比が 0 に近い値を取る可能性が高くなる．各リーフ矩形の形状が細長い長方形を描く，ということである．ここで，見やすさの観点で，矩形形状が正方形に近い方が良いことは Squarified Treemap などでも述べられている．本研究においては，矩形内に別のチャートを描くことを前提として設計しているため，矩形形状は細長い長方形よりも正方形の方が良い．しかし，逆にアスペクト比を可能な限り高めようとすると，充填率が下がってしまう．そのため，チャート全体の充填率と各リーフ矩形のアスペクト比のバランスをどうとるかが，本研究における課題の 1 つでもある．

4.3　チャート内の空白についての方針

提案するアルゴリズムを開発するにあたり，従来の Treemap の制約下では実現が難しいと考えた．本研究では，この問題点に対し，制約を緩めることにより，これを解決する．チャート内に空白ができることを許し，アルゴリズムの実現を目指す．

従来の Treemap は，矩形を分割していくアルゴリズムであったため，チャート内に空白は存在せず，充填率は常に 1 であった．しかし，リーフの矩形幅を均一にするにあたり，この方針を継続することは難しいと考えた．矩形を全て同じ方向に詰め込むものとすれば，全ノードの矩形幅は統一されるが，リーフ矩形は細長く，アスペクト比が低くなり，チャートを詰め込むには適さない形状となる．一方，リーフの矩形幅を均一に保ちながら，複数列に詰め込もうとすると，空白なしでは実現不可能な階層構造が存在してしまう．しかし，空間効率を考えると，充填率は高いほうが良い．

例えば，図 4.2 に示す様に，同じノードを親として持つ 3 つのノードがある．それぞれの重みが 3,4,5 である．リーフの矩形幅を均一に保とうとした際，3 つのノードを 1 列詰め込んだ場合は実現可能である．しかし，各リーフ矩形は細長い形状となり，その中にチャートを描くには適さない．次に，2 列に詰め込もうとした場合，従来の空白を許さないという制約下では実現できない．重みが 3 と 4 の矩形を同列に，その隣りに重みが 5 の矩形を，と詰め込むとすると，重みが 2 の分だけ足りず，空白ができてしまう．この例は，一般に十分想定され得る状況であり，無視できない問題である．そのため，本研究ではチャート内に空白ができることを許す方針とした．

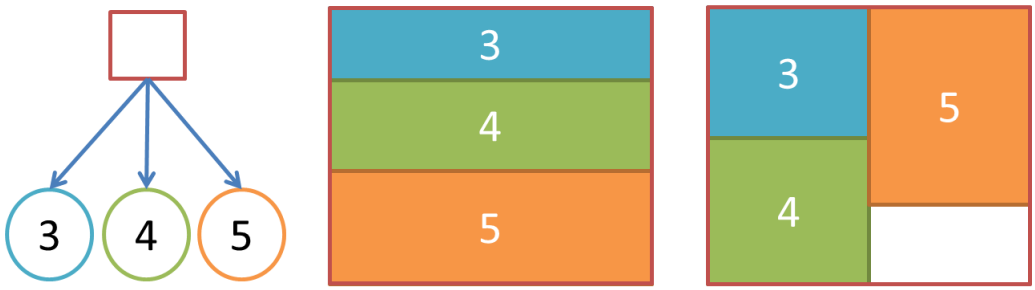

図 4.2: 空白をなくすことがが難しい例

4.4 制約条件の設定

- ノード同士の親子関係は矩形を入れ子状に配置する
- 親子関係にないノードの矩形同士は重ならないように配置する
- リーフの矩形幅は均一にする
- リーフ矩形の高さはリーフの持つ重みに比例する

第5章　アルゴリズムの開発

本アルゴリズムは，リーフ矩形の大きさを決める過程，各矩形の座標決定を行う過程，レイアウト結果を評価する過程，パラメータを変更する過程からなる．評価結果を元に詰め込み結果を繰り返し検討し，より良いレイアウト結果を導出するアルゴリズムである．処理の流れを図 5.1 に示す.

従来の Treemap は，領域を分割していく方針であり，階層構造のルートからリーフに向けて処理を行う，トップダウンのアルゴリズムであった．提案手法では，リーフの矩形幅を統一する．一般に，トップダウンで領域分割を行う方針でリーフの矩形幅を統一する場合，全て同じ方向に分割するのであれば，幅の統一は可能である．しかし，この場合，各リーフ矩形の形状は細長くなってしまう．リーフ矩形内に棒グラフのようなチャートを描くことを考えると，各矩形は正方形に近い方が良い．そこで，本アルゴリズムでは，階層構造のリーフからルートに向けて，ボトムアップで矩形を詰め込む，という処理を行う.

まず，対象となる階層構造に対し，リーフ矩形の大きさを決定する．リーフの重みを入力とし，各リーフを矩形として出力する.

次に，矩形とその入力順序，積み上げることが可能な高さを入力とし，矩形の詰め込みを行う．矩形の詰め込み過程は，節ノード毎に処理を行い，詰め込まれた矩形のレイアウト結果を出力する．各節ノードへの矩形の詰め込みは，長方形詰め込み問題の 1 つであるストリップパッキング問題として扱う．本アルゴリズムでは，既存のストリップパッキング問題の解法である NF 法を用いた.

また，この手法は，矩形の入力順序と積み上げることが可能な高さの最大値によりレイアウト結果が変化するレイアウト手法である．そのため，これら 2 つのパラメータを変化させながら繰り返しレイアウトを行うことで，より良いレイアウト結果が得られるのではないかと考えた.

ここで，上述の 2 つのパラメータは各節ノードに対する詰め込みが終わる度に変更できるように思える．矩形の入力順序に関しては，節ノード毎に矩形を詰め込む際に全ての順序を試す．その中で，最も充填率の高いレイアウトとなる順序を採用することで対応できる．しかし，このアルゴリズムでは，全ノードのレイアウトを計算し終えた後に表示領域へ適用させるためのサイズ調整を行う．このサイズ調整が終わるまでは，矩形の大きさや位置が確定しない．そのため，チャート全体のレイアウト結果に関わる，積み上げ可能な高さの最大値に関しては，節ノード毎に評価できない．本アルゴリズムでは，全ノードに対する矩形の詰め込みが終了した後に積み上げ可能な高さの最大値を変更する.

その後，全ノードに対する矩形の詰め込みを行った後，サイズ調整をした結果を入力とし，

レイアウト結果を評価する．そのために，レイアウト結果を評価するための関数を作成した．この関数はチャート全体の充填率とリーフ矩形のアスペクト比平均を元に作成した．

評価過程によりレイアウト結果の評価を行った後，パラメータの変更を行う．パラメータは，矩形を詰め込む過程で必要となる，矩形の入力順序と積み上げ可能な高さの最大値である．この 2 つのパラメータを節ノード毎に設定し直しし，再び矩形の詰め込み過程へ戻る．これを繰り返すことにより，より良い解へとたどり着ける．

なお，ストリップパッキング問題は NP に属する問題である．そのため，本研究では準最適解となるレイアウト結果を計算する．

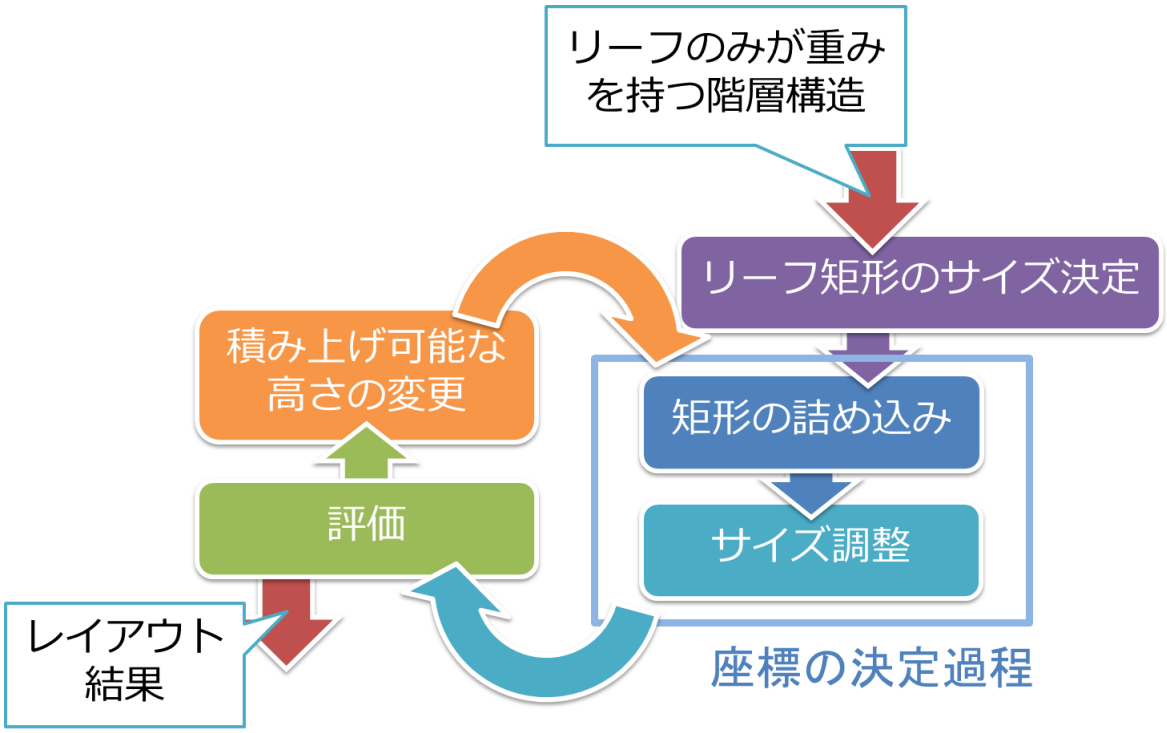

図 5.1: アルゴリズムの概要

5.1 リーフ矩形の決定

この過程では，対象となる階層構造のリーフの重みに着目し，重みに対応した大きさの矩形を形成し，出力する．矩形の幅は何れも統一するため，1 とする．高さは，その重みに比例させる．

なお，全ノードの矩形のレイアウトが決定した後，表示領域へ適用させるためのサイズ調整を行う．そのため，この過程で決定したリーフの大きさは後に変化する．

5.2 矩形の詰め込み過程

5.2.1 矩形の詰め込み

従来の Treemap は，予め与えられたルートの矩形を各ノードの重みにより分割していく手法であった．しかし，リーフ矩形の形状に制約を与えるに当たり，この方針で美的基準を考慮することは難しいと考えた．そこで本研究では，階層構造をルートからリーフに向けて辿るのではなく，リーフからルートに向けて辿り，レイアウトを行う方針とした．同じノードを親とするリーフ矩形を別の矩形に詰め込み，親ノードの矩形とする．これを再帰的に行うことによりチャート全体をレイアウトする手法をとる．

5.2.2 再帰的な矩形の詰め込み

本研究で開発した手法は，各節ノード毎にその子ノードを再帰的に詰め込んでいくことによりレイアウトを行う．複数の異なる形状の矩形を重なることなく配置する問題は，長方形詰め込み問題（rectangle packing problem）と呼ばれ，その中には様々な種類が存在する．この中でも，特にリーフ矩形の詰め込みは，詰め込まれる矩形の高さのみが可変な “ストリップパッキング問題” に該当する．本研究では，リーフ以外のノードの矩形を詰め込む問題に対しても，ストリップパッキング問題に帰着させている．ストリップパッキング問題は NP に属する問題であるため，最適解を導出することは難しく，準最適解を導出する．

また，矩形の詰め込みは各節ノード毎に行われるものである．そのため，階層構造全体を描画するために，この矩形の詰め込みを再帰的に行う必要がある．

5.2.3 ストリップパッキング問題

今堀ら [9] によると，形集合 J と幅　（固定），高さ y（可変）という大きさを持つ長方形の箱が一つ与えられる．すべての長方形を箱の中に重なりなく詰込むという制約条件のもとで，箱の高さ y を最小化する問題，をストリップパッキング問題と呼ぶ．つまり，幅のみが指定された箱に対して長方形群を詰め込み，いかに箱の高さを小さくできるか，という問題である．

これ以降，リーフ矩形の横幅の統一を考えるため，この問題における母材の幅と高さを逆として扱うが，これを逆とせず，高さの統一を図ることも原理的には可能である．

5.2.4 矩形の詰め込みについての方針

本研究では矩形の詰め込みをストリップパッキング問題として捉えると述べた．ストリップパッキング問題の解法には，母材の出来るだけ左下に詰め込んでいく手法やビンパッキング問題の解法の適用などがある．

母材の左下に詰め込んでいく手法の 1 つとして，Baker らの BL 法（bottom-left algorithm）[11] がある．BL 法は，矩形の各頂点を把握しておき，矩形を 1 つずつ詰め込んでいく．母材

の空いている空間の内，詰め込む矩形の入る最も左下の空間を探す．各矩形のレイアウトを確定する際には，他の先に詰め込まれた矩形の頂点を参照する必要があり，繰り返しの処理が必要となる．

ビンパッキング問題の解法は，Coffman ら [10] によって紹介されている．その 1 つとして，Johnson の NF 法（next-fit algorithm）[12] がある．このアルゴリズムは，矩形のレイアウトを確定する際，繰り返しの処理を必要としない．そのため，レイアウトに要する時間が短い．本研究では，この NF 法を採用し，矩形の詰め込みに使用する．

なお，ビンパッキング問題とは，複数の箱に対し予め与えられた長方形群を詰め込む際，可能な限り少ない数の箱の中に詰め込む，という問題である．

5.2.5 NF 法

矩形の詰め込み問題における解法の 1 つとして，NF 法（Next-Fit Algorithm）と呼ばれるものがある．この手法は，入力順序に依存はするものの，レイアウトを決める際の再帰的な処理が不要であるため，他の解法と比べて処理工程が少なく，効率が良い．入力値として，詰め込むべき幅が一定な矩形群，詰め込まれる矩形の幅を持つ．なお，詰め込まれる矩形の高さは可変であり，これが最小になるようなレイアウトを探す．

NF 法における矩形レイアウトの流れを図 5.2 に示す．まず，詰め込まれる矩形の左下に 1 番の矩形を置く．次に，先に詰め込んだ矩形の隣に 2 番の矩形が置くことが可能かを判断する．もし置くことが可能であれば，そこへ置く．置くことが出来なければ，横に並んだ矩形の中で，最も大きい高さに合わせ，境界線を引き，その境界線の左上に置く．以降に詰め込む矩形は境界線より上に置かれ，境界線より下の領域には何も詰め込むことはできない．この境界線により分割された領域をレベル，と呼ぶ．以上のように，直前に置いた矩形の隣に現在の矩形が置けるか否かを判断し，置くことが可能であればその場所に，置けなければ境界線を引き，次のレベルでその処理を再開する．この過程を繰り返す．

アルゴリズム

```
for(i=0; i < 詰め込む矩形の数 ; i++){
   if((積み上げ可能な高さ - (i-1) 番目の矩形高さ) > i 番目の矩形高さ){
      (i-1) 番目の矩形の下に i 番目の矩形を配置する;
   }else{
      (i-1) 番目の矩形が積まれている列の中で 1 番大きい横幅を探す;
      境界線を縦に引く;
      i 番目の矩形を，左の辺が境界線に接するように，左上に配置する;
   }
}
```

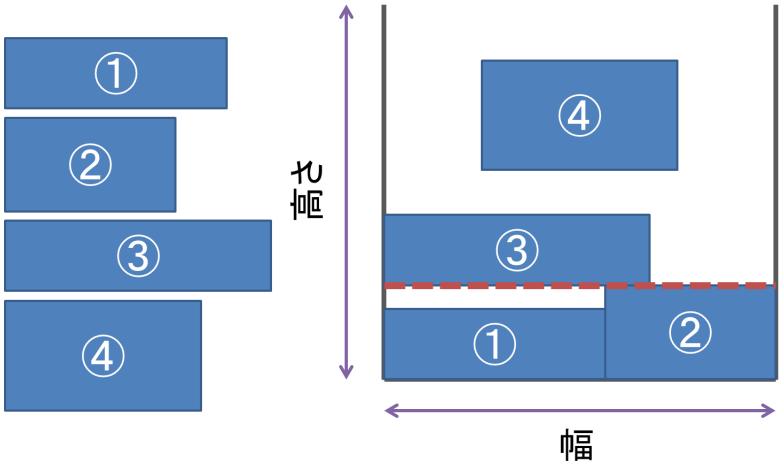

図 5.2: NF 法の説明

5.2.6　節ノードにおける矩形の詰め込み

子ノードとしてリーフのみを持つ任意の節ノードに対し，上述の NF 法を適用させる．NF 法は詰め込まれる矩形の幅が固定，高さが可変である．本研究ではリーフの矩形幅が均一であるため，幅を可変，高さを固定として扱う．詰め込まれる矩形の高さ（積み上げ可能な高さ）は予め設定しておく．また，詰め込む矩形の入力順序も予め設定しておく必要がある．

5.2.7　複数階層における矩形の詰め込み

節ノードを子ノードとして持つノードを含む階層構造に対する場合について述べる．子ノードがリーフのみの節ノードに関しては，先に述べた詰め込み過程を適用させる．

適用後，この節ノードの矩形を決定する作業が必要である．節ノードの矩形決定は，自身の持つ子ノードの矩形の座標を基に行われる．子ノードの中で最も小さい x 座標から最も大きい y 座標までを囲う矩形を節ノードの矩形とする．これにより，全ての子ノードを囲い込むことが可能な最小の矩形を生成することができる．

5.3　パラメータ変更による再レイアウト

先に述べたレイアウトアルゴリズムは，積み上げることが可能な高さや矩形の入力順序に依存するものであるとわかった．そのため，1 度処理を行っただけでは，より良い解に辿り着けるとは言い切れない．そこで，これらのパラメータを変更し，処理を繰り返すことによりより良い解を得ようと考えた．これらパラメータの決定方針を以下に示す．

5.3.1 入力順序の決定

NF 法は矩形の入力順序に依存するレイアウトアルゴリズムである．より良い解を得るためには，入力順序を変更し，再レイアウトを繰り返す必要がある．

このアルゴリズムでは節ノード毎に NF 法を用いる．そのため，入力順序の決定も節ノード毎に行われている．そこで，節ノード毎に入力順序を 1 つずつ変更していけば良いのではないか，と考えた．考えられ得るすべての入力順序で階層構造全体をレイアウトしたのであれば，その中から最も良いレイアウト結果を導出できるはずである．

5.3.2 積み上げ可能な高さの決定

NF 法を用いる際，積み上げ可能な高さは任意に予め設定しておくものとしていた．しかし，このパラメータの値によりレイアウト結果が大きく変化する．そのため，より良い解を得るために，積み上げ可能な高さを変更し，再レイアウトを繰り返す．

まず，積み上げ可能な高さのとり得る最小値を考える．自身の持つ子ノードを全て詰め込むためには，その中で最も高さの大きいものが入る高さでなければならない．そのため，積み上げ可能な高さの最小値は，自身の持つ子ノードの中で，最も大きい高さである．

次に，積み上げ可能な高さのとり得る最大値を考える．最も充填率が高いレイアウトは，全ての子ノードが同じ方向に並んだ場合である．つまり，全ての子ノードを縦に積み上げた際の高さが，積み上げ可能な高さのとり得る最大値である．

また，上述の範囲内での値の変化量を考える．積み上げ可能な高さは連続的に変化させることができれば全ての範囲を網羅できる．しかし，それでは計算量が膨大になってしまう．そこで今回は，自身の持つ子ノードの中で高さが最小のものを変化量とし，積み上げ高さ h を決定することとした．その式を式 (5.1) に示す．

式内の h_{min} は，自身の持つ子ノードの中で最も小さい高さを示す．これに，1 ずつ増える変数 i を掛け，積み上げ可能な高さを変化させる．また，h_{max} は，自身の持つ子ノードの中で最も大きい高さを示す．

$$h = h_{min} \cdot i + h_{max} \quad （h_{max} \leq h \leq h_{sum}）\tag{5.1}$$

5.4 評価過程

このアルゴリズムでは，パラメータを変更しつつ，繰り返しレイアウトを行う．複数のレイアウト結果が出力されるため，これを評価する必要がある．そこで，レイアウト結果を評価するための評価関数を設計した．この評価関数は，レイアウト結果が出る毎に呼び出され，計算される．そして，より良いレイアウト結果を 1 つだけ保存しておき，全てのレイアウト結果の計算が終了した後に取り出すものとした．

5.4.1 チャート全体の充填率

ルート矩形の中にリーフ矩形がどれだけ占めるかをチャート全体の充填率，と呼ぶ．充填率 f は式 (5.2) のように定義され，値のとり得る範囲は 1 以下の正数である．その式を以下に示す．

$$f = \frac{\sum \text{リーフ面積}}{\text{ルート面積}} \tag{5.2}$$

$f = 1$ の時，チャートには空白は無く，f　0 の時，チャートは空白だらけである．充填率は高い方が空間効率も良いため，f は 1 に近いほど良い．

5.4.2 リーフ矩形のアスペクト比平均

長辺と短辺の比をリーフ矩形のアスペクト比，と呼ぶ．また，全てのリーフ矩形のアスペクト比を平均したものをリーフ矩形のアスペクト比平均，と呼ぶ．アスペクト比 $r(l)$ およびアスペクト比平均 $\bar{r}$ は式 (5.3) およびの式 (5.4) に定義され，値のとり得る範囲はいずれも，0 を超え，1 以下である．その式を以下に示す．

$$r(l) = \frac{\text{矩形 } l \text{ の短辺}}{\text{矩形 } l \text{ の長辺}} \tag{5.3}$$

$$\bar{r} = \frac{1}{|L|} \sum_{l \in L} r(l) \qquad (0 < \bar{r} \leq 1) \tag{5.4}$$

$l(r) = 1$ の時，リーフ矩形は正方形であり，$l(r)$　0 の時，リーフ矩形は極端な長方形（酷く細長い形状）である．チャートを埋め込むことを考えた上で，リーフ矩形が極端な長方形であると見栄えが良くない．そのため，$l(r)$ は 1 に近いほど良い．

5.4.3 評価関数の設計

上述のチャート全体の充填率，リーフ矩形のアスペクト比平均を基に，評価関数 s を式 (5.5) のように定義する．w_1，w_2 は f，$\bar{r}$ にそれぞれ掛かる任意の重みであり，その値は実数である．

$$s = w_1 \cdot f + w_2 \cdot \bar{r} \tag{5.5}$$

なお，充填率とアスペクト比平均のとり得る値の範囲は同じであるため，w_1 と w_2 を同じ値とすれば，どちらかに偏ることなく，2 値が平均して高いレイアウトを出力する．また，充填率を重視したい場合は f に掛かる w_1 を，アスペクト比平均を重視したい場合は $\bar{r}$ に掛かる w_2 を大きくすることで実現可能である．

第6章　アルゴリズムの改良

6.1　高速化

これまで NF 法を用いたレイアウトアルゴリズムを提案してきた．しかし，このアルゴリズムは扱うデータの階層が深いほど，あるいは同じ親を持つノードの数が多いほど処理時間が爆発的に増えるものとわかった．そこで，高速化を図るために，リーフの矩形を詰め込む処理に着目した．NF 法は形状が様々な矩形を詰め込む際に用いられるものである．そのため，矩形幅が一定であると予め決められているリーフ矩形を詰め込む際にはより効率的なアルゴリズムがあると考えた．これを踏まえ，新たなレイアウトアルゴリズムを開発した．

6.1.1　積み上げ法

リーフ矩形を詰め込む際に用いるレイアウトアルゴリズムとして，積み上げ法を提案する．入力値として，詰め込むべき幅が一定な矩形群，詰め込まれる矩形の高さ（積み上げ可能な高さ）を持つ．なお，詰め込まれる矩形の幅は可変長であり，予め設定されていないものとする．

積み上げ法における矩形レイアウトの流れを図 6.1 に示す．まず，詰め込むべき矩形を詰め込まれる矩形内に高さの大きい順に横 1 列に並べる．すると，左から右へ下り階段のような形状に矩形が並ぶ．次に，右端の最も小さい 5 番の矩形を左端の最も大きいの矩形の上に積み上げられるか判断する．もし積み上げることが可能であれば 1 番の矩形の上に，そうでなければその右隣の 2 番の矩形の上に積み上げられるか判断する．ここでも同様に，積み上げることが可能であれば 2 番に大きい矩形の上に，そうでなければ右隣の 3 番へ，と繰り返し，積み上げ可能な場所を探す．もし自身の左隣である 4 番の矩形の上にも積み上げることができなかった場合，冒頭で並び替えた直後の状態が最適解となる．一方，その途中で積み上げることが可能な矩形が見つかった場合，最も小さい矩形は該当した矩形の上に置かれ，4 番目に大きい矩形の処理に移る．4 番の矩形は，5 番の矩形を積み上げることが可能であった矩形の右隣の矩形から積み上げることが可能かどうかの判断を行う．これは，大きい順に並べているため，最も小さい矩形を積み上げることができなかった矩形の上に 2 番目に小さい矩形を積み上げることは不可能だからである．同様に，4 番の矩形の処理が終わった後は 3 番の矩形，2 番の矩形，と小さい方から順に処理を行なっていく．最終的に，どの矩形の上にも積み上げることができない矩形が発見された場合，あるいは処理を行う矩形の上に別の矩形が積み上げられていた場合にその処理は終了となる．

アルゴリズム

```
高さが大きい順にソート;
for(i=0; i < 詰め込む矩形の数 ; i++){
   for(j=1 つ前の矩形が積み上げた矩形の右隣;
      積み上げる矩形と積み上げられる矩形が一致しない間実行; j++){
         if((積み上げ可能な高さ - 左から j 番目の矩形高さ) > 右から i 番目の矩形高さ){
         左から j 番目の矩形の上に右から i 番目の矩形を積み上げる;
      }
   }
}
```

この手法は順序依存がないため，NF 法の際に必要だった入力順序を変更しての再レイアウトを必要としない．そのため，NF 法と比べると大変処理効率のよいアルゴリズムである．しかし，これは幅が均一な矩形に特化したアルゴリズムであり，幅が異なる複数の矩形の詰め込みには適していない．なぜなら，幅が異なる矩形の場合，矩形の上に積み上げた別の矩形が，下の矩形よりはみ出したり，あるいは凹んだりすることが想定されるからである．先に述べた単純なアルゴリズムのまま適用させた場合，矩形同士が重なってしまったり，あるいは無駄な空白が生まれてしまう可能性がある．これを解決するためには，矩形の高さだけでなく幅についても意識しなければならず，また，入力順への依存も生じると考えられ得る為，処理効率が格段に低下する．そのため，リーフ矩形の詰め込みにのみ使用する．

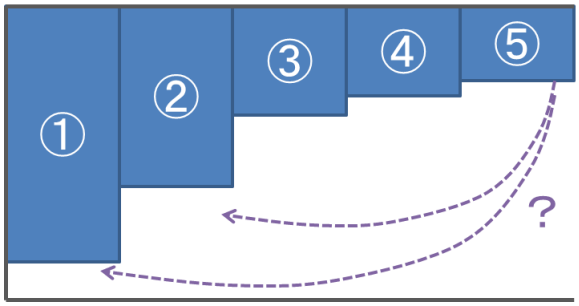

図 6.1: 積み上げ法の説明

6.1.2 NF 法と積み上げ法の組み合わせ

先に述べたとおり，積み上げ法は幅が一定の矩形に特価したレイアウトアルゴリズムである．そのため，本研究ではリーフを詰め込む過程においては積み上げ法，それ以外の過程においては NF 法を用いることを提案する．NF 法は入力順序に依存するレイアウトアルゴリズムであるため，全ての入力順序を試し，最良となるものを適用させる処理を繰り返すため，階

層が 1 つ増えただけであっても，その処理工程も爆発的に増加した．そのため，リーフ矩形を詰め込む過程のみを効率のよいアルゴリズムに変更しただけであっても，その処理時間の大幅な減少が期待される．

6.2 彩色の工夫

当初，矩形内の色については，リーフ矩形にのみ塗りつぶしの処理を行なっていた．従来の Treemap では，空白が存在しなかったため，リーフ矩形のみを塗りつぶすことで，チャート全体が色づく様になっていた．しかし，提案手法では，空白を許す方針としたため，リーフ矩形のみを塗りつぶしたとしても，無色の領域ができてしまった．この無色の領域が生じると，リーフと節ノードの間に大きな隔たりがあるように感じた．そのため，どのリーフがどの節ノードの子であるのか，という親子関係が識別し辛くなる．

そこでまず，同じノードを親として持つリーフに関しては，色相の近い色を設定することとする．その上で，節ノードには自身の子となる全ノードの色を平均し，それを少し明るくした色で塗りつぶすこととした．これにより，親子間は同系色で塗りつぶされるため，親子関係の識別が容易になると考えられる．また，チャート全体を色付けすることで，空白として視覚的に認識される領域の面積も減るのではないか，と考えている．

第7章　ユースケース

7.1　使用するデータについて

ユースケースを行うにあたり，矩形内に描くチャートの変化特徴が異なる 2 つのデータを使用した．1 つ目に，値がなだらかに変化するデータとして，都道府県別の降水量データを使用した．2 つ目に，値が極端に変化するデータとして，所属研究室内で作業の進行状況を管理するためのチケットデータである．これらは，何れもリーフ矩形内のチャートに描かれるデータの総和がリーフの重みとなる特徴を持つ．それぞれのデータの詳細について以下に述べる．

7.1.1　矩形内チャートの値がなだらかに変化するデータ

データの 1 つ目として，都道府県別の降水量データを挙げる．これは，気象庁が発表している気象情報の平年値（1981～2010 年）に記載されたものであり，各都道府県内の観測点 1 箇所ずつの月別降水量及び年間降水量を値として持つ．今回は，この中から東北地方，関東地方，中部地方のデータのみを選択し，扱った．

このデータが持つ階層としては，ルートが日本全体に当たり，その子ノードに関東や東海といった地方，その更に子ノードに各都道府県が存在する構造となっている．リーフとなる各都道府県は，観測地点における年間降水量と月別降水量の値を有する．

なお，提案手法を適用させた際，リーフの矩形面積は，各都道府県の観測点における年間降水量を示す．リーフの矩形内に描くチャートは月別降水量を示す棒グラフとし，横軸を時間，縦軸を降水量とする．月別降水量は各月毎に変化するが，極端な変化をするものではなく，なだらかに変化する．

7.1.2　矩形内チャートの値が極端に変化するデータ

2 つ目として，チケットデータを挙げる．これは，著者が所属する研究室において作業の進行状況などを管理するために用いられているデータである．各チケットは新規や進行中，終了といった各タスクの進行状況を示す値を持ち，担当者によって随時更新されている．

このデータが持つ階層としては，ルートが研究室全体に当たり，その子ノードにプロジェクト，その更に子ノードに担当するチームが存在する構造とした．リーフとなる担当チームは，該当するチケットの総数とその状況の値を有する．

なお，提案手法を適用させた際，リーフの矩形面積は，各チームが請け負うチケットの総数を示す．リーフの矩形内に描くチャートはチケットのステータスを示す棒グラフとし，横軸

をステータスの種類，縦軸をその数とする．各チケットの状況を示す値は，圧倒的に “終了” を示すものが多く，その他の値と比べて極端に変化する．

7.2　都道府県別降水量データの場合

矩形のレイアウトに提案手法を使用し，各リーフ矩形内に棒グラフを描画した．その結果を図 7.1 に示す．

矩形の幅が均一であるため，棒グラフの各棒の幅も均一である．そのため，棒の長さを比較することにより，リーフ矩形内の棒グラフ同士も容易に比較できる．

また，縦軸の目盛間隔は，棒グラフで表される月別降水量の総和と対応している．そのため，棒グラフと矩形の高さの対応にもあまり違和感はない．

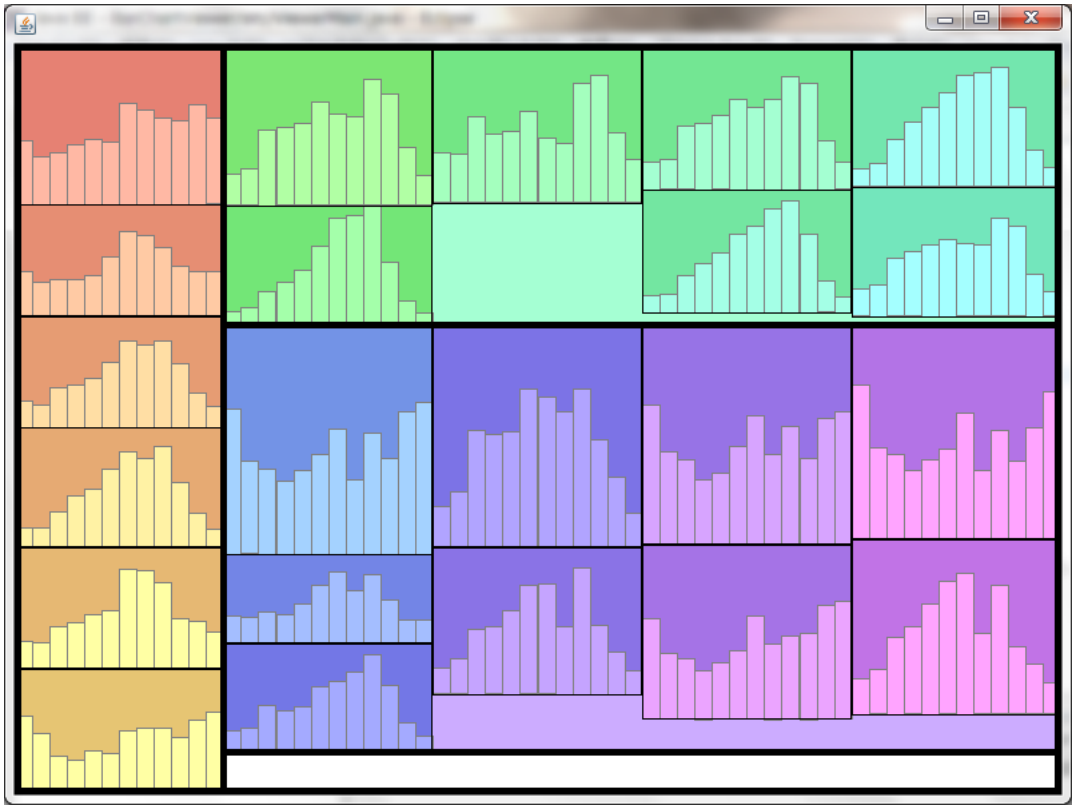

図 7.1: 都道府県別降水量データの提案手法への適用

7.3　チケットデータの場合

矩形のレイアウトに提案手法を使用し，各リーフ矩形内に棒グラフを描画した．その結果を図 7.2 に示す．

各リーフ矩形の幅は統一されているため，棒グラフの目盛間隔も均一である．リーフ同士の親子関係などに関わらず，棒グラフ同士を見たままのスケールで単純に比較することができる．

棒グラフ内の最大値が大きいもの程，矩形高さも大きくなっているため，棒グラフが潰れているということもない．

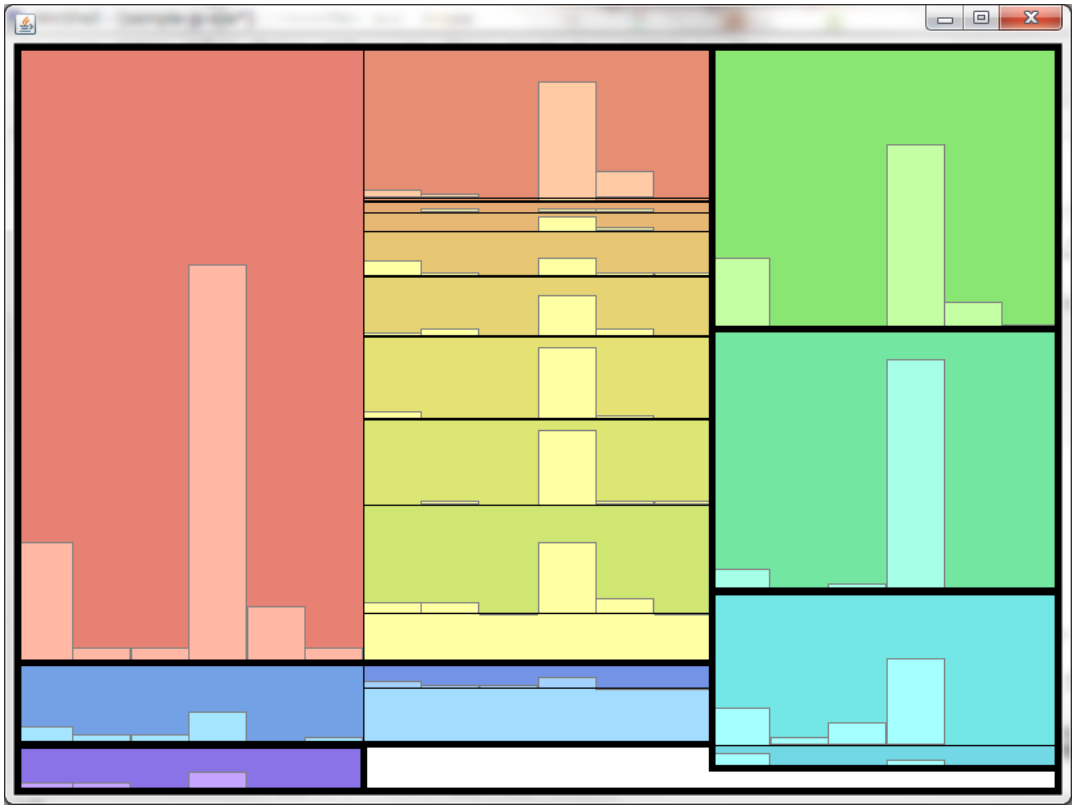

図 7.2: チケットデータの提案手法への適用

第8章　手法の評価

8.1　表現の評価

表現の評価として，ユースケースで使用した2つのデータに対し，新たに2つの表現手法を適用させ，比較を行なった．今回採用した表現手法は，提案手法に加え，Treemap，Squarified Treemap の3つである．Treemap は，予め決められた矩形領域を階層毎に分割方向を変えつつ，その矩形を分割していく手法である．Squarified Treemap は，予め決められた矩形領域を各矩形が可能な限り正方形に近づくように分割していく手法である．

また，上記の手法によりレイアウトされたチャートの各リーフ矩形内に棒グラフを描画した．矩形内に描くチャート（棒グラフ）の縦軸は，矩形の高さと矩形内チャートの各最大値の関係により決定し，チャート全体で統一した．横軸は，矩形の幅をそのデータ数で当分することにより矩形内チャート毎に決定した．

8.1.1　都道府県別降水量データの場合

Treemap を用いた場合

矩形のレイアウトに Treemap を使用し，各リーフ矩形内に棒グラフを描画した．その結果を図 8.1 に示す．

親ノードが同じリーフ同士の比較に関しては，その幅が等しいため，棒グラフ同士を比較することは容易い．一方，親が異なるリーフ同士に関しては，その幅は異なり，月毎の降水量よりも棒の面積に注目しやすい．

Squarified Treemap を用いた場合

矩形のレイアウトに Squarified Treemap を使用し，各リーフ矩形内に棒グラフを描画した．その結果を図 8.2 に示す．

矩形の幅が様々であり，それに伴い，棒の幅も様々である．そのため，やはり棒の面積に注目してしまい，棒グラフ同士の比較という観点では難しい．

また，縦軸の目盛間隔は統一して描画されているが，これにより縦長の矩形内に棒グラフを描いたものは，矩形上部がかなり余るような形になっている．つまり，棒グラフ全体が潰れたような形となっており，棒グラフ内での差が見えにくくなっている．

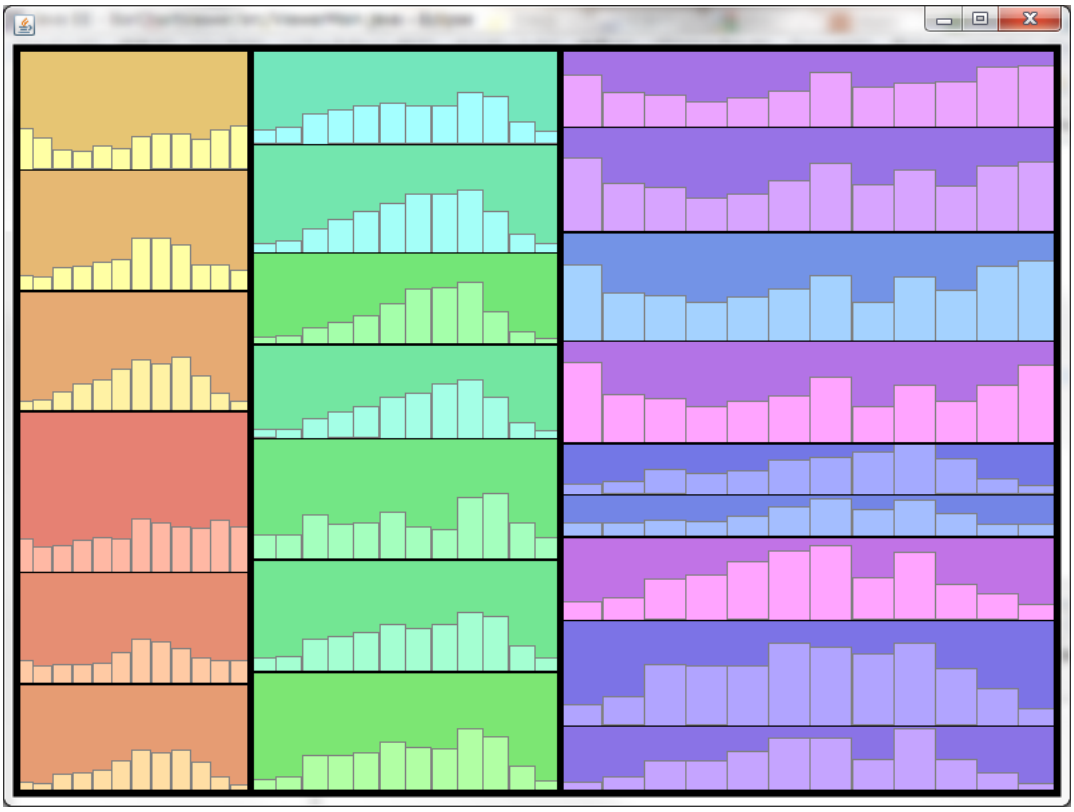

図 8.1: 都道府県別降水量データの Treemap への適用

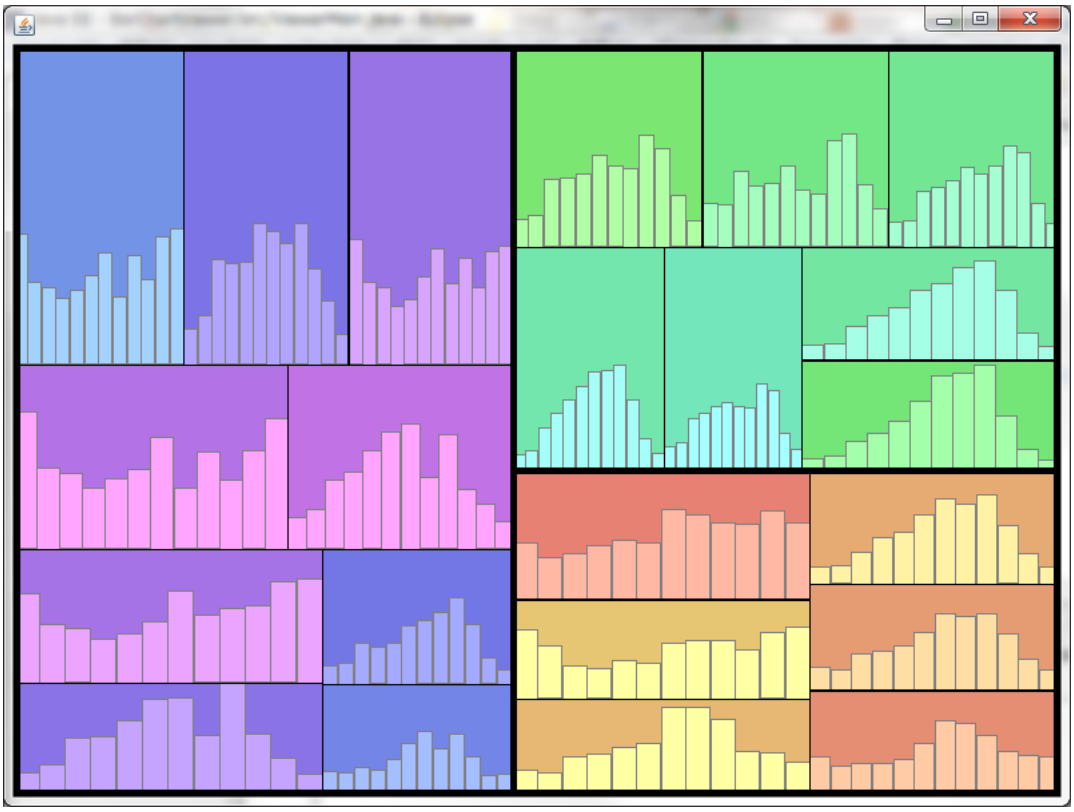

図 8.2: 都道府県別降水量データの Squarified Treemap への適用

3 つの手法の比較

上記 2 つの手法と提案手法の表現を比較すると，提案手法が最も適していると考えられる．

棒グラフは，本来棒の上辺の高さを比べるものである．しかし，Treemap および Squarified Treemap では，リーフの矩形幅が様々であるために，棒グラフを棒ではなく，面として認識し，その面積に注目しやすい．

Squarified Treemap では，一部の棒グラフが潰れるような形となってしまった．一方，提案手法では，各矩形の幅は等しく，棒グラフの各棒の幅も等しい．そのため，棒の高さ，というよりは長さでの比較とはなるが，異なるリーフ間での比較も他の 2 つと比べて容易である．また，矩形の高さは，棒グラフで表現される月別降水量の総和と対応しているため，棒グラフの高さにある程度合わせて伸縮されており，棒グラフが潰れてしまう問題も生じていない．

8.1.2 チケットデータの場合

Treemap を用いた場合

矩形のレイアウトに Treemap を使用し，各リーフ矩形内に棒グラフを描画した．その結果を図 8.3 に示す．

ある程度の幅を持ったリーフ矩形に関しては，降水量データの時と同様に，親ノードが同じリーフ同士の矩形内に描かれた棒グラフを比較することは容易であった．しかし，右寄りの幾つかの幅の小さいリーフ矩形に関しては，幅が小さすぎるためか，棒グラフの認識も難しい．親ノードが異なるリーフ同士の比較に関しても，幅が異なる．そのため，比較対象を棒の高さではなく面積として捉えやすく，正しくその値を比較することは難しい．

また，リーフ矩形の高さが棒グラフのデータに対応していない．そのため，特に縦長の矩形内に描かれた棒グラフは，棒グラフ全体が潰れたような形となっている．これが起因し，棒グラフ内での有意差を見出すことも難しい．

Squarified Treemap を用いた場合

矩形のレイアウトに Squarified Treemap を使用し，各リーフ矩形内に棒グラフを描画した．その結果を図 8.4 に示す．

降水量データの時と同様に，各リーフ矩形の幅が異なるため，それぞれに描かれた棒グラフ同士を比較しようとすると棒の高さではなく，面積で捉えやすい．そのため，この比較が正しく行われない可能性が十分に考えられ得る．

大半の棒グラフは，各棒グラフの中で最大値を取る棒の高さがリーフ矩形の半分の高さにも到達していない．縦軸の目盛間隔を統一するにあたり，大半の棒グラフが潰れた形となっており，そのリーフ矩形を生かしきれていない．

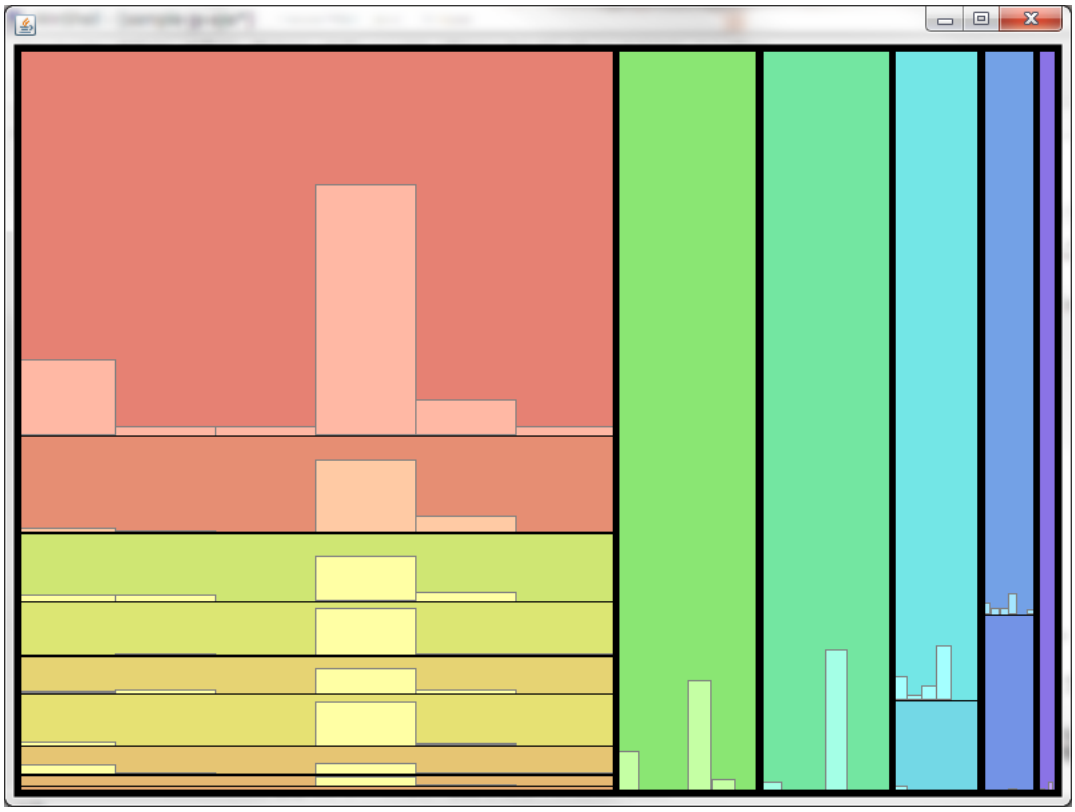

図 8.3: チケットデータの Treemap への適用

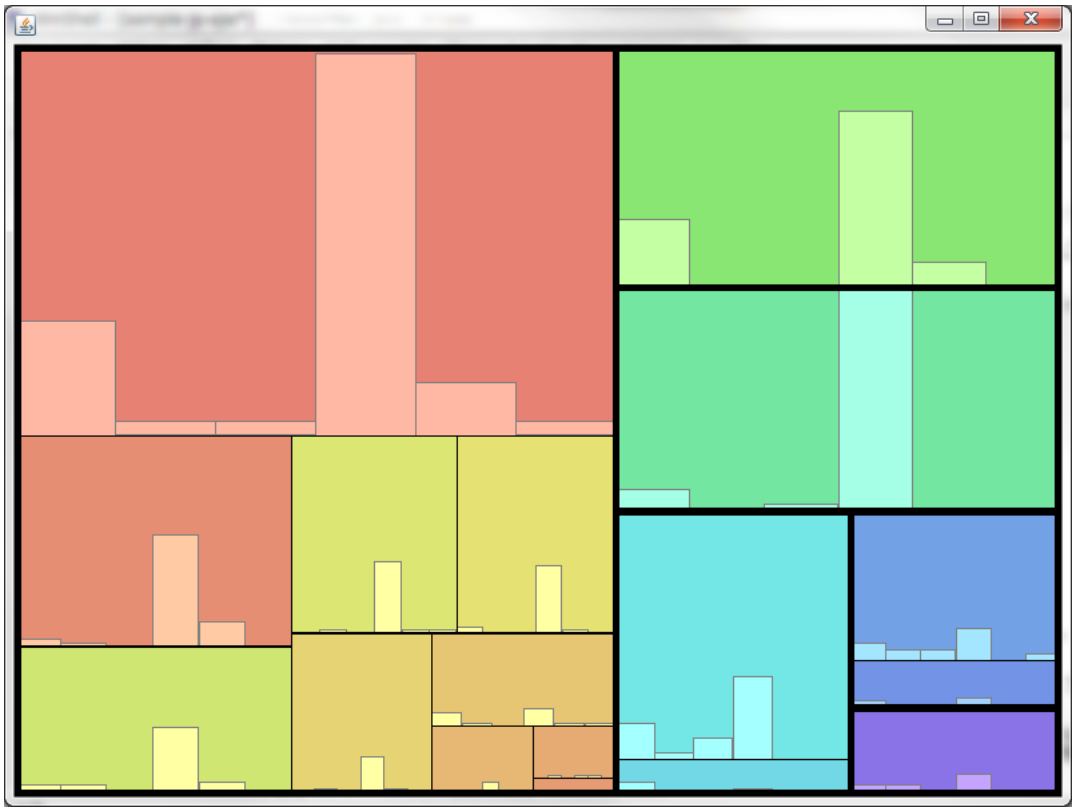

図 8.4: チケットデータの Squarified Treemap への適用

3 つの手法の比較

上記 2 つの手法と提案手法の表現を比較すると，チケットデータにおいてもやはり提案手法が最も適していると考えられる．

まず，矩形幅が異なることによる棒グラフ同士の比較の困難さについては，降水量データの時と同じである．そのため，これに関しては提案手法が適している．

今回，Treemap および Squarified Treemap 共に棒グラフがそのリーフ矩形の高さを生かしきれない状況が生じた．これは，棒グラフ内に描くデータが 1 つの値のみ極端に大きい値をとるためであると考えられる．リーフの重みが面積に対応している場合，その矩形が横長になるものがある．これが極端に横長であると，その棒グラフは潰れたような形になる．縦軸の目盛間隔はチャート全体を通して統一しているため，他の棒グラフも潰れたような形になる．これに比べ，提案手法では，リーフの重みを矩形高さにのみ対応させている．そのため，棒グラフの形状もある程度維持しつつ，その矩形高さを活かせたと考えられる．

8.1.3 考察

以上，2 つのデータを 3 つの手法に適用させ，比較を行なった．その結果，何れのデータにおいても，矩形内に描く棒グラフの比較しやすさ，という観点では提案手法が最も適しているとわかった．

1 つ目の理由として，棒グラフ同士を見たままのスケールで高さとして比較できる点が挙げられる．これは，矩形内に描く棒グラフの棒の幅がチャート全体で統一されていることによる．幅が統一されていない場合，棒の高さではなく棒の面積に注目しやすい．そのため，正しい比較を行うことが難しいと考えられる．

2 つ目の理由として，矩形の高さと棒グラフの最大値のバランスが良い点が挙げられる．これは，リーフの重みが矩形高さに対応していることによる．従来の Treemap はリーフの重みが面積に対応していた．そのため，棒グラフとして描くデータの最大値にかかわらず，その高さが決定される．これにより，縦軸の目盛間隔を統一するに当たり，チャート全体として見ると棒グラフが矩形の高さに対して小さく，潰れたような形となった，と考えられる．

8.2 アルゴリズムの評価

手法を評価するにあたり，都道府県の降水量データを使用した．ルートの子として，東北地方，関東地方，中部地方の地方が存在する．各地方は都道府県を子として持つ．この階層構造では，都道府県がリーフであり，リーフは月別降水量のデータを有する．また，リーフの持つ重みは年間降水量とする．

8.2.1　充填率とアスペクト比の関係

提案手法では，NF 法が矩形の入力順序に依存するレイアウトアルゴリズムであるため，入力順序を変更し，繰り返し再レイアウトを行うことにより，より良いレイアウト結果を得ようとする．この時に繰り返し計算される再レイアウトの結果から，各レイアウト結果のチャート全体の充填率とリーフ矩形のアスペクト比平均を計算した．これらの関係を図 8.6 に散布図として示す．横軸をチャート全体の充填率とし，縦軸をリーフ矩形のアスペクト比平均とする．なお，何れも最大値は 1 である．

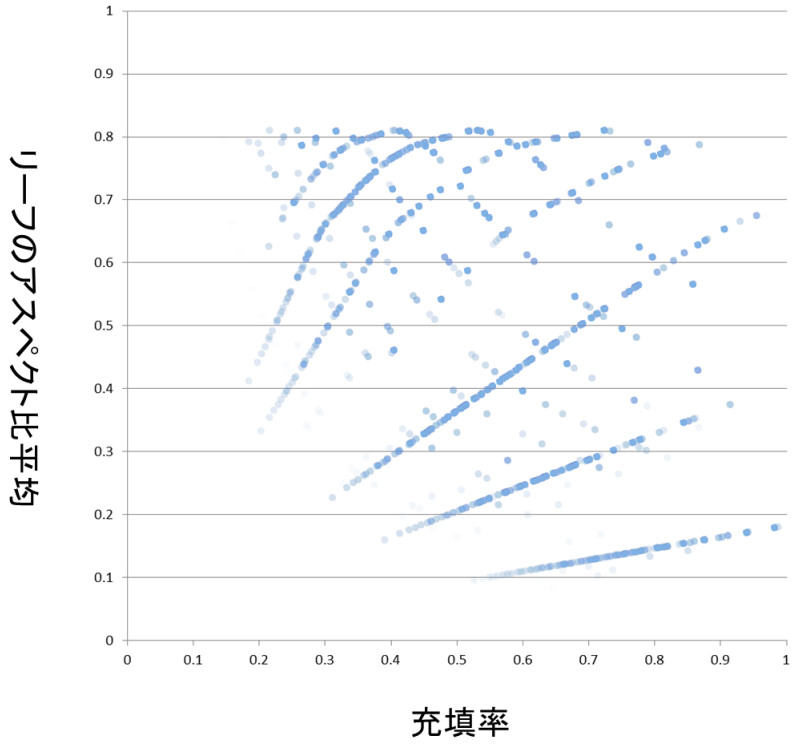

図 8.5: 充填率とリーフのアスペクト比平均

図 8.5 の散布図から，データがいくつかの系統に分かれていることがわかる．各系統は正規分布のような 1 箇所のみに極大値が存在する曲線を描いている．ここで，各軸共に 1 が最大値であり，また最良の値でもある．そのため，各系統の極大値，あるいは充填率の最も高い値をとる点の中に最良とされるレイアウト結果が存在すると考えられる．この中には，充填率は 0.8 を超え，アスペクト比平均は 0.8 に近い点や多少アスペクト比平均は落ちるが充填率が 0.9 を超える点も存在する．そのため，レイアウト結果の中には充填率とアスペクト比共に良いとされるものも存在すると考えられる．

各軸共に最良の値をとる点が存在するのであれば，それが最良のレイアウト結果であるといえる．しかし，図 8.5 からわかるように，各軸で最良の値を取る点は異なる．矩形を同じ方

向に並べた場合，充填率は最良となるが，アスペクト比平均が最良とはならないことからも，最良の値を取る点がこの点が軸ごとに異なることがわかる．

8.2.2　高速化による改善

処理速度の比較

処理速度を向上させるため，NF 法だけでなく，積み上げ法も用いるアルゴリズムへと改良を行なった．ここでは，高速化が図れたか否かを，両アルゴリズムを比較し，述べる．

まず，それぞれのアルゴリズムにおける処理時間を表に示す．今回の計測は，2.67GHz の CPU，6GB のメモリを積んだコンピュータで行なった．

表 8.1: 表の例

	NF 法のみ	NF 法＋積み上げ法
処理時間	128821277ms （約 35.8h）	3060ms

表からわかるように，アルゴリズムを改良したことによりその処理時間は格段に短くなっている．これは，入力順序への懸念からあらゆる入力順序でのレイアウトを試すこととしていた当初のアルゴリズムと比べ，改良後のアルゴリズムはリーフ矩形の詰め込みを積み上げ法で行うことにより，入力順序を気にする必要がなくなる．そのため，入力順序を変更しての再レイアウトが不要となり，今回の処理時間の短縮につながったと考えられる．

レイアウト結果の比較

実際に高速化がなされていたことは先に述べた．しかし，それに伴い，当初から重要視していたチャート全体の充填率やリーフ矩形のアスペクト比平均に何らかの影響が及ぶ可能性がある．この点についても両アルゴリズムを比較していく．

まず，図 8.5 に示した充填率とリーフ矩形のアスペクト比平均の関係を示すチャートに，改良後のアルゴリズムによる同様のチャートを重ねたものを図 8.6 に示す．上図のチャートは改良前のアルゴリズムによるものであり，図 8.6 のチャート内で青色の点で示す．改良後のアルゴリズムによるものは図 8.6 のチャート内で赤色で示す．なお，各軸は図 8.5 と同様に，横軸をチャート全体の充填率，縦軸をリーフ矩形のアスペクト比平均とし，いずれの値も 1 が最大値であり最良値とする．

図 8.6 からわかるように，改良後のアルゴリズムによる実行結果は，おおよそ改良前のものと一致する．繰り返しレイアウト処理を行なった回数が少ないため，データ数も明らかに少なく，完全一致とはいかない．しかし，充填率あるいはリーフ矩形のアスペクト比平均の値が良い点はあまり抜け落ちることなく，結果として出力されている．そのため，最良となる

点は落としてしまう可能性があるが，その値に近く，結果にあまり影響のない範囲の点は導出できると考えられる．

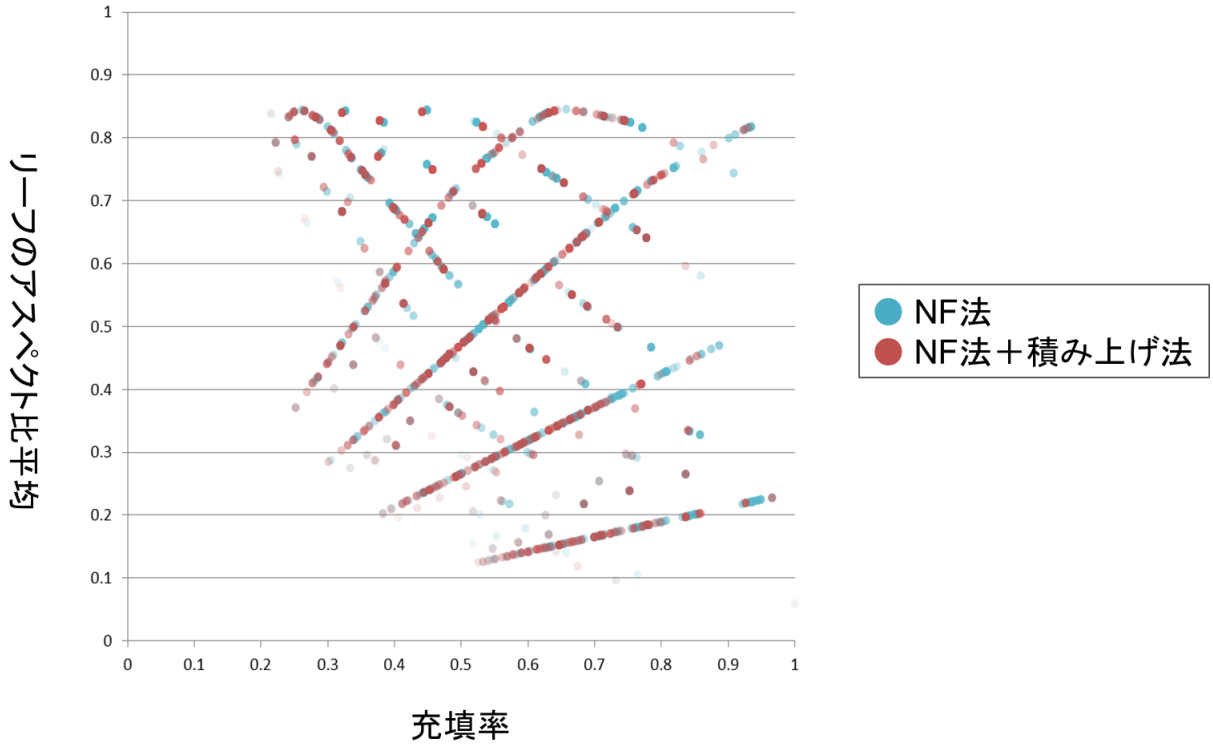

図 8.6: 2 つのアルゴリズム比較

また，両アルゴリズムでのレイアウト結果を下図に示す．なお，下図に示すレイアウト結果は充填率とアスペクト比平均にかける重みは同じものとした．今回はレイアウトに関する評価のため，彩色はランダムとした．

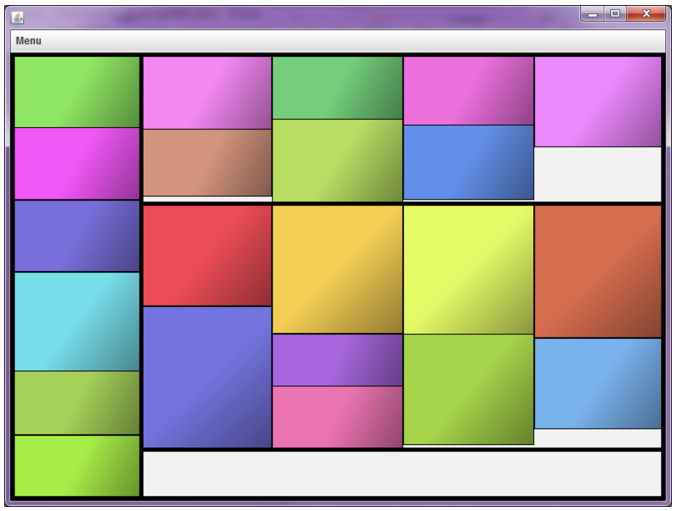

図 8.7: NF 法での描画結果

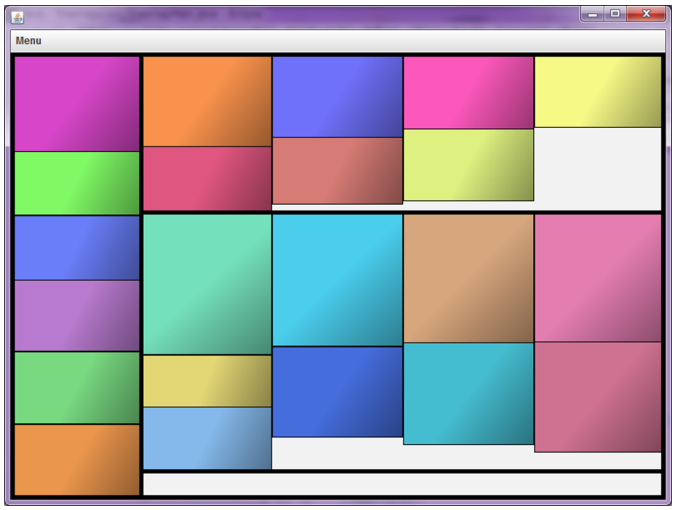

図 8.8: NF 法+積み上げ法での描画結果

第9章　おわりに

本研究では，階層構造における表現手法の 1 つである Treemap を基にし，リーフの矩形幅が均一な表現を開発した．この表現を使用することにより，リーフ矩形内に描かれたチャート同士を容易に比較することが可能となる．

従来の Treemap では，リーフ矩形の面積がその重さに対応している．しかし，リーフの矩形内に新たなチャートを描く際，各リーフ矩形内のチャートの軸の目盛間隔を統一することに適さない．そのため，各リーフ矩形内に描かれたチャート同士を比較することは難しい．本研究では，チャート同士の比較のしやすさに焦点を当て，リーフの矩形幅が均一な表現を提案し，アルゴリズムを開発した．

開発したアルゴリズムを用い，2 つのデータに対してリーフの矩形幅が均一なレイアウトを導出し，リーフ矩形内にチャートを描いた．また，同じデータに対し Treemap および Squarified Treemap のレイアウトを適用させ，リーフ矩形内にチャートを描いた．手法の評価として，これら 3 つの表現をチャート同士の比較の容易さ，という観点で比較を行なった．この結果からも，提案手法はリーフ矩形の形状を生かしながら，その中に新たなチャートを描くことが可能であるとわかった．

本研究では，矩形の詰め込み過程を “ストリップパッキング問題” に当てはめ，NF 法を用いることにより解決しようとした．しかし，その他にも解法は存在するため，今後，他の解法にも適用させてみたいと考えている．

以上，本研究では Treemap のリーフ矩形内にチャートを描く表現に着目した．特にリーフ矩形内のチャート同士の比較しやすさ，という観点に焦点を当て，これを改良する手法を開発した．その結果，Treemap や Treemap から派生したレイアウト手法よりも，チャート同士を比較しやすい，という結果が得られた．

謝辞

本研究を行うに当たり，指導教官である三末和男先生，志築文太郎先生，田中二郎先生をはじめ，高橋伸先生，Simona Vasilache 先生には，熱心かつ丁寧なご指導を賜りました．深く感謝致します．また，インタラクティブプログラミング研究室の皆様には，ゼミや日常生活の中で多くの貴重なご意見をいただきました．特に，NAIS チームの皆様には，研究生活全体を通して様々なご意見，ご指摘をいただきました．心より感謝しております．最後に，両親や友人をはじめ，大学生活でお世話になったすべての方に深く感謝申し上げます．本当にありがとうございました．

参考文献

[1] Johnson, B. and Shneiderman, B.“Tree-maps: A Space-filling Approach to the Visualization of Hierarchical Information Structures.” Proceedings of the IEEE Visualization ’91, pp.284-291, 1991.

[2] Dayal, U., Hao, M., Keim, D., and Schreck, T. “Importance driven visualization layouts for large time-series data.” In Proceedings of the IEEE Symposium on Information Visualization (InfoVis), IEEE Computer Society, pp.203-210, 2005.

[3] Bruls, D.M., C. Huizing, J.J. van Wijk.“Squarified Treemaps”. In: W. de Leeuw, R. van Liere (eds.), Data Visualization 2000, Proceedings of the joint Eurographics and IEEE TCVG Symposium on Visualization, pp.33-42, 2000.

[4] W. Wang, H. Wang, G. Dai, and H. Wang. “Visualization of large hierarchical data by circle packing.” In Proc. of ACM SIGCHI 2006, pp.517-520, 2006.

[5] B. Shneiderman and M. Wattenberg. “Ordered treemap layouts.” IEEE Transactions on Visualization and Computer Graphics, pp.73-78, 2001.

[6] J. Wood and J. Dykes. “Spatially ordered treemaps.” Visualization and Computer Graphics, IEEE Transactions on Visualization and Computer Graphics, , pp.1348-1355, 2008.

[7] Wattenberg, M.“Visualizing the Stock Market,”Proceedings of ACM CHI 99, Extended Abstracts, pp.188-189, 1999.

[8] 伊藤貴之, 小山田耕二, “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,” 可視化情報学会第 9 回ビジュアリゼーションカンファレンス, 2003.

[9] 今堀 慎治, 梅谷 俊治: “切出し・詰込み問題とその応用 － (2) 長方形詰込み問題－, オペレーションズ・リサーチ,” pp.335-340, 2005.

[10] E.G. Coffman, Jr., M.R. Garey, D.S. Jhonson: “Appears in Approximation Algorithms for NP-Hard Problems,” PWS Publishing, Boston, pp.46-93, 1996

[11] B.S. Baker, E.G. Coffman Jr. and R.L. Rivest: “Orthogonal packing in two dimensions,” SIAM Journal on Computing, pp.846-855, 1980.

[12] D.S. Johnson: “Near-optimal bin packing algorithms,” Ph.D.Thesis, MIT, Cambridge, MA, 1973.