NAIST-IS-DT9661023

博士論文

グラフの描画問題の計算複雑度に関する研究

林 邦彦

1999年2月8日

奈良先端科学技術大学院大学
情報科学研究科 情報処理学専攻

本論文は奈良先端科学技術大学院大学情報科学研究科に
博士 (工学) 授与の要件として提出した博士論文である。

林 邦彦

審査委員：　藤原 秀雄 教授
伊藤 実 教授
高橋 豊 教授
増澤 利光 助教授

グラフの描画問題の計算複雑度に関する研究*

林 邦彦

内容梗概

グラフは構造や関係を表現するための基礎的なモデルであり，データ構造や問題の記述など，計算機科学のさまざまな分野で用いられる．グラフを視覚化すること，つまり平面上に描画することは，その構造を理解するために有効な手段であるが，人手によるグラフの描画はグラフの規模の増大に伴って困難となるため，グラフを自動的に描画する手法が必要とされている．

本論文では，いくつかのグラフ族に対する描画問題を定式化し，その問題に対する計算複雑度およびアルゴリズムについて考察する．第1章では，本研究の目的と意義および背景について述べる．第2章では，根付き順序木の描画問題の一つとして，木構造図の描画問題について考察する．木構造図とは，節点を矩形で与えられる根付き順序木で，プログラム構造などの階層構造の表現のために幅広く用いられている．木構造図の描画手法の一つとして，海野らによって $O(n^3)$ 時間の描画アルゴリズムが提案されており，ここでは海野らと同じ描画を得るより効率的な時間計算量 $O(n\alpha(n,n))$(α はアッカーマン逆関数) のアルゴリズムを提案する．第3章では，根付き非順序木の描画問題について考察する．根付き非順序木は，根付き木のうち子の順序が指定されていないもので，この木に対する描画問題は過去の研究では扱われていなかった．ここではまず，描画に必要な美的制約を与え，その制約の下である幅以下の描画が存在するかどうかを判定する問題を定式化する．整数座標系，実数座標系のいずれの場合についても，この問題が NP 完全 であることを示す．更に，整数座標系の場合には，各節点の子の数を3以下に制限した場合についても NP 完全であることを示す．第4章では，一般

*奈良先端科学技術大学院大学 情報科学研究科 情報処理学専攻 博士論文，NAIST-IS-DT9661023, 1999年2月8日.

グラフの動的描画問題を取り上げる．動的描画とは，予め描画されたグラフに対してある編集操作が加えられた後，与えられた美的制約を満たしつつ節点の相対的な位置関係が大幅に変わらないように再描画を行うことである．ここではグラフの辺の接続関係を考慮しない場合を考える．節点を矩形と置き換えてグラフの再描画問題を矩形の再配置問題として取り扱う．予め配置されている矩形の集合に対して，直交順序を保存し，矩形同士が交差しない，という制約の下で，面積最小の再配置を求める問題について考察する．三末らはこの問題に対して $O(n^2)$ 時間の発見的アルゴリズムを提案している．ここではまず，ある面積以下で再配置可能かどうかを判定する問題が，整数座標系で NP 完全，実数座標系で NP 困難であることを示す．更に，面積最小の再配置を求める $O(n^2)$ 時間の発見的アルゴリズムを与え，このアルゴリズムで得られる再配置面積が三末らの手法による面積以下であることを証明する．また，ランダムな初期配置に対して計算機実験を行い，特に矩形数が多い場合に，三末らの結果に比べて 15% ～ 20% の面積で再配置できることを示す．最後に第 5 章で，以上の研究成果の結論を述べるとともに，今後の研究課題について述べる．

キーワード

グラフ，描画，制約，アルゴリズム, 計算量

Studies on Computational Complexity of Graph Drawing Problems*

Kunihiko Hayashi

Abstract

Graphs are a fundamental model to represent several structures and relations, which are used to describe data structures or problems. To visualize graphs, that is, to draw them on a plane is an effective method for understanding their structure. But it is very difficult to draw a graph manually as its size increases. Therefore, it is important to draw graphs automatically.

There are several classes for graphs. This dissertation considers interesting problems for the classes of graphs, and we discuss complexities and algorithms for these problems. In Chapter 1, we present motivation and background of the study. In Chapter 2, we discuss a drawing problem for rooted ordered trees, called *tree-structured diagrams*. A tree-structured diagram has vertices shaped as rectangles, and it has been widely used to represent hierarchical structure. As a drawing method of tree-structured diagrams, Unno et al. proposed an $O(n^3)$-time drawing algorithm. We improve the algorithm to obtain the same drawings as Unno's with time complexity $O(n\alpha(n,n))$ (α is a functional inverse of Ackermann's function). In Chapter 3, we discuss a drawing problem for rooted unordered trees. In an rooted unordered tree, the order of all children isn't specified. None of drawing problems of a rooted unordered tree has been considered. We first define a decision problem for finding a drawing under given aesthetics

*Doctor's Thesis, Department of Information Processing, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DT9661023, February 8, 1999.

whose width is less than a given integer, and we show that this problem is NP-complete in the cases of integer coordinate and real coordinate. In the case of integer coordinate, we also show that the problem is NP-complete even if the number of children of each vertex is restricted to be less than or equal to three. In Chapter 4, we consider a dynamic drawing problem of general graphs. For simplicity, we take no account of relationship of edge connections in graphs, and we treat a graph redrawing problem as a rectangle re-layout problem. For a given set of rectangles on a plane, we consider a problem of finding the minimum area layout of the rectangles that avoids intersections of the rectangles and preserves the orthogonal order. As a previous result, Misue et al. proposed an $O(n^2)$-time heuristic algorithm for the problem. We first show that the corresponding decision problem is NP-complete. We present an $O(n^2)$-time heuristic algorithm that gives a minimum area layout, and we prove that the area of our layout is same or smaller than Misue's. Furthermore, we show from experimental results that the layout area of our method is much smaller than Misue's, especially it is $15\% \sim 20\%$ of Misue's in the case of a large number of rectangles. In Chapter 5, we conclude the dissertation.

Keywords:

graph, drawing, aesthetics, algorithm, complexity

関連発表一覧

- 学術論文誌

  1. 林 邦彦, 増田 澄男, 田中 榮一, "木構造図の描写アルゴリズムの効率化," 電子情報通信学会論文誌 (A), Vol. J79-A, No. 3, pp. 669–679, 1996.
  2. 林 邦彦, 増田 澄男, "根付き非順序木の最小幅描写問題の計算複雑度," 電子情報通信学会論文誌 (A), Vol. J79-A, No. 11, pp. 1877–1885, 1996.
  3. 林 邦彦, 井上 美智子, 増澤 利光, 藤原 秀雄, "直交順序を保存する矩形の非交差配置問題," 電子情報通信学会論文誌 (D–I), (採録決定).

- 国際会議 (査読付き)

  1. K. Hayashi, M. Inoue, T. Masuzawa, H. Fujiwara, "A layout adjustment problem preserving orthogonal order," Proceedings of Sixth Symposium on Graph Drawing '98 (Lecture Notes in Computer Science 1574), pp. 183–197, 1999.

- 研究会報告

  1. 林 邦彦, 増田 澄男, 田中 榮一, "木の描写アルゴリズムの効率化," 電子情報通信学会技術報告, COMP94-56, pp. 21–30, 1994.
  2. 林 邦彦, 増田 澄男, "根付き非順序木の最小幅描写問題," 電気関係学会関西支部連合大会, G295, 1995.
  3. 林 邦彦, 井上 美智子, 増澤 利光, 藤原 秀雄, "直交順序を保存する矩形の非交差配置問題について," 電子情報通信学会技術報告, COMP97-65, pp. 41–48, 1997.

目 次

1 序論 **1**

1.1. グラフ描画問題 1

1.2. グラフの描画法 1

1.2.1 木 1

1.2.2 有向グラフ 2

1.2.3 無向グラフ 3

1.3. 根付き木の描画問題 4

1.3.1 木構造図の描画問題 4

1.3.2 根付き非順序木の描画問題 5

1.4. 動的描画法 5

1.4.1 矩形の再配置問題 7

1.5. 本論文の構成 8

2 木構造図の描画問題 **9**

2.1. 諸定義 9

2.2. 平行移動による描画 12

2.3. 相対座標と輪郭 14

2.3.1 相対座標 15

2.3.2 輪郭 16

2.3.3 節点および部分木の移動距離 18

2.4. 再描画の高速化 20

2.4.1 アルゴリズムの概略 20

2.4.2 移動の検出 20

2.4.3 相対座標および移動可能距離の計算 23

2.4.4 輪郭の合成 25

2.4.5 時間計算量 27

2.4.6 葉の $width_x$ が任意の場合 28

3 根付き非順序木の最小幅描画問題 **29**
3.1. 諸定義 29
3.2. 最小幅描画問題 30
3.3. 各節点の子の数に制限がない場合の NP 完全性 32
3.4. 各節点の子の数が 3 以下の場合の NP 完全性 37
3.4.1 3-SAT から問題 IUTD(D1) への変換 37
3.4.2 証明 44

4 矩形の最小面積非交差再配置問題 **49**
4.1. 諸定義 49
4.2. 矩形の最小面積非交差再配置問題 50
4.3. 矩形の最小面積非交差再配置問題の計算複雑度 51
4.3.1 3-SAT から問題 ILADR への帰着 51
4.3.2 証明 56
4.3.3 実数座標系における再配置問題 59
4.4. 再配置アルゴリズム 61
4.4.1 Push Force-Scan アルゴリズム 61
4.4.2 Push Force-Scan アルゴリズムの改良 64
4.4.3 アルゴリズムの有効性 66
4.4.4 配置例と実験結果 69

5 結論 **73**

謝辞 **75**

参考文献 **77**

図目次

1.1 動的描画の例 6

2.1 木の描画の例 10
2.2 平行移動による描画例 13
2.3 輪郭 $U(\overline{T}(w_k) \cup T(w_k))$ 17
2.4 節点と部分木の移動の例 19
2.5 手続き **SubOpt'** 21
2.6 輪郭の合成の例 26
2.7 葉の分割 28

3.1 WL_x と WR_x の例 30
3.2 T^iを構成する二つの木 34
3.3 問題 HPP から IUTD(D1) への変換例 35
3.4 $LT(y_{i,j})$ の構成 38
3.5 $VT(x_k)$ の構成 39
3.6 $VT(x_k)$ の二つの描画 39
3.7 $LT(y_{i,j})$ の二つの描画 ($y_{i,j} = x_k$の場合) 40
3.8 $LT(y_{i,j})$ の二つの描画 ($y_{i,j} = \overline{x_k}$の場合) 40
3.9 $LT(y_{i,j})$ の不当な描画の例 41
3.10 $CT'(F_i)$ の構成 41
3.11 $CT(F_i)$ の構成 42
3.12 $T(E)$ の構成 43
3.13 補題 3.3 の証明の説明図 45

4.1 $R(E)$ の初期配置$\pi_{R(E)}$の概略 51
4.2 R_i''および R'の配置 52
4.3 $VR(x_k)$ の配置 53
4.4 $LR(y_{i,j})$ の配置 ($y_{i,j} = x_k$の場合) 53
4.5 $LR(y_{i,j})$ の配置 ($y_{i,j} = \overline{x_k}$の場合) 54

4.6 $CR(F_i)$ の初期配置 55
4.7 R' と $CR(F_i)$ の配置 55
4.8 $R(E)$ の配置例 58
4.9 RLADR における $LR(y_{i,j})$ および $LR'(y_{i,j})$ の配置 ($y_{i,j} = x_k$の場合) 60
4.10 RLADR における $LR(y_{i,j})$ および $LR'(y_{i,j})$ の配置 ($y_{i,j} = \overline{x_k}$の場合) 60
4.11 アルゴリズム Horizontal-PFS 63
4.12 すき間が生じる例 64
4.13 アルゴリズム Horizontal-PFS′ 65
4.14 アルゴリズムの実行例 ($n = 30$) 70
4.15 実験結果 71

表目次

2.1 節点と部分木の移動の例のパラメータ値 19
2.2 輪郭の合成の例のパラメータ値 27

3.1 制約集合 D1 ～ D8 32

第 1 章 序論

1.1. グラフ描画問題

グラフは構造や関係を表現するための基礎的なモデルであり，データ構造や問題の記述など，計算機科学のさまざまな分野で用いられる．グラフを視覚化すること，つまり平面上に描画することは，その構造を理解するために有効な手段であるが，人手によるグラフの描画はグラフの規模の増大に伴って困難となるため，グラフを自動的に描画する手法が必要とされている．

グラフを自動的に描画する研究として，さまざまな手法が提案されている [3, 4, 5, 6, 19, 26, 32, 38, 39]．これらの描画法では，グラフを美しく描くために “美的制約 (aesthetic)” と呼ばれるいくつかの描画規則を設けている．また，グラフを紙や CRT の上に表示する際，描画領域に通常限界があることを考慮して，美的制約を満たす範囲で描画領域の小さな描画を得ることを目的としている場合が多い．

本論文では，いくつかのグラフのクラスに対して描画問題を定義し，問題の計算複雑度について考察する．

1.2. グラフの描画法

グラフにはさまざまなクラスがあり，各クラスごとに多くの描画法が提案されている．ここでは，各クラスごとの描画法について簡単に紹介する．

1.2.1 木

グラフの中で根付き木は最も基本的なものであり，また階層的構造の表現のためによく用いられる．そのため，根付き木の描画アルゴリズムについて数多くの研究がある．

木を描画するときには，通常，いくつかの制約の下で幅の小さな描画を得ることを目的とする．Wetherell，Shannon[43] は次の制約を導入し，その制約の下で

2 分木を描画する線形時間のアルゴリズムを提案した.

- 同じレベルにある節点は同一水平線上に置く (根は最も上の水平線に置く).
- 左の子は親の左側に，右の子は親の右側に置く.
- 親はその子たちの中央に置く.

更に，Reingold，Tilford[27] は二つの制約

- 同型な部分木は (平行移動を許す範囲で) 同じ描画とする.
- 左右反転した二つの部分木の描画は鏡像となるようにする.

を付け加え，“部分木の輪郭” を導入することによって，これらの制約の下での線形時間アルゴリズムを提案した．また，Supowit，Reingold[36] は上の五つの制約を満たす最小幅の描画を得る問題について考察し，その問題が実数座標系で多項式時間で解けることを，整数座標系では (対応する判定問題が)NP 完全であることを示した．土田 [41] は多分木の描画について，同様の問題の NP 完全性を示している.

1.2.2 有向グラフ

有向グラフには，閉路を含まない非閉路有向グラフと，閉路を含む一般有向グラフがある．有向グラフは通常，状態遷移図などのようにできるだけ多くの有向辺が同方向になるように描画されることが多く，根付き木と同様に，節点を等間隔に引かれた水平線上に並べて配置する，階層的描画が用いられる．このグラフの描画では，美的制約としては通常，以下の制約が用いられる.

- 交差する辺の数を最小にする.
- 同レベルの隣接する節点を最小距離以上分離する.
- 親は子の重心に置く.
- 閉路を含む場合は，逆方向に描かれる辺の数を最小にする.

一般的にこれらの制約をすべて満たすのは困難であり，交差辺最小化と帰還辺 (逆方向に描かれる辺) 最小化の問題は NP 困難であることが知られている．そのため，さまざまな発見的手法が提案されている．

杉山らは階層的描画法の基本となる STT 法を提案した [34]．この手法は，階層化，正規化，節点の配置順序の決定，節点の座標の決定，の四つのステップからなり，交差辺最小化と帰還辺最小化に対する発見的手法を用いている．この他，STT 法の効率改善 [13] や STT 法に対するブラウザの作成 [28] などが試みられている．

1.2.3 無向グラフ

無向グラフは，平面グラフと非平面グラフとに分けられる．

平面グラフの描画は平面埋め込み問題として，グラフ理論の分野で数多くのアルゴリズムが提案されている．描画を行うに先だって，与えられたグラフが平面グラフであるか否かを判定することが重要であり，Hopcroft らによって線形時間平面性判定アルゴリズムが提案されている [16]．

非平面グラフ (一般無向グラフ と呼ばれる) の描画では，美的制約としては通常

- 対称な部分は対称に描く．
- 辺の交差を最小にする．
- 辺の折れ曲がり数を最小にする．
- 辺の長さを一様にする．
- 節点の分布を一様にする．

などが用いられるが，各制約に対応する最適化問題は一般的に NP 困難であることが多く，これらの制約同士が競合して同時に最適性を満たすことが非常に難しい．そのため，無向グラフにおいても通常発見的手法によるアプローチとなる．代表的なものとして，辺を直線で描く直線描画と辺を格子上に描く直交格子描画がある．

直線描画の代表的なものとして，バネ埋め込み (spring-embedder) 法がある．Eades[9] は，力学的モデルを基礎とした発見的手法を提案した．この手法は，節点をリングに辺をバネに見立てて，リングに対する引力と斥力をすべての力が安定する方向に収束ように計算し，各節点を移動させる，というものである．この手法では非常に効率的に対称的な描画を得ることができる．更にこの変形として，鎌田ら [19, 20, 21, 37] は配置する節点の密度を一様にし，力のエネルギーを最小化するために Newton-Raphson 法を用いた手法を提案した．Fruchterman, Reingold[12] はこれらの手法を一般化してさまざまな力関数を与えて実験を試みている．この他，この手法の改良が数多く行われている [8, 11, 22, 29, 33]．また，角ら [35] は力指向アルゴリズムの性能評価を試みているが，扱う美的制約が競合しているため，理論的評価を行うことは非常に困難である．

直交格子描画では，格子上の交点に節点が配置される．この描画の代表的なものとして，グラフアルゴリズムに基づく発見的手法が提案されている [1, 2, 38]．これらの手法は，グラフの平面化，格子上への節点の配置，描画の最適化，の三つのステップからなる．この中で，交差辺の最小化，折れ曲がり数の最小化，描画面積の最小化などが考慮されている．

1.3. 根付き木の描画問題

本論文ではまず，根付き木の描画問題について考察する．根付き木は最も基本的なグラフの一つで，アプリケーションへの適用も非常に多く見られる．ここでは，根付き順序木と根付き非順序木の 2 種類の描画問題を扱う．特に，根付き順序木の描画問題では，その一つとして木構造図の描画問題を取り上げる．

1.3.1 木構造図の描画問題

1.2.1 節で挙げたようなアルゴリズムは一般の 2 分木若しくは多分木を対象としたものであるが，特に木構造をなすようなプログラム図式を念頭においた研究もある [15, 42]．それらの一つとして海野ら [42] は，(I) まず与えられた木の初期描画を求め，(II) 部分木の平行移動を繰り返し行うことによって幅のより小さな描画を得る，という $O(n^3)$ 時間アルゴリズム (n は木の節点数) を提案している．

このアルゴリズムでは，文献 [27, 36, 43] と異なり，

(i) 根を最も左に置き，各節点はその親より右に置く，

(ii) 各節点をあらかじめ指定された大きさの長方形によって表す．節点の縦方向の幅は任意の値をとってよいが，横方向の幅は葉を除いてすべて 1 である，

ものとしている．

第 2 章では，部分木の平行移動を行う際に輪郭を用いることにより，海野らのアルゴリズムと同じ描画をより高速に求める方法を示す．“ある定数 c が存在して，どの葉の横方向の幅も c を超えない” という仮定は現実的なものであると思われるが，この仮定の下では本方法の時間計算量は $O(n\alpha(n,n))$ である (n は節点数，α はアッカーマン逆関数 [40])．なお，文献 [23, 25, 27] 等のアルゴリズムでも部分木の輪郭を用いているが，描画を決定していく方法が異なるため，ここでは輪郭上の節点に付随させる情報を工夫している．

1.3.2 根付き非順序木の描画問題

根付き木のうち，各節点について子の順序が指定されているものを順序木と呼ぶのに対して，子の順序が指定されていないものを非順序木と呼ぶ．

根付き順序木の描画に関しては多くの研究があるが，根付き非順序木についての結果は見当たらない．第 3 章では，根付き非順序木に対して，根付き順序木で設けられていたものと同様の美的制約を考える．更に，それらの制約の 8 通りの集合を定義し，それぞれのもとである整数値以下の幅の描画が存在するかどうかを判定する問題 (最小幅描画問題) が NP 完全であることを示す．この結果は，根付き順序木の場合と異なり，実数座標系を用いた場合にも成立する．

1.4. 動的描画法

以上紹介したようなグラフの自動描画に関する研究の多くは，グラフの構造のみが与えられ，新たに描画を行うものである．しかし多くのアプリケーションでは，与えられた描画に対して，節点および辺の挿入や削除，節点が含む情報の変

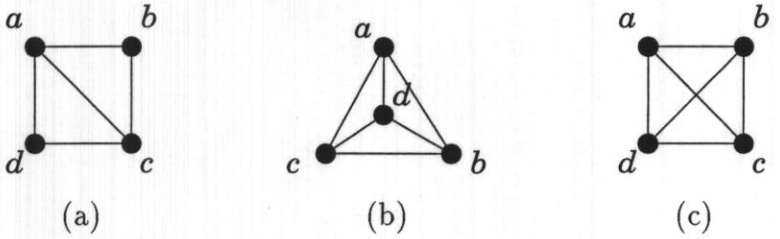


図 1.1 動的描画の例

更などの操作が加えられた後，元の描画の構造を保つように再描画する手法が有用である．これを動的描画法という [32].

新たに描画を行うアルゴリズムを再描画に用いることは，次の二つの点で問題を生じる [5].

- グラフが一部変更されたにもかかわらずすべての描画を行うため，新たに描画を行うアルゴリズムは変更されていない部分に対しても，より多くの資源や時間を要する．
- 新たに描画を行うアルゴリズムを再描画に用いると，節点の相対位置が全く変わってしまうことがあるため，元の描画との位置関係が把握できなくなることがある．

前者については，グラフのクラスごとにさまざまなアルゴリズムが提案されている．特に他の節点や辺に影響を与えにくい根付き木に関する動的描画法は数多くみられる．Moen[25] および松浦ら [23] は動的な木の描画アルゴリズムを提案した．Moen の方法は，節点の大きさや形状が異なるような場合も扱うことができる．

後者は，再描画を行う際には各節点間の位置関係が重要となる．例を図 1.1に示す．同図 (a) に (b, d) の辺を加えた後，ある平面描画アルゴリズムを実行すると，同図 (b) が得られる．しかしユーザの観点からすると，同図 (c) が得られた方が望ましい．このことから単純に，平面に描画することや辺の交差を減らすことを制約条件としてアルゴリズムを適用するだけでなく，動的描画を行う前後の節点の位置関係を考慮する必要がある．Eades ら [10] はこの問題に着目し，メンタルマップと呼ばれる節点間の位置関係に関する指標を提案し，次のような数学

的モデルを挙げている．

- 直交順序…再配置前後で節点の上下左右の関係を保存する．
- 近接関係…近接する節点は，再描画後も近接して配置する．
- トポロジー…再配置前後で配置によって定まる 2 次元平面上の領域間の隣接関係を保存する．

これらの指標を満たす再描画アルゴリズムがいくつか提案されている [24, 31]．

1.4.1 矩形の再配置問題

各節点が情報を保持しているようなグラフの場合，通常節点を矩形や円などで表し，内部に保持する情報を表示する．この情報に変更が加えられると，節点を拡張する必要があるため，他の節点と交差することがある．更に，描画領域は画面や紙面のように通常限界があるため，交差を解消するために節点を移動した結果，節点が領域外に押し出される可能性がある．全体の情報をできるだけ多く表示するためには，節点同士が交差せず，描画面積ができるだけ小さくなるような描画が得られることが望ましい．グラフ描画における矩形の再配置問題としては，レイアウト調整 (Layout Adjustment) 問題 [10] の他，点ラベル配置 (Point Feature Label Placement) 問題 [7] や，辺ラベル配置 (Edge Labeling Placement) 問題 [17, 18] などがある．

第 4 章では，節点の交差に重点を置き，単純化のためグラフの辺の接続関係を考慮せず，節点を矩形と置き換えてグラフの描画問題を矩形の配置問題として取り扱う．すなわち，直交順序を保存し，矩形の非交差を満たす面積最小の再配置を求める問題について考察する．三末ら [24] がこの問題に対する発見的手法を提案している．ここではまず，直交順序を保存する矩形の非交差最小面積再配置問題を定義し，この問題が整数座標系で NP 完全，実数座標系で NP 困難であることを示す．更に，文献 [24] と同じ漸近的時間計算量で，同文献で得られる面積以下の再配置を求める発見的再配置アルゴリズムを示す．このアルゴリズムの時間計算量は $O(n^2)$ である (n は矩形数)．また，ランダムな初期配置に対して計算機実験を行い，矩形数が多い場合に，文献 [24] の結果に対して 15 ～ 20% の面積で再配置できることを示す．

1.5. 本論文の構成

本論文の構成は以下の通りである．第 2 章では，根付き順序木の描画問題の一つとして，木構造図の描画問題について考察する．第 3 章では，根付き非順序木の描画問題について考察する．第 4 章では，一般グラフの動的描画問題の一つとして，予め配置されている矩形の集合に対して，直交順序を保存し，矩形同士が交差しない，という制約の下で，面積最小の再配置を求める問題について考察する．最後に第 5 章で，以上の研究成果の結論を述べるとともに，今後の研究課題について述べる．

第 2 章 木構造図の描画問題

本章では，木構造図の描画問題について述べる．以下，2.1節 で木に関する用語および本章で用いる表記法を定義し，2.2節 で海野らのアルゴリズムを紹介する．2.3 および 2.4節 では，海野らのアルゴリズムと同じ描画を効率的に求める方法を提案する (2.3節 では，部分木の輪郭上の節点が持つ情報など，データ構造について説明する．2.4節 では，新しいアルゴリズムを示し，その時間計算量を評価する)．

2.1. 諸定義

$T=(V,E)$ を木構造図 (以下，単に木と記す) とし，その根を rとする．各節点 $v \in V$について，二つの正の整数 $width_x(v)$ と $width_y(v)$ があらかじめ指定されているものとする．ここで扱う木構造図は順序木を基にしており，各節点の子の順序も与えられる．各節点 v の子を順に $v_1, v_2, \cdots, v_m$としたとき，$v_i (1 \leq i < m)$ を v_{i+1} の兄，v_{i+1} を v_i の弟と呼び，v_1を特に vの長男と呼ぶ．Tにおいて，節点 $v \neq r$の親を $pa^T(v)$，vを根とする部分木を $T(v)$ と書く．またvに対して，vの兄を根とする部分木すべてからなる森を部分森と呼び，$\overline{T}(v)$ で表す (図 2.1参照)．特に vが長男であれば$\overline{T}(v)=(\phi,\phi)$ である．Tの任意の二つの部分木 $T(v_a)=(V_a,E_a)$ と $T(v_b)=(V_b,E_b)$ に対して，演算 $\cup$を次のように定義する．

$$T(v_a) \cup T(v_b) \triangleq (V_a \cup V_b, E_a \cup E_b)$$

同様の定義を部分森に対しても行う．

r以外の Tの節点のうち，長男でないもの全体の集合を $YB(T)$ と表す．rから最も遠い葉への道の長さ (その上の辺の数) を Tの高さと呼ぶ．また，rから各節点 vへの道の長さを vのレベルといい，$level(v)$ と表す．二つの異なる節点 v，wについて，$level(v) \leq level(w) < level(v) + width_x(v)$ 若しくは $level(w) \leq level(v) < level(w) + width_x(w)$ が成立するとき，vと wとは互いにオーバーラップするということにする．

図 2.1に木の描画の例を示す．このように根は最も左に置き，それぞれの節点

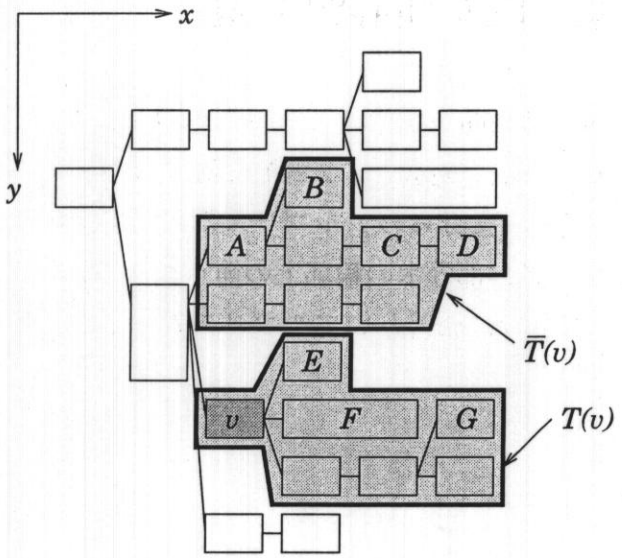


図 2.1 木の描画の例

はその親より右に置く．兄弟の節点を描画するときは長男を最も上に置くものとする．各節点 $v \in V$ は x 方向の幅 $width_x(v) - \delta$，y方向の幅 $width_y(v) - \delta$の長方形として描く．$\delta(< 1)$ をある小さな正の定数とする．但し，葉でない節点 (内部節点) の $width_x$は 1 であるものとする．各節点 vについて，それを示す長方形の左上の頂点の x 座標値，y座標値はともに正の整数であり，それぞれ$\pi_x(v)$，$\pi_y(v)$ と表す．節点の y座標値は，$\min\{\pi_y(v) \mid v \in V\} = 1$ となるように定めるものとする．このとき，木 Tの描画の y方向の幅 W_yは次のように定義することができる．

$$W_y \triangleq \max_{v \in V}\{\pi_y(v) + width_y(v)\} - 1.$$

木の描画について，文献 [42] と同様に，以下のような制約を考える．

(c1) 異なる節点は重ならない．異なる辺は (共有する端点での接触を除いて) 交差しない．任意の辺はその端点以外の節点と重ならない．

(c2) 同レベルの節点は同一 x 座標を持つ．ここでは各 $v \in V$について$\pi_x(v) = level(v) + 1$ とする．

(c3) 節点vの子を上から順に$v_1, \cdots, v_m$とする．このとき，$\pi_y(v) = \min\{\pi_y(v_1) + j, \pi_y(v_m)\}$ とする．ここでjは非負の整数 (定数) である．

ここでは，文献 [42] と同じ描画，すなわち，制約 (c1)〜(c3) を満たす描画を得ることを目的としているが，まず，上の制約に加えて，以下の制約 (c4),(c5) も満たす描画を初期描画として求める．

(c4) 同型の部分木は (平行移動が許される範囲で) 同じ描画をする．

(c5) 互いに節点を共有しないような部分木同士が同じy座標を共有しない．つまり，Tの二つの部分木 $T(v_a) = (V_a, E_a)$ および $T(v_b) = (V_b, E_b)$ に対して，v_aと v_bが互いに先祖–子孫の関係になく，かつ$\pi_y(v_a) < \pi_y(v_b)$ であるとき，$\max\{\pi_y(p) + width_y(p) \mid p \in V_a\} \leq \min\{\pi_y(q) \mid q \in V_b\}$ である．

制約 (c1)〜(c3) の下では，互いにオーバーラップする任意の 2 節点について，どちらがより上に配置されるかが一意に決まる．v, wを互いにオーバーラップする 2 節点とし，wが vより上に配置されるものとする．このとき，vの上辺のある点と wの下辺のある点とを結ぶ垂直線分 l が存在して，v, w以外のどの節点ともlが交差しないならば，wは vから可視であるということにする．節点 vから可視の節点すべてからなる集合を $Vis(v)$ と表す．この集合も，制約 (c1)〜(c3) の下では，Tの描画によらず一意に定まる．$0 \leq |Vis(v)| \leq width_x(v)$ である．

Tの任意の部分木 $T(v) = (V', E')$ について，$Vis(w) = \phi$若しくは $Vis(w) \cap (V - V') \neq \phi$であるようなすべての節点 $w \in V'$を $level(w) + width_x(w)$ の値の昇順に並べて得られる系列を $T(v)$ の上輪郭と呼び，$U(T(v))$ と表す．部分森に対しても同様に上輪郭を定義する．例えば図 2.1において，$U(T(v)) = [v, E, F, G]$, $U(\overline{T}(v)) = [A, B, C, D]$ である．

今，木 Tが平面上に描画されているものとする．このとき，各節点 vについて $u\_dist(v)$ を次のように定義する．

$$u\_dist(v) \triangleq \begin{cases} \pi_y(v) - 1 & (Vis(v) = \phi) \\ \min\{\pi_y(v) - (\pi_y(w) + width_y(w)) \mid w \in Vis(v)\} & (Vis(v) \neq \phi). \end{cases}$$

更に，部分木 $T(v) = (V', E')$ に対して移動可能距離 $d(v)$ を次のように定義す

る (文献 [42] では $Move_T(v)_\pi$という表記を用いている).

$$d(v) \triangleq \min_{w \in U(T(v))} \{u\_dist(w)\}.$$

2.2. 平行移動による描画

本節では，海野ら [42] のアルゴリズムを簡単に紹介する.

木 $T = (V, E)$ が与えられたとき，海野らのアルゴリズムはまず制約 (c1)~(c5) を満たす最小幅描画を $O(n)$ 時間で求める ($n \triangleq |V|$). この描画を $I(T)$ とし，$I(T)$ における各節点 vの y座標を$\pi_y^I(v)$ と表すことにする．次いで $I(T)$ に対して，制約 (c1)~(c3) を満たす範囲で更に y方向の幅 W_yが小さくなるように再描画を行う．これは，次の手続き **SubOpt**(v) を Tの根 rに対して ($v = r$として) 呼び出すことにより行う.

[**Procedure SubOpt**(v)]

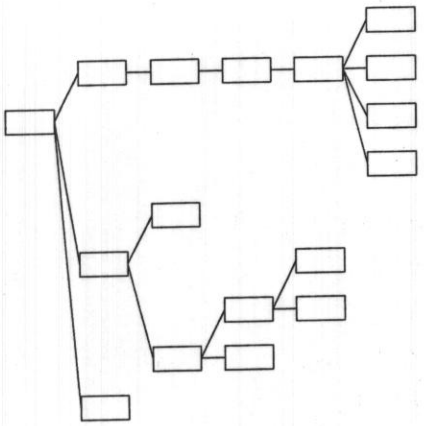
(a) vが葉の場合，(*) に従って vを平行移動する.

(b) vが内部節点の場合，vのすべての子 wについて (上の子から順に) **SubOpt**(w) を再帰的に実行し，その後 vの y座標を制約 (c3) に従って設定する．更に $T(v)$ 全体を (*) に従って平行移動する.

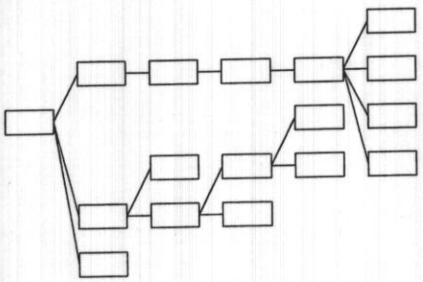
(*) **平行移動** $T(v)$ の根 vが長男であれば何もしない．vが長男でない場合には現時点での $d(v)$ を求め，$T(v)$ のすべての節点を $d(v)$ だけ上方へ移動する ($T(v)$ の節点の相対的な位置関係は変更されないことに注意).

[**例 2.1**] 図 2.2に描画例を示す．同図 (a) が初期描画 $I(T)$，同図 (b) が **SubOpt** で得られる最終的な描画であり，$j = 1$ である.

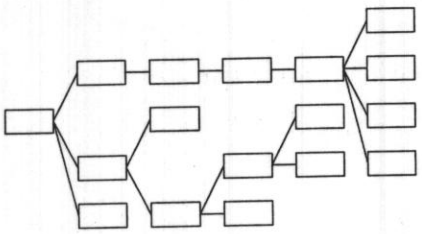
文献 [42] では，各節点 vに対し，$d(v)$ の計算に $O(n^2)$ 時間を，$T(v)$ の節点の平行移動に $O(n)$ 時間を使っており，再描画全体の実行時間が $O(n^3)$ となっている．上記のアルゴリズムにより得られる描画は，必ずしも制約 (c1)~(c3) の下で最小幅ではない (図 2.2(c) 参照).



(a) 初期描画



(b) 最終描画



(c) 制約 (c1)~(c3) の下での最小幅描画

図 2.2 平行移動による描画例

2.3. 相対座標と輪郭

本節と次節とにわたって，まず，すべての節点 $v \in V$ に対して $width_x(v) = 1$ である場合について，海野らの方法と同じ描画をより高速に求めるアルゴリズム **DRAW**(T) を提案する．$width_x$ が 1 でない葉が存在する場合については次節の最後で述べる．本節では，アルゴリズム **DRAW**(T) で用いるデータ構造について説明する．

アルゴリズム **DRAW**(T) は，海野らの方法と同様，制約 (c1)~(c5) を満たす最小幅描画 $I(T)$ を求めた後，ある再帰的手続きを根 r に対して呼び出すことによって再描画を行うものである．

2.2節で紹介した手続き **SubOpt** と区別するため，この新しい手続きを **SubOpt′** と表すことにする．**SubOpt′**は，次の 2 点を除いて **SubOpt** と同様のものである．

- 絶対座標ではなく，“相対座標” を用いることにより，平行移動による節点の y 座標の更新の回数を減らしている．
- 部分木および部分森の上輪郭を用いることにより，$d(v)$ の計算をより効率的に行っている．

SubOpt′は，**SubOpt** と同じ順序で再帰的に呼び出され，部分木の平行移動を行う (部分木の移動距離も同じである) ものであるから，再描画の結果得られる描画は海野らのアルゴリズムによるものと等しくなる．従って，アルゴリズム **DRAW**(T) が求める描画は前述の制約 (c1)~(c3) を満たすが，それらの下での最小幅描画ではない．

相対座標および輪郭は文献 [23, 25, 27] 等のアルゴリズムでも用いられている．これらのアルゴリズムはいずれも，木の節点を後行順に訪問し，各節点 v について，それを根とする部分木 $T(v)$ の描画 (節点間の相対的な位置関係) を定めていくというものである．

2 分木を対象とした文献 [27] のアルゴリズムでは，節点 v が左の子 v_l と右の子 v_r を持つならば，$T(v)$ の描画を定める際に，$T(v_l)$ の右側の輪郭と $T(v_r)$ の左側の輪郭とを用いて 2 節点 v_l と v_r の間の距離をできるだけ小さくなるように決める (文献 [27] では，根を最も上に置くような描画を求める)．この距離の計算が効率

的にできるように，文献 [27] では，各部分木の描画を決めた際，その左右の輪郭を求め，更に各輪郭上の各節点 wについて wと一つ次の節点の x 座標の差を保持するものとしている．

文献 [23] では多分木を扱っているが，そこでも各部分木の描画を求めるために文献 [27] と同様に輪郭を用いている．Moen[25] のアルゴリズムでは，節点が異なる大きさや形を持つことを許しているので，各輪郭を (節点の系列ではなく) 多角形として保持しているが，部分木の描画を定める方法は文献 [23, 27] と本質的に同様のものである．

以上の三つのアルゴリズムでは，$T(v)$ の描画を求めるときに $T(v)$ に属さない節点を考慮する必要はない．一方，海野らのアルゴリズムの再描画では，$T(v)$ の描画を定めるために，vの各子 $w \in YB(T)$ について $T(w)$ の移動可能距離 $d(w)= \min\{u\_dist(p) \mid p \in U(T(w))\}$ を計算するが，その際，一般には $T(v)$ に属さない節点の位置を考慮しなければならない．各 wについて，$T(w)$ の上輪郭 $U(T(w))$ と，$T(w)$ より上に配置されている部分全体の下側の輪郭を作っていたとしても，$U(T(w))$ 上の全節点について u_dist を求めるような方法では，$d(w)$ の計算に少なくとも $O(U(T(w))$ 上の節点数) の時間は要することになる．$\sum_{x \in YB(T)}(U(T(x))$ 上の節点数) $= O(n^2)$ であるから，再描画全体の時間計算量を $O(n^2)$ より少なくするためには，u_dist の無駄な計算を減らすことが本質的に必要である．手続き **SubOpt′**では，部分木および部分森の上輪郭だけを用い，更に輪郭上の各節点に付随させる情報を工夫することにより，この問題を解決している．

以下，2.3.1節で相対座標について，2.3.2節で輪郭上の節点が持つ情報について説明する．2.3.3節では，節点や部分木の移動が起こったときにその移動距離を記録しておくための変数を定義する．

2.3.1 相対座標

再描画においてある部分木 $T(v)$ の移動が発生したとすると，$T(v)$ に属するすべての節点の y座標が変わる．各節点 vについて次のような相対座標を求めることにしておけば，$T(v)$ を移動させるとき $\Delta\pi_y(v)$ の値のみを計算すればよい (各

節点の x 座標は変わらないので絶対座標のままでよい).

$$\triangle\pi_y(v) \triangleq \begin{cases} 0 & (v\text{が根}) \\ \pi_y(v) - \pi_y(pa^T(v)) & (v\text{が長男}) \\ \pi_y(v) - \pi_y(v\text{のすぐ上の兄}) & (v\text{が長男でない}). \end{cases}$$

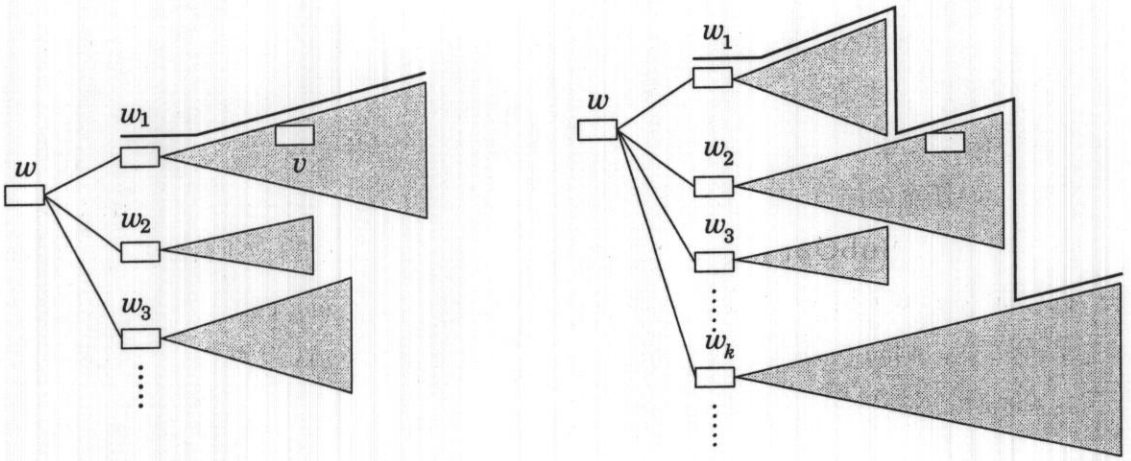
根でない節点 vについて $\triangle\pi_y(v)$ の計算は，vが長男ならば**SubOpt'**$(pa^T(v))$ において，vが長男でないならば**SubOpt'**(v) において行う (詳細は後述する)．再描画の終了後，任意の一つの節点の絶対座標を定めれば，すべての節点の絶対座標が決まることになる．制約 (c1)~(c5) を満たす最小幅描画 $I(T)$ において，y座標が 1 であった節点 (必ず存在する) は再描画後も最も上にあるので，それらの y座標を 1 とすることにより，すべての節点の絶対座標を正しく決めることができる．

2.3.2 輪郭

vを任意の節点とする．部分木 $T(v)$ の上輪郭 $U(T(v))$ は **SubOpt'**(v) の実行中に生成する．更に vが長男でない場合には，$T(v)$ の移動 (すなわち $\triangle\pi_y(v)$ の値の計算) を行った後で，後の便利のため，$\overline{T}(v) \cup T(v)$ の上輪郭を作っておく．部分木および部分森のこれらの輪郭はリストにより保持する．また輪郭上の節点数も合わせて記憶しておくものとする．部分木 $T(v)$ の上輪郭 $U(T(v))$ が $[p_1(=v), p_2, \cdots, p_l](l \geq 1)$ であるとしたとき，その部分列 $[p_i, \cdots, p_l](1 \leq i \leq l)$ を $U_s(T(v), p_i)$ と表すことにする．部分森の上輪郭についても同様の定義を行う．

各節点 vに対して，vから可視である節点を指すポインタ $u_{ptr}(v)$ を初期描画後に求めておく ($width_x(v) = 1$ であるので $|Vis(v)| \leq 1$ であることに注意)．但し，$Vis(v) = \phi$のとき $u_{ptr}(v) = null$とする．木の描画が定まっておりかつ $Vis(v) \neq \phi$ のとき，節点 vと $u_{ptr}(v)$ の間の距離が $u\_dist(v)$ になる．

SubOpt'(v) において $d(v)$ を計算するためには，$U(T(v))$ 上の各節点 p について $u\_dist(p)$ を求め，それらの最小値をとればよい．しかし前述のように，$\sum_{v \in YB(T)}(U(T(v))$ 上の節点数)$= O(n^2)$ であり，このような処理を単純に各節点について行えば，再描画に $O(n^2)$ 時間は要することになる．u_dist の無駄な計算を省くため，各節点 vに対して，三つの値 $u_{init}(v)$，$u_d(v)$，$u_{off}(v)$ を定義す



(a) $k = 1$　　(b) $k \neq 1$

図 2.3 輪郭 $U(\overline{T}(w_k) \cup T(w_k))$

る．$u_{init}(v)$ には，$I(T)$ における $u\_dist(v)$ の値を保持しておく．$u_d(v)$ の初期値は $u_{init}(v)$，$u_{off}(v)$ の初期値は 0 である．

変数 $u_d(v)$ は距離 $u\_dist(v)$ を表すためにまず用いるが，**SubOpt′**(v) 実行以前に節点 $u_{ptr}(v)$ が移動したときには，直ちに $u_d(v)$ の値を変えるのではなく，**SubOpt′**(v) が呼び出された直後に $u_{ptr}(v)$ の移動距離を計算し，$u_d(v)$ を更新する．**SubOpt′**(v) を終了した後には，$u_d(v)$ は以下のような値を持つ．vの先祖でありかつ **SubOpt′** の実行が終了していない節点のうちで，レベルが最大のものを wとする．wの子を上から順に $w_1, w_2, \cdots, w_m$とし，今 **SubOpt′**(w_1)，**SubOpt′**(w_2)，$\cdots$，**SubOpt′**$(w_k)(1 \leq k \leq m)$ までが実行終了しているとする．このときもし vが輪郭 $U(\overline{T}(w_k) \cup T(w_k))$ 上にあるならば (図 2.3参照)，$U_s(\overline{T}(w_k) \cup T(w_k), v)$ 上の節点の移動可能距離の最小値，すなわち $\min\{u\_dist(v') \mid v' \in U_s(\overline{T}(w_k) \cup T(w_k), v)\}$ は $u_d(v) - u_{off}(w_1)$ である (これが成立しているとき，$u_d(v)$ の値は $U(\overline{T}(w_k) \cup T(w_k))$ に関して正当である，ということにする)．

このように，$u_d(v)$ の値は初期値から変化する．**SubOpt′**(v) が呼び出された直後の更新方法は 2.4.2節で説明する．また，**SubOpt′**(v) の実行終了後の更新については $u_{off}(\cdot)$ の更新方法とともに 2.4.4節で述べる．

2.3.3 節点および部分木の移動距離

ここでは，節点や部分木の移動が生じたときにその移動距離を記録しておくための変数を定義する．

(i) 制約 (c3) による節点の移動

葉でない任意の節点 vについて，その子を上から順に $v_1,\cdots,v_m$とする．**SubOpt'**(v) が呼び出され，**SubOpt'**(v_1),$\cdots$,**SubOpt'**(v_m) の実行が終了した時点では，vの子の間の距離を示す相対座標値 $\triangle\pi_y(v_2)$, $\triangle\pi_y(v_3)$, $\cdots$, $\triangle\pi_y(v_m)$ が求められている．

このとき，制約 (c3) より $\pi_y(v) = \min\{\pi_y(v_1)+j, \pi_y(v_m)\}$ となる必要があるので，$\triangle\pi_y(v_1)$ は次式により定められる．

$$\triangle\pi_y(v_1) := -\min\{j, \sum_{i=2}^{m}\triangle\pi_y(v_i)\}.$$

この時点までに v_1が上方に移動していた距離を dd とすると，v_1の y座標$\pi_y(v_1)$ は$\pi_y^I(v_1) - dd$ となっているから，

$$\pi_y(v) = \pi_y^I(v_1) - dd - \triangle\pi_y(v_1).$$

よって $\triangle\pi_y(v_1)$ を決めることにより，節点 vが

$$\pi_y^I(v) - \pi_y^I(v_1) + dd + \triangle\pi_y(v_1)$$

だけ上に移動したことになる．この値を $m_{c3}(v)$ として記録しておくことにする．

任意の葉については m_{c3}の値を 0 とする．**SubOpt'** は長男を根とする部分木を単独で移動させないので，v_1が既に移動していたとすれば，それは制約 (c3) による移動である．従って，$m_{c3}(v)$ は次式により計算できることになる．

$$m_{c3}(v) := \pi_y^I(v) - \pi_y^I(v_1) + m_{c3}(v_1) + \triangle\pi_y(v_1).$$

(ii) 部分木の移動

vを任意の節点とし，x をその先祖のうちで長男でないものとする．**SubOpt'**(x) の実行時に $T(x)$ 全体が上方に移動した場合，節点 vもそれに伴って移動する．

部分木の移動距離を表すため，各節点 $v \in V$に対して，$m_{sbt}(v)$ を定義する．$m_{sbt}(v)$ の初期値は 0 であり，**SubOpt'**(v) において部分木 $T(v)$ 全体が移動したときには，その距離を $m_{sbt}(v)$ に格納する．

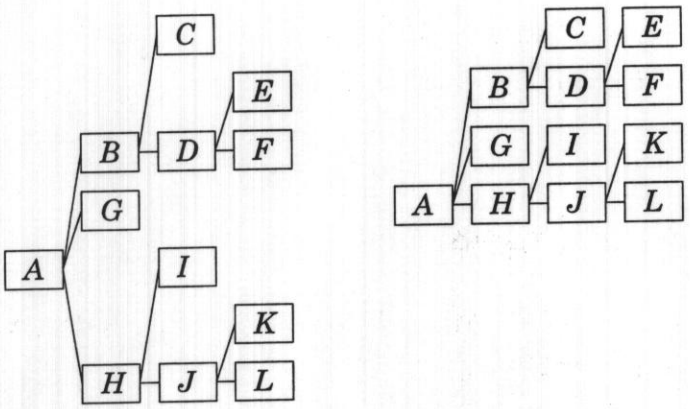


図 2.4 節点と部分木の移動の例

[例 2.2] 図 2.4(a) は，$j = 2$ であるときのある木の初期描画である．まず **SubOpt'**(D) の実行により部分木 $T(D)$ が上に 1 移動し $(m_{sbt}(D) = 1)$，その影響を受けて節点 Bが制約 (c3) により 1 移動する $(m_{c3}(B) = 1)$．更にそのため，$T(G)$ が上に 1 移動する $(m_{sbt}(G) = 1)$．以下，部分木の移動，制約 (c3) による節点の移動が続き，最終的な描画は図 2.4(b) のようになる．表 2.1に，各節点についての m_{c3},m_{sbt}の値，初期描画における y座標π_y^Iおよび最終描画における y座標π_y^Fを示している．

表 2.1 節点と部分木の移動の例のパラメータ値

	A	B	C	D	E	F	G	H	I	J	K	L
m_{c3}	1	1	0	0	0	0	0	1	0	0	0	0
m_{sbt}	0	0	0	1	0	0	1	2	0	1	0	0
π_y^I	5	3	1	3	2	3	4	7	5	7	6	7
π_y^F	4	2	1	2	1	2	3	4	3	4	3	4

2.4. 再描画の高速化

本節ではまず，すべての節点 vに対し $width_x(v)=1$ である場合についてのアルゴリズム **DRAW**(T) を示す．$width_x$が 2 以上の葉が存在する場合については本節の最後で述べる．

2.4.1 アルゴリズムの概略

アルゴリズム **DRAW**(T) の概略を以下に示す．また，**DRAW**(T) 内で呼ばれる手続き **SubOpt′** の概略を図 2.5に示す．

[**Algorithm DRAW**(T)]

(1) 初期描画 $I(T)$ を求める．

(2) (再描画のための前処理として) u_{ptr}，u_{init}，u_d，u_{off}，m_{sbt}などの初期設定を行う．

(3) 手続き **SubOpt′** を根 rに対して呼び出すことにより再描画を行う．

(4) $I(T)$ において y座標が 1 であった節点の y座標を 1 とすることにより，すべての節点の絶対座標を決める．

以下，2.4.2節 ～ 2.4.4節では，節点 vに対して **SubOpt′**(v) で行われる処理について順を追って説明する．アルゴリズム **DRAW**(T) の時間計算量については2.4.5節で述べる．

2.4.2 移動の検出

$u\_dist(v)$ は $I(T)$ では $u_{init}(v)$ であるが，$u_{ptr}(v) \neq null$ である場合には，**SubOpt′**(v) を実行する前に節点 $u_{ptr}(v)$ が上方に移動し，$u\_dist(v)$ の値が変わっていることがある．図 2.5に示したように，このような移動の検出を **SubOpt′**(v) が呼び出された直後に行い，$u_d(v)$ の値を更新する．$u_{ptr}(v) = null$ の場合には，この処理は必要ない．

[手続き SubOpt$'(v)$]

begin

1. **if** $(u_{ptr}(v) \neq null)$ **then**
2. $u_{ptr}(v)$ の移動距離を計算し，それを $u_d(v)$ に加える;
3. **if** (vは葉である) **then begin**
4. $U(T(v)) := [v]$; $m_{c3}(v) := 0$;

end

else begin

5. vの子を上から順に $v_1, \cdots, v_m$とする;
6. **for** $k := 1$ **to** m **do** SubOpt$'(v_k)$;
7. 制約 (c3) に従って $\triangle\pi_y(v_1)$ および $m_{c3}(v)$ を設定する;
8. $U(T(v))$ を作り，更にその輪郭に関して
   u_dの各値が正当なものになるように $u_d(v)$,$u_{off}(v)$ を更新する;

end;

9. **if** (vは長男でない) **then begin**
10. $T(v)$ の移動可能距離 $d(v)$ を計算し，$m_{sbt}(v)$ および $\triangle\pi_y(v)$ を設定する;
11. $U(\overline{T}(v) \cup T(v))$ を作り，更に u_d，u_{off}の必要な更新を行う;

end

end.

図 2.5 手続き SubOpt$'$

$u_{ptr}(v) = w$とする．**SubOpt'**(v) が呼び出されたとき，wがそれまでに移動した距離はwから順に親をたどることで計算できる．すなわち，wの先祖で既に**SubOpt'** の実行が終了しているものを$x_0(=w), x_1, \cdots, x_k$とすると，$w$の移動距離は $m_{c3}(w) + \sum_{i=0}^{k} m_{sbt}(x_i)$ である．しかし，ある節点の移動距離を求めた後，その真の子孫の移動距離が必要となった場合には，一度たどった道をもう一度たどることになる．このような無駄は，以下に述べるように，文献 [40] のアルゴリズムを用いることにより省くことができる．

まず，文献 [40] の結果のうち，必要な部分を簡単に紹介しておく．各節点 zに対して一つの整数 $lab(z)$ が定義されているような森 $F = (V_F, E_F)$ を考える．初期状態において $E_F = \phi$であり，各節点は単独で一つの木をなしている (その木の根となっている)．また各 $z \in V_F$について最初 $lab(z) = 0$ である．このような森に対して，次の 3 種類の指令からなる系列 Pが与えられるものとする．

EVAL(z) : Fにおいて zを含んでいる木の根 rを見つけ，zから rへの道上のすべての節点の lab の値の和を返す．

LINK(z, x) : 二つの木の根 z, x に対して，zが x の親となるように辺を加える．

UPDATE(z, γ) : ある木の根 zに対して，$lab(z)$ に整数γを加える．

文献 [40] の 2~3 章で示されているアルゴリズムは，系列 P中の各 EVAL に対してどのような値が返されるかを (オンラインで) 求めることができる．それに要する時間計算量は，P中の指令の総数を $|P|$ とすると

$$O((|P| + |V_F|) \cdot \alpha(|P| + |V_F|, |V_F|))$$

である．αは次のように定義される関数で，増加が極めて遅い (関数αおよび A の定義は文献 [40] に従っている)．

$$\alpha(a, b) = \min\{i \geq 1 \mid A(i, \lfloor 2a/b \rfloor) > \log_2 b\}.$$

ここで，整数 $i, h \geq 0$ に対してアッカーマン関数 $A(i, h)$ は以下のように定義さ

れる．

$$\begin{aligned} A(i,0) &:= 0, \\ A(0,h) &:= 2^h && (h \geq 1), \\ A(i,1) &:= A(i-1,2) && (i \geq 1), \\ A(i,h) &:= A(i-1, A(i,h-1)) && (i \geq 1, h \geq 2). \end{aligned}$$

移動の検出を行うために，上記のアルゴリズムを以下のように用いる．森Fの節点集合は木Tと同じくVとし，最初は$E_F = \phi$かつすべての$z \in V_F$について$lab(z) = 0$であるとする．各節点zについて**SubOpt'**(z)の実行が終了したときに，Fに対し指令 UPDATE$(z, m_{sbt}(z))$ および LINK$(pa^T(z), z)$ をこの順に与える．(**SubOpt'**(z)が終了する以前は，$pa^T(z), z$は共にFにおいて木の根である)．このようにしてFを更新していった場合，**SubOpt'**(v)が呼び出されたとき，節点$w(= u_{ptr}(v))$の先祖で既に**SubOpt'**の実行が終了しているものは，Fにおいてもwの先祖となっている．従って，wがそれまでに移動した距離を求めるためには，EVAL(w)を実行し，返された値に$m_{c3}(w)$を加えればよい．更に$u_d(v)$を更新するためには，単に

$$u_d(v) := u_d(v) + (w\text{の移動距離})$$

とすればよい．

以上述べた方法では，EVAL，LINK および UPDATE の実行回数の合計は$3n$を超えない．従って，次の補題が成立する．

[補題 2.1] 再描画全体における移動の検出 (**SubOpt'**の 1，2 行目) の時間計算量は$O(n\alpha(n,n))$である．

2.4.3 相対座標および移動可能距離の計算

節点vが葉である場合には，$U(T(v))$は$[v]$であり，$m_{c3}(v) = 0$である．これらは**SubOpt'**の 4 行目で正しく求められている．また，$T(v)$の移動可能距離$d(v)$は$u_{init}(v) + (u_{ptr}(v)$が移動した距離$)$であり，この値は既に$u_d(v)$に格納されている．$u_{off}(v) = 0$のままであるので，$U(T(v)) = [v]$に関して$u_d(v)$の値は正当である．$v$が長男でなければ$d(v)$だけ$v$を移動させる．その結果$u_dist(v) = 0$と

なるので，次のようにする (**SubOpt′**の 10 行目).

$$\begin{aligned} \triangle\pi_y(v) &:= 0, \\ m_{sbt}(v) &:= u_d(v). \end{aligned}$$

節点 vが葉でないものとし，その子を上から順に $v_1,\cdots,v_m$とする．節点 $u_{ptr}(v)$ の移動の検出を行って $u_d(v)$ を更新した後，**SubOpt′**(v_1), $\cdots$, **SubOpt′**(v_m) を実行する．これらが終了したときには，相対座標値 $\triangle\pi_y(v_2)$,$\triangle\pi_y(v_3)$,$\cdots$,$\triangle\pi_y(v_m)$ および $U(\overline{T}(v_m)\cup T(v_m))$ が求められている．更にこの輪郭上の各節点 p について，$u_d(p)$ の正当な値が得られている (すなわち，$U_s(\overline{T}(v_m)\cup T(v_m),p)$ 上の節点の移動可能距離の最小値は $u_d(p)-u_{off}(v_1)$ に等しい).

制約 (c3) より $\pi_y(v)=\min\{\pi_y(v_1)+j,\pi_y(v_m)\}$ となる必要があるので，2.3.3節で前述したように

$$\begin{aligned} \triangle\pi_y(v_1) &:= -\min\{j,\sum_{i=2}^{m}\triangle\pi_y(v_i)\}, \\ m_{c3}(v) &:= \pi_y^I(v)-\pi_y^I(v_1)+m_{c3}(v_1)+\triangle\pi_y(v_1) \end{aligned}$$

とする (**SubOpt′**の 7 行目). $\triangle\pi_y(v_1)$ を決めることにより $u\_dist(v)$ は $u_d(v)-m_{c3}(v)$ となっている．次に 8 行目では，$U(\overline{T}(v_m)\cup T(v_m))$ に左から vを追加することにより，$U(T(v))$ を得る.

$$\begin{aligned} temp &:= u_d(v)-m_{c3}(v), \\ u_d(v) &:= \min\{temp+u_{off}(v_1),u_d(v_1)\}, \\ u_{off}(v) &:= u_{off}(v_1) \end{aligned}$$

とすれば，上記のような更新を行った時点での $T(v)$ の移動可能距離 $d(v)$ は $u_d(v)-u_{off}(v)$ である．vが長男でないとき，この距離の分だけ $T(v)$ を移動させるので，10 行目では

$$\begin{aligned} m_{sbt}(v) &:= u_d(v)-u_{off}(v), \\ \triangle\pi_y(v) &:= temp+width_y(u_{ptr}(v))-\{u_d(v)-u_{off}(v)\} \end{aligned}$$

とする (この場合，$u_{ptr}(v)$ は vのすぐ上の兄を指す).

本節で述べた処理を節点 v に対して行うのに要する時間計算量は $O((v$の子の数$)+1)$ であり，木全体についての和は $O(n)$ である．すなわち，次の補題が成立する．

[補題 2.2] 手続き **SubOpt'** の 3～ 10 行目の時間計算量の (再描画全体についての) 総和は $O(n)$ である．

2.4.4 輪郭の合成

vが長男でない場合，$T(v)$ の移動により$\overline{T}(v) \cup T(v)$ の描画は実質的に決定するが，後に部分木 $T(pa^T(v))$ の移動可能距離を計算するため，$\overline{T}(v) \cup T(v)$ の上輪郭を求め，更にその上の各節点について正当な u_dの値を計算しておく．

輪郭$U(\overline{T}(v))$,$U(T(v))$ を$U(\overline{T}(v)) = [q_1, q_2, \cdots, q_k]$, $U(T(v)) = [p_1(=v), p_2, \cdots, p_l]$ とする．もし $k \geq l$ならば，$U(\overline{T}(v) \cup T(v)) = U(\overline{T}(v))$ である．逆に $k < l$の場合には，$U(\overline{T}(v))$ と $U(T(v))$ を次のようにして合成する．

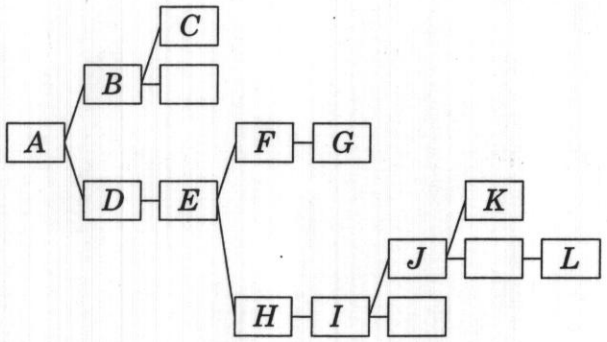
$U(T(v))$ 上の節点でレベルが $level(q_k)+1$ であるもの，つまり節点 p_{k+1}を合成点と呼ぶ．新しい輪郭 $[q_1, \cdots, q_k, p_{k+1}, \cdots, p_l]$ は，$U(\overline{T}(v))$ の末端に $U_s(T(v), p_{k+1})$ を追加して得られる．

既に $T(v)$ は $u_d(v) - u_{off}(v)$ だけ移動しているので，$u_{off}(v) := u_d(v)$ とすることにより，$U(T(v))$ 上の各節点 $p_i(1 \leq i \leq l)$ について $u_d(p_i)$ は ($U(T(v))$ に関して) 正当な値となる．次に，輪郭 $U(\overline{T}(v))$ 上の各節点 $q_i(1 \leq i \leq k)$ に対して，

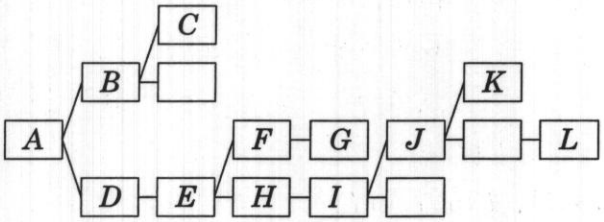
$$u_d(q_i) := \min\{u_d(q_i) - u_{off}(q_1) + u_{off}(v), u_d(p_{k+1})\}$$

と変更し，更に $u_{off}(q_1) := u_{off}(v)$ とする．これにより，$U(\overline{T}(v) \cup T(v))$ 上のすべての節点の u_dの値が正当となる ($U_s(T(v), p_{k+1})$ 上の節点の u_dを変更しなくてよい)．

各輪郭上の節点数を保持しているので，$U(\overline{T}(v) \cup T(v))$ を求めることは，$k \geq l$ のとき $O(1)$ 時間でできる．$k < l$のときには，$U(\overline{T}(v))$ と $U(T(v))$ を合成するが，その時間計算量は合成後の輪郭に属さない節点数 kに比例する．そのような節点 $p_1(=v), p_2, \cdots, p_k$は，後に輪郭の合成の対象とはならない．よって，次の補題が成立する．



(a)



(b)

図 2.6 輪郭の合成の例

[補題 2.3] 輪郭の合成 (**SubOpt′**の 11 行目) による時間計算量の (再描画全体についての) 総和は $O(n)$ である.

[例 2.3] 以下に輪郭の合成の例を示す. 図 2.6(a) は **SubOpt′** を再帰的に実行し, $T(B)$, $T(F)$, $T(H)$ の描画が完了したところである ($j = 1$ である). ここで $T(H)$ を移動させて (図 2.6(b)), $U(\overline{T}(H))$ と $U(T(H))$ の合成を行う. $T(H)$ を移動させる直前の各パラメータの値は表 2.2(a) の通りである.

$T(H)$ の移動可能距離は $u_d(H) - u_{off}(H) = 2$ である. $\triangle\pi_y(H)$ を 1 に設定した後, 上の手順に従って各値を計算する (Jが合成点である).

$$
\begin{aligned}
u_{off}(H) &:= u_d(H) = 2, \\
u_d(G) &:= \min\{u_d(G) - u_{off}(F) + u_{off}(H), u_d(J)\} = 3, \\
u_d(F) &:= \min\{u_d(F) - u_{off}(F) + u_{off}(H), u_d(J)\} = 3, \\
u_{off}(F) &:= u_{off}(H) = 2.
\end{aligned}
$$

表 2.2 輪郭の合成の例のパラメータ値

(a)

v	F	G	H	I	J	K	L
$u_d(v)$	2	2	2	2	3	3	4
$u_{off}(v)$	0	0	0	0	0	0	0

(b)

v	E	F	G	J	K	L
$u_d(v)$	3	3	3	3	3	4
$u_{off}(v)$	2	2	0	0	0	0

ここまでの更新を行った後，**SubOpt′**(H) を終了し，**SubOpt′**(E) の実行に戻る．節点 Eでは前節で述べたように，$m_{c3}(E) := 0$ とした後，$u_d(E) := \min\{u_d(E) + u_{off}(F), u_d(F)\} = 3$，$u_{off}(E) := u_{off}(F) = 2$ とする．この時点での各パラメータの値を表 2.2(b) に示す．ここで例えば，$U_s(T(E), F)$ および $U_s(T(E), L)$ 上の節点の移動可能距離の最小値は，それぞれ，$u_d(F) - u_{off}(E) = 1$，$u_d(L) - u_{off}(E) = 2$ となっている．

2.4.5 時間計算量

次の定理が成立する．

[定理 2.1] アルゴリズム **DRAW**(T) は $O(n\alpha(n,n))$ 時間で実行することができる．

(証明) ステップ (1) で初期描画 $I(T)$ を求めることは，文献 [42] にあるように $O(n)$ 時間で実行できる．またステップ (2) では，u_{ptr}, u_{init}, u_d, u_{off}, m_{sbt}および移動の検出のためのデータ構造の初期設定を行うが，これに要する時間計算量も $O(n)$ である．ステップ (3) において，手続き **SubOpt′** は各節点 vについて 1 回だけ呼び出され，$u_{ptr}(v)$ の移動の検出，$T(v)$ および$\overline{T}(v) \cup T(v)$ の上輪郭の構成，$T(v)$ の移動可能距離の計算などの処理が行われるが，補題 2.1~2.3(2.4.2節~2.4.4節) より，これらに要する時間計算量の総和は $O(n\alpha(n,n))$ である．すなわち，すべての $v \in V$についての **SubOpt′**(v) の時間計算量の和は $O(n\alpha(n,n))$ である．最後に，ステップ (4) ではすべての節点の絶対座標を計算するが，この処理を $O(n)$ 時間で行うことも容易である． □

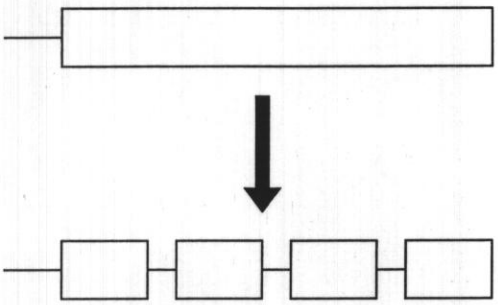


図 2.7 葉の分割

2.4.6 葉の $width_x$ が任意の場合

本節では，$width_x$ が 1 でないような葉が存在する場合について述べる．まず，$M_x \triangleq \max\{level(v) \mid v \in V\} + 1$ と定義する．明らかに $M_x \leq n$ である．もし T に $level(v) + width_x(v) > M_x$ であるような葉 v が存在するならば，そのような各葉 v に対して $width_x(v) := M_x - level(v)$ とする．その結果得られる木 T' に対して，海野らのアルゴリズムにより描画を行ったものとすると，各節点 v について最終的に求められる y 座標値は，T を描画したときの y 座標値に等しい．従って，T' を正しく描画することができれば十分である．

$width_x$ が 1 でないような葉が T' に存在するものとする．そのような各葉を図 2.7 のように $width_x$ 個の節点に分割すると，最も左にある節点を除いてすべて長男となる．これらの節点の $I(T')$ における y 座標はすべて等しくなる．更に再描画では，長男は真の先祖が移動したときのみそれに伴って移動するので，分割された節点が異なる y 座標を持つことはない．従って，再描画終了後，これらの節点を元の葉に戻すことは容易である．

葉の分割を行って得られる木の節点数を n' とすると，$n' = \sum_{v \in V} width_x(v) \leq n^2$ である．この木にアルゴリズム **DRAW** を適用すると，描画に要する時間は $O(n'\alpha(n', n'))$ となる．よって，次の定理を得る．

[定理 2.2] もしある定数 c が存在して，T の各葉 v について $width_x(v) \leq c$ であるならば，描画アルゴリズム全体を $O(n\alpha(n, n))$ 時間で実行することができる．

第 3 章 根付き非順序木の最小幅描画問題

本章では，根付き非順序木の最小幅描画問題について述べる．

以下，3.1節で本論文で用いる用語および記号を定義する．次に 3.2節では，制約の集合 D1 ～ D8 と，根付き非順序木の最小幅描画問題 UTD を定義する．3.3節では，D1 ～ D8 のそれぞれのもとで問題 UTD が NP 完全であることを証明する．更に 3.4節では，各節点の子の数を 3 以下に限定した場合の NP 完全性について述べる．

3.1. 諸定義

$T = (V, E)$ を任意の根付き非順序木とする．根から最も遠い葉への道の長さ（その上の辺の数）を Tの高さと呼び，$height(T)$ と書く．各節点 $v \in V$について，その子孫すべてからなる集合を $Des(v)$ と表す．$Des(v)$ の節点とそれらの間の辺からなる部分グラフを，vを根とする Tの部分木と呼び，$T(v)$ と書く．また，根から各節点 vへの道の長さを vのレベルと呼び，$level(v)$ と表す．

本論文では，根を最も上に配置して，その他の節点を親が上に子が下になるように配置することにする．木 Tを描画したとき，各節点 vについて，その x 座標値および y座標値をそれぞれ$\pi_x(v), \pi_y(v)$ と表す（整数座標系を用いるときには，これらの値は整数に限られる）．描画の幅は x 方向および y方向について 2 通りが考えられるが，y方向の幅は $height(T)$ によって決まる．x 方向の幅 $W_x(T)$ は次式で与えられ，これは描画に依存する．

$$W_x(T) \triangleq \max_{v,w \in V} \{\pi_x(v) - \pi_x(w)\}.$$

根付き非順序木 Tを描画したとき，各部分木 $T(v)$ に対して，vを中央として左側の幅 $WL_x(T(v))$ および右側の幅 $WR_x(T(v))$ を次のように定義する．

$$\begin{aligned} WL_x(T(v)) &= \max_{w \in Des(v)} \{\pi_x(v) - \pi_x(w)\}, \\ WR_x(T(v)) &= \max_{w \in Des(v)} \{\pi_x(w) - \pi_x(v)\}. \end{aligned}$$

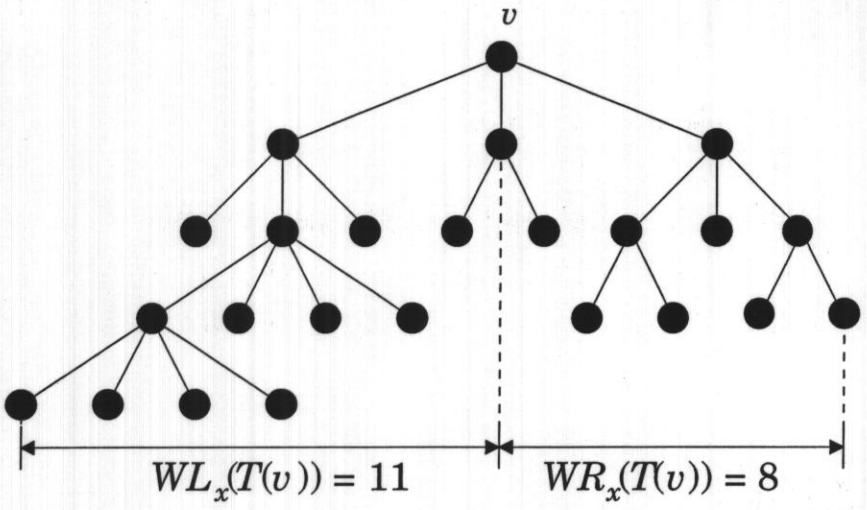


図 3.1 WL_x と WR_x の例

更に，これらをまとめて

$$W_x^*(T(v)) = (WL_x(T(v)), WR_x(T(v)))$$

と表すことにする．

$$WL_x(T(v)) + WR_x(T(v)) = W_x(T(v))$$

である．例えば，図 3.1の例では $W_x^*(T(v)) = (11, 8), W_x(T(v)) = 19$ である．

3.2. 最小幅描画問題

根付き非順序木を描画する際の制約として以下の六つを考える．これらはいずれも根付き順序木に対する制約として，従来から考えられていたものである．但し，(d3) および (d6) においては，節点 vの子 $v_1, v_2, \cdots, v_m$がこの順で左から描かれているものとする．

(d1) 異なる節点は重ならない．異なる辺は (共有する端点を除いて) 交差しない．また，任意の辺はその端点以外の節点と重ならない．

(d2) 同一のレベルの節点は同一の y座標を持つ．ここでは，各節点 $v \in V$に

ついて$\pi_y(v) = level(v)$とする．

(d3)　葉でない各節点vはその子の中央に配置する．すなわち，

$$\pi_x(v) = \frac{\pi_x(v_1) + \pi_x(v_m)}{2}$$

とする．

(d4)　互いに同型な部分木は（平行移動が許される範囲で）同じ描画を行う．

(d5)　同一レベルにある二つの節点p, qは，互いに距離2以上離して配置する（p, qは同一の親を持つとは限らない）．すなわち，

$$|\pi_x(p) - \pi_x(q)| \geq 2 \quad \text{但し } level(p) = level(q)$$

とする．

(d6)　共通の親を持つ子はすべて等間隔で配置する．すなわち，$i = 2, 3, \cdots, m-1$について，

$$\pi_x(v_{i+1}) - \pi_x(v_i) = \pi_x(v_i) - \pi_x(v_{i-1})$$

とする．

これらの制約からなる集合D1 ~ D8を表3.1のように定義する．制約(d1) ~ (d3)は木を描画する上で必須であり，(d4) ~ (d6)の有無で8通り定義する．表において“○”はその集合に含まれる制約を示している．例えば，制約集合D3は{(d1), (d2), (d3), (d4), (d6)}である．制約(d5)を設けない場合には，同一レベルの2節点間の距離は1以上とする．

本論文で扱う根付き非順序木の最小幅描画問題UTDは次のように記述できる．

[定義 3.1] (問題 UTD(D))　根付き非順序木Tと正の整数Bが与えられる．制約集合Dの制約をすべて満たすTの描画で，$W_x(T) \leq B$となるものが存在するか?

整数座標系を用いるときの問題UTD(D)をIUTD(D)，実数座標系を用いるときのUTD(D)をRUTD(D)と表す．

[補題 3.1] 問題IUTD(D1) ~ IUTD(D8)およびRUTD(D1) ~ RUTD(D8)はクラスNPに属する．

(証明)　UTDでは非順序木を対象としているが，まず子を複数持つ各節点に対して，子の順序を多項式時間で推測することができる．これにより，非順序木は順

表 3.1 制約集合 D1 ~ D8

制約集合	(d1)	(d2)	(d3)	(d4)	(d5)	(d6)
D1	○	○	○	○	○	○
D2	○	○	○	○	○	
D3	○	○	○	○		○
D4	○	○	○	○		
D5	○	○	○		○	○
D6	○	○	○		○	
D7	○	○	○			○
D8	○	○	○			

序木と同様に扱うことが可能となる．従って，文献 [36] と同様の議論により，問題 IUTD が整数計画問題に，RUTD が線形計画問題に，それぞれ変換できる．文献 [14, 30] より整数計画問題，線形計画問題がクラス NP に属することから，補題 3.1が成立する． □

以下では，まず 3.3節で IUTD(D1) の NP 完全性を証明する．この系として，IUTD(D2) ~ IUTD(D8) および RUTD(D1) ~ RUTD(D8) が NP 完全であることも分かる．更に 3.4節で各節点の子が 3 個以下の根付き非順序木に対して，IUTD(D1) が NP 完全であることを示す．

3.3. 各節点の子の数に制限がない場合の NP 完全性

本節では，問題 IUTD(D1) ~ IUTD(D8) および RUTD(D1) ~ RUTD(D8) が NP 完全であることを示す．

これらの問題の NP 困難性を証明するために，NP 完全問題であるハミルトン道問題 (Hamiltonian Path Problem，以下 HPP と略す) [14] を用いることにする．HPP とは，与えられたグラフ $G = (V_G, E_G)$ において，すべての節点を一度だけ通るような道 (ハミルトン道，以下 HP と略す) が存在するかどうかを判定する問

題である．

以下ではまず，このHPPから問題IUTD(D1)への変換を与える．$V_G = \{a_1, a_2, \cdots, a_n\}$ $(n = |V_G|)$ とする．グラフ G に対応して，根付き非順序木 T_{HP} および整数 B を構成する．木 T_{HP} は各節点 $a_i (1 \leq i \leq n)$ に対応する部分木 T^i を含んでおり，更に各 T^i は二つの部分 FT^i と ST^i からなる．

${}_nC_2 + 1$ 個の節点を作り，$v^i_{(a_1,a_2)}, v^i_{(a_1,a_3)}, \cdots, v^i_{(a_k,a_l)}, \cdots, v^i_{(a_{n-1},a_n)}, v^i_*$ とする $(1 \leq k < l \leq n)$．これらの節点すべてからなる集合を FV^i と表す．また，$v^i_{(a_l,a_k)}$ と表記した場合，$v^i_{(a_k,a_l)}$ を指すものとする．$v^i_{(a_k,a_n)}$ が $v^i_{(a_{k+1},a_{k+2})}$ の親となるように $(1 \leq k < n-1)$，$v^i_{(a_k,a_l)}$ が $v^i_{(a_k,a_{l+1})}$ の親となるように接続する $(1 \leq k < l \leq n-1)$．また，$v^i_{(a_{n-1},a_n)}$ が v^i_* の親となるように接続する．更に，各 $p (1 \leq p \leq n, p \neq i)$ について，G で節点 a_p と a_i の間に辺が存在しない場合にのみ，図 3.2(a) のように $v^i_{(a_p,a_i)}$ に二つの子を付け加える．以上のようにしてできる木を FT^i とし，その根を $r^i (= v^i_{(a_1,a_2)})$ とする．

$n + 1$ 個の節点を作り，$v^i_1, v^i_2, \cdots, v^i_n, v^i_{n+1}$ とする．これらの節点すべてからなる集合を SV^i と表す．各節点 v^i_j が v^i_{j+1} の親になるように接続する $(1 \leq j \leq n)$．また，節点 v^i_i にのみ図 3.2(b) のように二つの子を加える．このようにして構成した木を ST^i とする．

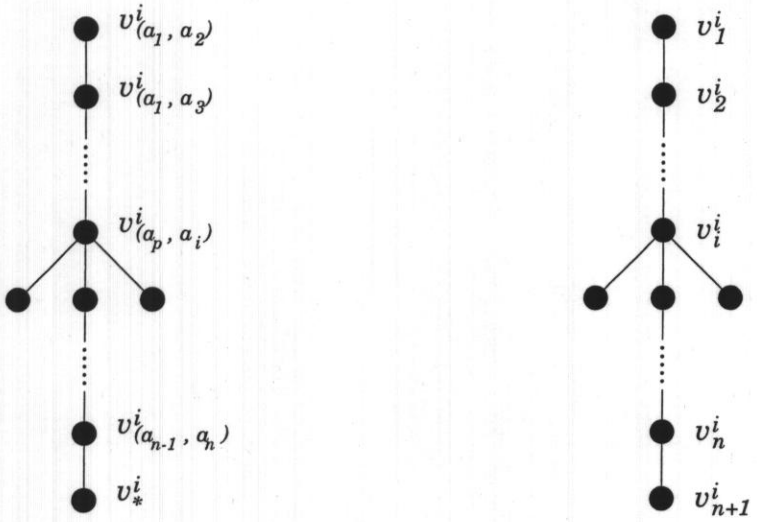
FT^i と ST^i とを，v^i_* が v^i_1 の親となるように接続することによって T^i を作る．制約 (d5) のもとでは，明らかに各 i について $W_x(T^i) \geq 4$ である．

各 T^i の根 r^i とそれらの共通の親となる節点 r_{HP} とを接続してできる木を T_{HP} とする．また，$B = 4n$ とする．G から T_{HP} および B を構成することは，明らかに $O(n^3)$ 時間でできる．

[例 3.1] 図 3.3に問題HPPからIUTD(D1)への変換例を示す．この場合，$n = 5$ であるから $B = 20$ となる．

[補題 3.2] グラフ $G = (V_G, E_G)$ にHPが存在するとき，かつそのときに限り，制約 (d1)～(d6) のもとで T_{HP} は幅 $4n$ 以下で描画可能である．

(証明)　まず，グラフ G においてHPが存在すると仮定する．HPの節点列を $[a_{m_1}, a_{m_2}, \cdots, a_{m_n}]$ とする．このとき，根 r_{HP} の子の順序が左から順に $r^{m_1}, r^{m_2}, \cdots, r^{m_n}$ となるように，T_{HP} を描くことを考える．各部分木 $T^{m_i} (1 \leq i \leq n)$ の描画は，制約 (d1) ～ (d6) を満たし，かつ $W_x(T^{m_i}) = 4$ となるようにする（$FV^i \cup SV^i$



(a) FT^i (b) ST^i

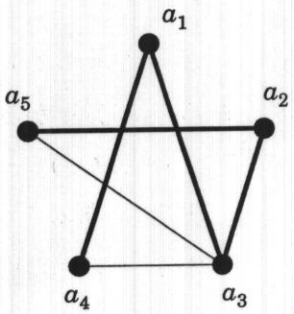
図 3.2 T^iを構成する二つの木

のすべての節点が同じ x 座標を持つようにする）．また，各 $i(1 \leq i < n)$ について，r^{m_i}と $r^{m_{i+1}}$の間隔が 4 となるように T^{m_i}を配置する．このとき明らかに，ST^{m_i}と $ST^{m_{i+1}}$の節点が制約 (d5) に違反することはない．$[a_{m_1}, a_{m_2}, \cdots, a_{m_n}]$ が G の HP であるので，各 i について $(a_{m_i}, a_{m_{i+1}}) \in E_G$．よって T_{HP}において，$v^{m_i}_{(a_{m_i},a_{m_{i+1}})}$と $v^{m_{i+1}}_{(a_{m_i},a_{m_{i+1}})}$は共に一つしか子を持たない．従って，$FT^{m_i}$と $FT^{m_{i+1}}$の同じレベルの節点が三つの子をもつことはない．r^{m_i}と $r^{m_{i+1}}$との間隔を 4 としているので，FT^{m_i}と $FT^{m_{i+1}}$の節点が制約 (d5) に反することもない．

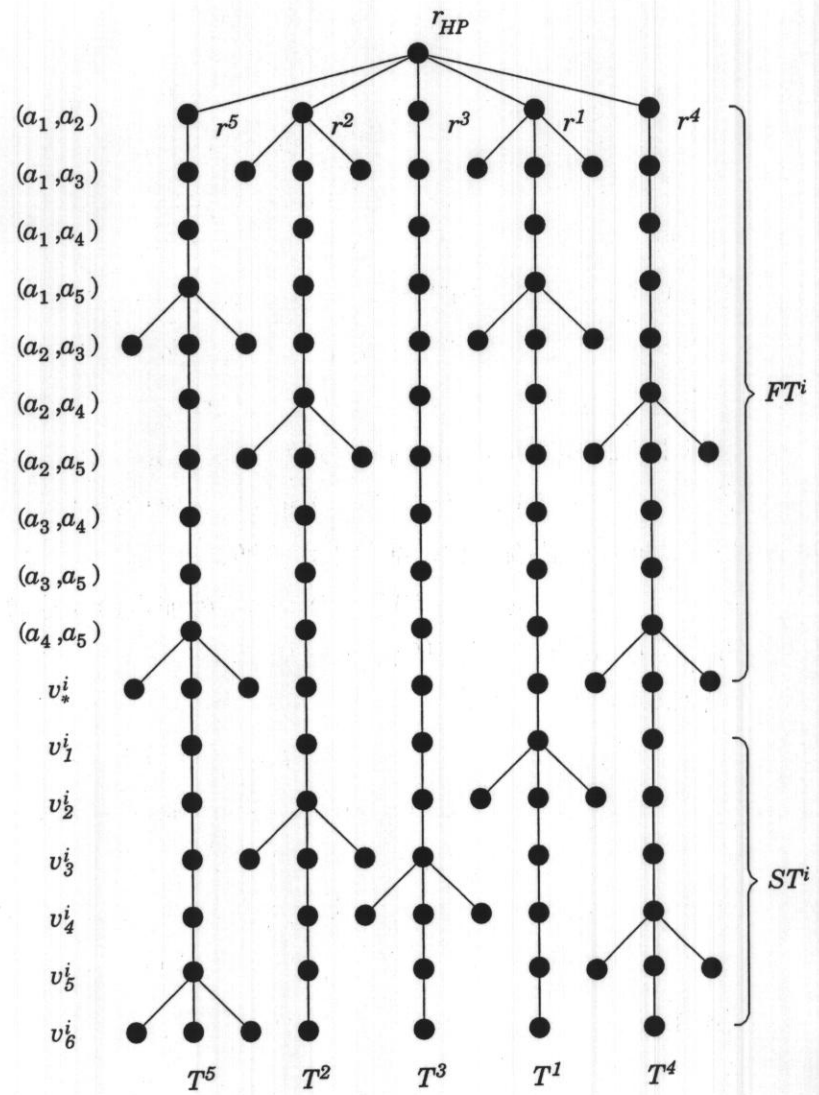
以上より，グラフ G に HP が存在すれば，T_{HP}は幅 $2 + 4(n-1) + 2 = 4n$ 以下で描画可能である．

次に，幅 $4n$ 以下の T_{HP}の描画が存在すると仮定する．そのような任意の描画 $\widehat{T_{HP}}$において，r_{HP}の子を根とする部分木が左から順に $T^{m_1}, T^{m_2} \cdots, T^{m_n}$と並んでいるものとする．一般性を失うことなく，$\widehat{T_{HP}}$ における節点の x 座標の最小値を 0 とする．各 $i\,(1 \leq i \leq n)$ について $W_x(ST^{m_i}) = 4$ であり，$\pi_x(v^{m_i}_1) = 4i - 2$ であることが容易に分かる．

今，FV^{m_1} のある節点 v の x 座標が 0 若しくは 1 であると仮定する．v の子



(a) グラフ G



(b) T_{HP} の描画の例

図 3.3 問題 HPP から IUTD(D1) への変換例

孫で子を三つ持つ節点のうち，v に最も近いものを w とする．w の三つの子のうち 2 番目に置かれているものの x 座標も 0 若しくは 1 であるから，左端の子の x 座標は 0 未満となる．これは，$\widehat{T_{HP}}$ における節点の x 座標の最小値が 0 であることに矛盾する．よって，FV^{m_1} のすべての節点の x 座標は 2 以上である．

次に，FV^{m_2} のある節点 v' の x 座標が 6 未満であると仮定する．v' の子孫で子を三つ持つ節点のうち，v' に最も近いものを w' とする．$w' \in FV^{m_2} \cup SV^{m_2}$ である．このとき，v' から w' への道上のすべての節点，および w' の三つの子のうち 2 番目に置かれているものの x 座標は 6 未満であり，w' の左端の子の x 座標は 4 未満となる．$w' \in FV^{m_2}$ であるならば，w' の左端の子と，同一レベルの FV^{m_1} の節点との間隔が 2 未満であることになり，$\widehat{T_{HP}}$ が制約 (d5) を満たしていることに矛盾．また，$w' \in SV^{m_2}$ であるならば，$\pi_x(v_1^{m_2}) = 6$ であることに矛盾する．よって，FV^{m_2} のすべての節点の x 座標は 6 以上である．

以上の議論を左の部分木から順に（$i = 3, 4, \ldots, n$ に対して）行うことにより，各 i に対して FV^{m_i} のすべての節点の x 座標が $4i-2$ 以上であることが分かる．更に，同様の議論を右の部分木から順に（$i = n, n-1, \ldots, 1$ について）行うことにより，各 i に対して FV^{m_i} のすべての節点の x 座標が $4i-2$ 以下であることも導かれる．従って，任意の i $(1 \leq i \leq n)$ について，FV^{m_i} の任意の節点の x 座標は $4i-2$ である．

各 i $(1 \leq i < n)$ について，FV^{m_i} と $FV^{m_{i+1}}$ の間隔が 4 であるので，三つの子を持つような節点が同じレベルで並んでいない．つまり，各 i について，$v^{m_i}_{(a_{m_i}, a_{m_{i+1}})}$ と $v^{m_{i+1}}_{(a_{m_i}, a_{m_{i+1}})}$ のどちらかは子を一つしか持っていない．ここで G の節点列 $P = [a_{m_1}, a_{m_2}, \cdots, a_{m_n}]$ を考えると，木 T_{HP}の構成法より明らかに，各 i について $(a_{m_i}, a_{m_{i+1}}) \in E_G$．すなわち，$P$は G の HP である．□

[例 3.2] 図 3.3(a) のグラフ G は HP$[a_5, a_2, a_3, a_1, a_4]$ を持つ．木 T_{HP}は，根の子を r^5, r^2, r^3, r^1, r^4の順に並べることにより，同図 (b) のように幅 $B = 20$ で描画できる．

補題 3.1, 3.2より，次の定理が導かれる．

[定理 3.1] 問題 IUTD(D1) は NP 完全である．

以下では，D1 以外の制約集合に対する問題 IUTD および実数座標系における問題 RUTD について考える．

[系 3.1] 問題 IUTD(D2) ~ IUTD(D8) は NP 完全である.
(証明) D1 から制約 (d4) 若しくは (d6) を除いた場合 (D2, D5, D6) にも，上で述べた各部分木 T^iの描画幅 $W_x(T^i)$ の最小値は 4 となり，D1 と同様に証明可能である．制約 (d5) を除いた場合 (D3, D4, D7, D8) については，$W_x(T^i)$ の最小値が 2 となるが，$B = 2n$ とすることで，同様に証明できる． □

[系 3.2] 問題 RUTD(D1) ~ RUTD(D8) は NP 完全である.
(証明) 補題 3.1より RUTD(D1) ~ RUTD(D8) はクラス NP に属する．また，NP 困難であることも，T_{HP}の構成を定理 3.1 若しくは系 3.1で述べたのと同じとすることにより，整数座標系の場合と同様に証明可能である． □

3.4. 各節点の子の数が 3 以下の場合の NP 完全性

本節では，各節点の子が3 個以下の場合について，問題 IUTD(D1) が NP 完全であることを示す．証明には，文献 [36, 41] と同様，代表的な NP 完全問題である 3-SAT(3-Satisfiability; 3-充足可能性問題)[14] を用いているが，ここでは各節点の子の順序に自由度があることに注意する必要がある．

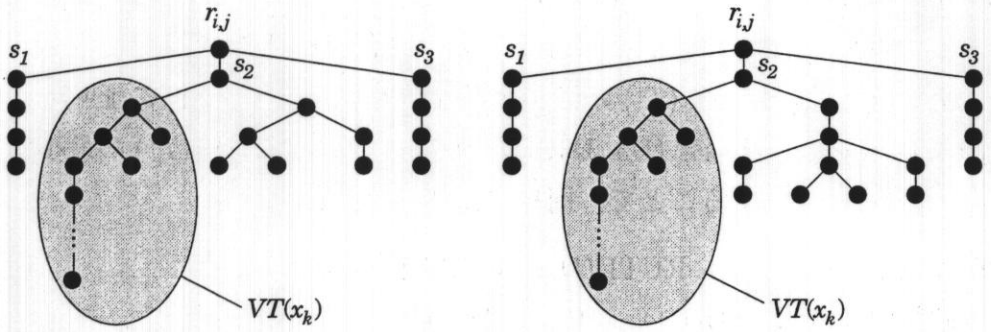
3.4.1 3-SAT から問題 IUTD(D1) への変換

3-SAT とは次のような問題である．

$X = \{x_1, x_2, \cdots, x_n\}$ をブール変数の集合とする．各変数 $x \in X$ またはその否定 $\bar{x}$ をリテラル（literal）と呼ぶ．有限個のリテラルの論理和を節（clause）と呼ぶ．有限個の節 $F_1, F_2, \cdots, F_m (m \geq 1)$ の論理積をブール代数式 Eとする．E の各変数 $x \in X$に true または false を適切に割り当てることによって，Eを true とすることができるとき，Eは充足可能であるという．

充足可能性問題 (SAT) とは，ブール代数式 Eが与えられたときに，Eが充足可能であるかどうかを判定する問題である．3-SAT とは，SAT において各節のリテラルを 3 個に制限したものである．各節 $F_i (1 \leq i \leq m)$ に含まれるリテラルを $y_{i,1}, y_{i,2}, y_{i,3}$と表す．

以下では，この 3-SAT から $B = 104$ に対する問題 IUTD(D1) への変換を与



(a) $y_{i,j} = x_k$　　(b) $y_{i,j} = \overline{x_k}$

図 3.4　$LT(y_{i,j})$ の構成

える．

3-SAT の入力となるブール代数式 Eに対応して，根付き非順序木 $T(E)$ を構成する．木 $T(E)$ は各節 F_iに対応する部分木 $CT(F_i)$ を含み，更に $CT(F_i)$ は各リテラル $y_{i,j}$に対応する部分木 $LT(y_{i,j})$ を含んでいる．

$LT(y_{i,j})$ の構成は，図 3.4のように 2 通り考える．あるブール変数 $x_k \in X$に対して，図 3.4(a) が $y_{i,j} = x_k$の場合であり，同図 (b) が $y_{i,j} = \overline{x_k}$の場合である．いずれの木にも x_kに対応する部分木 $VT(x_k)$ が含まれている．$LT(y_{i,j})$ の根を $r_{i,j}$と表し，その三つの子 s_1, s_2, s_3を図に示すように定める．

$VT(x_k)$ は図 3.5のように構成する．節点 q_0から順に $q_1, q_2, \cdots, q_k$までの $k+1$個の節点を同図のように接続したものを含んでいる．$x_k = x_{k'}(1 \leq k, k' \leq n)$ であるとき，かつそのときに限り $VT(x_k)$ と $VT(x_{k'})$ は同型である．

$VT(x_k)$ について，その根 p の二つの子の間隔を d_pと表すことにする．整数座標系を用いていることと制約 (d3) より，d_pの値は偶数でなければならない．d_pが 2 の場合と 4 の場合の $VT(x_k)$ の描画の例を図 3.6に示す．

$LT(y_{i,j})$ を描画する際には，制約 (d3) および (d6) に従って，各節点 vの子を等間隔に配置し，vをそれらの中央に配置する．また制約 (d5) に従って，同一レベルの 2 節点間の距離は 2 以上としなければならない．

d_pの値と $LT(y_{i,j})$ の描画の幅との間には次の関係 (*1) が成り立つ．

(*1) $y_{i,j} = x_k$の場合，d_pが 2 であれば $LT(y_{i,j})$ 全体の描画幅を 14 にすることが

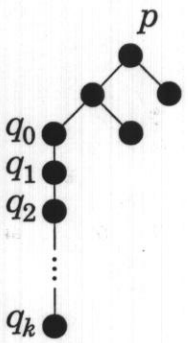


図 3.5 $VT(x_k)$ の構成

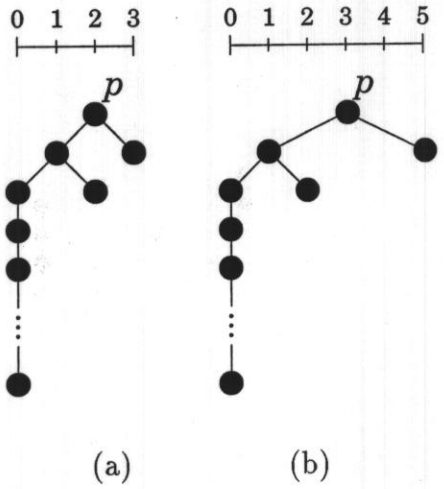


図 3.6 $VT(x_k)$ の二つの描画

できる（図 3.7(a)）．一方，d_pが 4 であれば $LT(y_{i,j})$ の描画幅は 16 以上を要する（図 3.7(b)）．d_pが 6 以上の場合も同様である．同様に，$y_{i,j} = \overline{x_k}$ の場合，d_pが 4 であれば $LT(y_{i,j})$ 全体の描画幅を 14 にすることができる（図 3.8(b)）．一方，d_pが 2 であれば $LT(y_{i,j})$ の描画幅は 16 以上を要する（図 3.8(a)）．d_pが 6 以上の場合も同様である．

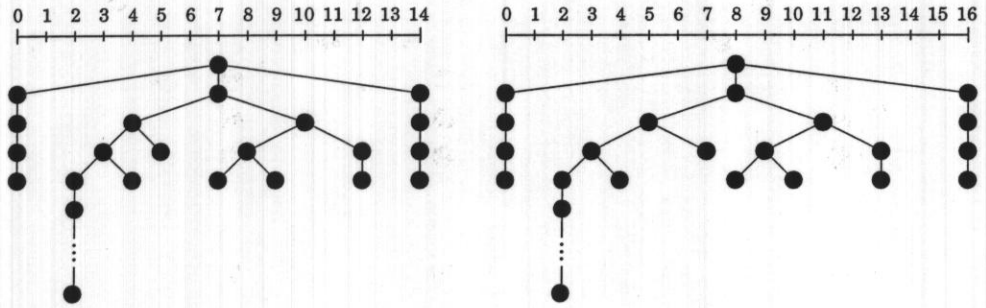
図 3.4に示したように，$LT(y_{i,j})$ の根 $r_{i,j}$は三つの子 s_1, s_2, s_3を持つ．$LT(y_{i,j})$ の描画で s_2が s_1と s_3の間に置かれていないものは不当であるということにする．このとき，次の (*2) が成立する．

(*2) $LT(y_{i,j})$ の任意の不当な描画の幅は 19 以上である (図 3.9).

また，$LT(y_{i,j})$ の描画幅の最小値について，次の (*3) が成立する．

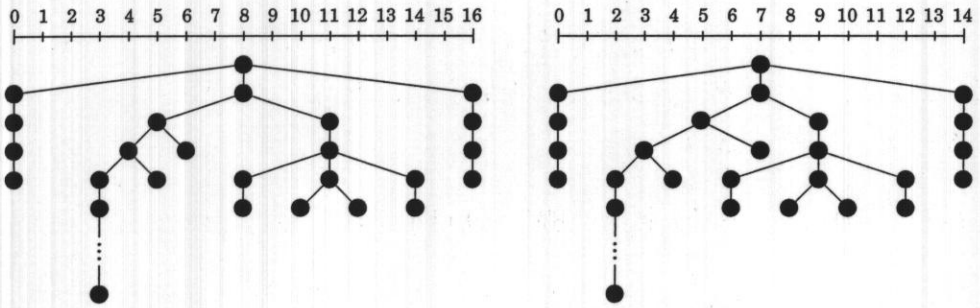
(*3) $LT(y_{i,j})$ の描画幅の最小値は 14 であり，幅 15 の描画は存在しない．

各 $LT(y_{i,j})(j = 1,2,3)$ の根 $r_{i,j}$と，それらの共通の親となる節点を図 3.10のように接続し，部分木 $CT'(F_i)$ を作る．更に，$CT'(F_i)$ の二つのコピーと，$n+6$ 個の節点を図 3.11のように接続して $CT(F_i)$ を作る．ここで，$CT(F_i)$ の根となる節点を w_iとし，$n+6$ 個の節点のうち最も下にあるものを w_i^*とする．



(a) (b)

図 3.7 $LT(y_{i,j})$ の二つの描画 ($y_{i,j} = x_k$の場合)



(a) (b)

図 3.8 $LT(y_{i,j})$ の二つの描画 ($y_{i,j} = \overline{x_k}$の場合)

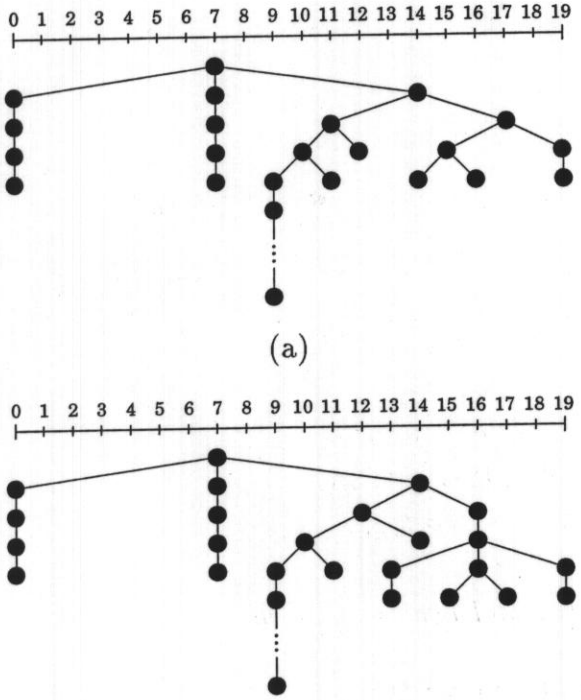


図 3.9 $LT(y_{i,j})$ の不当な描画の例

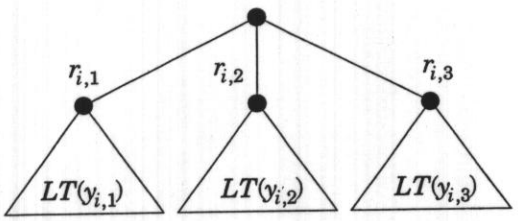


図 3.10 $CT'(F_i)$ の構成

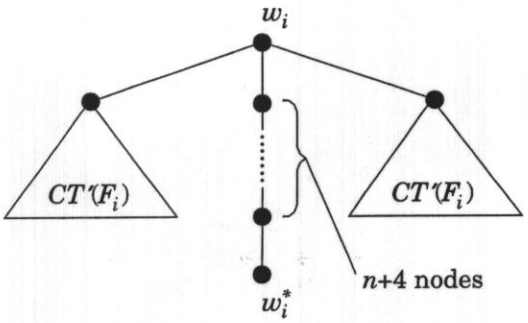


図 3.11 $CT(F_i)$ の構成

木 $T(E)$ は，m 個の $CT(F_i)$ を，各 $i(1 \leq i \leq m-1)$ に対して w_i^* が w_{i+1} の親となるように接続して得られる（図 3.12）．

$y_{i,j} = x_k$ 若しくは $y_{i,j} = \overline{x_k}$ であるとき，図 3.4より

$$height(LT(y_{i,j})) = height(VT(x_k)) + 2$$

である．従って，$CT'(F_i)$ の高さは

$$\begin{aligned} height(CT'(F_i)) &= \max_{j=1,2,3}\{height(LT(y_{i,j}))\} + 1 \\ &\leq \max_{1\leq k\leq n}\{height(VT(x_k))\} + 3 \\ &= (n+2) + 3 \\ &= n + 5 \end{aligned}$$

より，$n+5$ 以下である．よって，$T(E)$ における w_iのレベルを lとすると，図 3.11より，$CT'(F_i)$ の最も下にある節点のレベルは最大 $l + 1 + (n+5) = l + n + 6$ となる．一方，w_i^*のレベルは $l + n + 5$ なので，図 3.12より $CT'(F_{i+1})$ の根のレベルは $l + n + 7$ となる．このように，$CT'(F_i)$ の最も下にある節点のレベルは $CT'(F_{i+1})$ の根のレベルより小さいので，部分木 $CT'(F_i)$ と $CT'(F_{i+1})$ の節点が重ならないことが保証される．

各 $CT(F_i)$ は $O(n)$ 個の節点を持ち，$T(E)$ は m 個の $CT(F_i)$ を接続したものであるので，明らかに $T(E)$ の構成は $O(nm)$ 時間で可能である．

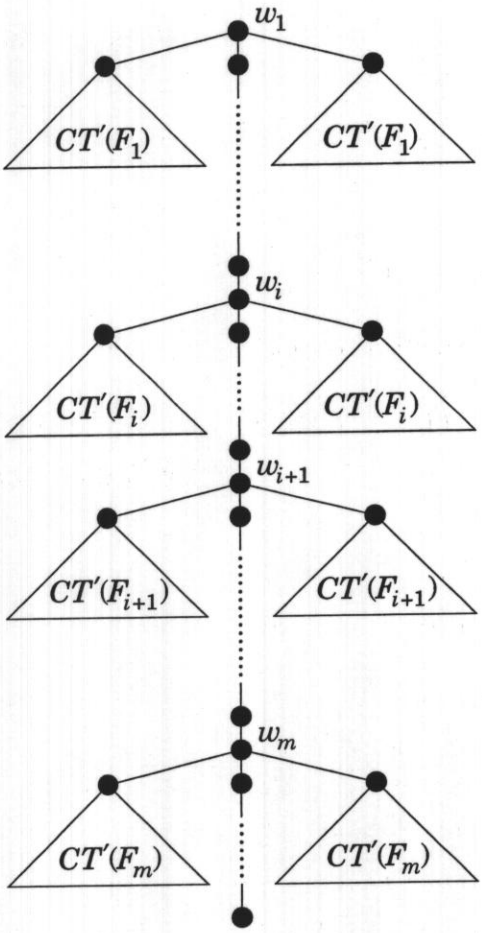


図 3.12 $T(E)$ の構成

3.4.2 証明

ここでは，ブール代数式 Eが充足可能であり，かつそのときに限り，$T(E)$ が幅 104 以下で描画可能であることを示す．

[補題 3.3] 各部分木 $LT(y_{i,j})$ の描画を図 3.7若しくは図 3.8 のように行ったものとする．このとき，任意の $CT'(F_i)$ について，三つの部分木 $LT(y_{i,j})$ のうち少なくとも一つが幅 14 の描画を持つならば，$W_x(CT'(F_i)) \leq 50$ となるように描画可能である．

(証明) 図 3.7, 3.8のように各 $LT(y_{i,j})$ の描画を行うと，$W_x(LT(y_{i,j}))$ は 14 か 16 であり，WL_xと WR_xの値によって，次の 2 通りに分類することができる．

1. $W_x^*(LT(y_{i,j})) = (7,7)$，図 3.7(a) および 3.8(b)
2. $W_x^*(LT(y_{i,j})) = (8,8)$，図 3.7(b) および 3.8(a)

仮定より，$LT(y_{i,1}) \sim LT(y_{i,3})$ の少なくとも一つは (1) の描画を持つことになる．従って，三つの $LT(y_{i,j})$ における W_x^*の可能な組み合わせは，

⟨ 1 ⟩ (7, 7) が三つ

⟨ 2 ⟩ (7, 7) が二つ，(8, 8) が一つ

⟨ 3 ⟩ (7, 7) が一つ，(8, 8) が二つ

の 3 通りのみである．制約を満たしながら，三つの $LT(y_{i,j})$ を図 3.13のように並べることにより，$W_x(CT'(F_i)) \leq 50$ とすることができる．□

任意の $CT'(F_i)$ について，$LT(y_{i,1}) \sim LT(y_{i,3})$ の描画がいずれも不当なものでなければ，制約 (d5) より，それらは 2 以上ずつ離して配置する必要がある．一方，例えば $LT(y_{i,1})$ と $LT(y_{i,2})$ の描画が不当なものであるとき，それらの一部の節点が同じ x 座標を持つ可能性がある．しかし，(*2) で述べたように不当な描画の幅は大きくなるため，次の補題が成立する（証明は省略する）．

[補題 3.4] ある $CT'(F_i)$ の描画の幅が 50 以下であるとする．このとき，$LT(y_{i,1}) \sim LT(y_{i,3})$ の描画はいずれも不当ではない．

補題 3.3，3.4より，次の補題が導かれる．

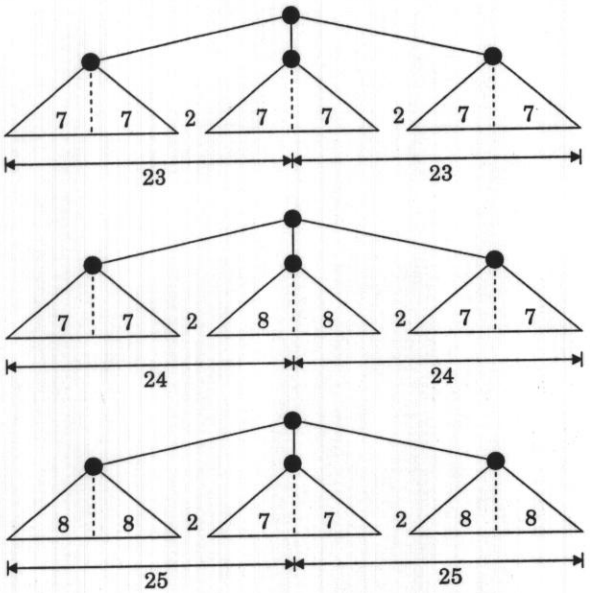


図 3.13 補題 3.3 の証明の説明図

[補題 3.5] ブール代数式 E が充足可能であるとき，かつそのときに限り，$W_x(T(E)) \leq 104$ となる木 $T(E)$ の描画が制約 (d1) ~ (d6) のもとで可能である．

(証明) まず，E が充足可能であると仮定し，E を true とするような，各変数への任意の真偽割り当てについて考える．

各リテラル $y_{i,j}(1 \leq i \leq m, j = 1,2,3)$ に対して，$y_{i,j}$ = true であるとき $W_x(LT(y_{i,j})) = 14$，$y_{i,j}$ = false であるとき $W_x(LT(y_{i,j})) = 16$ となるように，$LT(y_{i,j})$ の描画を定める．具体的には，$y_{i,j} = x_k$ であるとき，x_k = true であれば図 3.7(a) のように，x_k = false であれば同図 (b) のように描画する．また，$y_{i,j} = \overline{x_k}$ であるときには，x_k = true ならば図 3.8(a) のように，x_k = false ならば同図 (b) のように描く．

仮定より，各節 $F_i(1 \leq i \leq m)$ に含まれる三つのリテラル $y_{i,1} \sim y_{i,3}$ のうち少なくとも一つが true となっている．よって，以上のようにして各 $LT(y_{i,j})$ の描画を決めれば，補題 3.3より，$W_x(CT'(F_i)) \leq 50$ となるように $CT'(F_i)$ を描くことができる．

このCT'(F_i)の描画を元に，制約(d3)，(d5)，(d6)に従って，図3.11のように部分木を並べてCT(F_i)を描くことにより，

$$\begin{aligned} & W_x(CT(F_i)) \\ & \leq \ W_x(CT'(F_i)) + 2 + 2 + W_x(CT'(F_i)) \\ & \leq \ 50 + 2 + 2 + 50 = 104 \end{aligned}$$

とすることができる．このとき，$WL_x(CT(F_i)) \leq 52$，$WR_x(CT(F_i)) \leq 52$ となる．

最後に，$w_1 \sim w_m$が同じx座標を持つようにして，$CT(F_1) \sim CT(F_m)$の描画を接続する（図3.12参照）．このとき，

$$\begin{aligned} WL_x(T(E)) \ &\leq \ \max_{1 \leq i \leq m} \{WL_x(CT(F_i))\} \leq 52, \\ WR_x(T(E)) \ &\leq \ \max_{1 \leq i \leq m} \{WR_x(CT(F_i))\} \leq 52 \end{aligned}$$

より，$W_x(T(E)) \leq 104$ となる．従って，$T(E)$は幅104以下で描画可能である．

次に，幅が104以下である$T(E)$の描画が存在すると仮定する．そのような描画において，明らかに各$CT(F_i)$の描画幅も104以下である．各$CT(F_i)$の描画では，節点w_iとw_i^*とをつなぐ道が二つの$CT'(F_i)$の間に描かれている．制約(d5)より，同一レベルの節点間の間隔が2以上必要であることを考慮すると，各$CT'(F_i)$の描画幅は50以下となる．$CT'(F_i)$は三つの$LT(y_{i,j})$を含むが，補題3.4よりそれらの描画はいずれも不当ではない．よって制約(d5)より，三つの$LT(y_{i,j})$の描画の間隔は2ずつ必要であり，描画の幅の和は$50-2-2=46$以下となる．従って，(*3)より，$LT(y_{i,1}) \sim LT(y_{i,3})$のうち，少なくとも一つの描画幅は14である．

各変数に対して，次の(**)に従ってtrue若しくはfalseと対応させるものとする．

(**) 描画幅14の各$LT(y_{i,j})$に対して，$y_{i,j}$をtrueとする．すなわち，$y_{i,j} = x_k (\in X)$であればx_kをtrueとし，$y_{i,j} = \overline{x_k}$であればx_kをfalseとする．その結果値が定まらない各変数に対しては，任意にtrue若しくはfalseを割り当てる．

あるi, j, i', j'について，$y_{i,j} = x_k$，$y_{i',j'} = \overline{x_k}$であり，かつ$LT(y_{i,j})$と$LT(y_{i',j'})$の描画幅が共に14になったと仮定する．このとき，前述の(*1)より，$LT(y_{i,j})$

における $VT(x_k)$ の根の子の間隔は 2 であり，$LT(y_{i',j'})$ における $VT(x_k)$ の根の子の間隔は 4 である．これは制約 (d4) に反する．よって，$LT(y_{i,j})$ と $LT(y_{i',j'})$ の幅が共に 14 になることはなく，(**) により一つの変数に true および false の両方が割り当てられることはない．

(**) により各変数への真偽割り当てを行ったとき，各 F_iはすべて true となり，Eが true となるので充足可能である． □

補題 3.1, 3.5より，次の定理が導かれる．

[定理 3.2] 与えられた木 T の各節点の子の数が 3 以下である場合に対する問題 IUTD(D1) は NP 完全である．

第 4 章 矩形の最小面積非交差再配置問題

本章では，直交順序を保存する矩形の最小面積非交差再配置問題について述べる．

以下，4.1で本論文で用いる用語および記号を定義し，4.2で矩形の最小面積非交差再配置問題 LADR を定義する．4.3では問題 LADR が整数座標系で NP 完全であることを証明する．更に 4.4では，三末らのアルゴリズムを紹介し，このアルゴリズムより面積の小さい再配置を求めるアルゴリズムを提案する．

4.1. 諸定義

Rを任意の n 個の矩形 $v_1, v_2, \cdots, v_n$からなる矩形集合とする．各矩形 v_iはそれぞれ正整数の x 方向の幅 w_i，y方向の幅 h_iを持つ．矩形 v_iを $\langle w_i, h_i \rangle$ と表すことがある．ここで，矩形集合 Rの 2 次元空間上の配置を考える．矩形集合 Rの配置 π_Rは，整数座標系を用いるとき写像 $\pi_R : R \to \mathcal{Z}^2$，実数座標系を用いるとき写像 $\pi_R : R \to \mathcal{R}^2$ で表され，各矩形 $v_i \in R$の中心の x 座標値，y座標値がそれぞれ x_i，y_iのとき，$\pi_R(v_i) = (x_i, y_i)$ と定義する．ここで $\mathcal{Z}^2$，$\mathcal{R}^2$はそれぞれ，2 次元整数座標空間，2 次元実数座標空間である．本論文で扱う配置では，矩形 v_iの長さ w_i(x 方向の幅) の辺は 2 次元平面の x 軸と水平になるように置くものとし，矩形の回転は許さない．

Rに含まれる矩形 v_iの，配置π_Rにおける左右の境界の x 座標をそれぞれ $left_\pi(v_i)$，$right_\pi(v_i)$，上下の境界の y座標をそれぞれ $top_\pi(v_i)$，$bottom_\pi(v_i)$ と定義する．π_R についても同様に，次のように定義する．

$$
\begin{aligned}
left(\pi_R) &= \min_{v_i \in R} left_\pi(v_i) \\
right(\pi_R) &= \max_{v_i \in R} right_\pi(v_i) \\
top(\pi_R) &= \min_{v_i \in R} top_\pi(v_i) \\
bottom(\pi_R) &= \max_{v_i \in R} bottom_\pi(v_i)
\end{aligned}
$$

更に，π_Rの x 方向の幅，y方向の幅をそれぞれ $W_x(\pi_R)$，$W_y(\pi_R)$ と表し，次のよ

うに定義する.

$$\begin{aligned} W_x(\pi_R) &= right(\pi_R) - left(\pi_R) \\ W_y(\pi_R) &= bottom(\pi_R) - top(\pi_R) \end{aligned}$$

矩形と同様に，π_Rに対して $\langle W_x(\pi_R), W_y(\pi_R)\rangle$ という表記を用いる．π_Rの面積 $S(\pi_R)$ を $S(\pi_R) = W_x(\pi_R) \cdot W_y(\pi_R)$ と定義する.

4.2. 矩形の最小面積非交差再配置問題

矩形の最小面積非交差再配置問題を定義する.

[定義 4.1] (最小面積非交差再配置問題) 矩形集合 Rとその配置π_Rおよび正の整数 Kが与えられる．任意の2つの矩形が同じ中心座標を持たないとする．すなわち，任意の $v_i, v_j \in R(i \neq j)$ について，$\pi_R(v_i) \neq \pi_R(v_j)$ である．このとき，以下の制約 (1),(2) を満たし，かつ $S(\pi'_R) \leq K$となる Rの配置π'_Rが存在するかどうかを判定する問題を，最小面積非交差再配置問題という．但し，以下の制約 (1),(2) では，各 $v_i \in R$について$\pi_R(v_i) = (x_i, y_i)$，$\pi'_R(v_i) = (x'_i, y'_i)$ とする.
(1) 再配置前後で直交順序を保存する．つまり，各矩形 $v_i, v_j \in R$に対して，次式が成り立つ.

$$\begin{aligned} x_i < x_j &\Leftrightarrow x'_i < x'_j\ , \\ x_i = x_j &\Leftrightarrow x'_i = x'_j\ , \\ y_i < y_j &\Leftrightarrow y'_i < y'_j\ , \\ y_i = y_j &\Leftrightarrow y'_i = y'_j\ . \end{aligned}$$

(2) 再配置後，矩形は互いに交差しない．つまり，各矩形 $v_i, v_j \in R(i \neq j)$ に対して次式が成り立つ.

$$|x'_i - x'_j| \geq \frac{w_i + w_j}{2} \quad \text{または} \quad |y'_i - y'_j| \geq \frac{h_i + h_j}{2}$$

以下では，最小面積非交差再配置問題を LADR と表し，整数座標系を用いるとき，特に ILADR と表す.

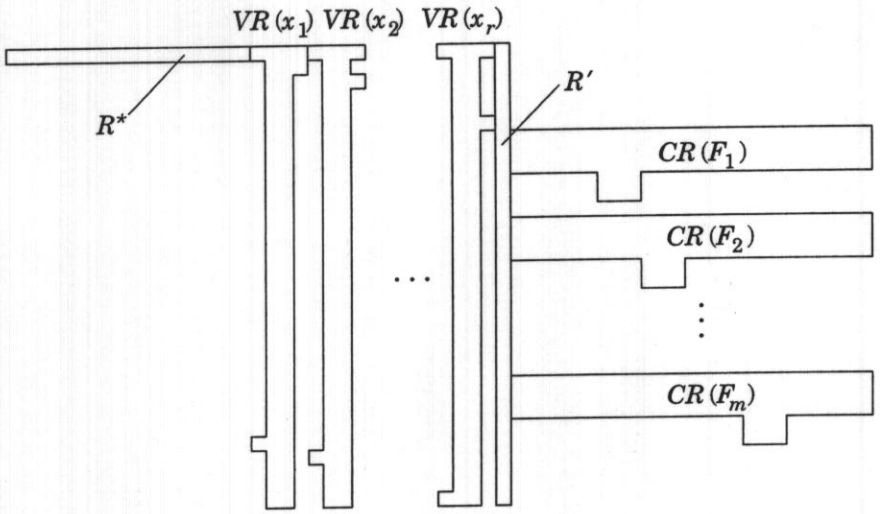


図 4.1 $R(E)$ の初期配置$\pi_{R(E)}$の概略

4.3. 矩形の最小面積非交差再配置問題の計算複雑度

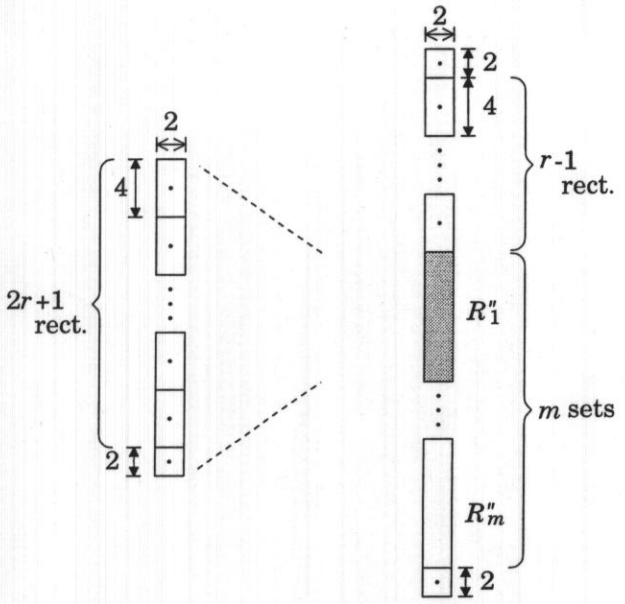
本節では，問題ILADRがNP完全であることを示す．ILADRがNPに属するのは明らかである．ここでは，まずILADRのNP完全性を証明するために，NP完全問題である3-SAT[14]をILADRに帰着する．

3-SATについては，で記述した通りであるが，ここではブール変数の集合Xはr個の変数を含むものとする．つまり，$X = \{x_1, x_2, \cdots, x_r\}$とする．

4.3.1 3-SAT から問題 ILADR への帰着

3-SATから問題ILADRへの帰着性を示す．

ブール代数式Eから矩形集合$R(E)$とその初期配置$\pi_{R(E)}$を定義する．最も左上の矩形の左上角を座標原点$(0,0)$とする．$R(E)$は各ブール変数x_kに対応する矩形集合$VR(x_k)$，各節F_iに対応する矩形集合$CR(F_i)$，矩形集合R'および矩形R^*からなる．$R(E)$の初期配置$\pi_{R(E)}$の概要を図4.1に示す．$\pi_{R(E)}$は3-SATの各ブール変数x_kがすべてtrueである場合に対応する．以下，矩形集合$R(E)$とその初期配置$\pi_{R(E)}$について順に述べる．なお，矩形集合$VR(x_k)$に対し，初期配



(a) R''_i (b) R'

図 4.2 R''_iおよびR'の配置

置$\pi_{R(E)}$の定義域を$VR(x_k)$に制限した初期配置$\pi_{R(E)}|_{VR(x_k)}$を，ここでは簡単のため単に$\pi_{VR(x_k)}$と表記する．$R(E)$に含まれる他の矩形集合に対しても同様の表記を用いる．

R'は$VR(x_k)$や$CR(F_i)$の配置を制限するためのもので，矩形集合$R''_i(i = 1,\cdots,m)$を含んでいる．R''_iおよびR'の初期配置を図4.2に示す．図の中で用いているrは3-SATにおけるブール変数の個数である．

各変数x_kに対応する矩形集合$VR(x_k)$の初期配置$\pi_{VR(x_k)}$を図 4.3(a) に示す．$VR(x_k)$はm個の$\langle 2, 2(4r+3)\rangle$の矩形$vc_{k,i}$を含んでおり，初期配置ではこれらは縦にすき間なく配置される．$\pi_{R(E)}$では，各$VR(x_k)$に対して$top(\pi_{VR(x_k)}) = top(\pi_{R'})$となるように配置する (図 4.1参照)．

次に，各節F_iに対応する矩形集合$CR(F_i)$について述べる．まず，$CR(F_i)$に含まれる各リテラル$y_{i,j}(j = 1,2,3)$に対応する矩形集合$LR(y_{i,j})$について述べる．$LR(y_{i,j})$は2つの種類がある (図 4.4, 4.5)．いずれの場合も，$LR(y_{i,j})$は$\langle 2,2\rangle$の矩形$vs_{i,j}$を含んでいる．$LR(y_{i,j})$の初期配置$\pi_{LR(y_{i,j})}$はそれぞれ，ある変数x_kに

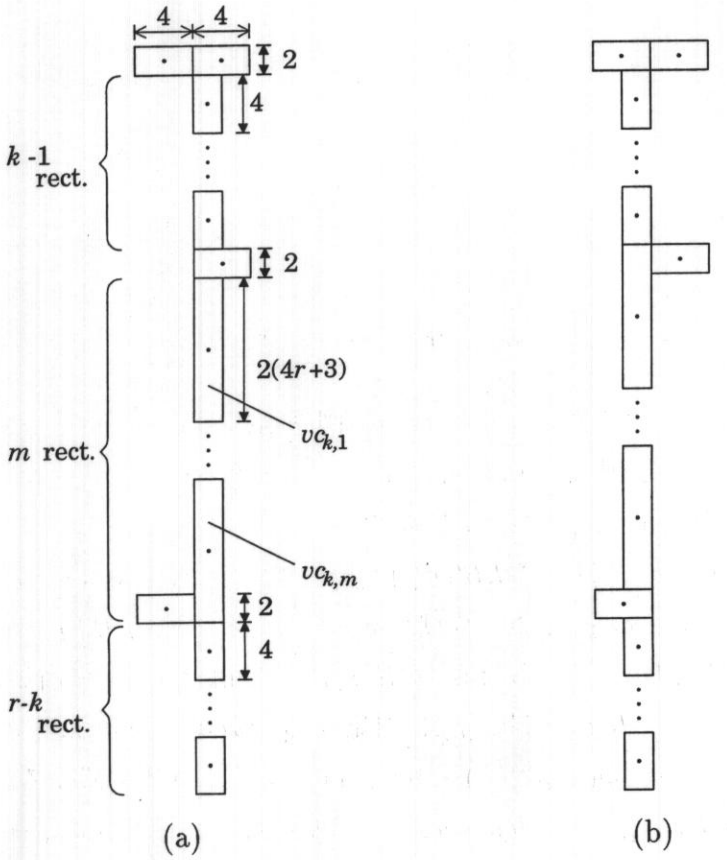


図 4.3 $VR(x_k)$ の配置

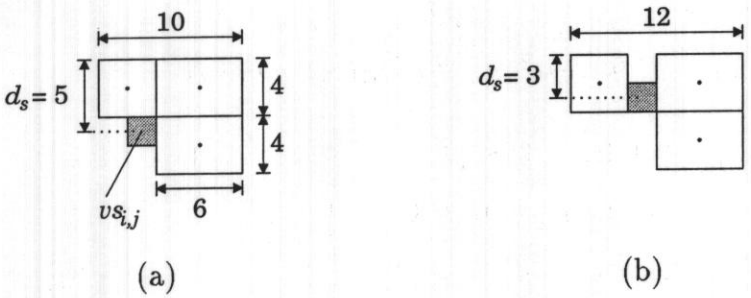
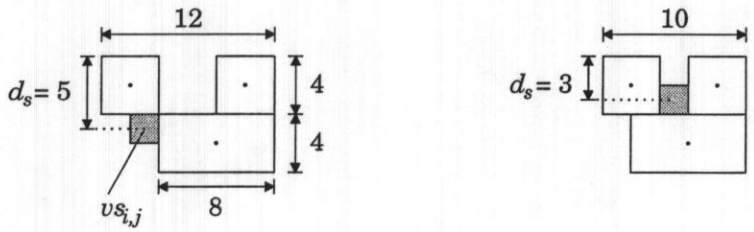


図 4.4 $LR(y_{i,j})$ の配置 ($y_{i,j} = x_k$の場合)



(a) (b)

図 4.5 $LR(y_{i,j})$ の配置 ($y_{i,j} = \overline{x_k}$の場合)

対して，対応するリテラルが $y_{i,j} = x_k$ならば図 4.4(a)，$y_{i,j} = \overline{x_k}$ならば図 4.5(a) とする．$\pi_{LR(y_{i,j})}$の上端の境界の y座標と $vs_{i,j}$の y座標との差を $d_s(\pi_{LR(y_{i,j})})$ とおく．但し，ここでは $d_s(\pi_{LR(y_{i,j})})$ を単に d_sと表すことにする．

$CR(F_i)$ は，各 $LR(y_{i,j})$ および $LR'(y_{i,j})$ (j=1,2,3)，更に $\langle 36(m+i-2), 4\rangle$ の矩形 vl_i，$\langle 36(2m-i-1)+12, 4\rangle$ の矩形 vr_i を含んでいる．$LR'(y_{i,j})$ は $\langle 4,4\rangle$ の矩形 $ve_{i,j}$と $\langle 6,4\rangle$ の矩形 $vf_{i,j}$ からなる．$CR(F_i)$ の初期配置$\pi_{CR(F_i)}$を図 4.6に示す．$CR(F_i)$ に含まれる各 $LR(y_{i,j})$ は，$y_{i,j}$に対応する変数が x_kまたはその否定ならば，$top(\pi_{LR(y_{i,j})}) = bottom(\pi_{LR'(y_{i,j})}) + 4(k-1)$ となるように配置する．$LR'(y_{i,j})$ と $LR(y_{i,j})$ は，$left(\pi_{LR(y_{i,j})}) = left_\pi(ve_{i,j})$，$right(\pi_{LR(y_{i,j})}) = right_\pi(vf_{i,j})$ となるように配置する．

各 $VR(x_k)$，$CR(F_i)$，R'および R^*から $R(E)$ の初期配置$\pi_{R(E)}$を図 4.1のように作る．$\pi_{R(E)}$では，$top_\pi(vl_i) = top(\pi_{R''_i})$ となるように $CR(F_i)$ と R'を配置する (図 4.7参照)．このとき，$\pi_{LR(y_{i,j})}$の $vs_{i,j}$以外の各矩形の y座標は，R'に含まれるある矩形の y座標と等しい．$y_{i,j} = x_k$またはその否定であるとすると，$\pi_{LR(y_{i,j})}$に含まれる $vs_{i,j}$と$\pi_{VR(x_k)}$に含まれる $vc_{k,i}$の y座標はそれぞれ，

$$
\begin{aligned}
(vs_{i,j}\text{の } y\text{座標}) &= 2 + 4(r-1) + \{4(2r+1)+2\}(i-1) + 4 + 4(k-1) + 5 \\
&= 4(r+k) + 2(4r+3)(i-1) + 3 \\
(vc_{k,i}\text{の } y\text{座標}) &= 2 + 4(k-1) + 2 + 2(4r+3)(i-1) + (4r+3) \\
&= 4(r+k) + 2(4r+3)(i-1) + 3
\end{aligned}
$$

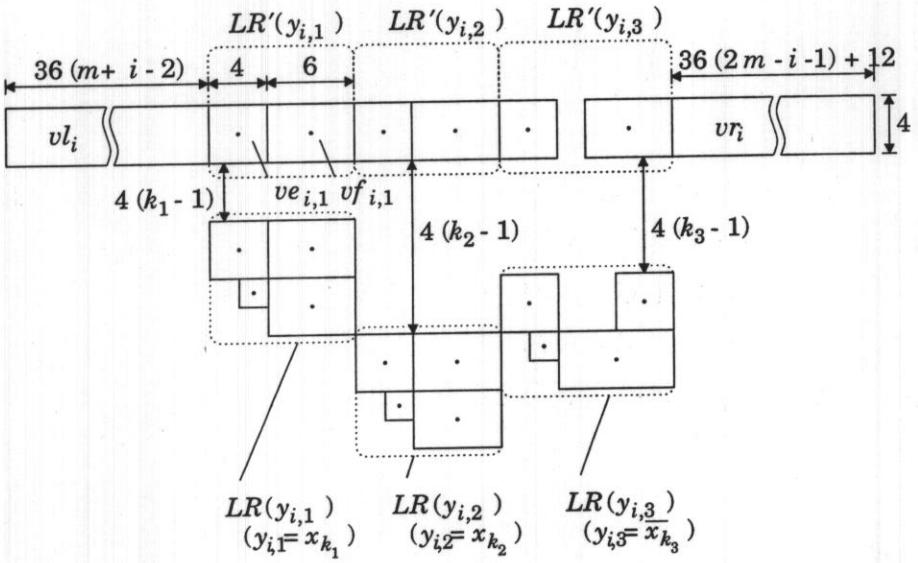


図 4.6 $CR(F_i)$ の初期配置

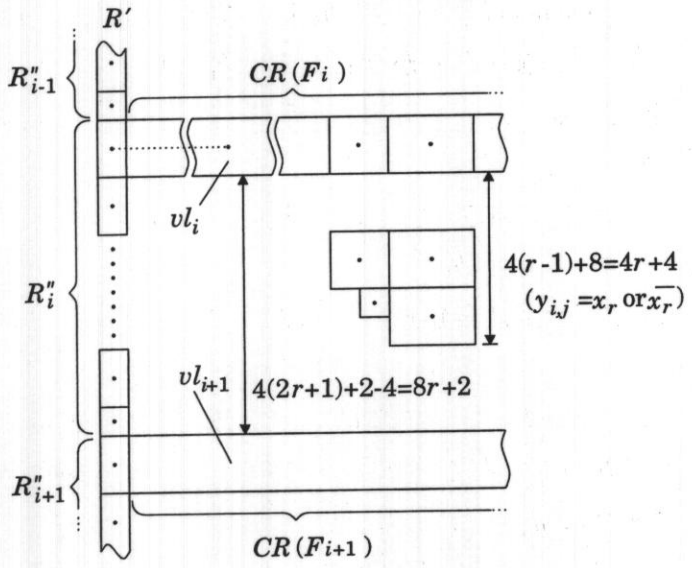


図 4.7 R' と $CR(F_i)$ の配置

であるから，$vs_{i,j}$と$vc_{k,i}$は同じy座標をもつ．従って，制約 (1) を満たす再配置においてもこれらは同じy座標をもつ．また，$CR(F_i)$と$CR(F_{i+1})$の配置では，それらに属する矩形が交差しないように十分なy座標の間隔を取っている (図 4.7参照)．更に$\pi_{R(E)}$では，$\pi_{VR(x_1)}$の左側に，上端の境界が同じになるように$\langle 32mr+8r, 2\rangle$の矩形$R^*$を配置する．

各$VR(x_k)$および$CR(F_i)$は，それぞれr，mに関する多項式の大きさおよび個数の矩形を含み，$R(E)$はそれらをそれぞれ，r個，m個含んでいるので，$R(E)$とその初期配置$\pi_{R(E)}$は多項式時間で作成できる．

Eの各変数にtrueまたはfalseを割り当てることによって得られる$R(E)$の再配置を$\pi'_{R(E)}$とする．$\pi'_{R(E)}$は直交順序を保存し，矩形の非交差を満たすように作成する．

4.3.2 証明

[補題 4.1] Eが充足可能ならばそのときに限り，$R(E)$の配置$\pi'_{R(E)}$が存在し，$\pi'_{R(E)}$は制約 (1),(2) を満たし，$W_x(\pi'_{R(E)}) \leq 32mr+108m+16r-60$，$W_y(\pi'_{R(E)}) = 8mr+6m+4r$ である．

(証明) ($\Rightarrow$) まず，Eが充足可能であると仮定する．このとき，Eを充足可能にするような各変数への真偽割り当てが存在する．各変数x_kの割り当てに対応する$VR(x_k)$の配置を決める．x_k = true ならば図 4.3(a)，x_k = false ならば同図 (b) とし，$R(E)$の配置$\pi'_{R(E)}$を考える．図 4.3(a) の初期配置に対して，同図 (b) の配置が直交順序を満たしていることに注意する．このとき，$W_x(\pi'_{VR(x_k)}) = 8$，$W_y(\pi'_{VR(x_k)}) = 8mr+6m+4r$である．

各$VR(x_k)$の配置より，R'は各矩形がすき間なく縦列に配置され，これにより各$LR(y_{i,j})$ $(1 \leq i \leq m, j = 1,2,3)$に属する各矩形の$y$座標が決定する．$W_y(\pi'_{LR(y_{i,j})}) = 8$であり，$d_s = 5$または3で幅が最小となる配置は，図 4.4および図 4.5で示す配置のみである．リテラル$y_{i,j}$に対応する変数がtrueであれば$d_s = 5$，falseであれば$d_s = 3$となる．このとき，$y_{i,j}$の値がtrueであれば図 4.4(a) または図 4.5(b) のいずれかの配置となり，$W_x(\pi'_{LR(y_{i,j})}) = 10$となる．また，$y_{i,j}$の値がfalseのときは$W_x(\pi'_{LR(y_{i,j})}) = 12$となる．

仮定より，各F_iの3つのリテラルのうち少なくとも一つはtrueであるので，

$W_x(\pi'_{LR(y_{i,j})}) \leq 34$ となり，$W_x(\pi'_{CR(F_i)}) \leq 108m-62$ となる．よって $R(E)$ 全体は，$W_x(\pi'_{R(E)}) \leq 32mr+16r+2+(108m-62) = 32mr+108m+16r-60$，$W_y(\pi'_{R(E)}) = W_y(\pi'_{VR(x_k)}) = 8mr+6m+4r$ となる．図 4.8に $E=(x_1 \vee x_2 \vee x_3) \wedge (\overline{x_2} \vee \overline{x_3} \vee \overline{x_4}) \wedge (\overline{x_1} \vee \overline{x_4} \vee x_2)$ の場合の $R(E)$ の配置例を示す．

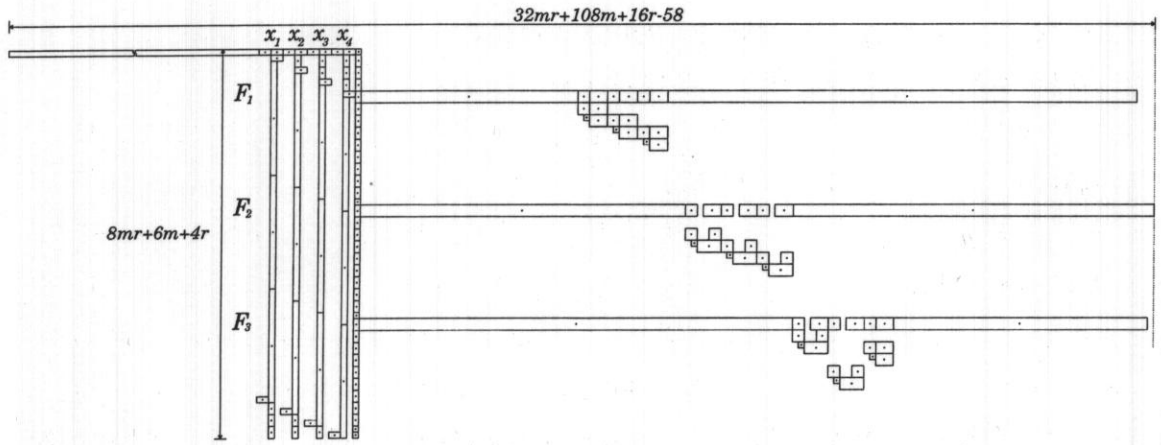
($\Leftarrow$) 次に，$W_x(\pi'_{R(E)}) \leq 32mr+108m+16r-60$，$W_y(\pi'_{R(E)}) = 8mr+6m+4r$ の $R(E)$ の配置$\pi'_{R(E)}$が存在すると仮定する．

F_iに含まれるリテラルが x_kとその否定$\overline{x_k}$の両方を含んでいるならば，F_iは任意の割り当てで true である．F_iに含まれるリテラルがすべて異なる変数の場合を考える．各 $y_{i,j}(j=1,2,3)$ は x_{k_j}または$\overline{x_{k_j}}$とする．$VR(x_{k_j})$ と $LR'(y_{i,j})$ の$\pi'_{R(E)}$における幅の和は，$W_x(\pi'_{R(E)}) - \{\sum_{k \neq k_1,k_2,k_3} W_x(\pi_{VR(x_k)}) + W_x(\pi_{R'}) + (vl_{i,j}$の幅$) + (vr_{i,j}$の幅$)\}$ であるから，i に関わらず $(32mr+108m+16r-60) - \{32mr+8r+8(r-3)+2+36(m+i-2)+36(2m-i-1)+12\} = 58$ 以下である．よって，少なくとも 1 つの jに対して，$VR(x_{k_j})$ と $LR'(y_{i,j})$ の W_xの和が 19 以下である．それぞれ $W_x(\pi'_{VR(x_{k_j})}) \geq 8$，$W_x(\pi'_{LR'(y_{i,j})}) \geq 10$ なので，$W_x(\pi'_{VR(x_{k_j})}) \leq 9$，$W_x(\pi'_{LR'(y_{i,j})}) \leq 11$ である．

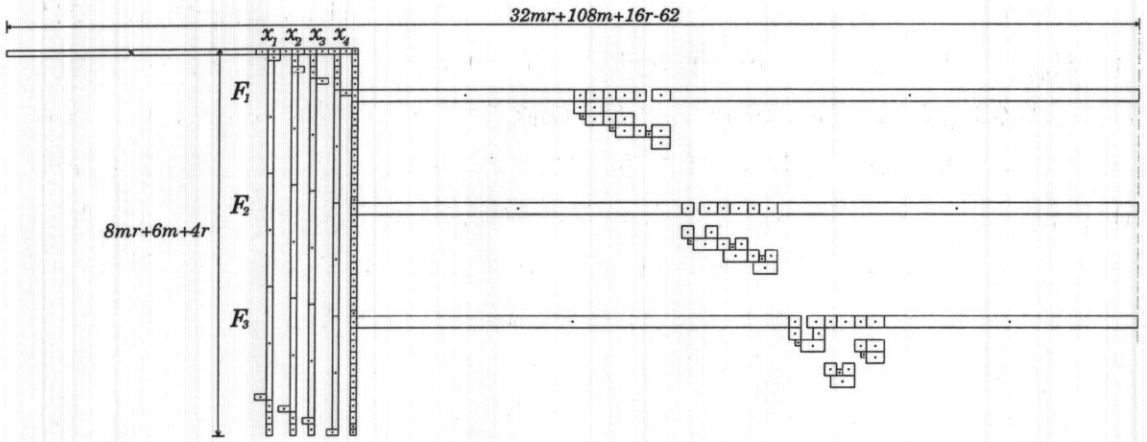
初期配置πでは，$W_y(\pi_{VR(x_{k_j})}) = 8mr+6m+4r$である (図 4.3)．再配置$\pi'$では $W_y(\pi'_{R(E)}) = W_y(\pi'_{VR(x_{k_j})})$ かつ $W_x(\pi'_{VR(x_{k_j})}) \leq 9$ であり，図 4.3より，$vc_{k_j,1},\cdots,vc_{k_j,m}$は必ず y方向に間隔が開くことなく配置され，それぞれの y座標は図 4.3(a) または (b) のいずれかである．また R'に関しても，$W_y(\pi_{R'}) = 8mr+6m+4r = W_y(\pi'_{R(E)})$ であることから，再配置π'_Rでは R'は y方向に間隔が開くことなく配置され，このことにより各 $LR(y_{i,j})$ $(1 \leq i \leq m, j=1,2,3)$ に属する $vs_{i,j}$以外の矩形の y座標は初期配置と変わらず，また $W_y(\pi'_{LR(y_{i,j})}) = 8$ となる．

$y_{i,j}$が x_kまたは$\overline{x_k}$であるとする．このとき，$vs_{i,j}$と $vc_{k,i}$の y座標が等しいことより，$VR(x_k)$ に属する $vc_{k,1},\cdots,vc_{k,m}$の y座標が図 4.3(a) ならば，$LR(y_{i,j})$ に対して $d_s=5$，同図 (b) ならば $d_s=3$ となる (図 4.4，図 4.5参照)．$d_s=5$ のとき $y_{i,j}=\overline{x_k}$ならば，$W_x(\pi'_{LR(y_{i,j})}) \geq 12$ となり，このとき $W_x(\pi'_{LR'(y_{i,j})}) \geq 12$ となる．また，$d_s=3$ のとき $y_{i,j}=x_k$ならば，同様に $W_x(\pi'_{LR'(y_{i,j})}) \geq 12$ となる．従って，$W_x(\pi_{LR'(y_{i,j})}) \leq 11$ であれば，$d_s=5$ のとき $y_{i,j}=x_k$，$d_s=3$ のとき $y_{i,j}=\overline{x_k}$である．

ここで，$vc_{k,1},\cdots,vc_{k,m}$の y座標が図 4.3(a) または (b) のいずれかである x_kに対



(a) 初期配置 ($x_1 = x_2 = x_3 = x_4 = \text{true}$)



(b) 再配置 ($x_1 = x_2 = \text{true}, x_3 = x_4 = \text{false}$)

図 4.8 $R(E)$ の配置例

して，図 4.3(a) であれば $x_k = \text{true}$，同図 (b) であれば $x_k = \text{false}$ の割り当てを考える．このとき，各 F_i に対してある j が存在し，$W_x(\pi'_{VR(x_{k_j})}) \leq 9$，$W_x(\pi'_{VR(x_{k_j})}) \leq 11$，$y_{i,j} = x_{k_j}$ または $\overline{x_{k_j}}$ である．$W_x(\pi'_{VR(x_{k_j})}) \leq 9$ より，$vc_{k_j,1}, \cdots, vc_{k_j,m}$ の y 座標は，図 4.3(a) または (b) である．図 4.3(a) ならば，$x_{k_j} = \text{true}$ および $d_s = 5$ となり，$W_x(\pi'_{LR'(y_{i,j})}) \leq 11$ より，$y_{i,j} = x_{k_j}$，$y_{i,j} = \text{true}$ である．また，同図 (b) ならば同様に，$x_{k_j} = \text{false}$，$d_s = 3$ であり，$y_{i,j} = \overline{x_{k_j}}$，$y_{i,j} = \text{true}$ である．

よって，各 F_i の少なくとも 1 つのリテラルは true となり，E は true となるので充足可能である． □

[補題 4.2] E が充足可能ならばそのときに限り，配置 $\pi'_{R(E)}$ が存在し，$\pi'_{R(E)}$ は制約 (1),(2) を満たし，$S(\pi'_{R(E)}) \leq (32mr + 108m + 16r - 60)(8mr + 6m + 4r)$ である．

(証明) ($\Rightarrow$) $S = (32mr + 108m + 16r - 60)(8mr + 6m + 4r)$ とおく．E が充足可能ならば，補題 4.1 より，配置 $\pi'_{R(E)}$ が $S(\pi'_{R(E)}) \leq S$ でできる．

($\Leftarrow$) $S(\pi'_{R(E)}) \leq S$ ならば $R(E)$ の定義より $W_x(\pi'_{R(E)}) \geq 32mr + 108m + 16r - 64$，$W_y(\pi'_{R(E)}) \geq 8mr + 6m + 4r$ である．$W_y(\pi'_{R(E)}) > 8mr + 6m + 4r$ ならば，

$$\begin{aligned} S(\pi'_{R(E)}) &\geq (32mr + 108m + 16r - 64)(8mr + 6m + 4r + 1) \\ &= (32mr + 108m + 16r - 60)(8mr + 6m + 4r) \\ &\quad -4(8mr + 6m + 4r) + 32mr + 108m + 16r - 64 \\ &= S + 84m - 64. \end{aligned}$$

$m \geq 1$ なので $84m - 64 > 0$ となり，$S(\pi'_{R(E)}) > S$ となる．

よって，$S(\pi'_{R(E)}) \leq S$ ならば，$W_y(\pi'_{R(E)}) = 8mr + 6m + 4r$ であり，$W_x(\pi'_{R(E)}) \leq 32mr + 108m + 16r - 60$ であるので，補題 4.1 より E は充足可能である． □

ILADR が NP に属することは明らかなので，補題 4.2 より次の定理が導かれる．

[定理 4.1] ILADR は NP 完全である．

4.3.3 実数座標系における再配置問題

RLADR も ILADR と同様に 3-SAT から変換することができる．

RLADR では，$LR(y_{i,j})$ および $LR'(y_{i,j})$ を 図 4.9, 4.10 とする．いずれの場合も，$LR(y_{i,j})$ は $\langle 2,2 \rangle$ の矩形 $vs_{i,j}$ を含んでいる．$LR(y_{i,j})$ の初期配置 $\pi_{LR(y_{i,j})}$ はそれぞ

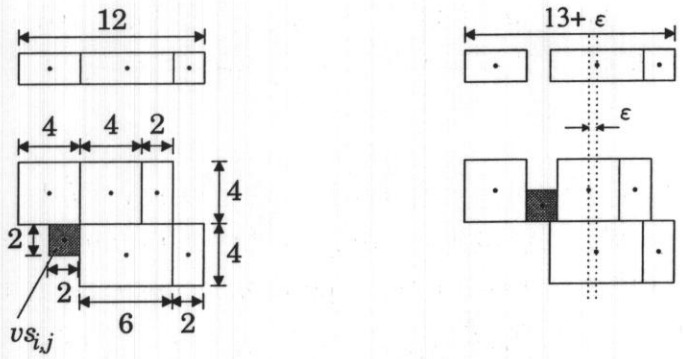


図 4.9 RLADR における $LR(y_{i,j})$ および $LR'(y_{i,j})$ の配置 ($y_{i,j} = x_k$の場合)

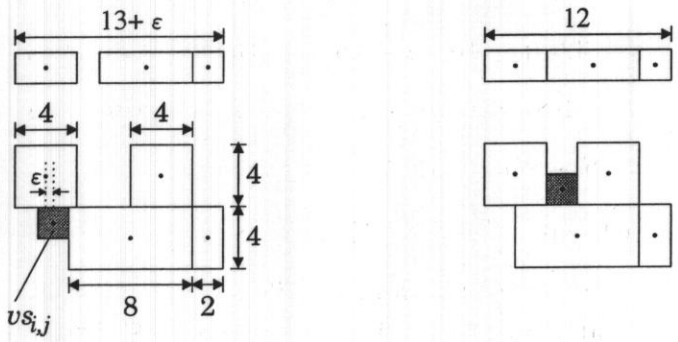


図 4.10 RLADR における $LR(y_{i,j})$ および $LR'(y_{i,j})$ の配置 ($y_{i,j} = \overline{x_k}$の場合)

れ，ある変数x_kに対して，対応するリテラルが$y_{i,j} = x_k$ならば図 4.9(a)，$y_{i,j} = \overline{x_k}$ならば図 4.10(a) とする．実数座標系であるから，図 4.9(b) および 4.10(a) では，直交順序を満たす範囲で矩形の座標差を 1 以下にすることができる．$W_y(\pi_{LR(y_{i,j})}) = 8$ であり，幅が最小となる配置は，図 4.9および図 4.10で示す配置のみである．このとき，$y_{i,j}$の値が true であれば図 4.4(a) または図 4.5(b) のいずれかの配置となり，$W_x(\pi_{LR'(y_{i,j})}) = 12$ となる．また，$y_{i,j}$の値が false のときは $W_x(\pi_{LR'(y_{i,j})}) = 13 + \epsilon$ となる ($\epsilon << 1$)．

この $LR(y_{i,j})$ および $LR'(y_{i,j})$ を用いることで，ILADR の場合と同様に，3-SAT から RLADR に変換することができる．よって，次の定理が成り立つ．

[定理 4.2] RLADR は NP 困難である．

4.4. 再配置アルゴリズム

文献 [24] では，矩形集合とその配置が与えられたときに，直交順序を保存した面積最小の非交差再配置を求める発見的手法として，アルゴリズム **PFS**(Push Force-Scan) が提案されている．本章では，**PFS** アルゴリズムを基にした，面積最小の再配置を求める発見的アルゴリズム **PFS′**を示す．

4.4.1 Push Force-Scan アルゴリズム

アルゴリズム **PFS** は矩形間の交差を除くために，矩形間力と呼ばれる値を用いる．矩形間力は任意の 2 つの矩形間で定義されるベクトルである．矩形 v_i，v_j 間の 矩形間力 $f_{i,j}$は，これらの矩形が交差するとき，v_i に対して v_jを相対的に $f_{i,j}$だけ移動すれば，v_i,v_jが交差しないことを保証する．$f_{i,j}$の向きは，v_i，v_j間の交差を除くだけでなく，過去の実験などで得られた経験に従って，直交順序を保存してできるだけ配置面積が小さくなるように選ばれる．

ここではまず，矩形間力および 本節で用いる記号について定義する．矩形集合 R およびその配置π_Rは与えられているものとする．

配置π_Rにおける矩形 $v_i(\in R)$ の中心座標を (x_i, y_i) と表す．すなわち，$\pi_R(v_i) =$

(x_i, y_i) とする．矩形 $v_i, v_j (\in R)$ 間の座標の差 $\triangle x_{i,j}$,$\triangle y_{i,j}$ を次のように定義する．

$$\begin{aligned} \triangle x_{i,j} &= x_j - x_i \\ \triangle y_{i,j} &= y_j - y_i \end{aligned}$$

矩形 v_i, v_j が交差しているならばそのときに限り，

$$|\triangle x_{i,j}| < \frac{w_i + w_j}{2} \quad \text{かつ} \quad |\triangle y_{i,j}| < \frac{h_i + h_j}{2}$$

が成り立つ．L を v_i および v_j の中心座標を結ぶ直線とする．交差を解消するために，直交座標を満たす範囲で，v_i, v_j が交差せずに接触するまで，L に沿って v_j を移動させるとする．このとき，移動後の v_j の座標を (x'_j, y'_j) とする．矩形間力 $f_{i,j} = (f^x_{i,j}, f^y_{i,j})$ は，(x_j, y_j) から (x'_j, y'_j) へのベクトルとして，以下のように定義する．$g_{i,j}$ を L の勾配，すなわち，$g_{i,j} = \triangle y_{i,j} / \triangle x_{i,j}$ とする．$\triangle x_{i,j} = 0$ ならば $g_{i,j} = \infty$ とする．また，$G_{i,j}$ を $(h_i + h_j)/(w_i + w_j)$ とする．

a) v_i, v_j が y 方向の境界で接触する場合，すなわち，$G_{i,j} \geq g_{i,j} > 0$，$-G_{i,j} \leq g_{i,j} < 0$，$g_{i,j} = 0$ のいずれかの場合，

$$f^x_{i,j} = \frac{\triangle x_{i,j}}{|\triangle x_{i,j}|} \left(\frac{w_i + w_j}{2} - |\triangle x_{i,j}| \right), f^y_{i,j} = f^x_{i,j} \cdot g_{i,j}.$$

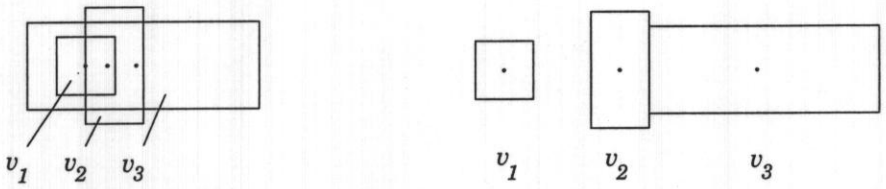
b) v_i, v_j が x 方向の境界で接触する場合，すなわち，$(G_{i,j} < g_{i,j})$ かつ $(g_{i,j} > 0)$，または $(-G_{i,j} > g_{i,j})$ かつ $(g_{i,j} < 0)$ の場合，

$$f^y_{i,j} = \frac{\triangle y_{i,j}}{|\triangle y_{i,j}|} \left(\frac{h_i + h_j}{2} - |\triangle y_{i,j}| \right), f^x_{i,j} = \frac{f^y_{i,j}}{g_{i,j}}.$$

次に，**PFS** の詳細について述べる．**PFS** は制約 (1),(2) を満たす再配置を $O(n^2)$ 時間で求める $(n = |R|)$．**PFS** では，まず x 方向の移動である Horizontal-PFS，次に y方向の移動である Vertical-PFS を実行する．Horizontal-PFS では，任意の矩形対 v_i,v_jに対し，v_i から見て v_jを少なくとも $f_{i,j}$移動させることを保証する．Vertical-PFS は 適用する方向を除いて Horizontal-PFS と同じであるため，ここでは Horizontal-PFS についてのみ説明する．

[**Algorithm** Horizontal-PFS]
begin
  $i := 1$;
  $l := 1$;
  while $(i \leq n)$ **do begin**
    $k := \max\{j | x_i = x_j\}$; /*$x_i = x_{i+1} = \cdots = x_k$ */
    $\delta := 0$;
    if $(x_i > x_1)$ **then**
      $\delta := \max(0, \max_{l \leq m < i \leq j \leq n} f^x_{m,j})$;
    for $j := i$ **to** n **do**
      $x_j := x_j + \delta$;
    $l := i$;
    $i := k + 1$;
  end;
end.

図 4.11 アルゴリズム Horizontal-PFS



(a) 初期配置　　(b) **PFS** による配置

図 4.12 すき間が生じる例

Horizontal-PFS の概要を図 4.11に示す．ここで，$x_1 \leq x_2 \leq \cdots \leq x_n$ と仮定する．Horizontal-PFS では $v_1, v_2, \cdots, v_n$ の順で各矩形の再配置における x 座標を決める．同じ x 座標を持つ矩形の x 座標は同時に決まる．同じ x 座標を持つ矩形 $v_i, \cdots, v_k$ の x 座標を決めるとき，これらの矩形だけを動かすのではなく，すべての $v_j (i \leq j \leq n)$ を同時に移動させる．この移動距離は，while ループの一回前に移動した矩形 v_m $(l \leq m < i)$ と v_j $(i \leq j \leq n)$ 間の矩形間力 $f^x_{m,j}$ の最大値とする．

PFS は 正方向のみの移動に制限しているが，この制限を解消するために，三末らは別の手法である *Push-Pull Force-Scan* アルゴリズムを提案している．このアルゴリズムは負方向の移動を許しているが，非交差を保証していない．よって，ここではこのアルゴリズムについては議論しない．

4.4.2 Push Force-Scan アルゴリズムの改良

PFS ではいくつかの場合において，矩形間に余分なすき間を生じる．図 4.12の例において，同図 (a) は初期配置，(b) は **PFS** による再配置を表している．この例では，v_2 と v_3 は $f^x_{1,3}$ によって右に移動し，その後，v_3 が $f^x_{2,3}$ によって再び移動するため，v_1 と v_2 の間に余分なすき間を生じる．

ここでは，**PFS** で得られた配置面積以下の配置を得るアルゴリズム **PFS′**を提案する．**PFS′**の計算量は **PFS** と同じく $O(n^2)$ である．**PFS′**は **PFS** と同様に Horizontal-PFS′, Vertical-PFS′ の順で実行する．ここでは Horizontal-PFS′についてのみ説明する．

再び，図 4.12の例について考える．この場合，v_2 に対しては $f^x_{1,2}$ による移動，

[**Algorithm** Horizontal-PFS′]

begin

$i := 1;$

$\sigma := 0;$

$lmin := 1;$

while $(i \leq n)$ **do begin**

$k := \max\{j | x_i = x_j\}$; /* $x_i = x_{i+1} = \cdots = x_k$ */

$\gamma := 0;$

if $(x_i > x_1)$ **then**

$\gamma := -\infty;$

for $m := i$ **to** k **do begin**

$\gamma'' := \max_{1 \leq j \leq i-1} (\gamma_j + f^x_{j,m});$

$\gamma' :=$

$$\begin{cases} \sigma & \text{if } l\_bnd(v_m, x_m) + \gamma'' < l\_bnd(v_{lmin}, x_{lmin}) \\ \gamma'' & \text{otherwise} \end{cases}$$

$\gamma := \max(\gamma, \gamma');$

end;

for $m := i$ **to** k **do begin**

$\gamma_m := \gamma;$

$x_m := x_m + \gamma_m;$

if $l\_bnd(v_m, x_m) < l\_bnd(v_{lmin}, x_{lmin})$ **then**

$lmin := m;$

end;

$\sigma := \sigma + \max(0, \max_{i \leq m \leq k < j \leq n} f^x_{m,j});$

$i := k + 1;$

end;

end.

図 4.13 アルゴリズム Horizontal-PFS′

v_3 に対しては $\max\{f^x_{1,2}+f^x_{2,3}, f^x_{1,3}\}$ による移動を考えれば十分である．**PFS′**はこの考え方を一般化したものである．$x_1 \leq x_2 \leq \cdots \leq x_n$ と仮定する．Horizontal-PFS′の概要を図 4.13に示す．関数 $l\_bnd(v_i, x_i)$ は矩形 v_iが座標 x_iをもつときの左端の x 座標である．Horizontal-PFS′ では $v_1, v_2, \cdots, v_n$ の順で再配置における矩形の x 座標を決める．同じ x 座標を持つ矩形の x 座標は同時に決まる．同じ x 座標を持つ矩形 $v_i, \cdots, v_k$ の x 座標を決めるとき，移動距離はある特別な場合を除いて $v_1, v_2, \cdots, v_k$ にのみ依存する．これが **PFS** とは異なる部分である．ここでは $v_i, \cdots, v_k$ の x 座標を決める方法について述べる．$v_1, v_2, \cdots v_{i-1}$ の x 座標が既に決まっていると仮定する．γ_j を **PFS′**によって v_j が x 方向に移動する距離とする．後で述べる特別な場合を除いて，Horizontal-PFS′ は$\gamma_m (i \leq m \leq k)$ を $\gamma_j + f^x_{j,m}$ の最大値として決定する．$(1 \leq j < i \leq m \leq k)$

特別な場合について述べる．σ_m を，**PFS** において $v_m (i \leq m \leq k)$ が右に移動した距離とする．**PFS′**では，矩形 v_m が γ_m だけ移動することによって，v_m の左境界が x 座標の決まっている v_m以外の矩形より左になるように置かれることがある．このとき，配置面積は**PFS′**よりも大きくなる可能性があるため，このような場合に限って，移動距離として γ_m の代わりに **PFS** で用いられる σ_m を用いることにする．

4.4.3 アルゴリズムの有効性

ここでは，**PFS′**が制約 (1),(2) を満たし，**PFS′**による配置面積が **PFS** による配置面積以下であることを示す．

π'_R と π''_R をそれぞれ **PFS**，**PFS′**を実行した後の R の配置とする．x_i，x'_i，x''_i をそれぞれ初期配置，π'_R，π''_R における v_i の x 座標とおく $(1 \leq i \leq n)$．**PFS** における v_iの x 移動距離σ_iは，

$$\begin{aligned} \sigma_i &= \delta_0 + \delta_1 + \cdots + \delta_{i-1} \\ \delta_i &= \begin{cases} 0 & \text{if } i = 0 \text{ or } x_i = x_{i+1} \\ \max(0, \max\limits_{l \leq m \leq i < j \leq n} f^x_{m,j}) & \text{if } x_i < x_{i+1} \end{cases} \end{aligned}$$

l は $x_i = x_m$ を満たす最小の m である．一方，**PFS′**における v_i の移動距離 γ_i

は 以下のように計算する.

$$\begin{aligned}
\gamma_i'' &= \max_{1\le j<l}(\gamma_j + f^x_{j,i}) \\
\gamma_i' &= \begin{cases} \sigma_i & \text{if } l_bnd(v_i, x_i) + \gamma_i'' < \min_{j<l} l_bnd(v_j, x_j + \gamma_j) \\ \gamma_i'' & \text{otherwise} \end{cases} \\
\gamma_i &= \max_{x_i = x_m} \gamma_m'
\end{aligned}$$

l は $x_i = x_m$ を満たす最小の m である.

[補題 4.3] すべての $i(1 \le i \le n)$ について, (a) $\sigma_i \ge \gamma_i''$ および (b) $x_i' \ge x_i''$ が成り立つ.

(証明) i に関する帰納法で証明する. まず, $x_1 = x_i$ である i について, $\gamma_i = \sigma_i'' = \sigma_i = 0$ が成り立つ. 故に, $x_i' = x_i + \sigma_i = x_i + \gamma_i = x_i''$ が成り立つ. $x_l, \cdots, x_k$ を同じ x 座標を持つ最大の列とする. $j < l$ である j について, $x_j' \ge x_j''$, すなわち, $\sigma_j \ge \gamma_j$ と仮定する. $l \le i \le k$ であるすべての i について, x_iの x 座標はすべて等しいので, $\gamma_i = \gamma_l$ および $\sigma_i = \sigma_l$ が成り立つ. よって, $\sigma_l \ge \gamma_l$ を示せば十分である.

$l \le i \le k$ である i について,

$$\gamma_i'' = \max_{1\le j<l}(\gamma_j + f^x_{j,i}) \le \max_{1\le j<l}(\sigma_j + f^x_{j,i})$$

が成り立つ. l_jおよび k_j を それぞれ, $x_{l_j} = x_j$ および $x_{k_j} = x_j$ を満たす最小および最大の添字とおく.

$$\begin{aligned}
f^x_{j,m} &\le \max_{l_j \le j' \le k_j} f^x_{j',m} \\
&\le \max_{l_j \le j' \le k_j < m' \le n} f^x_{j',m'} \qquad (\because k_j < m) \\
&\le \max(0, \max_{l_j \le j' \le k_j < m' \le n} f^x_{j',m'}) \\
&= \delta_{k_j}
\end{aligned}$$

$\sigma_j \le \sigma_{k_j}$ より, $\sigma_i \ge \gamma_i''(l \le i \le k)$ であることを示す.

$$\begin{aligned}
\gamma_i'' &\le \max_{1\le j<l}(\sigma_j + f^x_{j,i}) \\
&\le \max_{1\le j<l}(\sigma_j + \delta_{k_j})
\end{aligned}$$

$$
\begin{aligned}
&\leq \max_{1\leq j<l}(\sigma_{k_j}+\delta_{k_j})\\
&\leq \sigma_{k_j+1}\\
&\leq \sigma_l=\sigma_i
\end{aligned}
$$

$\sigma_i \geq \gamma''_i$ より，$\sigma_i \geq \gamma'_i$ が成り立つ．$\sigma_l = \cdots = \sigma_k$ より，$\gamma_i = \max_{l\leq m\leq k}\gamma'_m \leq \max_{l\leq m\leq k}\sigma_m = \sigma_i$ である．よって，$x'_i \geq x''_i$ が成り立つ． □

[補題 4.4] π'_R および π''_R は次の式を満たす．

$$W_x(\pi'_R) \geq W_x(\pi''_R) \text{ かつ } W_y(\pi'_R) \geq W_y(\pi''_R)$$

(証明) $W_x(\pi'_R) \geq W_x(\pi''_R)$ についてのみ証明する．π'_R の左境界をなす矩形を $v_{l'}$，右境界をなす矩形を $v_{r'}$ とおく．同様に，π''_R に対しても，$v_{l''}, v_{r''}$ を定義する．このとき，

$$
\begin{aligned}
left_{\pi'}(v_{l'}) &\leq left_{\pi''}(v_{l''}) \text{ かつ}\\
right_{\pi'}(v_{r'}) &\geq right_{\pi''}(v_{r''})
\end{aligned}
$$

を示せば十分である．

$x_{l''} = x_1$ ならば，$\gamma_{l''} = \sigma_{l''} = 0$ である．このとき，$left_{\pi'}(v_{l'}) \leq left_{\pi'}(v_{l''}) = left_{\pi''}(v_{l''})$ が成り立つ．$x_{l''} \neq x_1$ の場合を考える．矩形 $v_{l''}$ は 同じ x 座標を持つ矩形 $v_{l_{l''}}, \cdots, v_{k_{l''}}$ の中で最も幅の大きいものである．$lmin''$ を **PFS′**が $\sigma_i (i = 1, \cdots, l_{l''}-1)$ を決めた後の 変数 $lmin$ の値とおく．$left_{\pi''}(v_{l''}) \leq left_{\pi''}(v_{lmin''})$ より，**PFS′**は $l\_bnd(v_{l''}, x_{l''}) + \gamma''_{l''} < l\_bnd(v_{lmin''}, x_{lmin''} + \gamma''_{lmin''})$ であることを検出し，$\gamma_{l''} = \sigma_{l''}$ とする．これにより，$left_{\pi'}(v_{l'}) \leq left_{\pi'}(v_{l''} = left_{\pi''}(v_{l''}))$ である．

補題 4.3より，$right_{\pi'}(v_{r''}) \geq right_{\pi''}(v_{r''})$ が成り立つ．よって，$right_{\pi'}(v_{r'}) \geq right_{\pi'}(v_{r''}) \geq right_{\pi''}(v_{r''})$ である． □

次に，アルゴリズム **PFS′**による再配置が 2 つの制約を満たすことを示す．

[補題 4.5] 任意の 2 つの矩形 $v_i, v_j \in R (i \leq j)$ に対して，$\gamma_j - \gamma_i \geq f^x_{i,j}$ が成り立つ．

(証明) $x_i = x_j$ ならば，$\gamma_i = \gamma_j$ かつ $f^x_{i,j} = 0$ であるので，与式は成り立つ．$x_i < x_j$ の場合を考える．l および k をそれぞれ，$x_l = x_j$ および $x_k = x_j$ を満た

す最小および最大の添字とおく．$l \leq m \leq k$ であるすべての m に対して，

$$\gamma''_m = \max_{1 \leq i' < l} (\gamma_{i'} + f^x_{i',m}) \geq \gamma_i + f^x_{i,j}$$

が成り立つ．補題 4.3より，$\gamma''_m \leq \sigma_m$ が成り立ち，更に $\gamma_j = \max_{l \leq m \leq k} \gamma'_m$ かつ $\gamma'_m =$ (σ_m または γ''_m) である．よって，$\gamma''_m \leq \gamma_j$ であるから，$\gamma_j \geq \gamma_i + f^x_{i,j}$ が成り立つ．
□

[補題 4.6] アルゴリズム **PFS′**は直交順序を保存する (制約 (1))．
(証明) $x_i = x_j$ ならば，$x''_i = x''_j$ が成り立つ．ここでは $x_i \neq x_j$ の場合を考えるが，一般性を失うことなく $x_i < x_j$ と仮定してもよい．補題 4.5より，$\gamma_j - \gamma_i \geq f^x_{i,j}$ が成り立つ．$f^x_{i,j}$の定義より，$x_i \leq x_j + f^x_{i,j}$．よって，

$$x''_i = x_i + \gamma_i \leq x_i + \gamma_j - f^x_{i,j} \leq x_j + \gamma_j = x''_j$$

が成り立つ．
□

$i < j$である 2 つの矩形 v_i, v_j に対して，補題 4.5および $f^x_{i,j}$の定義より，**PFS′** による配置は制約 (2) を満たすことが言え，次の補題が成り立つ．

[補題 4.7] アルゴリズム **PFS′**は矩形の非交差を保証する．

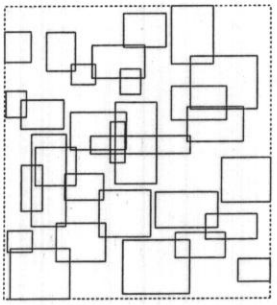
以上より，次の定理が成り立つ．

[定理 4.3] **PFS′**は時間計算量 $O(n^2)$ で再配置を行い，再配置後の矩形の配置は制約 (1),(2) を満たし，配置面積は **PFS** による再配置後の配置面積以下である．

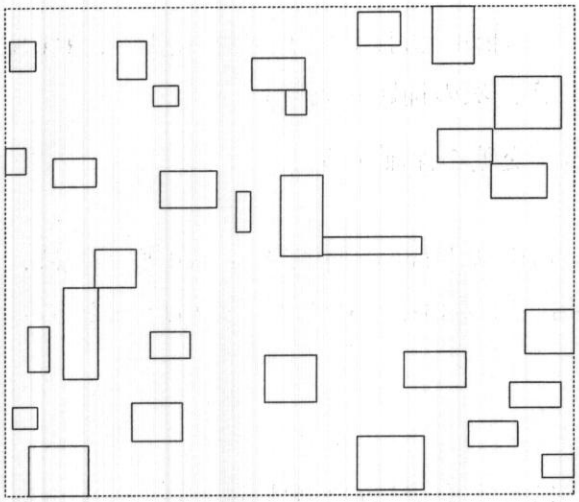
4.4.4 配置例と実験結果

まず，ある矩形集合 Rとその配置π_Rに対する **PFS** と **PFS′**の配置例を示す．図 4.14(a) は元の配置π_R，同図 (b)(c) がそれぞれ **PFS**，**PFS′**実行後の配置π'_Rである．

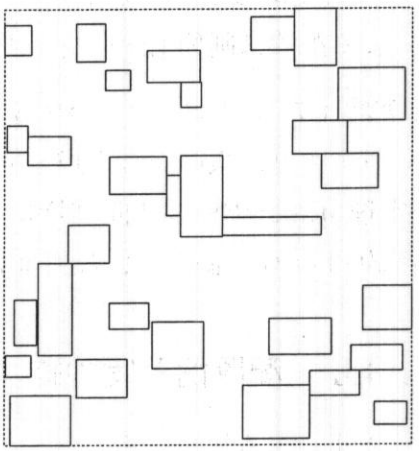
更に，ランダムに生成した矩形集合に対して **PFS** および **PFS′**を適用する実験を行った．実験には Sun Ultra-2 を用いた．図 4.15に実験結果を示す．初期配置は，200 × 200 の範囲にランダムに矩形を配置したものを用いる．同図 (a) は，横軸に矩形の個数を取った場合の **PFS** に対する **PFS′**の面積比である．矩形サイズは一辺が 50 ～ 100 の範囲でランダムに選んだものである．これらの値は，



(a) 元の配置π_R

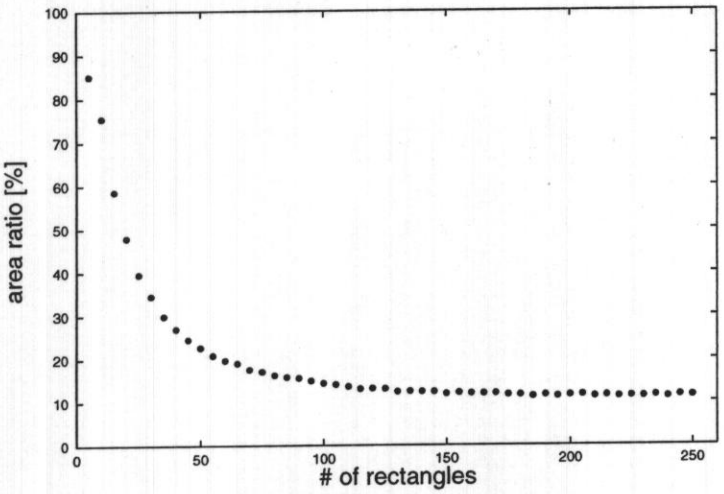


(b) **PFS** の結果

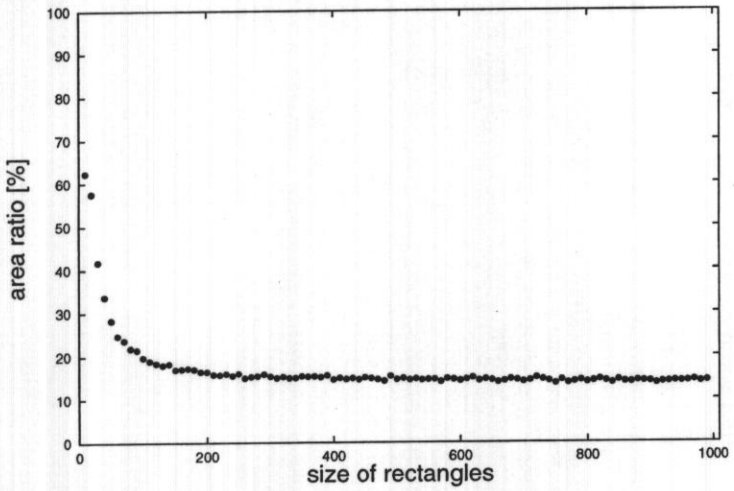


(c) **PFS′**の結果

図 4.14 アルゴリズムの実行例 ($n = 30$)



(a) 矩形数を変化させた場合



(b) 矩形サイズを変化させた場合

図 4.15 実験結果

50 の矩形集合を独立に作成して実験し，結果の平均値を取ったものである．実験に要した計算時間は，50 回の試行に対していずれも 1 秒以下である．この結果より **PFS′**は，特に矩形の個数が多い場合に対して，**PFS** による配置の面積の 15 ～ 20% の面積で再配置できることがわかる．同図 (b) は，横軸に矩形サイズを取った場合の **PFS** に対する **PFS′**の面積比である．矩形の個数を 50，矩形サイズは一辺が (値)±10 の範囲でランダムに選んだものである．このことから，矩形の個数が同じ場合には，矩形のサイズの変化は余り影響しないことが言える．

第 5 章 結論

本論文では，グラフの描画問題について根付き木に関する描画問題および一般無向グラフの動的描画問題を取り上げ，それぞれの問題の計算複雑度について考察した．

まず，根付き順序木の描画問題として，木構造図の描画アルゴリズムを取り上げた．再描画の際に部分木の輪郭を用い，その上の頂点に付随させる情報を工夫することにより，文献 [42] のアルゴリズムと同じ描画をより効率的に求める方法を示した．ある定数 c が存在してすべての葉の $width_x$ が c 以下である場合，本方法の時間計算量は $O(n\alpha(n,n))$ である (n は節点数)．

次に，根付き非順序木の描画問題として，最小幅描画問題を定義し，制約のいくつかの組み合わせのもとで，整数座標系，実数座標系のどちらを用いる場合でも NP 完全であることを示した．特に，整数座標系を用いるときには，各節点の子の数が 3 以下の木に限定しても NP 完全となる場合があることを示した．

最後に，一般無向グラフの動的描画問題として，矩形の最小面積再配置問題を取り上げた．ここでは，直交順序を保存する矩形集合に対して，矩形が交差しない最小面積再配置問題を定義し，この問題が整数座標系では NP 完全であること，実数座標系では NP 困難であることを示した．更に，矩形の非交差を満たす面積最小化のための発見的再配置手法の一つとして，文献 [24] のアルゴリズムと同じ漸近的時間計算量で，面積が小さいか等しい再配置を求める方法を示し，実験によって多くの場合に本手法による再配置の面積が小さくなることを確認した．

本論文では，与えられた制約を満たす最小幅および最小面積の描画を得る問題の計算複雑度について考察した．この他，実際に描画を得る上では “描画の見やすさ” を評価することも重要である．しかし，描画の見やすさを評価するというアプローチは描画の制約条件を如何に決めるかに依存するだけでなく，ユーザの認知にも依存するため，理論的な評価を行うのは困難な部分である．そのため，ユーザの認知に適応する，制約条件および評価基準の制定が不可欠である．更に，ほとんどの場合は NP 困難であることが予想されるため，より良い発見的アルゴリズムの開発が今後の課題となる．

謝辞

本研究の全過程を通じて，本研究の機会を与えて下さるとともに，適切な御指導と御助言を賜わりました藤原秀雄教授に深く感謝の意を表します．

本研究の副指導教官として，御指導頂きました伊藤実教授並びに高橋豊教授に深く感謝致します．

本研究の全過程を通じて，方針や問題点などのあらゆる面において，懇切丁寧に直接的な御指導をして頂きました増澤利光助教授に深く感謝致します．

本研究の全過程を通じて，様々な面でお世話になり，熱心な御討論，御協力を頂きました井上美智子助手に深く感謝致します．

本研究を進めるにあたって，日頃から様々な面で暖かい御助言，御援助を頂きました井上智生助手に深く感謝致します．

博士前期課程にて本研究を進めるにあたって，研究方針や問題点などの面において，御指導をして頂きました神戸大学工学部電気電子工学科田中榮一教授 (現·武庫川女子大学教授) 並びに増田澄男助教授に深く感謝致します．

また，日頃より有益な御討論，御助言を頂きました情報論理学講座の皆様に深く感謝致します．

参考文献

[1] C. Batini, E. Nardelli, and R. Tamassia. A layout algorithm for data flow diagrams. *IEEE Transactions on Software Engineering*, Vol. SE-12, No. 4, pp. 538–546, 1986.

[2] C. Batini, M. Talamo, and R. Tamassia. Computer aided layout of entity relationship diagrams. *The Journal of Systems and Software*, Vol. 4, pp. 163–173, 1984.

[3] G. Di Battista, editor. *Proceedings of the 5th Symposium on Graph Drawing (GD '97)*. Springer-Verlag, 1997.

[4] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. Algorithms for drawing graphs:an annotated bibliography. *Computational Geometry: Theory and Applications*, Vol. 4, pp. 235–282, 1994.

[5] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *GRAPH DRAWING – Algorithm for the visualization of graphs.* Prentice-Hall, 1999.

[6] F. J. Brandenburg, editor. *Proceedings of Symposium on Graph Drawing (GD '95)*. Springer-Verlag, 1995.

[7] J. Christensen, J. Marks, and S. Shieber. An empirical study of algorithms for point feature label placement. *ACM Transactions on Graphics*, Vol. 14, No. 3, pp. 203–232, 1995.

[8] R. Davidson and D. Harel. Drawing graphs nicely using simulated annealing. Technical Report CS 89-13, Department of Applied Mathematics and Computer Science, The Weizmann Institute of Science, Rehovot, 1989.

[9] P. Eades. A heuristic for graph drawing. *Congressus Numerantium*, Vol. 42, pp. 149–160, 1984.

[10] P. Eades, W. Lai, K. Misue, and K. Sugiyama. Preserving the mental map of a diagram. In *Proceedings of COMPUGRAPHICS '91*, pp. 34–43, 1991.

[11] A. Frick, A. Ludwig, and H. Mehldau. A fast adaptive layout algorithm for undirected graphs. In R. Tamassia and I. Tollis, editors, *Proceedings of Symposium on Graph Drawing '94 (Lecture Notes in Computer Science 894)*, pp. 388–403. Springer-Verlag, 1994.

[12] T. M. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Software Practice and Experience*, Vol. 21, No. 11, pp. 1129–1164, 1991.

[13] E. R. Gansner, E. Koutsofios, S. C. North, and K-P. Vo. A technique for drawing directed graphs. *IEEE Transactions on Software Engineering*, Vol. 19, No. 3, pp. 214–230, 1987.

[14] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, New York, 1979.

[15] 郷信義, 岸本美紀, 宮寺庸造, 岡田直之, 土田賢省, 夜久竹夫. Hichart プログラム図式の生成方法. 情報処理学会論文誌, Vol. 31, No. 10, pp. 1463–1473, 1990.

[16] J. Hopcroct and R. E. Tarjan. Efficient planarity testing. *Journal of the ACM*, Vol. 21, No. 4, pp. 549–568, 1974.

[17] K. G. Kakoulis and I. G. Tollis. On the edge label placement problem. In S. North, editor, *Proceedings of Symposium on Graph Drawing '96 (Lecture Notes in Computer Science 1190)*, pp. 241–256. Springer-Verlag, 1996.

[18] K. G. Kakoulis and I. G. Tollis. An alogirhtm for labeling edges of hierarchical drawings. In G. Di Battista, editor, *Proceedings of Symposium on Graph Drawing '97 (Lecture Notes in Computer Science 1353)*, pp. 169–180. Springer-Verlag, 1997.

[19] T. Kamada. *Visualizing abstract objects and relations.* World Scientific, 1989.

[20] 鎌田富久. グラフ描画アルゴリズム. *bit*, Vol. 23, No. 3, pp. 284–289, 1991.

[21] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letter*, Vol. 31, pp. 7–15, 1989.

[22] P. Luders, R. Ernst, and S. Stille. An approach to automatic display layout using combinatorial optimization algorithms. *Software Practice and Experience*, Vol. 25, No. 11, pp. 1183–1202, 1995.

[23] 松浦敏雄, 中村亨, 谷口健一, 増田澄男. 概形表示および部分拡大表示機能を有する木構造グラフエディタの作成とその評価. 電子情報通信学会論文誌, Vol. J75-D-I, No. 7, pp. 459–468, 1992.

[24] K. Misue, P. Eades, W. Lai, and K. Sugiyama. Layout adjustment and the mental map. *Journal of Visual Languages and Computing*, Vol. 6, pp. 183–210, 1995.

[25] S. Moen. Drawing dynamic trees. *IEEE Software*, Vol. 7, No. 4, pp. 21–28, 1990.

[26] S. North, editor. *Proceedings of Symposium on Graph Drawing (GD '96).* Springer-Verlag, 1996.

[27] E. M. Reingold and J. S. Tilford. Tidier drawings of trees. *IEEE Transactions on Software Engineering*, Vol. SE-7, pp. 223–228, 1981.

[28] L. A. Rowe, M. Davis, E. Messinger, C. Meyer, C. Spirakis, and A. Tuan. A browser for directed graphs. *Software Practice and Experience*, Vol. 17, No. 1, pp. 61–76, 1987.

[29] 佐野達郎. グラフ構造を考慮した無向グラフ自動描画法. 電子情報通信学会論文誌, Vol. J80-D-II, No. 3, pp. 772–782, 1997.

[30] A. Schrijver. *Theory of Linear and Integer Programming.* John Wiley and Sons, New York, 1986.

[31] M. D. Storey and H. A. Muller. Graph layout adjustment strategies. In F. J. Brandenburg, editor, *Proceedings of Symposium on Graph Drawing '95 (Lecture Notes in Computer Science 1027)*, pp. 487–499. Springer-Verlag, 1995.

[32] 杉山公造. グラフ自動描画法とその応用 — ビジュアルヒューマンインタフェースのための基礎技法. 計測自動制御学会出版, 1993.

[33] K. Sugiyama and K. Misue. A simple and unified method for drawing graphs: Magnetic-spring algorithm. In R. Tamassia and I. Tollis, editors, *Proceedings of Symposium on Graph Drawing '94 (Lecture Notes in Computer Science 894)*, pp. 364–375. Springer-Verlag, 1994.

[34] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on System, Man and Cybernetics*, Vol. 11, No. 2, pp. 109–125, 1981.

[35] 角浩二, 田中寿俊, 榎原博之, 中野秀男. グラフ描画アルゴリズムの性能評価. 電子情報通信学会論文誌, Vol. J79-A, No. 3, pp. 680–686, 1996.

[36] K. J. Supowit and E. M. Reingold. The complexity of drawing trees nicely. *Acta Informatica*, Vol. 18, pp. 377–392, 1983.

[37] 鈴木和彦, 鎌田富久, 榎本彦衛. 単純無向グラフ自動描画アルゴリズム. コンピュータソフトウェア, Vol. 12, No. 4, pp. 45–55, 1995.

[38] R. Tamassia, G. Di Battista, and C. Batini. Automatic graph drawing and readability of diagrams. *IEEE Transactions on System, Man and Cybernetics*, Vol. 18, No. 1, pp. 61–79, 1988.

[39] R. Tamassia and I. G. Tollis, editors. *Proceedings of the DIMACS International Workshop on Graph Drawing (GD '94).* Springer-Verlag, 1994.

[40] R. E. Tarjan. Applications of path compression on balanced trees. *Journal of the ACM*, Vol. 26, No. 4, pp. 690–715, 1979.

[41] K. Tsuchida. The complexity of tidy drawings of trees. In S. Suzuki, editor, *Topology and Computer Science*, pp. 487–520, Tokyo, 1987. Kinokuniya.

[42] 海野浩, 安斎公士, 小倉耕一, 西野哲朗, 中西美智子, 夜久竹夫. 木構造図式の描画問題. 情報処理学会論文誌, Vol. 33, No. 7, pp. 879–886, 1992.

[43] C. Wetherell and A. Shannon. Tidy drawings of trees. *IEEE Transactions on Software Engineering*, Vol. SE-5, pp. 514–520, 1979.