設 計 自 動 化 58－1
（1991. 7. 12）

拡張平面掃引法によるコンパクション制約グラフ生成

粟島 亨† 佐藤 政生†† 大附 辰夫†

†早稲田大学理工学部
††拓殖大学工学部

拡張平面掃引法によるコンパクション制約グラフ生成手法を提案する．本手法は，shadow-propagation法におけるshadow-frontを，拡張平面掃引に基づいて一括管理する手法であり，ワークリストの実装に平衡二分木を用いることで，O(nlogn)の手間で制約グラフを生成することができる．ここに，nはレイアウト・データの要素数である．単層のレイアウト・データに対して，従来用いられてきた直交平面掃引法との比較実験を行ったところ，両者はほぼ同等の性能であることが確認された．一方，多層レイアウトの場合には，直交平面掃引法が単層の場合の計算複雑度を保てないのに対し，本手法は，単層の場合と同様の手間で制約グラフを生成することができる．

Constraint Graph Generation Based on an Enhanced Plane Sweep Method

Toru Awashima† Masao Sato†† Tatsuo Ohtsuki†

† School of Science and Engineering, Waseda University
3-4-1 Ohkubo, Shinjuku-ku, Tokyo 169, Japan
†† Faculty of Engineering, Takushoku University
815-1 Tatemachi, Hachioji-shi, Tokyo 193, Japan

A constraint graph generation algorithm for layout compaction based on an enhanced plane-sweep method is proposed. The algorithm can maintain multiple shadow-fronts simultaneously by introducing an enhanced plane-sweep concept. Using a balanced search tree as a work-list, the algorithm runs in O(nlogn) time, where n is the number of layout elements. In case of single layer, experimental results show that the performance of the algorithm is approximately equal to that of the traditional perpendicular plane-sweep method. However, in case of multiple layers, the complexity of the perpendicular plane-sweep could fall into $O(n^2)$ while the proposed algorithm guarantees O(nlogn) execution time.

1. はじめに

レイアウトコンパクションの分野では，制約グラフに基づいた手法が主流となっている．また，制約グラフによるレイアウト表現は，チャネルコンパクションのように純粋な幾何学的処理に帰結できる場合をのぞいて，本質的に必要なものであるとの指摘もなされている[1]．

コンパクションのための制約グラフ生成(constraint graph generation)はそれ自体興味深い問題であり，さまざまな研究が行われてきた[2~10]．制約グラフ生成においては，処理効率だけでなく生成されたグラフの非冗長性が重要となる．なぜなら，制約グラフ生成に引き続いて行なわれるコンパクション処理の手間は，制約グラフの枝数に比例するため，冗長な制約をできるだけ削減することが，コンパクション処理全体の効率向上につながるからである．

本稿では，コンパクション制約グラフ生成についてまとめるとともに，拡張平面掃引法(enhanced plane sweep method)[11]によってこの処理を効果的に行う手法を提案する．提案する手法は，最も初期に提案された制約グラフ生成手法であるshadow propagation法[2,3]におけるshadow frontの概念を発展させたものと位置づけることができる．

本手法は，データ構造に平衡二分木を用いることによって，レイアウト要素数をnとしたとき，O(nlogn)の手間で制約グラフを生成できる．特に，単層レイアウトモデルの場合，上記の手間で，非冗長な制約グラフを生成する．これは，現在まで最良の手法とされてきた直交平面掃引法(perpendicular plane sweep method)[7~9]と理論的には同等の性能であることを意味する．単層の問題について比較実験を行なったところ，処理時間の点では拡張平面掃引法が，使用メモリの点では直交平面掃引法がやや優れていたものの，基本的には同等の性能であることが確認できた．

一方，多層レイアウトモデルの場合には，直交平面掃引法が単層の場合の計算複雑度を保つことができないのに対して，拡張平面掃引法は単層の場合と同様の手間と使用メモリで制約グラフを生成することができる．

2. レイアウトモデル

本稿では多層のレイアウトモデルを扱う．各層上のレイアウト要素は長方形で表現され，オブジェクトと呼ばれる．以下で対象とするのは，リーフセルレベルのコンパクションである．したがって，各オブジェクトはマスクの幾何学パタンを表すものであって，シンボリックなものではない．議論を簡単にするため，同一層上のオブジェクトは互いに重なることを許されず，オブジェクトの，どの垂直（水平）辺も等しいx(y)座標を持たないものと仮定する．

層i上のオブジェクトと層j上のオブジェクト間の最小間隔規則は$d_{ij}(\geqq 0)$で表わされ，同一層上のオブジェクト間の最小間隔規則は，$d_{ii}(\geqq 0)$で表される．単層のみを対象とする場合は，便宜上その層番号を1とみなすこととし，層上の最小間隔をd_{11}と表記する．また，以下ではレイアウトを上方から下方へ圧縮するy-コンパクションのための制約グラフ生成に議論を統一する．

一般に，コンパクション制約は以下の3種に分類できる．

1）接続制約(connectivity constraints)

配線と端子，配線とビアなど回路的な接続関係を保つために設けられる制約である．オブジェクト同士の接続関係は回路の接続情報（ネットリスト）から知ることができるので，接続制約は自明な制約であるといえる．

2）分離制約(separation constraints)

異なるオブジェクトが，許される最小間隔を越えて近づくことを禁止する制約である．この制約の有無はオブジェクトの相対位置関係によって変わるため，コンパクションの度に，レイアウト情報を直接解析して生成する必要がある．

3）付加制約(additional constarints)

上記2種以外に設けられる何らかの付加的制約である．回路設計上の理由から，設計者によって設定されることが多いと考えられる．

上述のように，接続制約は自明なものである．また，付加制約は設計者によって与えられるもので，しかもその数は通常少ないと考えられる．したがって制約グラフ生成において最も重要なのは，分離制約の生成である．そこで以下では，接続制約及び付加制約を対象外とし，分離制約グラフの生成に焦点を当てる．

分離制約を設けるべきオブジェクト対としてあきらかなのは，コンパクション方向で互いに対面しているものである．このようなオブジェクト間に設ける制約を直接的分離制約と呼ぶことにする（図1(a)参照）．

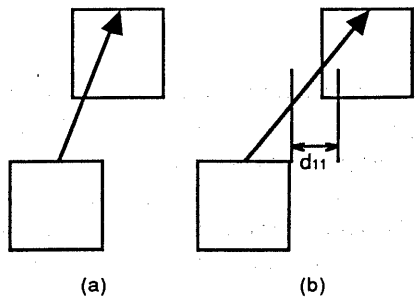


図1　分離制約

しかし，コンパクション後のレイアウトの許容性を保証するためには直接的分離制約だけでは不十分であり，図1(b)のような間接的な分離制約をも考慮する必要があ

る．これは，水平方向の距離が最小間隔未満であるようなオブジェクト間に設ける制約である．間接的分離制約を導入することで，垂直方向のコンパクション処理による水平方向での最小間隔違反の発生を回避することができる．

次節では，提案されている分離制約グラフ生成手法をいくつか紹介する．各手法の処理効率とともに，生成された制約グラフの冗長性にも注目する．

3．コンパクション制約グラフ生成

ここでいうコンパクション制約グラフとは，各オブジェクトが節点に，オブジェクト間の分離制約の有無が枝の有無に，最小間隔規則が枝の重みに対応するような有向グラフのことである．また，ある制約を取り除いてもコンパクションの結果に影響が及ばないとき，その制約は冗長であるという．コンパクション制約グラフの例を図2に示す．図中破線で描かれた制約は冗長なものである．

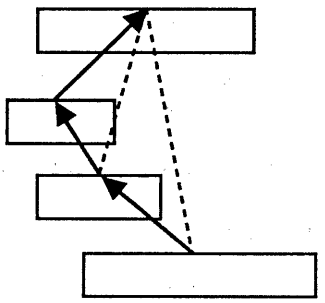


図2　制約グラフ

分離制約は全ての異なるオブジェクト間に存在し得るので，全オブジェクト数をnとすれば，$O(n^2)$が制約数（すなわち制約グラフの枝数）の上界となることはあきらかである．したがって，最も単純な制約グラフ生成の方法は，全てのオブジェクト対に関して分離制約の有無を調べるという方法である．しかし，これでは常にn^2に比例する手間がかかってしまい効率が悪い．また，生成されたグラフは冗長な制約枝を多く含んでいると考えられる．そこで，処理の効率化と冗長な制約の排除の2点からいくつかの手法が提案されてきた．以下，提案された順にしたがい，代表的な3つの制約グラフ生成手法を紹介する．

3.1 Shadow propagation 法[2],[3]

分離制約は，コンパクション方向で互いに可視であるようなオブジェクト対に設ければ十分であることから，制約数の削減を図った手法である．具体的には，現在注目しているオブジェクトから，コンパクション方向と逆方向，すなわち上方向に領域を掃引していくことで互いに可視なオブジェクトを特定し，制約を生成する．間接的分離制約を生成するため，オブジェクトを水平両方向にd_{11}ずつ伸ばした範囲で，領域を掃引する．掃引は，掃引線（これを特にshadow frontと呼ぶ）の全区間が他のオブジェクトによって遮断されるまで続ける．このような処理を全てのオブジェクトに対して逐次的に行う．手法の概念図を図3に示す．計算の手間はあきらかに$O(n^2)$となるが，実験的には$O(n^{1.5})$程度であることが報告されている．

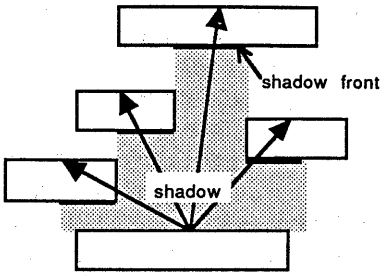


図3 Shadow-propagation法

3.2 Intervening Group 法[4]~[6]

冗長な制約数をさらに削減するという観点からshadow propagation 法に改良を加えた手法である．互いに可視であるようなオブジェクト対であっても，その間の領域に別のオブジェクトが存在すれば，そのオブジェクトを仲介にした分離制約がすでに存在するはずである（そのようなオブジェクトを起点とした処理はすでに終了しているとする）．そこで，shadow propagationを行う際に，掃引の起点となったオブジェクトから現在対象としているオブジェクトへ至る制約グラフ上の径路を探索する．そのような径路が存在すれば，新たに制約を付加する必要はないと結論できる．

径路探索の深さに制限を課さない場合，この方法は常に非冗長な制約グラフを生成することができる．しかし非常に手間がかかるため，通常は探索の深さを2～4程度に制限している．探索の深さを2に制限したときの手法の概念図を図4に示す．図中破線は，太線で示された径路の探索によって冗長であると判断された制約枝である．この手法も実験的に$O(n^{1.5})$程度の手間で処理を行えることが報告されている．

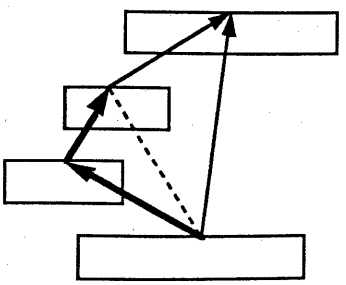


図4 Intervening group法

3.3 直交平面掃引法[7]~[9]

上記2つの方法が，いずれもコンパクション方向と逆方向，すなわち垂直上方向に領域を掃引するという処理に基づいていたのに対し，この手法は領域全体を，コンパクション方向と直交する方向に，ただ1度だけ掃引することで全ての制約を生成する．手法の概念図を図5に示す．図のように，y-コンパクションの場合，領域を左から右へ垂直な掃引線によって掃引し，垂直方向で対面しているオブジェクト間に制約を設けていく．直交平面掃引法は，問題を単層に限った場合，生成される制約グラフが非冗長である（すなわち制約数が最小である）という特徴を持つ．このとき，制約数はオブジェクト数の線形オーダーであり，その上界は2n-2であることが知られている．また，ワークリストに平衡二分木を用いることで，処理の複雑度をO(nlogn)に抑えることができる．

以下ではまず単層の場合について直交平面掃引法のアルゴリズムを示し，次に多層に適用する場合の問題点を列挙する．

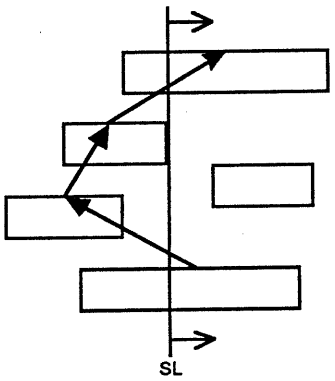


図5　直交平面掃引法

3.3.1 単層の場合のアルゴリズム

直交平面掃引法で間接的分離制約を扱うためには，以下のいずれかの処置が必要である．

1）全オブジェクトを水平両方向に$d_{11}/2$ずつ伸長するという前処理によって全ての間接的制約を直接的制約に変換する（文献[7]~[9]がこれに準ずる）．

2）掃引線を幅$w=d_{11}$のスリットに置き換えて掃引を行うことによって，間接的分離制約を設けるべきオブジェクト対の探索を可能とする（文献[9]等に類似の手法を見ることができる）．

どちらの方法も基本的には等価であるが，1）の方法では，伸張する幅によって，オブジェクトに重なりが生じてしまう危険性があり，望ましくない．したがって，2）の方法すなわち幅wのスリットによる掃引法を用いることにする．

ここでは，オブジェクトの上辺のみを対象として処理すれば十分である．まず，領域の最下部に全x区間にわたる仮想的なオブジェクトR_0を設ける．R_0は制約グラフのソースとなる．次に，オブジェクト集合を保持するイベントリストELを上辺の左端のx座標の昇順に整列する（定義より，左端点のx座標が同じオブジェクトは存在しない）．以下にアルゴリズムの概略を示す．なお，簡単のためスリット幅w=0，すなわち$d_{11}=0$とする．ここに，SLとはスリットと交差するオブジェクトの上辺をそのy座標をキーとして保持するワークリストであり，PQとはその右端点のx座標をキーとして保持するプライオリティキューである．

直交平面掃引によるアルゴリズム（単層の場合）

[Step1]
ELからオブジェクトの上辺を1つ取り出す（eと呼ぶ）．ELが空なら終了．
[Step2]
PQが空ならStep4へ．PQの先頭要素e'の右端のx座標がeの左端のx座標より大きければStep4へ．
[Step3]
e'の右端の上方および下方に隣接する辺がStep4によって記憶していた辺と一致したら，対応するオブジェクト間に制約を設ける．e'をSLおよびPQから削除してStep2へ．
[Step4]
eをそのy座標をキーとしてワークリストSLに挿入する．このとき上方に隣接している辺を記憶しておく．また，下方に隣接している辺がeを指すように記憶を更新する．eを右端点のx座標をキーとしてPQに挿入する．Step1へ．

上記のアルゴリズムは，挿入時に隣接していたオブジェクトを記憶するという一連の操作(Step3，Step4)によって，冗長な制約の生成を禁止している．

いま，SLおよびPQに隣接要素を双方向ポインタで結んだ平衡二分木を用いるとすると，挿入・削除をO(logn)の手間で，隣接要素の探索をO(1)の手間で行うことができる．したがって，手法全体の手間をO(nlogn)で抑えることができる．

3.3.2 多層の場合のアルゴリズム

Lengauerは多層の場合の制約グラフ生成に対して，直交平面掃引法を用いた以下のようなアプローチを提案している[9]．

直交平面掃引によるアルゴリズム（多層の場合）

[Step1]
全ての層について，単層の場合と同様に制約グラフを生成する．

[Step2]
全ての層対について，１つの仮想的な層を設け，対応する２層の全てのオブジェクトを写像する．このような仮想的な層全てについて，単層の場合と同様に制約グラフを生成する．このとき，同一の層に属するオブジェクト間には制約を設けないようにする（すでにStep1で生成されているため）．
[Step3]
Step1,Step2で生成した全ての制約グラフを併合する．

Step2では，制約を設けるべき異なる層のオブジェクト対が必ずしも隣接していないため，最悪の場合１回の探索にO(n)の手間がかかってしまう．したがって処理全体の複雑度は$O(n^2)$となる．

上記以外のアプローチとして，全ての層および層対に対する処理を一括して行う手法も提案されている[10]．制約が存在し得るオブジェクト対をもれなく探索するためには，スリット幅$w=\max(d_{ii}, d_{ij})$とする必要がある．したがって，層iに注目したとき常に$w \geq d_{ii}$となって，制約を設けるべきオブジェクト対の隣接を保証できず，処理の複雑度は,やはり$O(n^2)$となってしまう．図６に，このような場合の例を示す．図中網掛けされた部分は，スリットを表わしている．この例で，実際に制約を設けるべきオブジェクト対はAとDであるが，SL内でAの上方に隣接しているのはBである．

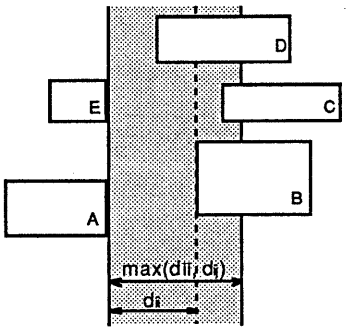


図６　スリットと交差するオブジェクト

４．拡張平面掃引による制約グラフ生成

領域全体を上方から下方へ，コンパクション方向に沿ってただ１度だけ掃引することによって，全ての制約を生成してしまおうというのが，拡張平面掃引法の基本的考え方である．Shadow propagation法は，最も古典的な制約生成手法であるが，shadow frontを効率的に管理することが困難であるという理由であまり用いられていなかった．しかし，shadow frontの概念は拡張平面掃引法[11]におけるprevious boundaryとして一般化することが可能である．Previous boundary（以下PBと呼ぶ）とは平面掃引法における通常の掃引線から掃引済みの領域方向に可視であるような図形の境界辺を保持するワークリストのことである．拡張平面掃引法に基づいて制約グラフ生成を行なえば，処理の複雑度を直交平面掃引法と同じO(nlogn)に抑えることができる．

Shadow propagation法は１つのオブジェクトに注目し，そのオブジェクトを起点として平面を掃引するという処理を全てのオブジェクトについて逐次的に行なっていた．一方，拡張平面掃引法では，全オブジェクトを対象として掃引を行い，制約生成に必要なshadow frontはPBとして一括して管理することができるのである．いいかえれば，PBは全ての未掃引オブジェクトのshadow frontの和をとったものであるといえる．図７に拡張平面掃引法の概念図を示す．

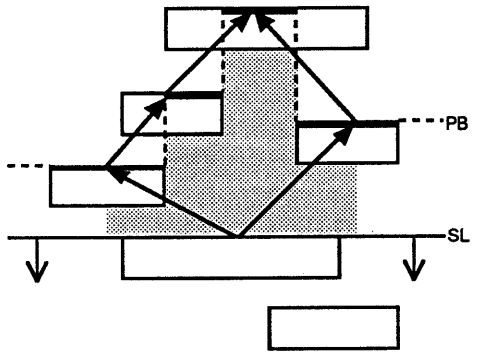


図７　拡張平面掃引法

以下では，まず単層の場合のアルゴリズムを示し，次に多層の場合でも処理の手間が変わらないことを示す．

４．１ 単層の場合のアルゴリズム

まず，領域の最下部に全x区間にわたる仮想的なオブジェクトR_0を設ける．これが制約グラフのソースとなる．次にオブジェクトを保持するイベントリストELを上辺のy座標の降順に整列する．以下にアルゴリズムの概略を示す．

拡張平面掃引によるアルゴリズム（単層の場合）

[Step1]
ELからオブジェクトの上辺を１つ取り出す（eと呼ぶ）．ELが空なら終了．
[Step2]
ワークリストPBによって，辺eを左右にd_{11}ずつ伸張した範囲の上方向に可視な境界辺を探索し，対応するオブジェクトを特定する．オブジェクト間の制約が冗長でないと結論できるときのみ制約を生成する．探索した範囲に存在する境界辺に対して後述の更新操作を行う．
[Step3]
eをその左端のx座標をキーとしてPBに挿入し，Step1へ．

Step2について詳しく説明する．いま辺eと垂直方向で対面しているPB内の境界辺（掃引済みオブジェクトの上辺の全体あるいは一部分）との関係は，$d_{11}=0$としたとき，

図8に示した(a)～(d)の4通りに分けられる．(a)の場合PB内の境界辺の右端が対応するオブジェクトの端点と一致するとき，eと境界辺に対応するオブジェクトの間には他のオブジェクトが存在しないことがいえ，非冗長な制約の存在を結論できる．(b)の場合は左端点，(c)は両端点がオブジェクトの端点と一致することが条件となる．(d)のように，eを完全に被覆するようなものに対しては，無条件に制約を設けることができる．

PB内の境界辺を更新は以下のように行う．(a)の場合右端のx座標をeの左端のx座標に書き換える．(b)の場合は左端のx座標をeの右端のx座標に書き換える．(c)の場合は境界辺をPBから削除する．(d)の場合は境界辺を二分割し，端点の座標を書き換える（新たな境界辺の挿入を伴う）．

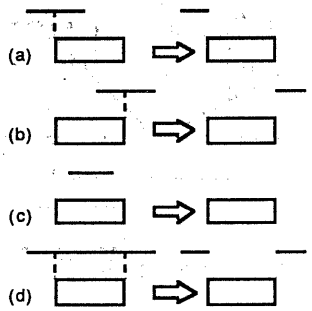


図8　PBの更新操作

PBに，隣接する節点を双方向のポインターで結んだ平衡二分木を用いれば，挿入，削除，探索をO(log gn)の手間で，隣接要素の探索をO(1)で行なえる．Step2は要素の列挙を含んでいるが，列挙された要素は直ちに削除あるいは更新されることを考えると，手法全体の計算複雑度をO(nlogn)で抑えることができる．

4.2 多層の場合のアルゴリズム

拡張平面掃引法によって，多層レイアウトの制約グラフを生成するには以下のようにする．

拡張平面掃引によるアルゴリズム（多層の場合）

全ての層についてPBを設ける．層iに対応するPBをPB_iと呼ぶ．あるオブジェクトを起点とする探索は全てのPBについて行う．層iのオブジェクトからPB_jを探索するときは，その探索範囲をd_{ij}によって動的に決定する．

上記のアルゴリズムは，PBを層の数だけ用意する点を除いて，単層の場合と全く同じである．したがって全ての基本処理を高々層数に応じた回数だけ行えばよいので，処理全体の手間はO(nlogn)で抑えられる．

直交平面掃引法と拡張平面掃引法の決定的な相違点は，掃引の方向である．前者がコンパクションの方向と直交する方向に掃引を行うのに対し，後者はコンパクション方向に沿って掃引を行う．このことは見方を変えると，間接的分離制約の扱いの違いとしてとらえることができる．直交平面掃引法は，各オブジェクトの挿入，削除のタイミングすなわち掃引のイベントそのものを制御することで対処しているため，処理が煩雑となってしまう．一方，拡張平面掃引法においては，掃引のイベント制御そのものは不変であり，ワークリストの探索範囲を動的に制御することで柔軟に対処している．拡張平面掃引法の優位性は，多層の場合，特に層毎あるいは層対毎に最小間隔規則が異なるような場合，顕著な計算効率の差として現れる．

5. 計算機実験結果

単層の場合について直交平面掃引法と拡張平面掃引法の比較実験を行なった．実験には100,000×100,000グリッドの領域に，乱数によって重なりのない長方形を128～16,384個発生させたデータを用いた．d_{11}=0とした．実験プログラムはSun-4/110（7MIPS程度）上にC言語で実装した．

5.1 データ構造

全てのワークリストは隣接要素同士を双方向のポインターで結んだ平衡二分木を用いて実装した．したがって，ワークリストに対する要素の挿入，削除，探索は要素の総数をnとしたとき，O(log gn)の手間で行える．さらに，双方向のポインタにより，隣接要素を一定時間で探索することが可能である．

平衡二分木としてはred-black tree[12]を採用した．Red-black treeは半平衡木の一種であり，その呼称は各節点に赤か黒の色をつけることによって木の平衡状態を管理していることに由来するものである．

5.2 制約数

実験は単層の問題を対象としているので，非冗長な制約グラフの生成が保証されている．したがって生成される制約グラフは，いずれの手法によるものも一致しなくてはならない．実験の結果，全ての場合について生成された制約グラフは一致した．図9にオブジェクト数と制約数の関係を示す．比較のため，全ての可視対数と，非冗長な制約グラフの枝数の理論的な上界をあわせて示す（枝数の上界は，オブジェクト数nのとき2n-2となることが知られている）．図から，制約数はオブジェクト数に対して線形に増加していることがわかる．実際の制約数は全可視対の数と比べると，おおむね2分の1程度となっている．また非冗長な制約数の理論的な上界値を30%程度下回っている．

図10にn=128の場合の制約グラフ出力例を示す．図10(a)は全ての可視オブジェクト対に制約枝を設けたものであり，同図(b)は拡張平面掃引法によって生成した制約

枝である．一見して，制約数が大幅に減少していることがわかる．(b)のグラフは周囲長が3であるようなループすなわち三角形の成分を含んでいない．制約グラフにおいて，三角形の成分を含まないことと，非冗長であることは同義である[7],[8]．

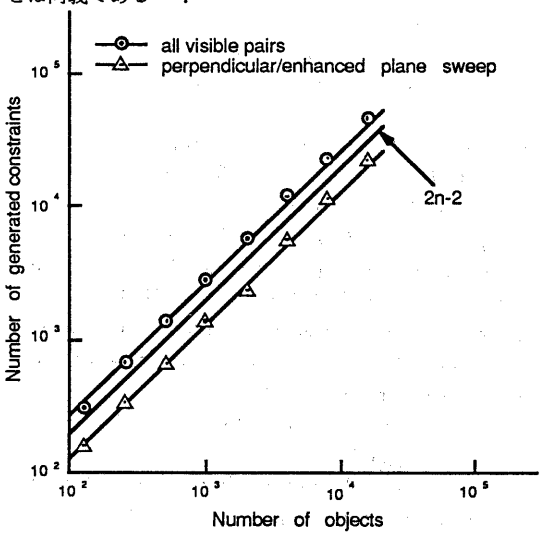
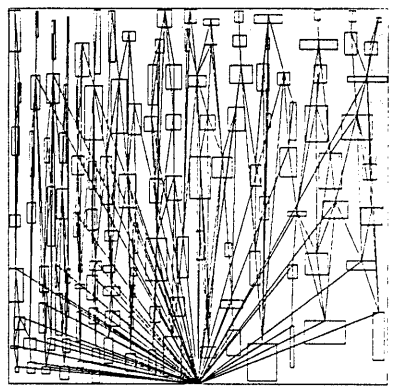
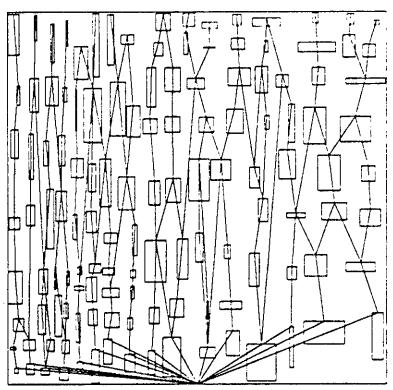


図9　オブジェクト数と制約数の関係



(a)



(b)

図10　制約グラフの出力例(n=128)

5.3 処理時間

図11にオブジェクト数と処理時間の関係を示す．ただし，ここでいう処理時間とは制約グラフ生成に要した時間であり，データの読み込み，イベントリストの整列にかかった時間は省いてある．直交平面掃引法，拡張平面掃引法のいずれのグラフも傾きは約1.05であり，オブジェクト数に対してほぼ線形に近い時間で処理を終えていることがわかる．絶対的な処理時間でみると，拡張平面掃引法は直交平面掃引法に対しておよそ25％程度高速に処理を行なっている．

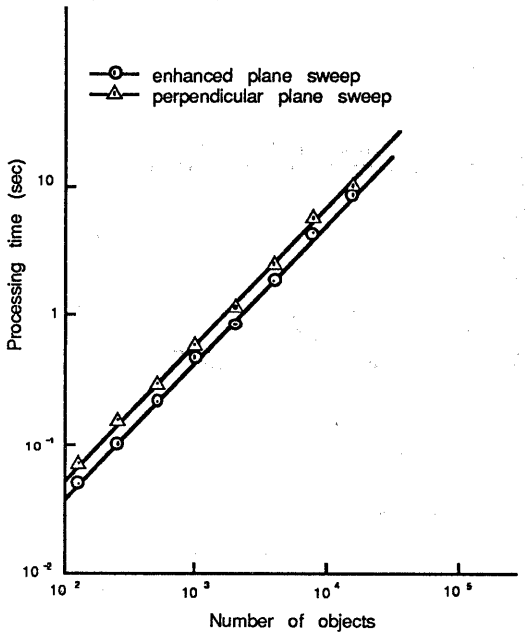


図11　オブジェクト数と処理時間の関係

5.4 メモリ使用量

ここでいうメモリ量とは，各ワークリストを構成する基本要素の総数のことである．図12に各ワークリストの基本要素の構造を示す．図中SL Nodeのcand_ptrとは，挿入時に隣接していたオブジェクト，すなわち制約を設けるべき候補となったオブジェクトを指すポインターである．図13に，処理終了までに記録されたメモリ量の最大値を示す．直交平面掃引法，拡張平面掃引法とも，グラフの傾きはおよそ0.46である．平面掃引法に基づいたレイアウトアプリケーションに必要なワークリストのサイズは，経験的に全オブジェクト数の平方根に比例する程度だといわれており，実験結果はこれと合致する．絶対量でみると，拡張平面掃引法は，直交平面掃引法の3～4倍のメモリを要している．

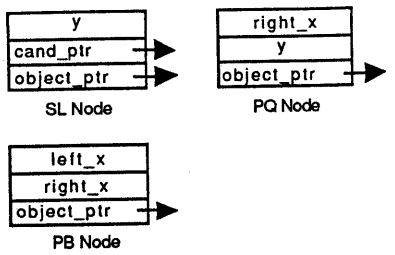


図１２　ワークリストの基本要素

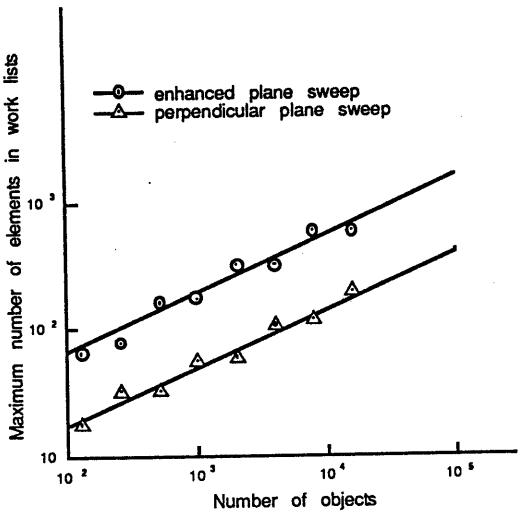


図１３　オブジェクト数と使用メモリの関係

６．おわりに

拡張平面掃引法による制約グラフ生成について論じた．単層レイアウトを対象とした場合，本手法の理論的計算複雑度，実験的評価は，ともに直交平面掃引法と同等であった．しかし，より一般的な多層のレイアウトモデルを対象とした場合には，少なくとも理論的計算複雑度において拡張平面掃引法が優れていることがあきらかになった．今後は，多層レイアウトにおける拡張平面掃引法の優位性を計算機実験によって確認していく必要があろう．

本稿のレイアウトモデルでは，同一層上でのオブジェクト同士の重なりを禁止していた．しかし異なるオブジェクトであっても，それぞれが同一ネットの構成部分であるような場合にはコンパクションの際に生じる重なりを許すのが普通である．重なりを許すか許さないかを判定するためには，オブジェクトの属しているネットの論理的な情報にアクセスする必要がある．この点について，拡張平面掃引法は，あるオブジェクトを起点とする制約を一度に生成してしまうため，データベースへのアクセス頻度をを最小化できるという利点を持っている．

文献

[1] 佐藤，山元 他: ジョグ挿入を伴ったチップ・コンパクションアルゴリズム，情処研報 AL-16-11(1990).

[2] Boyer, D. G.: Symbolic Layout Compaction Review, *Proc. of 25th ACM/IEEE DA Conf.*, pp.383-389 (1988).

[3] Preas, B. T. amd M. J. Lorenzetti ed.: *Physical Design Automation of VLSI System*, Benjamin/Cummings Publishing (1988).

[4] Kingsley, C.: A Hierarchical, Error-Tolerant Compactor, *Proc. of 21st DA Conf.*, pp.126-132(1984).

[5] Do, J. and W. M. Dawson: A Well-Behaved IC Layout Compactor, *Proc. of VLSI85*, pp.277-285 (1985).

[6] Hedges, T. et al.: Bitmap Graph Build Algorithm for Compaction, *Proc. of IEEE ICCAD*, pp.340-342 (1985).

[7] Doenhaedt, J. and T. Lengauer: Algorithmic Aspect of One-Dimensional Layout Compaction, *IEEE Trans. on CAD*, Vol. 6, No. 3, pp.863-878(1987).

[8] Lengauer, T.: *Combinatorial Algorithms for Integrated Circuit Layout*, John Wiley & Sons (1990).

[9] Burns, J. L. and A. R. Newton: Efficient Constraint Generation for Hierarchical Compaction, *Proc. of IEEE ICCC*, pp.197-200 (1987).

[10]宮下: 詳細コンパクションにおける制約グラフ生成手法，信学春期全大 A-100 (1991).

[11]Sato, M. and T. Ohtsuki: Enhanced Plane Sweep Method for LSI Pattern Design Problems, 信学技法 CAS86-199(1987).

[12]Tarjan, R. E.: *Data Structures and Network Algorithms*, Society for Industrial and Applied Mathematics(1983).