設 計 自 動 化 76－3
（1995．7．20）

Minimum Spanning Tree に基づいた配置改善手法

新田 泉，澁谷 利行，河村 薫
{nitta, shibu, kawamura}@flab.fujitsu.co.jp
富士通研究所
川崎市中原区上小田中 1015

本論文では，Minimum Spanning Tree(MST) の配線長を改善する配置改善手法である **Minimum Spanning Tree Reduction Pairwise Interchange** 手法 (MRPI) について述べる．MRPI 法は Mincut による初期配置の後に行なわれ，セルのペア交換によって MST の配線長を改善する．MRPI 法は，各セルについて配線長を改善する領域を improvement area と定義し，交換を行なうセルの探索を improvement area を用いて行なうことにより，改善が行なわれるセルの組合せを高速かつ効率的に検出する手法である．SOG ゲートアレイの実データを用いて実験を行ない，配線長の改善率，処理時間を評価し，有効性を示した．

A detail placement technique improving wire length estimated by Minimum Spanning Tree

Izumi Nitta, Toshiyuki Shibuya and Kaoru Kawamura
{nitta, shibu, kawamura}@flab.fujitsu.co.jp
Fujitsu Laboratories Ltd.
1015, Kamikodanaka, Nakahara-ku, Kawasaki, Kanagawa, 211 Japan

In this paper, we present a **Minimum Spanning Tree Reduction Pairwise Interchange**(MRPI) method as an improvement placement after min-cut partitioning.

The MRPI method improves wire length of Minimum Spanning Tree (MST) by pairwise interchanging of cells.

The idea of the improvement area is introduced as an idea of searching for the cell pairs to reduce wire length of MST.

Good experimental results have been observed for wire length and excution time.

1 はじめに

VLSIの自動配置において，Mincutは代表的な手法として用いられている[3] [4][11]. Mincutでは，カット数が最小になるように回路をtop-downに分割していく．Mincutが広く用いられている理由として，カット数を最適化することによって，結合の強いセルの集合がまとまって置かれやすく，結果として配線長が間接的に最適化されること，ネットの混雑度が分散されること，また，top-downの処理によって高速であることが考えられる．

しかし，配線あるいはレイアウト全体からみると，配置においてはカット数だけでなく，配線長の最適化やタイミング制約を満足することも要求される．そこで，Mincutを基本とした配置では，配線長やタイミング制約を考慮した処理を1.パーティショニングの各階層，および，2.最終的なパーティショニングが得られた後にそれぞれ行なう必要がある．1については，すでにクラスタリング[1][13]やタイミングドリブン配置[2][5][12]等において提案されている．また，2についても詳細配置の手法が提案されている[14].

本稿では，2の局面において，すなわちMincutによって得られた配置結果に対して，配線長を改善する手法について提案している．Mincut後に配線長を改善する理由として，配線に与える効果が考えられる．実際にSOGの実データについて，Mincutによる配置結果に対して自動配線を行なって評価したところ，未配線が多く残るような難しいデータでも，配置後の論理配線長を数%減少させるだけで未配線の数が0に近くなる場合が多くあることがわかっている．

提案する手法では，Mincutで得られた配置結果にセルの局所移動を繰り返して配線長改善を行なう．ここで，Mincutの利点を活かすために，局所移動がMincutによるパーティショニングを大きく変えないことと，処理全体にかかる時間がMincutの時間に比べて充分に短いことが要求される．Mincutの結果が充分最適化されているので配線長改善率はわずかとなる可能性がある．しかし，大規模なデータや配線の難しいデータについては，わずかな改善でもその後の配線処理に効果があると考えられる．

従来の局所改善の方法としては，次の2つが代表的である．

1. Pairwise Interchange(PI)[6] セルのペア交換を繰り返して配線長を改善する方法である．

   PIにSimulated Annealing[10]を適用すればさらによい解が得られる可能性が高くなるが，処理時間はPIよりもさらに長くなる．

2. Force Directed Pairwise Relaxation(FDPR)[6][9]

   選択したセルをその重心の近傍に存在するセルとペア交換することによって配線長を改善する方法．[7]によればMincut後の配置改善手法として高速で効果的な手法であり，数%の配線長改善が報告されている．

これらの方法では一般に，Bounding Boxと呼ばれる，ネットに接続する端子を囲む最小矩形の周長の1/2で配線長を定義している．Bounding Boxの利点は計算時間が短いことであるが，多端子のネットに対して実配線長との誤差が大きいという欠点があり，多端子ネットが多く存在するデータでは，Bounding Boxで配線長を改善しても，実配線に反映されるとはかぎらない．

実配線に最も近い配線長の評価方法としてはSteiner Minimum Treeが考えられる．しかし，多端子のネットの場合にはNP-hardな問題となり[8]，大規模回路でこれを用いるのは非現実的である．Steiner Minimum Treeの近似として，Single Trunk Steiner Tree(STST), Minimum Spanning Tree(MST)がある．SOGでは，配線チャネルが厳密に定義されていないことを考慮すると，STSTの構造はMSTに比べて誤差が大きいと思われる．そこで，我々はSOGの配線長改善という点からMSTを配線長のコストとして用いることにした．

改善にかかる時間をMincutに比べて充分短くする必要を考慮すると，FDPRのように解の探索空間を絞る方法が有効である．その場合，なるべく多くの良い解が存在するような探索空間を選ばなくてはならない．そこで我々は，MSTにつ

いて，FDPRの重心に相当するものが定義できないか考えた．あるセルについて，このセルに接続するMSTの各枝をベクトルと見なして，これらのベクトルの和の方向にこのセルを移動すると，MSTのコストを減らすことができる．

この考えをもとに，本稿ではMSTのコストを改善する領域improvement areaを定義し，このimprovement areaをFDPRの重心の代わりに用いてMSTの配線長改善を行なう手法である，**Minimum spanning tree Reduction Pairwise Interchange**(MRPI)手法を提案する．

以下，第2章ではimprovement areaの考え方を，第3章ではMRPIのアルゴリズムを説明し，第4章において実データを用いた実験結果を示し，improvement areaとMRPIの有効性を評価する．第5章でまとめをする．

2 improvement areaの考え方

2.1 準備

まず，以下の議論で扱うMSTの説明をする．

- $G_n(V_n, E_n)$：ネットnに接続するセル集合をV_nとしたときの，V_nの各セルを節点と見なした完全グラフ．E_nはV_nに含まれる任意の2つのセルを結ぶ枝の集合．
- $T_n(V_n, E'_n)$：ネットnのMST．このとき，$G_n(V_n, E_n)$において，$E'_n \subseteq E_n$である．

  2つのセルu, vがMSTの枝$e \in E'_n$の両端の節点のとき，uとvは隣接するという．

次に，MSTのコストの定義について説明する．今，ネットnのMST $T_n(V_n, E'_n)$上のある枝$e \in E'_n$の両端の節点u，vがそれぞれ，配置領域$\Re$内の座標(u_x, u_y)，(v_x, v_y)に置かれているとき，枝eのコストは2点間のマンハッタン長，

$$|e| = |v_x - u_x| + |v_y - u_y|$$

で定義する．MST $T_n(V_n, E'_n)$のコストは，E'_nの各枝のコストの和，

$$L(T_n) = \sum_{e \in E'_n} |e|$$

と定義する．

図1(a)にMSTの例を示す．図においてv_1, v_2, v_3はv_0に隣接するセルであり，e_1, e_2, e_3はMSTの枝を表す．ここで，v_0の位置を始点とし，v_1, v_2, v_3

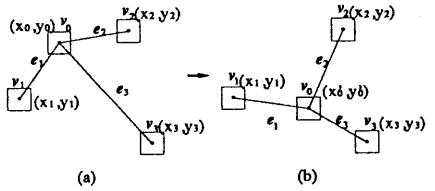

図1: improvement areaの考え方

の各位置を終点とするベクトルを考える．これらのベクトル和の方向にv_0を移動させて図1(b)のようにすると，枝e_1, e_2, e_3のコストの和が減少し，v_0に接続するネットのMSTのコストの総和が減少する．この考えをもとに，セルv_0とMST上でv_0に隣接するセルv_1, v_2, v_3の位置関係から，v_0のimprovement areaを定義する．

2.2 improvement areaの定義

Definition 1 あるセルv_0と接続するネットを$n_i \in N_{v_0}(i = 1, \cdots, m)$とし，そのMSTを$T_i(V_i, E'_i)$とする．$T_i(V_i, E'_i)$上で$v_0$と隣接するセルを$v_i(i = 1, \cdots, m)$とし，$v_i$の位置を$(x_i, y_i) \in \Re$とする．$v_0$に接続する枝，すなわち$v_0$と$v_i$を両端に持つ枝を$e_i = (v_0, v_i)$と表し，その集合を

$$E_{v_0} = \{e_i | (i = 1, \cdots, m)\}$$

と表す．E_{v_0}の各枝のコストの和は，

$$\sum_{i=1}^{m} |e_i| = \sum_{i=1}^{m} (|x_i - x_0| + |y_i - y_0|)$$

である．ここで，MST $T_i(V_i, E'_i)$を変えずに，v_0を位置$(x, y) \in \Re$に移動させると，E_{v_0}の各枝のコストの和は，x, yの関数として，

$$\sum_{i=1}^{m} |e_i| = \sum_{i=1}^{m} (|x_i - x| + |y_i - y|) \equiv C(x, y)$$

と表せる．ここで，*improvement area*は，

$$C(x, y) \leq C(x_0, y_0) \quad (1)$$

となる領域をA，すなわち，

$$A \equiv \{(x, y) \in \Re \mid C(x, y) \leq C(x_0, y_0)\} \quad (2)$$

を improvement area と定義する.

次に，あるセルをその improvement area 内に移動すると，このセルにつながるネットの MST のコストの総和が減少することを示す.

Theorem 1 あるセル v_0 の座標を (x_0, y_0) とし，v_0 に接続するネット $n_i \in N_{v_0}$ の MST を $T_i(V_i, E_i')$ とする. v_0 をその improvement area A 内の点 $(x, y) \in A$ に移動した後のネット n_i の MST を $T_i'(V_i, E_i'')$ とすると，

$$\sum L(T_i') \leq \sum L(T_i) \qquad (3)$$

が常に成り立つ.

Proof 1 **Theorem 1** を証明する.

step1 セル v_0 に接続するすべてのネットの MST の構造が不変の場合 ($T_i(V_i, E_i') = T_i'(V_i, E_i'')$) は，定義 (1) より自明.

step2 v_0 の移動で 1 本以上のネットの MST の構造が変わる場合.

セル v_0 は移動するが，MST の構造は不変の状態 T_i''' を考える. T_i''' と T_i のコストの差は，v_0 に接続する枝のコストだけが変化しているから，式 *1* より，

$$\begin{aligned} &\sum L(T_i''') - \sum L(T_i) \\ =\ & C(x, y) - C(x_0, y_0) \leq 0 \end{aligned} \qquad (4)$$

移動後に更新した MST が T_i' であるから，

$$L(T_i') \leq L(T_i''') \qquad (5)$$

したがって，全ネットの MST のコストの和について，

$$\sum L(T_i') \leq \sum L(T_i''') \qquad (6)$$

が成り立ち，(4),(6) より (3) が成り立つ.

以上から，**Theorem 1** が証明される. □

次に，コスト $C(x, y)$ の変化，すなわち，$C(x, y) - C(x_0, y_0)$ について次のことがいえる.

Theorem 2

$$\begin{aligned} D_x(x) &\equiv C(x, y_0) - C(x_0, y_0), \\ D_y(y) &\equiv C(x_0, y) - C(x_0, y_0) \end{aligned}$$

とおくと，improvement area は次の式，

$$C(x, y) - C(x_0, y_0) = D_x(x) + D_y(y) \leq 0 \quad (7)$$

を満たす (x, y) である.

この定理の証明は省略する. $D_x(x), D_y(y)$ はそれぞれ x,y のみの関数なので，式 (7) を満たす (x, y) は $D_x(x), D_y(y)$ をそれぞれ独立に計算して求めることができる.

2.3 improvement area の計算方法

Theorem 2 を用いて，improvement area を計算する方法について説明する.

節点 v の x 座標が x のとき，v から左側へ伸びる枝の数を $l(x)$，右側へ伸びる枝の数を $r(x)$ とする. 次に，節点 v の x 座標を $x+1$ に移動すると，右側の枝は 1 ずつ長さが減り，左側の枝は 1 ずつ長さが増える. したがって，

$$D(x+1) = D(x) + l(x) - r(x)$$

x の負方向への移動も同様に考えると，$D_x(x)$ は，

$$D_x(x) = \begin{cases} 0 \ (x = x_0) \\ D_x(x-1) + l(x-1) - r(x-1) \ (x > x_0) \\ D_x(x+1) + r(x+1) - l(x+1) \ (x < x_0) \end{cases}$$

となる. $D_y(y)$ も同様に計算できる.

図 2 に improvement area の例を示す. 図の左上部分は，あるセルとそれに接続する MST の枝である. セルを現在位置より x, y 方向それぞれ独立に移動したときの，$D_x(x)$ および $D_y(y)$ をセル位置に対応させて示している.

図の破線で囲んだ多角形領域が式 (7) の示す improvement area である. しかし，多角形の領域はその計算とデータ管理のコストが高くなる. そこで，今回の評価における実装は，この多角形領域の部分集合である，

$$\{(x, y) \mid D_x(x) \leq 0 \text{かつ} D_y(y) \leq 0\} \qquad (8)$$

となる矩形領域とし，improvement area の計算とデータ管理を簡単にした．図2の斜線部分が，実装において採用した improvement area である．以下の議論では improvement area に式(8)の定義を用いる．

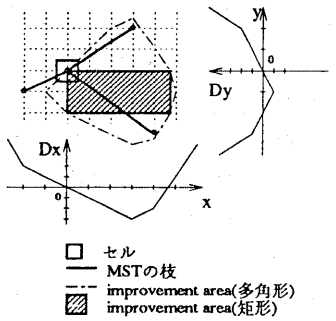

図 2: improvement area の例

2.4 improvement area を用いたセルの交換

ここで，improvement area を用いたペア交換による MST 配線長を改善する方法を説明する．

選択されたセル A について，その improvement area I_A を求め，交換する相手のセルは I_A の中にあるものに限定する．このとき，セル A，B の関係は図3に示すような2つの場合に分けられる．

1. セル A，B ともに，相手のセルの improvement area の中にある場合 (図 3(a)).

   セル A，B ともに MST のコストは減る．

2. セル B はセル A の improvement area の中にあり，セル A はセル B の improvement area の外にある場合 (図 3(b)).

   セル A の MST のコストは減るが，セル B の MST のコストは増えるため，全体の配線長の増減は判定できない．そこで，この場合は，セル A，B それぞれについて MST の隣接セルまでの枝のコストの変化のみを計算して，その和が減少する場合に交換後の位置とする．

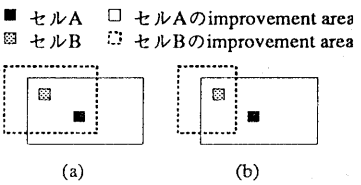

図 3: 交換を試みるセルペアの条件

3 MRPI アルゴリズム

MRPI では，improvement area を用いて MST の配線長を改善するセルペアを探索し，その交換を繰り返して配線長の改善を行なっている．

MRPI 全体の処理の流れを図 4に示す．入力は

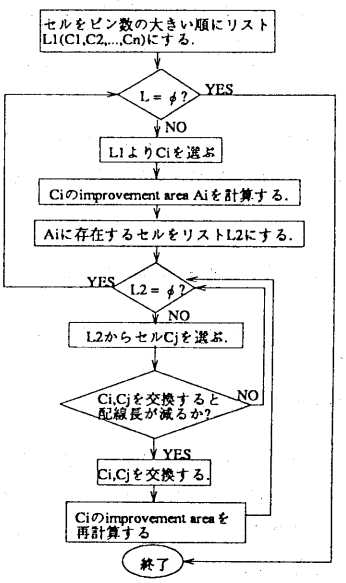

図 4: MRPI の処理の流れ

セル位置，ネットリスト，および各ネットの MST である．まず，セルをピン数の大きい順にソートしたリスト L1 を作り，次にピン数の大きいセルから順にセル C_i を選び，improvement area A_i を計算する．L2 は A_i 内にあるセルのリストであり，C_i と L2 のすべてのセルとの交換を行なう．

ここで，交換が成立するたびに動いたセルに接続するネットの MST の木を再構成する必要があ

る．しかし，交換毎に再構成の処理を行なうと時間がかかるため，同一セルがα回以上位置を更新したときに再構成を行なうようにした．後述の評価では$\alpha=2$としている．

4 評価

4.1 実験方法

実装したMRPIについて，SOGゲートアレイの実データを用いて評価を行なった．表1に実験を行なったデータのサイズを示す．計算機はSun4-

表 1: 実験データ

回路	セル数	ネット数	ピン数
A	962	4800	1599
B	3389	17087	4889
C	9021	12240	37748
D	9328	32967	11701
E	33829	38575	128719

1000ワークステーションを使用した．以下，実験結果の表2〜4中の時間の単位は秒，また改善率は次の式，

(改善前配線長)−(改善後配線長)/(改善前配線長)(%)

で計算される．配線長は断りのないかぎりMSTで計算している．

MRPIとの比較手法は次の3つとした．

1. PIセルをピン数の大きい順に選択し，そのセルと残りのセルとのすべての組合せについてペア交換を試み，配線長が減る場合に交換後の位置に更新する．

   PIでは，すべての組合せについて交換を試行するので，計算時間が膨大となり大規模回路では非現実的である．そこで，本実験では規模の大きい回路D,Eについては以下に説明するNPIで実験した．

2. NPI(Neighborhood Pairwise Interchange)

   PIと手順は同じだが，交換相手のセルを，選択したセルの近傍にあるセルに限定する．ここで，近傍は，そのセルを中心とする一辺2α(grid)の矩形と定義する．αの値は，MRPIで計算されるimprovement areaの大きさの平均値を選んでいる．

3. FDPR

   セルとその重心を近傍にあるセルとのペア交換を繰り返す．手順は次の通りである．

   step1 ピン数の大きい順にセルC_iを選択する．

   step2 セルC_iの重心を計算する．

   step3 C_iの重心を中心とする1辺2αの正方形内のセルC_kとの交換を試み，配線長が減る場合に位置を更新する．αはNPIと同様にimprovement areaの大きさの平均値を選んでいる．

   step4 交換成立の場合はセルC_kの重心を計算し*step3*を行なう．不成立の場合はC_iの重心近傍の他の候補セルを探索する．

   step5 交換可能なペアがなくなるまで*step3*〜*4*を繰り返す．

   step6 *step1*〜*5*を繰り返す．

4.2 MRPIの配線長改善能力

MRPIでは，FDPRにおける重心に相当するimprovement areaを用いて，探索空間を限定して高速化を図りつつ，MSTのコストを改善するような組合せをより多く試行して改善率の向上を図っている．

まず，MRPIの配線長改善能力について調べるために，ランダムな配置結果に対してMRPIとFDPRを試み，比較した．この結果を表2に示す．表には，改善後の論理配線長，初期配置に対する配線長改善率，および配置改善の時間を示している．ただし，ここでは，MRPIではMSTで，FDPRではBounding Boxで配線長計算を行なっている．MRPIはFDPRに対して改善率，交換回数ともに多くなる．速さではFDPRが勝っているが，単位時間当たりの交換数を比較すると，MRPIの方が良い．

次に，Mincutによる配置結果に対するMRPIの配線長改善能力について調べた．MRPIの初期配置として[15]で提案されているMincutによる配置(以下IPと呼ぶ)を用いた．MRPI，FDPR，

表 3: Mincut による初期配置に対する結果 (時間： sec 配線長:10^3grid)

	IP		MRPI				FDPR				PI(NPI)			
	時間	配線長	時間	配線長	改善率	交換数	時間	配線長	改善率	交換数	時間	配線長	改善率	交換数
A	71.9	247.0	6.2	240.0	3.26	56	2.8	248.4	-0.57	108	6000.6	224.4	9.15	358
B	469.8	1686.7	22.93	1662.6	1.43	427	6.72	1676.8	0.58	265	227502.6	1656.1	6.93	2808
C	2555.6	5261.1	64.0	5046.9	2.40	1542	181.9	5162.8	0.16	1585	595922.6	4858.1	6.05	5354
D	2407.7	6401.0	1647.0	6015.3	6.03	7148	14.9	6399.8	0.02	1006	5913.3	6141.2	4.06	12616
E	35975.8	15578.7	13821.2	15211.8	2.36	15966	241.8	15549.0	0.19	4323	83397.8	15093.5	3.11	17695

表 2: ランダムな初期配置の改善結果 (時間： sec)

	MRPI			
	改善率	時間	交換数	交換数/秒
A	44.6	198.2	3065	15.4
B	45.7	1570.0	24976	15.9
C	52.1	13829.3	145130	10.5

	FDPR			
	改善率	時間	交換数	交換数/秒
A	22.9	63.9	531	8.3
B	15.4	661.8	2102	3.2
C	17.3	3258.3	11786	3.6

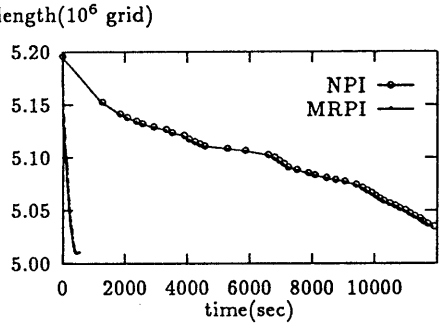

図 5: MRPI と NPI の配線長の時間変化 (回路 A)

PI(NPI) の実行時間，論理配線長，IP に対する改善率，交換回数を表 3に示す．ただし，ここでは配線長をすべて MST で計算している．

ランダムな初期配置に比べると，配線長改善率が 1～6% とわずかなのは，Mincut によって，結合の強いセルの集合がまとまって置かれることにより各セルの improvement area がそのセルのごく近傍となるためである．しかし，大規模回路では総配線長の数 % は数十万 grid に相当するので，未配線が多く出るような配線が難しい回路に対して効果はあると考えられる．

FDPR では，回路 A,C で MRPI と比べて交換数が多いにもかかわらず改善率が低くなった．BoundingBox のコスト改善と MST のコスト改善が一致しないことを示しているといえる．

NPI では，すべてのセルに対して一律の領域を近傍として探索しているため，無効となる交換の試行回数が MRPI よりも多く，時間がかかってしまう．回路 A における MRPI と NPI の配線長の時間変化を図 5に示す．改善の速さは MRPI の方がかなり速く，MRPI は improvement area を用いることによって，NPI のように単純に近傍を探索するよりも，改善率の良くなるセルペアを高速に検出できることがわかる．

4.3 実配線への効果

次に，IP，および IP+MRPI の配置結果に対して配線を行なった．表 4に配置後の論理配線長，実配線長，未配線数，配置時間，配線時間を示す．MRPI によって実配線長において 1% 前後の改善が得られ，また MRPI の方が未配線数が少ない結果となった．MST での改善が実配線に対しても効果があることがわかる．

5 まとめ

本稿では Mincut による配置結果を元に MST 配線長を改善する配置改善手法について述べた．各セルについて，MST 配線長を改善する領域 improvement area を定義し，これを用いて，交換によって MST 配線長を改善するセルの組合せを効率的に検出する方法を示した．実装した MRPI を SOG の実データに対して試み，配線長改善手法と

表 4: 配線結果 (時間 : sec 配線長: grid)

	IP					IP+MRPI				
	論理配線長	実配線長	# 未配線	配置時間	配線時間	論理配線長 / 改善率	実配線長	# 未配線	配置時間	配線時間
A	247038	300346	6	71.9	2062.4	238991(3.26)	288486	0	78.1	2398.1
B	1686703	1729821	8	469.8	17561.8	1662569(1.43)	1707552	6	492.7	19729.8
C	5261163	5257232	10	2813.4	33351.0	5046914(2.40)	5225292	4	2877.5	32027.0

しての有効性を示した.

MRPI の実行時間は Mincut に比べて短くはなるが, 大規模な回路では, その改善率に比べて時間がかかり過ぎている. この問題を解決策として, 平面走査法を用いて improvement area の重なりを探索することによって高速化を試みている. また, MRPI は配置領域全体に配線長改善を行なっているが, 領域全体でなく配線長がクリティカルとなるような部分への改善も可能である. improvement area への移動は局所的なものなので, 他の領域に影響を及ぼさずに, 単純な近傍への移動よりも効率的に改善が行なえると考えられる. これらの検討が今後の課題である.

謝辞

日頃御指導・御助言をいただく, 富士通研究所 CAD 研究部津田部長に深謝いたします. また, 本研究に当たり, 多大なるご協力をいただいた, 富士通 LSICAD 開発部の皆様に深く感謝いたします.

参考文献

[1] J. Cong and M. Smith. "A Parallel Bottom-Up Clustering Algorithm with Applications to Circuit Partitioningin VLSI Design". In *Proc. 30th DAC*, pp. 755–760, 1993.

[2] W. E. Donath, R. J. Norman, B. K. Agrawal, S. E. Bello, S. Y. Han, J. M. Kurtzberg, P. Lowy, and R. I. McMillan. "Timing Driven Placement Using Complete Path Delays". In *Proc. 27th DAC*, pp. 84–89, 1990.

[3] A. E. Dunlop and B. W. Kernighan. "A Procedure for Placement of Standard-Cell VLSI Circuits". *IEEE Trans. on Computer-Aided Design*, Vol. CAD-4, No. 1, pp. 92–98, January 1985.

[4] C. M. Fiducca and R. M. Mattheyses. "A Linear-Time Heuristic for Improving Network Partitions". In *Proc. 19th DAC*, pp. 175–181, 1982.

[5] T. Gao, P. M. Vaidya, and C. L. Liu. "A New Performance Driven Placement Algorithm". In *Proc. ICCAD*, pp. 44–47, 1991.

[6] M. Hanan, Peter K. Wolef, and Barbara J. Agule. A Study of Placement Techniques. *J.Design Automation and Fault Tolerant Computing*, Vol. 1, No. 1, pp. 28–61, October 1976.

[7] Mark R. Hartoog. Analysis of Placement Procedures for VLSI Standard Cell Layout. *23rd DAC*, pp. 314–319, 1986.

[8] M.R.Gareyand D.S.Johonson. The Rectlinear Steiner tree Problem is NP-complete. *SIAM Journal Applied Mathematics*, Vol. 32, pp. 826–834, 1977.

[9] S. Goto and T. Matusda. Partitioning, Assignment and Placement. In *Layout Design and Verification*, volume 4, chapter 5. North Holland, 1986.

[10] C. Sechen. *VLSI Placement and Global Routing Using Simulated Annealing*. Kluwer Academic Publishers, 1988.

[11] H. Shiraishi and F. Hirose. "Efficient Placement and Routing Techniques for Master Slice LSI". In *Proc. 17th DAC*, pp. 191–197, 1980.

[12] 三島英樹, 若林真一, 吉田典可. "タイミング制約を考慮したセル配置の一手法". In *VDL91-82*, pp. 17–24, 1991.

[13] 川本貞行, 若林真一, 小出哲士, 吉田典可. "ハイパーグラフ分割のための動的クラスタリングに基づくヒューリスティックアルゴリズム". In *VDL94-64*, pp. 7–12, 1994.

[14] 長谷川隆. "ネット緩和法を用いた配置改善アルゴリズム". In *VDL90-63*, pp. 17–24, 1990.

[15] 澁谷利行, 河村薫. "Hill-Climbing を用いたパーティショニング最適化手法". In *VDL93-74*, pp. 1–8, 1993.