情報処理学会論文誌 Vol. 52 No. 2 756–765 (Feb. 2011)

3次元格子スタイナー木を求める並列遺伝的アルゴリズム

大村道郎[†1] 瀬能浩史[†2,*1] 上田真琴[†2]

近年の大規模集積回路における製造技術の進歩にともない，MCM に加えて，回路素子自体を 3 次元に集積化するものや，複数のダイを貼り合わせて 3 次元化する 3 次元 VLSI に関する研究が発表されるようになってきている．これらの LSI では，3 次元の配線が重要になってくる．格子スタイナー木は，LSI 概略配線設計等にも応用される重要な問題の 1 つであり，ナノ CMOS 時代の配線に関しては，折れ曲がり（ビア）が増えると，配線遅延が増え回路の性能を落とす等，様々な悪影響を及ぼす．しかし，最小限の折れ曲がりを持つ格子スタイナー木では障害物を柔軟に避けることができない．本論文では，空間上に 3 次元座標を持つ点 S の集合，それらを結ぶユークリッド最小全域木，障害物が与えられたとき，木の枝を X 軸，Y 軸，および Z 軸に平行な線分に置き換え，最小 +1 の折れ曲がりを使うことにより，配線長が短く，障害物がある場合は柔軟に避ける，3 次元最小格子スタイナー木を求める並列遺伝的アルゴリズムを提案する．本手法では評価基準に，層を積み重ねる Z 軸方向の長さに重みを付けた木の長さの合計と木の直径の線形和を用いた．本論文では提案する手法と性能評価のために行った実験結果について述べ，手法の有効性を示す．

Parallel Genetic Algorithm for 3-D Rectilinear Steiner Tree

Michiroh Ohmura,[†1] Hirofumi Senou[†2,*1] and Makoto Ueda[†2]

As manufacturing technology has advanced in recent years, 3-D VLSI in which circuits or dies are piled on top of each other, MCM and so on have been the focus of attention. A rectilinear Steiner tree is one of the most important problems that are applied to the global routing in LSI or other designs. In the routing design, wire bends (vias) had various bad influences such as propagation delay and therefore degrade circuit performance. However, rectilinear Steiner trees with the minimum number of bends cannot avoid obstacles flexibly. In this paper, we propose a parallel genetic algorithm which can obtain a rectilinear Steiner tree with short wire length and can avoid obstacles flexibly by using minimum+1 bends. In our method, given a set of points S in the space, a Euclidean minimum spanning tree for S, and obstacles, each edge of the Euclidean spanning tree is replaced by the segments which are parallel to the X-axis, the Y-axis, or the Z-axis. For the evaluation criteria, a linear sum of the wire length and diameter of the rectilinear Steiner tree is used. The experimental results to show the effectiveness of the proposed method are also shown.

1. はじめに

最小格子スタイナー木[1)] を求める問題は，LSI 概略配線設計等にも応用される重要な問題の 1 つである．この問題は一般的に，現実的な時間で解決することが困難な NP 困難[2)] と呼ばれる問題に属するため，様々なヒューリスティック手法が用いられる．障害物が与えられる場合もますます重要な問題となっており，これまで多くの手法が提案されているが，通常の最小格子スタイナー木よりさらに難しい問題となるため，多項式時間の最適アルゴリズムは存在しそうにない[3),4)]．

障害物を考慮した 2 次元の格子スタイナー木に関する研究として，文献 3) ではトラックグラフと蟻コロニー最適化アルゴリズムに基づいた手法を提案しており，ビア数の最小化についても考慮している．しかし，3 次元のスタイナー木については議論していない．

文献 4) では CDCTree を提案することにより，文献 3) より短い配線長を求めている．文献 5)，6) は迷路法に基づいており，文献 7) では OARG というグラフを提案し，3 ステップで効率良く配線長の短い格子スタイナー木を求めている．また，文献 8) では Geosteiner に基づき，最適解を求めている．文献 4)–8) では，3 次元のスタイナー木や，ビア数に対応する折れ曲がり数の制限は考慮していない．

文献 9) では層ごとの配線方向を設定し，各層に 2 次元の障害物が存在する多層配線モデルを用いており，ビア数の最小化についても考慮している．しかし，通常の 3 次元スタイナー木にそのまま適用することは難しい．

†1 広島工業大学工学部
Faculty of Engineering, Hiroshima Institute of Technology
†2 広島工業大学大学院工学研究科
Graduate School of Engineering, Hiroshima Institute of Technology
*1 現在，新川電機株式会社
Presently with Shinkawa Electric Co., Ltd.

遺伝的アルゴリズムに基づく手法としては，文献 10)，11) 等がある．文献 10) では格子スタイナー木のセットを求めており，エルモア遅延も取り入れている．文献 11) では平面上に与えられた各点を結ぶ全域木を求め，その枝を 1 本の水平配線と 1 本の垂直配線に置き換えることで，格子スタイナー木を求めている．文献 10)，11) では，3 次元のスタイナー木や，ビア数に対応する折れ曲がり数の制限は考慮していない．また，文献 3)–11) のいずれの文献においても，アルゴリズムの並列化に関しては議論されていない．

近年の大規模集積回路における製造技術の進歩にともない，MCM [12] に加えて，回路素子自体を 3 次元に集積化するものや，複数のダイを貼り合わせ，スルーシリコンビア（TSV）を用いてダイの間を接続する 3 次元 VLSI に関する研究が行われるようになってきている[13),14)]．これらの LSI では 3 次元の端子位置を考慮した 3 次元の配線[15)] が重要となってくるとともに，3 次元の障害物も考慮する必要がある．また，ナノ CMOS 時代のこれらの配線に関して，折れ曲がり数に対応するビア数が増えると，配線遅延が増え，回路の性能を落とすと考えられ，ネットごとのビア数の最小化が重要になる[12)]．他にも製造歩留まりの低下，チップ面積の増加，コストの増加等の悪影響があるため，これまでビア数最小化に関して多くの研究が発表されてきた[16)]．したがって，折れ曲がりを制限した格子スタイナー木の構成が有効であると考えられる．しかし最小限の折れ曲がりを持つ格子スタイナー木[11)] を単純に 3 次元化した手法では障害物を柔軟に避けることができない．

本論文では，空間上に 3 次元座標を持つ点 S の集合，それらを結ぶユークリッド最小全域木，障害物が与えられたとき，木の枝を X 軸，Y 軸，および Z 軸に平行な線分に置き換えることにより，3 次元最小格子スタイナー木を求める並列遺伝的アルゴリズムを提案する．このとき最小 +1 の折れ曲がりを使い，共有部分が長くなりそうな分岐点を求めてそこで折れ曲がることにより配線長を短くしようと試み，障害物がある場合はより柔軟に障害物を避ける．本手法では評価基準として，層（2.1 節参照）を積み重ねる Z 軸方向の長さに重みを付けた木の長さの合計と木の直径の線形和を用いた．

実験の結果，まず障害物がない場合，文献 11) を単純に 3 次元化した従来手法に比べて，提案手法では平均 2.0%コストが小さい 3 次元格子スタイナー木を求めることができた．また小さいデータについて配線長の最適解と比較した場合，従来手法では平均 2.6%増加したのに対し，提案手法ではわずか平均 0.6%の増加に抑えることができた．さらに障害物がある場合，従来手法に比べて，2.9%コストが小さい 3 次元格子スタイナー木を求めることができた．本論文では，提案するアルゴリズムと性能評価のために行った実験結果について述べる．

2. 準　　備

初めに準備として，まず 3 次元 VLSI について述べ，次に本論文で問題の入力として仮定するユークリッド全域木，さらに問題の出力となる 3 次元格子スタイナー木について説明する．

2.1　3 次元 VLSI

本論文で考える 3 次元 VLSI では，複数のダイを貼り合わせることにより 3 次元に積層し，スルーシリコンビア（TSV）を用いてダイの間を接続する．以下では 1 枚のダイのことを単に層と呼ぶ．**図 1** の例では 3 層が積み重なっており，TSV に対応する Z 方向の配線はそれらの層を貫通することができる．ここでは，大まかな配線経路（概略配線）としての格子スタイナー木を考えるので，単純化し，配線は十分細かい 3 次元の立方格子上で行うものと仮定するが，以降では単に 3 次元空間と記す．

2.2　ユークリッド全域木

重み付き無向グラフを $G = (V, E, w)$ で表す．ここで V は節点の集合，E は枝の集合，w は各辺 $e \in E$ に重みを割り当てる関数とする．

3 次元空間上に点 v の集合 S が与えられるとする．各点 v_i の座標を (x_i, y_i, z_i) とするとき，v_i，v_j 間のユークリッド距離 YD_{ij} を $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}$ と定義する．また，S に含まれる点の総数を $|S|$ と表す，

S に属する任意の 2 点 v_i，v_j 間のユークリッド距離 YD_{ij} をグラフ G の枝の重みに対応させると，$|S|$ 個の節点からなる完全グラフを構成することができる．このグラフ上での全域木を，S に対するユークリッド全域木（Euclidean spanning tree）と呼ぶ．

本論文では，ナノ CMOS 時代の高性能な配線を実現するため，空間上に与えられた 3 次元座標を持つ点だけではなく，配線形状を考慮し，なんらかの基準で最小化された，それら

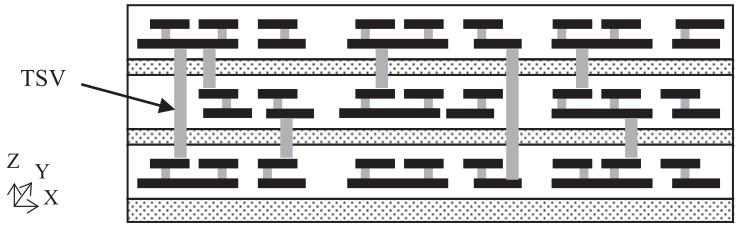


図 1　3 次元 VLSI と TSV
Fig. 1　3-D VLSI and TSV.

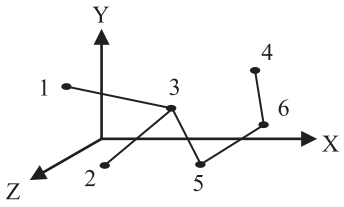


図 2　ユークリッド全域木
Fig. 2　Euclidean spanning tree.

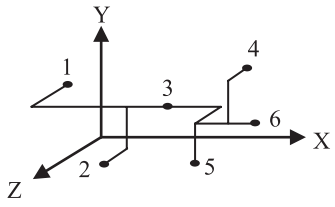
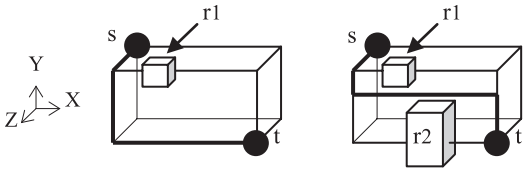


図 3　格子スタイナー木
Fig. 3　Rectilinear Steiner tree.



(a)　折れ曲がり数最小　　(b) 折れ曲がり数最小＋ 1

図 4　折れ曲がりと障害物
Fig. 4　Number of bends and obstacles.

の点を結ぶユークリッド全域木が与えられると仮定する．

［**例 1**］　本論文で仮定するユークリッド全域木の例を**図 2** に示す．　□

2.3　格子スタイナー木

3 次元空間上で 2 つの点を接続する X 軸，Y 軸，または Z 軸に平行な線分を，ここでは単に，線分と呼ぶことにする．このとき，連結，非サイクルで，端点でしか交わらない線分の集まりを格子木（rectilinear tree）と呼ぶ．また，3 次元空間上に n 個の点の集合 S が与えられるとする．それら n 個の点が，含まれるいずれかの線分の端点となる格子木を，格子スタイナー木（rectilinear Steiner tree）と呼ぶ[1]．

格子スタイナー木において，木を構成する線分の長さの和を配線長 L と呼び，それが最小となるものを最小格子スタイナー木（rectilinear Steiner minimum tree）と呼ぶ．

また，格子スタイナー木において，任意の 2 端子間に対する経路の長さの最大値を直径 ϕ と定義する．この直径は格子スタイナー木が表す配線のうち，遅延の最大値に関係し，ϕ が小さいと，端子間の遅延の差が減少する．配線長 L を最小化しただけでは，特定の端子間の配線にしわ寄せが行き，大きな遅延によって全体の性能を落とす可能性があるので，直径 ϕ を小さくすることによって遅延の差が大きくならないようにしている．本論文では，この格子スタイナー木について議論する．

［**例 2**］　本論文で議論する格子スタイナー木の例を**図 3** に示す．　□

3.　問題の定式化

スタイナー木は，LSI の概略配線等に応用される．3 次元の配線において，層に対して垂直（Z 軸方向）に通過する配線は，同一層内（X 軸，Y 軸方向）の配線に比べてコストがかかると考えられる．そこで，本論文では，直径と配線長に対して，Z 方向の配線に重み k（$\geqq 1$）をかけることにより，このコストを考慮する．また，配線長と直径を同時に考慮するため，本論文では，$f = c1 \cdot L + c2 \cdot \phi$ を評価の基準とする．ただし，L は総配線長，ϕ は直径，$c1$，$c2$ は定数であり，$c1 \gg c2$ とする．

また，ナノ CMOS 時代のこれらの配線に関しては，折れ曲がり数に対応するビア数が増えると，配線遅延が増え，回路の性能を落とすと考えられる[12] ため，折れ曲がりの少ない配線が求められる．

一般に，ユークリッド全域木の枝が，X 軸，Y 軸，または Z 軸に平行な同一直線上にある場合，その枝自身が，1 本の線分となり，折れ曲がりなしで結ぶことができる．ユークリッド全域木の枝が，XY 平面，YZ 平面，または ZX 平面に平行な同一面上にある場合，2 本の線分を用いて，折れ曲がり 1 回で結ぶことができる．一般に 3 次元のユークリッド全域木の枝は，x 線分，y 線分，z 線分の 3 本を用いて，折れ曲がり 2 回で結ぶことができる．

ユークリッド全域木の各枝 $\{s,t\}$ に対し，点 s，t（$\in S$）を結ぶ線分の折れ曲がりを最小（2 以下）に制限した場合，折れ曲がり点は**図 4** (a) に示す，s，t を囲む最小の直方体の角の部分に存在する．もし，折れ曲がりを 1 つ追加すれば，同図 (b) に示すように，さらに柔軟に障害物（図中の r1，r2）を避けることができる．このとき節点 s，t を含む最小の直方体の外を迂回しないとすると，折れ曲がり点は，直方体の辺上に存在する．これは，面内部や直方体内部で折れ曲がると 3 回の折れ曲がりでは接続できないことから明らかである．

［**問題 4ST**］　入力として，① 3 次元座標を持つ点 S の集合と，② それらを結ぶ 3 次元のユークリッド全域木，③ 直方体の障害物の集合 R が与えられる．このとき，以下の条件を満たし，目的関数を最小化する，3 次元格子スタイナー木を求めよ．

（条件）　① 3 次元格子スタイナー木を構成する線分は，障害物を通過しない．

② ユークリッド全域木の各枝 $\{s,t\}$ に対し，点 s，t（$\in S$）を結ぶ線分の折れ曲がり数は最小 +1 以下．③ 線分は，s，t を囲む最小の直方体の外に出ない．

（目的関数）　$f = c1 \cdot L + c2 \cdot \phi$（ただし，$L$，$\phi$ は Z 方向の重みを考慮した配線長，直径であり，$c1$，$c2$ は定数で，$c1 \gg c2$ とする.）　□

4. 提案手法

4.1 コード化

提案する遺伝的アルゴリズムでは，与えられた 3 次元ユークリッド全域木の枝 $\{s,t\}$ を，x 線分，y 線分，z 線分に置き換える．今 $s = (x1, y1, z1)$，$t = (x2, y2, z2)$ とするとき，$x1 \neq x2$，かつ，$y1 \neq y2$，かつ，$z1 \neq z2$ である場合，x 線分，y 線分，z 線分をそれぞれ 1 本ずつ用いれば，折れ曲がり数最小（この場合 2 回）で置き換えることができる．この xyz，xzy，yxz，yzx，zxy，zyx の 6 通りの順列を遺伝子の前半とする．

問題の条件は，折れ曲がり数最小 +1 以下なので，もう 1 本線分を用いることができる．そこで，提案手法では，x 線分，y 線分，または z 線分のいずれか 1 本を 2 本に分割する．X 軸，Y 軸，および Z 軸に平行な 4 つの線分の順列として，x が 2 回現れる場合，$4!/2! = 12$ 通りあるが，このうち，xx のように続く場合を除くと，次の 6 通りが得られる．① $xy\boldsymbol{X}z$，② $xyz\boldsymbol{X}$，③ $xz\boldsymbol{X}y$，④ $xzy\boldsymbol{X}$，⑤ $yxz\boldsymbol{X}$，⑥ $zxy\boldsymbol{X}$．y，z が 2 回現れる場合についても同様で，計 18 通りの順列が考えられる．これを先ほどの x，y，z 線分の 6 通りの順列を表す遺伝子前半に対して，重複しないよう，3 つずつ振り分け，遺伝子の後半に対応させる．順列と遺伝子を**表 1** に示す．ただし，図中，文中の太字は重複している文字を表す．

したがって，各個体の遺伝子は 2 つの部分から構成される．各遺伝子の前半は 0〜5 の数字で，基本的な 6 通りの線分の順列を表し，後半は 0〜2 の数字で，分割した線分を用いた折れ曲がり方を表す．これを使ってできるだけ線分を共有し，障害物がある場合はそれを避ける．

提案手法では，ユークリッド全域木の枝を折れ曲がりの少ない 4 つの線分に置き換えるとき，すべての順列について置き換える代わりに，表 1 の遺伝子によって線分の順列を決める．また，遺伝子では決まらない，1 つの線分を分割する座標については，5 章の最適解との比較で説明する 3 次元の Hanan グリッド上のあらゆる位置で線分を分割する代わりに，4.2 節で説明する分岐点を求めてそこで線分を分割する．

4.2 分岐点と手法の概要

今，共通の節点 s を持つユークリッド全域木の 2 つの枝 $a_i = \{s,t\}$，$a_j = \{s,u\}$ を，x，y，または z 軸に平行な線分で接続する場合を考える．ここでは遺伝子を考えず，線分が通る可能性がある直方体のみに注目する．

節点 s，t を含む最小の直方体と節点 s，u を含む最小の直方体の位置関係は，以下の (a)〜(d) に分けて考えることができる（**図 5** 参照）．以下では線分をまったく共有できない (a) の場合を除く．

表 1　順列と遺伝子
Table 1　Permutations and genes.

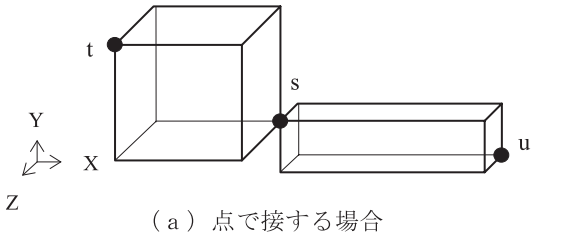
前半＼後半	0	1	2
0 (xyz)	xy**X**z	xyz**X**	xyz**y**
1 (xzy)	xz**X**y	xzy**X**	xzy**Z**
2 (yxz)	yx**y**z	yxz**y**	yxz**X**
3 (yzx)	yz**y**x	yzx**y**	yzx**Z**
4 (zxy)	zx**Z**y	zxy**Z**	zxy**X**
5 (zyx)	zy**Z**x	zyx**Z**	zyx**y**

(a) 2 つの直方体が点 s のみで接している場合：線分をまったく共有できない．
(b) 節点 s を含む 1 辺のみで接している場合：最大で線分 1 本の共有ができる．
(c) 節点 s を含む面で接している場合：最大で線分 2 本の共有ができる．
(d) 節点 s を含む直方体どうしが重なる場合：最大で線分 3 本の共有ができる．

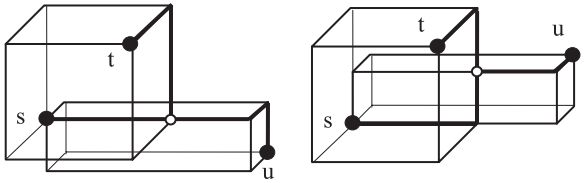
節点 s，t，u の座標をそれぞれ (xs, ys, zs)，(xt, yt, zt)，(xu, yu, zu) とする．x 座標について非減少順にソートした結果を $(x1, x2, x3)$，y 座標，z 座標についても同様であるとすると，2 つの枝 a_i，a_j に対する分岐点 D_{ij} を $(x2, y2, z2)$ と定義する．

枝が a_i，a_j の 2 つしかなく分岐点 D_{ij} が与えられる場合を考える．このとき提案手法において，分岐点 D_{ij} は線分を分割する座標を与える．すなわち，x 線分は $x2$ で，y 線分，z 線分は，それぞれ $y2$，$z2$ で分割する．ただし，線分の順序は遺伝子に依存し，D_{ij} は遺伝子を考慮せずに決めているので，D_{ij} の位置で必ず分岐できるわけではない．

［**例 3**］　**図 6** において，$\{s,t\}$ の遺伝子が “$xzy\boldsymbol{Z}$” であり z 線分が $z2$ の位置で分割され，$\{s,u\}$ の遺伝子が “$xz\boldsymbol{X}y$” であり x 線分が $x2$ の位置で分割されるとき，折れ曲がる各線

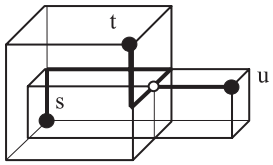


（a）点で接する場合



（b）辺で接する場合

（c）面で接する場合



（d）重なる場合

図 5　2 つの直方体の位置関係
Fig. 5　Relationship of the positions between two rectangular solids.

分の座標は，以下のようになる．ここで，$zs < zu < zt$ より $z2 = zu$，また $xs < xt < xu$ より $x2 = xt$ となる．この例では，実際に分岐する点 $(x2, ys, z2)$ は D_{ij} とは異なる．

$\{s, t\}$：x 線分 (xs, ys, zs)–(xt, ys, zs)，z 線分 (xt, ys, zs)–$(xt, ys, z2)$，
　　y 線分 $(xt, ys, z2)$–$(xt, yt, z2)$，z 線分 $(xt, yt, z2)$–(xt, yt, zt)

$\{s, u\}$：x 線分 (xs, ys, zs)–$(x2, ys, zs)$，z 線分 $(x2, ys, zs)$–$(x2, ys, zu)$，
　　x 線分 $(x2, ys, zu)$–(xu, ys, zu)，y 線分 (xu, ys, zu)–(xu, yu, zu)　□

一般に 1 つの枝 a_i に対し，複数の枝 $a_m, \cdots, a_j, \cdots, a_n$ との間に複数の分岐点 $D_{im}, \cdots, D_{ij}, \cdots, D_{in}$ が求まる．1 つの枝 a_i に対し，分岐点 $D_{im}, \cdots, D_{ij}, \cdots, D_{in}$ に対する線分の分割をすべて行うと時間がかかる．そこで，提案手法では 1 つの枝 a_i ごとに，

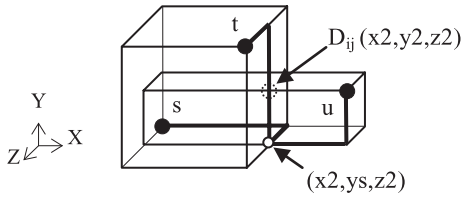


図 6　分岐点
Fig. 6　Branch point.

他の枝とは関係なく，1 つの分割位置を決定したい．その分割位置を与える分岐点を D_i と表す．以下に，分岐点 D_i の求め方を説明する．

今，共通の節点 s の座標を (x_s, y_s, z_s)，D_{ij} の座標を (x_D, y_D, z_D) するとき，s，D_{ij} 間の（3 次元の）マンハッタン距離を $|x_s - x_D| + |y_s - y_D| + |z_s - z_D|$ と定義する．このとき s，D_{ij} 間のマンハッタン距離が大きくなるほど，共有する線分を長くできる可能性がある．

今，ユークリッド全域木の枝の集合 A に属するすべての枝の組に対し，共通点 s と分岐点間のマンハッタン距離が最も大きくなる分岐点を D_{pq} とする．このとき，枝の組 $\{a_p, a_q\}$ は分岐点 D_{pq} で線分の分割座標を与えると，共有線分を長くできる可能性がある．本手法では a_p に対する分岐点 D_p を，他の枝に関係なく，この D_{pq} とする．また a_q に対する分岐点 D_q も，他の枝に関係なく，この D_{pq} とする．枝の集合 A から a_p，a_q を除き，同様の手順を繰り返し，1 つの枝 a_i に対する 1 つの分岐点 D_i を求める（手続き Find_D 参照）．

提案手法では，求まった D_i の座標を用いて各枝 a_i ごとに線分の分割座標を決める．すなわち枝 a_i に対する分岐点 D_i の座標を $(x2, y2, z2)$ とすると，x 線分は $x2$ で，y 線分，z 線分は，それぞれ $y2$，$z2$ で分割する．a_i に対し分岐点 D_i が決められていない場合は，分割位置を線分の終端とし（すなわち分割せず），単純に最小の折れ曲がりで接続する．手続き Find_D を以下に示す．

［**手続き Find_D**］
ユークリッド全域木の枝の集合を A とする．
S1：A に属する任意の枝の組 $\{a_i, a_j\}$ に対し，もし共通の節点 s を持ち，かつ，(a) でなければ，分岐点 D_{ij} を求める．
S2：s，D_{ij} 間のマンハッタン距離の非増加順に D_{ij} をソートし，得られた系列を E とする．
S3：系列 E の先頭から D_{ij} を取り出し，a_i，a_j がともに A の要素であれば S3.1，S3.2 を実行する．

S3.1：D_{ij} を a_i に対する分岐点 D_i とし，同様に D_{ij} を a_j に対する分岐点 D_j とする．

S3.2：集合 A から a_i，a_j を，系列 E から D_{ij} を取り除く．

S4：系列 E が空であれば終了，そうでなければ S3 へ．

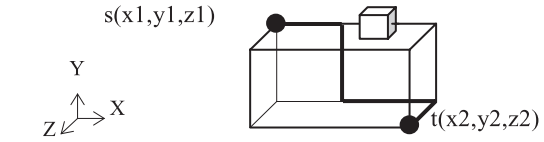
枝の総数を $|A|$ とすると，枝の組の総数は $|A| \times (|A|-1)/2$ となる．それらに対し定数回の計算で分岐点 D_{ij} を求めるには $O(|A|^2)$ の計算時間がかかる．さらにそれらをソートし，定数回の計算で D_i を求めるには，$O(|A|^2 \log |A|^2)$，すなわち $O(|A|^2 \log |A|)$ の計算時間がかかる．

障害物がある場合は分割位置を移動して，それを避ける．2 点 s，t の座標を $(x1, y1, z1)$，$(x2, y2, z2)$ とし，遺伝子前半が “0” (xyz) の場合を考える．遺伝子後半も “0” のとき，置き換える線分は，$(xy\boldsymbol{X}z)$ となり，x 線分を，障害物に当たる手前の座標 xr で分割する（**図 7** (a) 参照）．s，t 間の各線分は次のようになる．

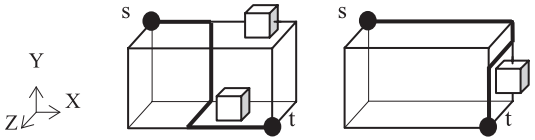
$\{s, t\}$：x 線分 $(x1, y1, z1)$–$(xr, y1, z1)$，y 線分 $(xr, y1, z1)$–$(xr, y2, z1)$，
　　x 線分 $(xr, y2, z1)$–$(x2, y2, z1)$，z 線分 $(x2, y2, z1)$–$(x2, y2, z2)$

同様に，遺伝子が “01” $(xyz\boldsymbol{X})$ のとき，および “02” $(xyz\boldsymbol{Y})$ のときの例を，それぞれ図 7 (b)，(c) に示す．

以下に提案する並列遺伝的アルゴリズム ST4B の概要を示す．なお，母集団の分割は島モデル[17] に基づいている．



（a）遺伝子 ”00” (xy**X**z)



（b）　遺伝子 "01" (xyz**X**)　　（c）　遺伝子 "02" (xyz**Y**)

図 7　障害物と遺伝子
Fig. 7　Obstacles and genes.

［**アルゴリズム ST4B**］

S1：初期集団をランダムに生成する．

S2：母集団を，島と呼ばれる複数のサブ母集団に分割する．

S3：各島ごとに，与えられた世代数（50 世代）に達するまで，以下の S3.1–S3.2 を繰り返す．

S3.1：与えられた回数（10 回）だけ S3.1.1–S3.1.7 を繰り返す．

S3.1.1：手続き Find_D によってユークリッド全域木の各枝 a_i に対する分岐点 D_i を求める．

S3.1.2：ユークリッド最小全域木の 2 点 s，t 間を遺伝子に基づき X 軸，Y 軸，および Z 軸に平行な線分の並びに置き換える．このとき分岐点 D_i が決められている場合は，その座標を使って線分を分割し，決められていない場合は分割位置を線分の終端とする（すなわち分割しない）．

S3.1.3：障害物をチェックする．線分が障害物を通過するとき，線分の分割座標を移動しそれを避ける．もし避けることができなかったときは，配線長を無限大とする．

S3.1.4：各個体の適応度を求める．

S3.1.5：適応度により，次世代の個体を選択する．

S3.1.6：交叉により，個体を 2 個ずつ組み合わせ，新しい個体を生成する．

S3.1.7：ある確率で突然変異させる．

S3.2：移住を行う（**図 8** 参照）．

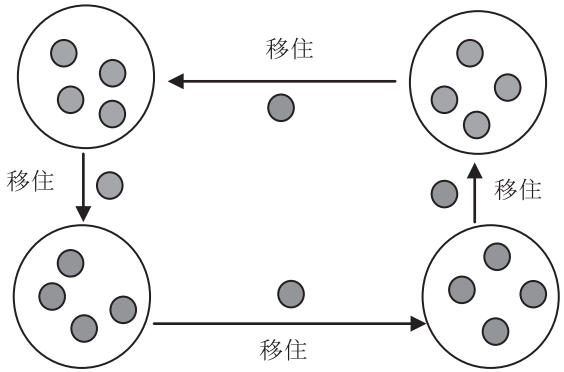


図 8　島（サブ母集団）と移住
Fig. 8　Islands and migration.

S4：すべての島において，最も適応度の高いスタイナー木の座標を出力し，終了する．

S3.1.3 の障害物チェックでは，各線分に対し，各障害物（直方体）の座標で比較して，X 座標，Y 座標，および Z 座標で同時に重なりを持つかどうかを，1 つずつ総当たりで調べる．ユークリッド全域木の節点数を n とすると，その枝数は $n-1$ となり，ある遺伝子によって置き換えられる線分数は $4(n-1)$ で $O(n)$ となる．一方障害物の数を R とすると座標の数はその定数倍で，結局，ある 1 つの遺伝子に対し，障害物チェックに $O(nR)$ の計算時間がかかる．ただし，平面走査法[18] 等を応用して探索範囲を制限すると，もっと効率良くチェックできると考えられる．

本手法で用いる遺伝的アルゴリズムでは，以下の遺伝的操作を用いる．まず，選択では，エリート保存戦略を用いる．次に交叉では，一点交叉を用いる．交叉確率は 0.6 と設定した．さらに突然変異では，任意の確率により，0〜5，および 0〜2 の数字で表される遺伝子を強制的に変化させる操作を行う．突然変異の起こる確率は 0.05 に設定した．

5. 実験結果

提案する並列遺伝的アルゴリズムおよび従来手法を，Intel Core 2 Duo（2.4 GHz），2 GByte メモリの計算機 1 台をサーバノード，同計算機 2 台（計 4 コア）を計算ノードとし，Pelican HPC v1.7 上で，Open MPI 1.2.3，GNU C++ 4.3.1 を用いてノード数 4 の島モデルとして実現した．各手法における GA では，個体数を 20×4，世代数 50 とし，移住は 10 世代ごとに行った（S3.1 の回数 $= 10$）．

まず，障害物がない場合について，折れ曲がり 2 回以下で接続する従来手法[11] を単純に 3 次元化した手法と本手法を比較した．実験は節点数 $n = 5$ から 150 の，乱数を用いて作成した 3 次元座標を持つユークリッド最小全域木それぞれ 10 個ずつに対し，配線長 L，直径 ϕ，問題の目的関数に対応するコスト $Cost$，実行時間 $Time$ (sec) の各平均について比較した．ここで，配線長の Z 軸方向の係数は 2.0，目的関数の係数は，$c1 = 10.0$，$c2 = 1.0$ とした．実験の結果，従来手法に比べてコストで平均 2.0%小さい 3 次元格子スタイナー木を求めることができた（**表 2** 参照）．これは，提案手法で手続き Find_D を用いて決めた，1 つ余分な折れ曲がり座標を使うことにより，線分の共有部分を長くすることができ，結果としてコストを小さくすることができたためと考えられる．

また，小さいデータについては配線長 L の最適解とも比較した．一般に格子スタイナー木の最適解は，3 次元以上についても Hanan グリッド上に存在する[19]．そこで枝を置き換える線分のすべての並びに対し，与えられたユークリッド全域木の節点座標から求めた 3 次元の Hanan グリッドで線分を分割することにより最適解を求めた．ここで，配線長の Z 軸方向の係数は 1.0 とした．なお，最適解は，同計算機 24 台（計 48 コア）を計算ノードとした Pelican HPC v1.7 上で，48 ノードの並列計算で求めた．節点数 $n = 5$ から 8 の同様のデータに対し，従来手法では最適解に比べて配線長で平均 2.6%の増加で 3 次元格子スタイナー木を求めていたのに対し，提案手法では最適解に比べて平均 0.6%の増加に抑えることができた（**表 3** 参照）．問題の定式化では最小 +1 の折れ曲がりを許しているが，+1 の折

表 2　障害物がない場合
Table 2　Testing cases with no obstacles.

n	Conventional method				Proposed method			
	L	ϕ	Cost	Time	L	ϕ	Cost	Time
5	318.4	266.2	3450.2	0.086	312.5	219.4	3344.4	0.151
6	398.5	273.0	4258.0	0.116	393.1	248.0	4179.0	0.171
7	443.5	312.5	4747.5	0.139	439.0	290.5	4680.5	0.219
8	466.6	349.6	5015.6	0.139	455.8	310.4	4868.4	0.245
9	551.2	374.3	5886.3	0.197	542.4	348.8	5772.8	0.297
10	583.1	366.9	6197.9	0.215	574.3	340.9	6083.9	0.331
15	772.6	475.8	8201.8	0.496	762.2	456.5	8078.5	0.582
50	1764.5	732.0	18377.0	10.258	1736.6	690.4	18056.4	10.544
100	2700.7	932.0	27939.0	76.801	2654.8	888.2	27436.2	77.105
150	3470.4	908.9	35612.9	255.156	3419.0	861.6	35051.6	256.591

表 3　最適解との比較
Table 3　Comparison with optimal solutions.

n	Opt. value		Conventional method		Proposed method	
	L	Time	L	Time	L	Time
5	238.1	0.167	245.4	0.077	239.8	0.133
6	290.4	11.518	296.7	0.077	291.8	0.193
7	328.7	753.339	335.7	0.131	331.3	0.239
8	345.4	37114.050	356.1	0.149	347.0	0.240

表 4　障害物がある場合
Table 4　Testing cases with obstacles.

n	Conventional method				Proposed method			
	L	ϕ	Cost	Time	L	ϕ	Cost	Time
5	319.7	266.5	3463.5	0.106	312.5	218.6	3343.6	0.230
6	398.9	273.4	4262.4	0.125	393.1	250.7	4181.7	0.250
7	445.2	315.7	4767.7	0.193	437.9	292.1	4671.1	0.276
8	475.4	360.8	5114.8	0.209	461.4	311.5	4925.5	0.298
9	560.3	382.4	5985.4	0.260	540.1	349.4	5750.4	0.359
10	586.3	369.7	6232.7	0.300	576.7	337.7	6104.7	0.394
15	784.5	481.9	8326.9	0.544	766.0	452.8	8112.8	0.682
50	1794.3	742.2	18685.2	10.438	1738.5	692.9	18077.9	10.844
100	2748.4	948.8	28432.8	77.236	2668.0	900.9	27580.9	77.664
150	3521.0	949.7	36159.7	255.574	3424.1	880.8	35121.8	257.517

れ曲がり座標には自由度があり，自明には求まらない．最適解ではすべての座標候補を試しており，節点数 8 では 37,114 秒，すなわち 10 時間以上かかっている．提案手法では，手続き Find_D によってこの座標を求め，最適に近い解を得ている．一方，従来手法では自明で求まる最小の折れ曲がり座標の組合せのみで格子スタイナー木を構成したため，最適解からのずれが大きくなったと考えられる．

次に同じデータに対し，乱数を用いて障害物を 150 個発生させた場合について実験を行った．これ以降，係数は再び，配線長の Z 軸方向は 2.0，目的関数は $c1 = 10.0$，$c2 = 1.0$ とした．従来手法に比べてコストは平均 2.9%小さく，障害物を避けた 3 次元格子スタイナー木を求めることができた（**表 4** 参照）．これは提案手法において障害物がある場合においても，手続き Find_D と障害物の座標による移動で決めた 1 つ余分な折れ曲がり座標を使い，線分の共有部分を長くすることができたためと考えられる．**図 9** に与えられた節点数が 100 のときの障害物と得られた格子スタイナー木を示す．ここで細い斜めの線は与えられたユークリッド全域木を，太い線は得られた 3 次元格子スタイナー木を表す．

さらに障害物を増やした場合について実験を行った（**表 5** 参照）．障害物を増やしていくと，避けきれずに解が得られない場合が増えてくる．与えられるユークリッド全域木の節点数を 150 とし，障害物 R を 50 から 2,100 まで増やしたときに，それぞれ 10 個のデータに

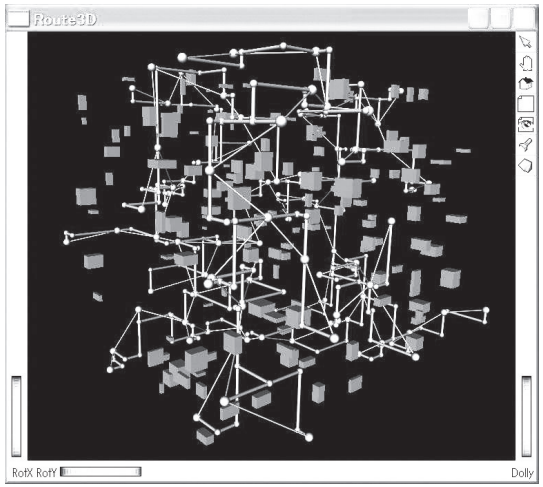


図 9　障害物と格子スタイナー木
Fig. 9　Obstacles and rectilinear Steiner tree.

表 5　さらに障害物がある場合
Table 5　Testing cases with more obstacles.

Conventional (n=150)		Proposed (n=150)	
R	Rate	R	Rate
50	100%	150	100%
100	100%	300	100%
150	100%	500	90%
200	70%	800	90%
250	50%	1000	80%
300	30%	1200	70%
350	10%	1500	50%
400	10%	1800	20%
450	10%	2000	10%
500	0%	2100	0%

表 6　非並列との比較
Table 6　Comparison with non-parallel version.

n	Proposed, Non-parallel				Proposed, Parallel			
	L	ϕ	Cost	Time	L	ϕ	Cost	Time
5	312.5	219.3	3344.3	0.253	312.5	218.6	3343.6	0.230
6	393.3	249.8	4182.8	0.316	393.1	250.7	4181.7	0.250
7	438.1	292.6	4673.6	0.420	437.9	292.1	4671.1	0.276
8	462.5	313.2	4938.2	0.506	461.4	311.5	4925.5	0.298
9	540.8	343.1	5751.1	0.646	540.1	349.4	5750.4	0.359
10	576.1	339.7	6100.7	0.759	576.7	337.7	6104.7	0.394
15	766.4	448.3	8112.3	1.846	766.0	452.8	8112.8	0.682
50	1745.3	692.4	18145.4	42.190	1738.5	692.9	18077.9	10.844
100	2667.3	910.5	27583.5	310.029	2668.0	900.9	27580.9	77.664
150	3428.1	886.6	35167.6	1019.445	3424.1	880.8	35121.8	257.517

対し何個の解を得ることができたかを回避率 *Rate* として，実験を行った．従来手法では，100%の回避率が得られたのは障害物の数が 150 までで，500 以上では解がまったく得られなかった．それに対し提案手法では，障害物の数が 300 まで 100%の回避率が得られ，さらに 2,000 までは 1 つ以上解が得られた．これは提案手法において，手続き Find_D と障害物の座標による移動で決めた 1 つ余分な折れ曲がり座標を使い，十分に障害物を回避することができたためと考えられる．

最後に，提案手法における並列化の効果について実験を行った．なお，文献 11) では並列化に関しては議論されていない．個体数を 20×4，世代数を 50 とし，10 世代ごとに移住を行った並列の場合と，個体数を 80，世代数を 50 で非並列の場合を比較した（**表 6** 参照）．非並列の場合に比べて，並列の場合，10 個のデータのうち 8 個のデータについて解がわずかに良くなったが，平均するとわずか 0.08%程度の違いであった．計算時間については，n が 10 以下の小さいデータについては，すべて高速化率が 2.0 未満，並列化効率が 0.5 未満であったが，n が 15 以上のデータについてのみ平均をとると，高速化率が 3.6，並列化効率が 0.9 となり，非常に効率良く計算ができていることが確認できた．

以上の実験結果により，提案手法が有効であることを検証することができた．

6. まとめと今後の課題

本論文では，空間上に 3 次元座標を持つ点 S の集合，それらを結ぶユークリッド最小全域木，障害物が与えられたとき，木の枝を X 軸，Y 軸，および Z 軸に平行な線分に置き換え，最小 +1 の折れ曲がりを使うことにより，共有部分が長くなりそうな分岐点を求めてそこで折れ曲がることにより配線長を短くしようと試み，障害物がある場合はより柔軟に障害物を避ける，3 次元最小格子スタイナー木を求める並列遺伝的アルゴリズムを提案した．

実験の結果，文献 11) を単純に 3 次元化した従来手法に比べて，障害物がない場合，最適解と比較した場合，障害物がある場合，障害物が非常に多い場合のいずれでも，提案手法が有効であることを検証した．また，並列化の効果も示した．今後の課題としては，アルゴリズムの高速化等がある．

参 考 文 献

1) Prömel, H.J. and Steger, A.: *The Steiner Tree Problem: A Tour through Graphs, Algorithms and Complexity*, Vieweg (2002).
2) Garey, M.R. and Johnson, D.S.: *COMPUTERS AND INTRACTABILITY: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company (1979).
3) Hu, Y., Jing, T., Hong, X., Feng, Z., Hu, X. and Yan, G.: An-OARSMan: Obstacle-Avoiding Routing Tree Construction with Good Length Performance, *Proc. ASP-DAC'05*, IEEE, pp.7–12 (2005).
4) Shi, Y., Jing, T., He, L., Feng, Z. and Hong, X.: CDCTree: Novel Obstacle-Avoiding Routing Tree Construction based on Current Driven Circuit Model, *Proc. ASP-DAC'06*, IEEE, pp.630–635 (2006).
5) Hentschke, R.F., Narasimham, J., Johann, M.O. and Reis, R.L.: Maze Routing Steiner Trees with Effective Critical Sink Optimization, *Proc. ISPD'07*, ACM, pp.135–142 (2007).
6) Li, L. and Young, E.F.Y.: Obstacle-avoiding Rectilinear Steiner Tree Construction, *Proc. ICCAD'08*, IEEE, pp.523–538 (2008).
7) Liu, C.-H., Yuan, S.-Y., Kuo, S.-Y. and Wang, S.-C.: High-performance obstacle-avoiding rectilinear Steiner tree construction, *Trans. Design Automation of Electronic Systems*, Vol.14, No.3, pp.45:1–45:29 (2009).
8) Li, L., Qian, Z. and Young, E.F.Y.: Generation of Optimal Obstacle-Avoiding Rectilinear Steiner Minimum Tree, *Proc. ICCAD'09*, pp.21–25 (2009).
9) Liu, C.-H., Chou, Y.-H., Yuan, S.-Y. and Kuo, S.-Y.: Efficient Multilayer Routing Based on Obstacle-Avioding Preferred Direction Steiner Tree, *Proc. ISPD'08*,

pp.118–125 (2008).
10) Wakabayashi, S.: A Genetic Algorithm for Rectilinear Steiner Tree Problem in VLSI Interconnect Layout, *IPSJ Journal*, Vol.43, No.5, pp.1315–1322 (2002).
11) Hare, R.M. and Julstrom, B.A.: A Spanning-Tree-Based Genetic Algorithm for Some Instances of the Rectilinear Steiner Problem with Obstacles, *Proc. ACM SAC'03*, pp.725–729 (2003).
12) Sherwani, N., Bhingarde, S. and Panyam, A.: *Routing in the Third Dimension*, IEEE Press (1995).
13) Xie, Y., Cong, J. and Sapatnekar, S. (Eds.): *Three-Dimensional Integrated Circuit Design – EDA, Design and Microarchitectures*, Springer (2010).
14) Pavlidis, V.F. and Friedman, E.G.: *THREE-DIMENSIONAL INTEGRATED CIRCUIT DESIGN*, Morgan Kaufman (2009).
15) Tong, C.C. and Wu, C.-L.: Routing in a Three-Dimensional Chip, *IEEE Trans. Comput.*, Vol.44, No.1, pp.106–117 (1995).
16) Sarrafzandeh, M. and Wong, C.K.: *AN INTRODUCTION TO VLSI PHYSICAL DESIGN*, McGraw Hill (1994).
17) Whitely, D., Rana, S. and Heckendorn, R.B.: The Island Model Genetic Algorithm: On Separability, Population Size and Convergence, *Journal of Computing and Information Technology*, Vol.7, pp.33–47 (1998).
18) 築山修治：アルゴリズムとデータ構造の設計法，コロナ社 (2003).
19) Snyder, T.L.: On the Exact Location of Steiner Points in General Dimension, *SIAM J. Comput.*, Vol.21, No.1, pp.163–180 (1992).

(平成 22 年 6 月 14 日受付)
(平成 22 年 11 月 5 日採録)

大村　道郎（正会員）

1985 年広島大学工学部第二類（電気系）卒業．1991 年同大学大学院工学研究科博士課程後期単位修得後退学．同年広島工業大学工学部助手，講師を経て，2000 年同大学助教授（准教授）．工学博士．VLSI 設計自動化，VLSI 設計教育に関する研究に従事．電子情報通信学会，IEEE，ACM 各会員．

瀬能　浩史（正会員）

2008 年広島工業大学工学部電気・ディジタルシステム工学科卒業．2010 年同大学大学院工学研究科博士前期課程修了．現在，新川電機株式会社勤務．在学中は，VLSI 設計自動化に関する研究に従事．

上田　真琴

2010 年広島工業大学工学部電気・ディジタルシステム工学科卒業．現在，同大学大学院工学研究科博士前期課程在学中．VLSI 設計自動化に関する研究に従事．