Vol.2010-SLDM-143 No.12
2010/1/26

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の改良

瀬能浩史[†]　上田真琴[†]　大村道郎[†]

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ひとつである．また，ナノＣＭＯＳ時代のこれらの配線に関しては，折れ曲がりが増えると，タイミングの見積もりなどに悪影響を及ぼすと考えられる．著者らは，空間上に３次元座標を持つ点の集合，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，及び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，より柔軟に障害物をよける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ているが，本稿ではこれを改良し，障害物がない場合でも，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る．評価のために行った実験結果についても述べる．

Improvement of the Parallel Genetic Algorithm for 3-D Rectilinear Steiner Tree

Hirofumi Senou[†], Makoto Ueda[†]
and Michiroh Ohmura[†]

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, if the number of wire bends increases, it will prevent efficient estimation of wiring distance and signal timing. We have already proposed a parallel genetic algorithm which can avoid obstacles flexibly and obtain the 3-D minimum rectilinear Steiner tree by using the minimum+1 bends. In this paper, we improve this algorithm in order to obtain the rectilinear Steiner tree with shorter wire length by calculating the branch point to share edges as much as possible and bending there, even if there is no obstacle. The experimental results are also shown.

1. はじめに

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ひとつである．この問題は一般的に，現実的な時間で解決する事が困難な「ＮＰ困難」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ときも，多項式時間の最適アルゴリズムは存在しそうにない[1],[2]．一方，近年の大規模集積回路における製造技術の進歩に伴い，ＭＣＭ[3]や，回路素子自体を３次元に集積化し，複数の能動層を持つ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[4]．これらのＬＳＩでは，３次元の配線が重要になってくる．

また，ナノＣＭＯＳ時代のこれらの配線に関しては，折れ曲がりが増えると，タイミングの見積もりなどに悪影響を及ぼす[5]と考えられるため，折れ曲がりを制限した格子スタイナー木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[6]では障害物を柔軟に避けることができない．

著者らは，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，及び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，より柔軟に障害物をよける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ている[7]が，本稿ではこれを改良し，障害物がない場合でも，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る．評価基準には，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．本稿では，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

始めに，準備として，本稿で問題の入力として仮定するユークリッド全域木について述べる．次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 ユークリッド全域木

グラフ G=(V,E,w)が与えられるとする．ここで，V は節点の集合，E は枝の集合，w は各辺 e∈E に重みを割り当てる関数とする．G の全域木のうち，辺の重みの和が最小となるものを最小全域木（minimum spanning tree）と呼ぶ．

３次元空間上に点 v の集合Ｓが与えられるとする．Ｓに含まれる点の総数を｜S｜と表す．このとき，Ｓに属する任意の２点 v_i, v_j 間の距離を，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．このグラフ上での最小全域木を，Ｓに対するユークリッド最小全域木（Euclidean minimum spanning tree）と呼ぶ．

[†] 広島工業大学工学部
Faculty of Engineering, Hiroshima Institute of Technology

本稿では，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した，それら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］ 本稿で仮定するユークリッド全域木の例を図１に示す．

2.2 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．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個の点が端点となり，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，格子スタイナー木（rectilinear Steiner tree）と呼ぶ．また，格子スタイナー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．本稿では，この格子スタイナー木について議論する．

［例２］本稿で議論する格子スタイナー木の例を図２に示す．

3. 問題の定式化

スタイナー木は，ＬＳＩの概略配線などに応用される．３次元の配線において，層を垂直（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，本稿では，直径と配線長に対して，Ｚ方向の配線に重み k (≧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，本稿では，$f = c_1 \cdot L + c_2 \cdot D$ を評価の基準とする．但し，Ｌは総配線長，Ｄは直径，c_1，c_2は定数であり，$c_1 >> c_2$とする．

また，ナノＣＭＯＳ時代のこれらの配線に関しては，折れ曲がりが増えると，タイミングの見積もりなどに悪影響を及ぼすと考えられる[5]ため，折れ曲がりの少ない配線が求められる．ユークリッド全域木の枝が，Ｘ軸，Ｙ軸，またはＺ軸に並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，折れ曲がり無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．

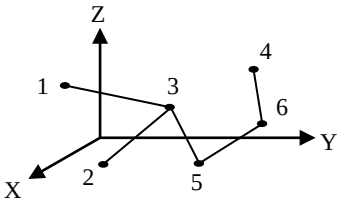


図１ユークリッド全域木

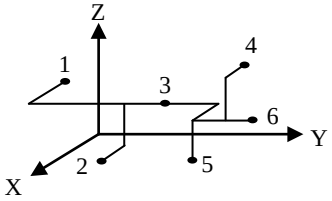


図２　格子スタイナー木

ユークリッド全域木の各枝 {s, t} に対し，点 s, t (∈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３（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をよけることができる．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，①３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）①３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 (∈Ｓ)を結ぶ線分の折れ曲がり数が最小＋１まで．③線分は，s，t を囲む直方体の外には出ない．

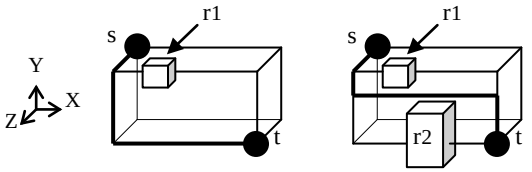
［目的関数］ $f = c_1 \cdot L + c_2 \cdot D$ （但し，L, D はＺ方向の重みを考慮した配線長，直径であり，c_1, c_2は定数で，$c_1 >> c_2$とする．）

4. 手法の概要

4.1 コード化

提案手法では，ｘ線分，ｙ線分，またはｚ線分を分割する．Ｘ軸，Ｙ軸，及びＺ軸に平行な４つの線分の順列として，ｘが２回現れる場合で，xx のように続く場合を除くと，次の６通りが得られる[7]．①xyxz，②xyzx，③xzxy，④xzyx，⑤yxzx，⑥zxyx．

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 xyz，xzy，yxz，yzx，zxy，zyx 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．



（ａ）折れ曲がり数２　　（ｂ）折れ曲がり数３
図３　折れ曲がりと障害物

従って，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，障害物がある場合はその避け方，ない場合は線分を共有するための折れ曲がり方を表す．

4.2 手法の概要

（１）障害物がある場合：

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**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４参照）．s，t 間の各線分は次の様になる．

{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**X**)のとき，および“02”(xyz**Y**)のときの例を，それぞれ図５（ａ），（ｂ）に示す．

（２）障害物がない場合：

著者らの以前の手法[7]では，障害物がある場合のみ，追加の折れ曲がりを使ってそれを避け，節点同士を接続していた．しかし，本手法では，障害物がない場合でも，共有部分が長くなりそうな分岐点を求めて追加の折れ曲がりを用い，配線長を短くしようと試みる．

今，a_i={s,t}，a_j={s,u}間のように，共通の節点 s を持つユークリッド全域木の２つの枝を，x，y，または z 軸に平行な線分で接続する場合を考える．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．共有できない（ａ）の場合を除いて，{s,t}，{s,u}間で枝を共有するとき，分岐点の候補は，両方の直方体の重なった部分の辺上に存在する（図６参照）．

（ａ）２つの直方体が点 s のみで接している場合，線分を全く共有できない．
（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 を含み直方体を共有するよう重なる場合，最大で線分３本を共有することができる．

（ｂ）－（ｄ）のいずれの場合でも，分岐点の座標は，次のように求められる．節点 s，t，u の座標をそれぞれ(x_s,y_s,z_s)，(x_t,y_t,z_t)，(x_u,y_u,z_u)とする．x 座標について非減少順にソートした結果を(x_1,x_2,x_3)とすると，x 線分を分割する分岐点の座標は x_2 となる．y 座標，z 座標についても同様である．

表１　順列と遺伝子

後半 / 前半	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**

（太字は重複している文字）

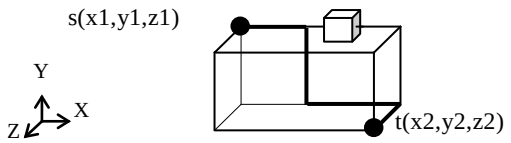
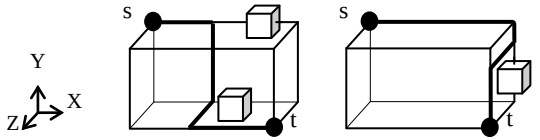
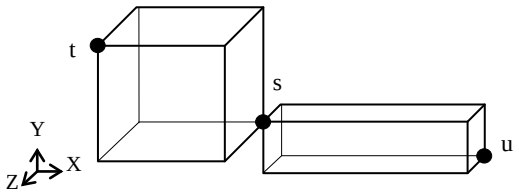


図４　遺伝子“00”（xy**X**z）

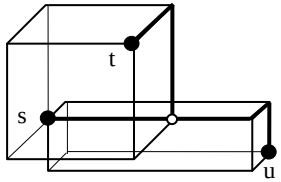


（ａ）遺伝子“01”（xyz**X**）　（ｂ）遺伝子“02”（xyz**Y**）
図５　障害物と遺伝子

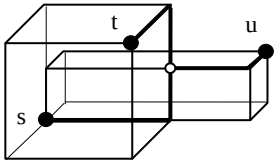
Vol.2010-SLDM-143 No.12
2010/1/26



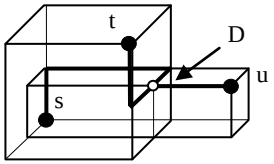
(a) 点で接する場合



(b) 辺で接する場合



(c) 面で接する場合



(d) 重なる場合

図６　直方体の位置関係

[例３] 図７において，{s,t}の遺伝子が“xzyZ”であり z2 の位置で分割され，{s,u}の遺伝子が“xzXy”であり x2 の位置で分割されるとき，折れ曲がる各線分の座標は，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．

{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)

今，２枝間で最も共有する線分が長くなる場合について考える．（ｄ）の場合において更に遺伝子の組み合わせがいい場合，座標 (x2,y2,z2) がそのような分岐点となり，これを分岐点Ｄとする（図６（ｄ）参照）．一般にひとつの枝に対し，複数の枝との間に複数のＤが求まるが，ここでは，共有する線分の長さの順に，Ｄを共有する２つの

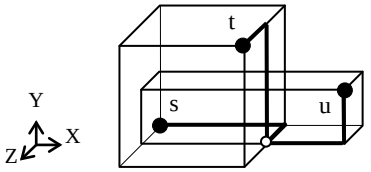


図７　分岐点

枝に排他的に関連付けることにより，各枝に対しひとつＤが決まるようにする（手続き Find_D）．各枝では，このＤの座標(x2,y2,z2)を使って，遺伝子により x 線分を分割する場合には x2 を，y 線分，z 線分を分割する場合には，それぞれ y2,z2 を用いる．Ｄが関連付けられていない場合は，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 を持てば分岐点$D_{i,j}$を求める．

S2: s,Ｄ間の３次元マンハッタン距離の非増加順に$D_{i,j}$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$D_{i,j}$を取り出し，ai, aj が A の要素であれば，分岐点$D_{i,j}$を枝 ai, aj に関連付ける．

S4: 集合 A から ai，aj を，系列Ｅから$D_{i,j}$を取り除く．

S5: 系列Ｅが空であれば終了，そうでなければ S3 へ．

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．

[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: 各島毎に，結果が基準を満たすまで，以下の S3.1-S3.2 を繰り返す．

S3.1: 与えられた世代数だけ 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にＤを関連付ける

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，及びＺ軸に平行な線分の並びに置き換える．このときＤの座標を元に線分の分割座標を決定する．

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これを避ける．もし避けることができなかったときは，配線長を無

Vol.2010-SLDM-143 No.12
2010/1/26

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 n 個選択する．

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: ある確立で突然変異させる．

S3.2: 移住を行う（図８参照）．

S4: 全ての島において，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．

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，単純交叉を用いる．交叉確率は 0.6 と設定した．更に突然変異では，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. 評価実験

提案する遺伝的アルゴリズムおよび従来手法を Intel Core 2 Duo (2.4GHz)をサーバノード，Intel Core i7-940 (2.93GHz)を計算ノードとした 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 10，世代数 50 とし，移住は 10 世代ごとに行った．

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[7]と本手法を比較した．実験は節点数 5 から 200 の，乱数を用いて作成した３次元座標を持つ点と，ユークリッド最小全域木 10 組ずつに対し，配線長 L，直径 D，問題の目的関数に対応するコスト Cost，実行時間 Time (sec)について比較した．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）．

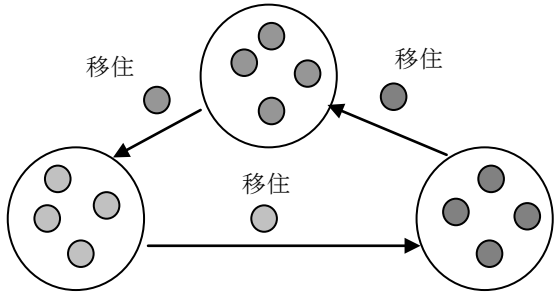


図８　島（サブ母集団）と移住

表２　障害物がない場合

n	従来手法				提案手法			
	L	D	Cost	Time	L	D	Cost	Time
5	318.4	246.7	3450.2	0.08	312.5	239.2	3344.4	0.08
6	398.5	298.6	4258	0.1	393.1	293.7	4180.8	0.12
7	442.8	335.7	4739.8	0.12	439.1	332	4679.7	0.13
8	467.1	360.1	5021.1	0.15	456.5	346.9	4867.6	0.14
9	551.1	414.6	5887	0.21	542.4	407.4	5767.3	0.2
10	575.4	434.4	6116.7	0.217	565.5	424.7	5988.3	0.209
15	772.6	573.9	8201.8	0.553	762.2	563.8	8078.5	0.535
50	1763.6	1338.2	18365	15.4	1734	1311	18018.5	15.4
100	2655.1	2014.7	27461.7	114	2606	1978.2	26940.7	114
150	3470.4	2623.5	35612.9	364	3419	2586.1	35051.6	380
200	4301	3228.8	44016.1	902	4238	3174.7	43329.2	897

表３　最適解

n	最適解		従来手法		提案手法	
	L	Time	L	Time	L	Time
5	310.9	0.244	318.4	0.08	312.5	0.08
6	389.6	17.5	398.5	0.1	393.1	0.12
7	435.9	1083	442.8	0.12	439.1	0.13
8	452.9	44087	467.1	0.15	456.5	0.14

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[8]．そこで枝を置き換える線分の全ての並びに対し，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．

なお，計算機は Intel Core2 Duo 24 台を用い，並列計算により求めた．節点数 5 から

Vol.2010-SLDM-143 No.12
2010/1/26

表４　障害物がある場合

n	従来手法				提案手法			
	L	D	Cost	Time	L	D	Cost	Time
5	319.7	247.3	3463.5	0.11	312.5	239.2	3343.6	0.12
6	398.9	298.7	4262.4	0.13	393.1	293.7	4179.1	0.14
7	445.5	340.1	4770.7	0.22	437.6	330.5	4666.9	0.18
8	475.5	364.8	5117.6	0.19	461.3	350.6	4922.4	0.25
9	559.3	421.5	5974.2	0.3	539.6	404.9	5735.9	0.26
10	581.3	436.8	6180	0.267	567.4	425.7	6003.7	0.283
15	784.5	580.5	8326.9	0.646	766	565.1	8112.8	0.623
50	1787	1355.3	18609.5	16	1739	1321.2	18081.7	16.2
100	2711.6	2059.1	28046.1	108	2627	1998.2	27148.7	115
150	3521	2657	36159.7	382	3424	2587	35121.8	381
200	4345.1	3253.9	44543.9	890	4237	3182.1	43308.7	897

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 2.3%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 0.74%の増加で抑えることができた（表３参照）．

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．従来手法に比べてコストは平均 3.0%程度小さく，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）．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示す．

6. まとめと今後の課題

本稿では，空間上に与えられた３次元のユークリッド全域木および障害物が与えられたとき，最小＋１の折れ曲がりを使い障害物を柔軟に避ける３次元最小格子スタイナー木を求めるアルゴリズムを改良し，さらに配線長の短い解を得る並列遺伝的アルゴリズムについて述べた．実験の結果，本手法の有効性が確認できた．今後の課題としては，アルゴリズムの高速化などがある．

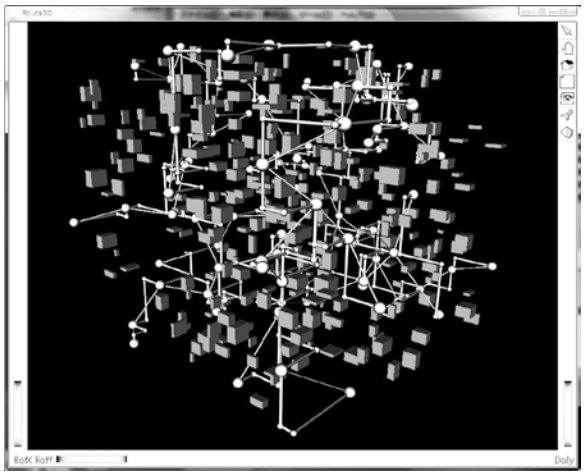


図９　障害物と格子スタイナー

参考文献

1) Y. Shi, T. Jing, L. He, Z. Feng, and X. Hong: “CDCTree: novel obstacle-avoiding routing tree construction based on current driven circuit model,” Proc. of ASP-DAC, pp. 630-635, 2006.
2) Y. Hu, T. Jing, X. Hong, Z. Feng, X. Hu, and G. Yan: “An-OARSMan: obstacle-avoiding routing tree construction with good length performance,” Proc. of ASP-DAC, pp. 7-12, 2005.
3) N.Sherwani, S. Bhingarde, and A. Panyam: “Routing in the Third Dimension,” IEEE Press, 1995.
4) C.C. Tong and C-L. Wu: “Routing in a three– dimensional chip,” IEEE Trans. on Comp., Vol.44.1, p8.106-117, 1995.
5) C. Kodama and K. Fujiyoshi, “Minimizing the number of empty rooms on floorplan by dissection line merge,” IEICE Trans. Inf. & Syst., vol. E88-D, no.7, pp.1389-1396, July 2005.
6) R. M. Hare and B. A. Julstrom, “A spanning-tree-based genetic algorithm for some instances of the rectilinear Steiner problem with obstacles,” Proc. of the 2003 ACM symposium on Applied 3 computing, pp.725-729, 2003.
7) 戸津川祐志，瀬能浩史，大村道郎，“障害物を考慮した３次元格子スタイナー木を求める遺伝的アルゴリズム”，情報処理学会　DAシンポジウム2007論文集，pp127 (2007).
8) T. L. Snyder, “On the exact location of Steiner points in general dimension,” SIAM J. Comput, Vol. 21, No. 1, pp. 163-180, 1992.