2005－DBS－137（Ｉ）（11）
2005／7／13

放射状の木の自動生成アルゴリズム

草苅　良至†　　新里　善美††　　能登谷淳一†　　笠井　雅夫†

† 秋田県立大学システム科学技術学部電子情報システム学科
由利本荘市土谷字海老の口 84-4
†† 秋田県立大学大学院システム科学技術研究科電子情報システム学専攻
由利本荘市土谷字海老の口 84-4
E-mail: †{kusakari,m06b007,notoya,kasai}@akita-pu.ac.jp

あらまし　近年, 難しい問題に対して, 問題例 (インスタンス) の自動生成アルゴリズムの研究が行なわれている．一方, Alt らは木状折れ線図形の直線化可能性の判定問題が PSPACE 困難であることを示している [2]．ここで, 木状折れ線図形の直線化可能性問題とは,「平面上の木に対して, 各辺の長さを変化させず辺同士を交差させずに, 2 次元平面上の連続的な変形で直線状にできるのか?」を判定する問題である．この問題に対して, 限定されたクラス “放射状の木” が考えられている [9]．そこで本稿では，放射状の木を自動生成するアルゴリズムを与える．提案アルゴリズムは平面走査法に基づいており, n 個のバーを持つ放射状の木を，$O(n)$ の領域量を用いて $O(n \log n)$ 時間で生成する．

キーワード　インスタンス生成, PSPASE 困難, NP 困難, 折れ線図形, 直線化, 単調な木, 放射状の木

An Algorithm for Generating Radial Trees

Yoshiyuki KUSAKARI†, Yoshimi NIISATO††, Junichi NOTOYA†, and Masao KASAI†

† Department of Electronics and Information Systems, Faculty of Systems Science and Thechnology, Akita Prefectural University
†† Department of Electronics and Information Systems, Graduate School of Systems Science and Thechnology, Akita Prefectural University
E-mail: †{kusakari,m06b007,notoya,kasai}@akita-pu.ac.jp

Abstract　Recently, some algorithms have been developed for generating instances of difficult problems, such as NP-hard or PSPACE-hard. Alt *et.al.* have showed that the decision problem for flattening the tree linkage, that is, whether the tree linkage could be flattened or not without crossing any two bars [2]. For such problem, the “radial tree” is considered [9]. In this paper, we give an algorithm for generating a radial tree T, in time $O(n \log n)$ using space $O(n)$ if T has n joints. Our algorithm is a plane sweep type algorithm using a circle instead of a line.

Key words　instance generation, PSPACE-hard, NP-hard, linkage, flattening, monotone tree, radial tree

1. はじめに

近年, 難しい問題に対して, 問題例 (インスタンス) の自動生成アルゴリズムの研究が行なわれている．広く楽しまれているパズルやゲームの計算量は, NP 困難,PSPACE 困難のものが多い．これらの難しい計算量を持つ問題で

は, 問題例生成アルゴリズムそのものに意義があると考えられる. 自動生成アルゴリズムによって, 多種多量の問題例を作ることが可能となるからである. 例えば, 「倉庫番」問題は PSPACE-完全問題である [5] が, 村瀬らによって「倉庫番」問題生成法が提案されている [12]. また, 実用上の組合せ問題の多くは NP 完全問題であるが, インスタンス自動生成法を通して, 問題の構造的複雑さを解明しようという試みも行なわれている. 例えば, ある種のスケジューリング問題はグラフの彩色問題として定式化できる [1] が, グラフの彩色問題は NP 完全である. 水野らは, グラフの 3 彩色問題に対して, 判定困難なインスタンスの自動生成アルゴリズムを提案している [11]. また, このような自動生成法で得られる多種多量の問題例は, 困難な問題を解くアルゴリズムや各種ヒューリスティクスの動作検証および性能評価のためにも用いることができる.

一方, 折れ線図形の連続変形に関する様々な問題が, 近年計算幾何学の分野でさかんに研究されている [2]〜[4], [6], [8], [9], [14]. 折れ線図形とは, 直線分の集合から成り, 直線分同士が端点で接続されている図形である. 折れ線図形を構成する直線分はバーと呼ばれ, バーの端点はジョイントと呼ばれる. 折れ線図形の平面連続変形とは, 各バーの長さを変化させることなく, バー同士を交差させないように, 2 次元平面上においてバーを連続的に動かす変形である. この折れ線図形の連続変形問題の一つに, 木状折れ線図形の直線化可能性問題がある.「どんな木でも平面連続変形で直線状にできるのか？」という基本的な問題がこれまでに研究されてきた. この基本的な問題に対しては, 否定的な結果が得られている. すなわち, 直線化できない木状折れ線図形がいくつか発見されている [2], [3], [6], [9]. 例えば, 図 1 のような折れ線図形は直線化不可能である [3]. また, Alt らは木の直線化可能性の判定問題が PSPACE 困難であることを示している [2]. PSPASE 困難の問題は NP 完全問題よりはるかに難しいと考えられるので, 一般的な木の直線化可能性を判定する効率の良い (多項式時間の) アルゴリズムの作成は絶望的である.

しかし, 木状折れ線図形の部分クラスを考えることにより, 直線化可能性が効率良く判定できる可能性がある. 例えば, 根から各ジョイントまで結ぶ線分列がすべて単調であるような木状折れ線図形である “単調な木” は, 平面連続変形で直線化可能である [8]. すなわち, 木の単調性は直線化可能であるための十分条件である. 単調な木の例を図 2 に示す. しかしながら, 限定したクラスの中でも, 直線化不可能である木状折れ線図形もいくつか発見されている. 例えば, 次数 3 のジョイントが唯一であるように木を限定したとしても, 直線化不可能な木が存在することが示めされている [6]. なお, ジョイントの次数がすべて 2 以下の場合は, 木状折れ線図形は道状折れ線図形となり, 必ず直線化することができる. このことは, 「カーペンターズルール」と呼ばれていた未解決問題であったが, 2000 年に肯定的に証明されている [4], [14]. 単調な木の定義を自然に変更することにより, “放射状の木” というクラスが得られる. しかし, 放射状の木というクラスにおいては, 直線化不可能な木が発見されている [9]. ここで, 放射状の木とは放射単調性を満す木状折れ線図形のことである. (放射単調性の厳密な定義は 2 節で与える.) 単調性と放射単調性の定義は非常に類似しているにも関わらず, 片方は直線化可能であるための十分条件であり, もう一方は十分条件ではない. このように, 木状折れ線図形の直線化問題を解決するために, 木の単調性と放射単調性が本質的役割を果たす可能性がある.

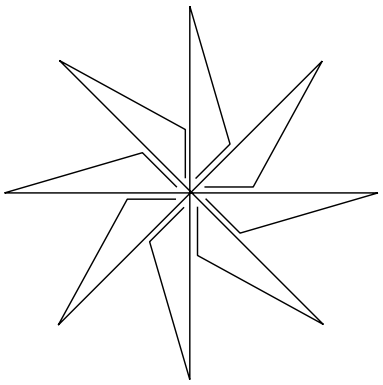

図 1　直線化不可能な木の例 [3]
Fig. 1　An example of locked tree [3].

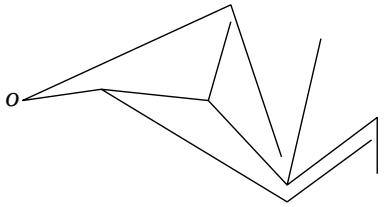

図 2　単調な木の例 [8]
Fig. 2　An example of monotone tree [8].

本研究では, 単調性および放射単調性が直線化可能性問題にどのように関係しているのかを解明することを目指す. 一般の木の直線化可能性判定問題は PSPACE 困難の問題であるが, クラスを限定したときの計算量については以前不明である. 限定クラスに対する木の直線化の判定

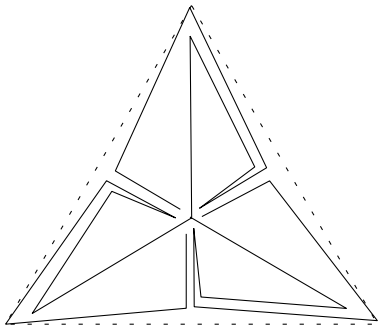

図 3　次数 3 のジョイントが 1 点の木 [6]
Fig. 3　A locked tree with only one degree-3 joint [6].

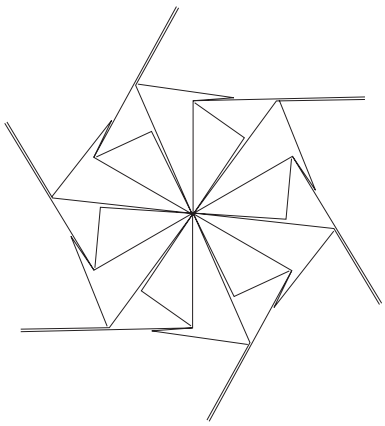

図 4　直線化不可能な放射状の木 [9]
Fig. 4　A locked radial tree [9].

問題の解決や, 直線化アルゴリズムの開発, 検証, 評価のためには, 限定クラスに属する多種多量のインスタンスが必要である. そこで本稿では, 放射単調性を持つ限定クラスのインスタンスを自動生成するアルゴリズムを与える. すなわち, “放射状の木” の自動生成アルゴリズムを与える. なお, 単調な木を自動生成するアルゴリズムは, 既に草苅らにより提案されている [10].

本稿は，次のような構成である．2 節では，用語と問題の定義を与える．3 節では，放射状の木の自動生成アルゴリズムを示し，アルゴリズムの解析を行なう．4 節では，3 節のアルゴリズムの実装について述べる．5 節はむすびである．

2. 準　　備

本節では，用語と問題の定義を与える．

折れ線図形$L=(J,B)$ とは，直線分の集合 B とその端点の集合 J から成る図形である．J の各点を**ジョイント**と呼び，B の各線分を**バー**と呼ぶ．また，バーはジョイントに**接続**しているものとする．このような接続関係だけに注目し，座標等の幾何情報を取り除いたグラフを，折れ線図形 L の**構造グラフ**といい，$G(L)=(V(J),E(B))$ と表す．折れ線図形は，構造グラフの平面への埋め込みとみなすこともできる．任意の 2 つのバー $b,b'\in B$ が交差しないとき，折れ線図形 $L=(J,B)$ は**単純**であるという．本稿では特に断わらない限り，折れ線図形は単純であるとする．**木状折れ線図形**$T=(J,B)$ とは，その構造グラフ $G(T)=(V(J),E(B))$ が根付き木であるような折れ線図形である．これ以降，木状折れ線図形を単に**木**と呼ぶ．本稿では議論を単純化するために，木 T の**根**は $o\in\mathbb{R}^2$ に埋め込まれているものとし, 単に o と表わす. 任意のジョイント $j\in J$ に対して，j を根とする部分木を $T(j)$ と書く．T 上の点 p から点 q までを結ぶ T 上の道を $T[p,q]$) と書く．任意のジョイント $j\in J$ に対して，j の子の集合を $N(j)\subset J$ と書く．子の数 $|N(j)|$ をジョイント j の**次数**といい $d(j)$ と書く．次数が 0 であるようなジョイントを**葉**といい，根と葉以外のジョイントを**内点**という．ジョイント j とその子ジョイント $j'\in N(j)$ に接続するバーを $b=(j,j')$ と書く．任意のジョイント $j\in J-\{o\}$ に対して, バー $b=(j',j)$ を j の**親バー**という．葉以外のジョイント $j\in J$ に対して, 子ジョイントから接続しているバー $b=(j,j')$ を j の**子バー**という．バー b の長さを $|b|$ と書く．

ここで, まず単調な木の定義を与える．構造グラフ $G(P)$ が (有向) 道であるような折れ線図形 $P=(J,B)$ を**道**と呼ぶ．$o\in\mathbb{R}^2$ を始点とする道 P に対して，x 軸に垂直な任意の直線と P との交点が 1 点か 1 つの線分であり, o の x 座標が最小であるとき, P を **(x 方向に) 単調な道**という．木 $T=(J,B)$ に対して，根 o から任意の点 $p\in T-\{o\}$ までの T 上の部分道 $T[o,j]$ が単調であるとき，T を **(x 方向に) 単調な木**という [8]．次に, 単調な木の定義を自然に変更することによって, 放射状の木の定義を与える．根 o を始点とする道 P に対して，o を中心とする任意の同心円 C と P との交点が高々1 つであるとき，P は **(o に関する) 放射状の道**という．道 P が放射状であるとき, P は**放射 (単調) 性を満たす**ともいう．木 $T=(J,B)$ に対して，根 o から任意の点 $p\in T-\{o\}$ までの T 上の部分道 $T[o,p]$ が放射状であるとき，T を**放射状の木**という [9].

点 $p\in\mathbb{R}^2$ に対して，直交座標系の x 成分，y 成分をそれぞれ $x(p)$，$y(p)$ と書き, 極座標系の r 成分 (長さ成分)

を $r(p)$, θ 成分 (方向成分) を $\theta(p)$ と書く．2 つの線分 s, s' に対して，s と s' の交点を $c(s,s')$ と書く．点 $p \in \mathbb{R}^2$ から $d(\in (-\pi,\pi])$ 方向に伸びる半直線を p を始点とする**光線**といい $R_d(p)$ と書く．ジョイント $j \in J$ に対して, j からの光線 $R_d(j)$ を $d = d_s$ から $d = d_t$ まで反時計廻りに順に移動させたときに通過する領域を, **(j の) 扇**といい, $w_j[d_s, d_t]$ と書く．o を中心とした半径 r の円を $C(r)$ と書く．また, 円 C の半径を $r(C)$ と書く．$C(r)$ の外部の領域を $\breve{C}(r)$ と書く．すなわち,

$$\breve{C}(r) = \{p \in \mathbb{R}^2 : |op| \geqq r\}$$

である.

本稿では，乱数を用いて，多種多様な放射状の木を生成するアルゴリズムを与える．放射状な木を特徴づけるパラメータとしては，ジョイント数，次数分布，バー長分布等様々なものが考えられる．本稿では，主にジョイント数に注目するものとする．すなわち, 放射状の木の生成問題とは，自然数 $n \in \mathbb{N}$ が与えられたとき，n 個のジョイントを持つ単調な木 T を生成する問題である．なお，放射状の木の構造グラフは木であるので，バーの数は常に $n-1$ である [7].

3. 放射状の木を生成アルゴリズム

本節では，放射状の木を生成するアルゴリズムを示す.

3.1 放射状の木の性質

放射状の木に対して, 次の補題が成り立つ.

[補題 1] 木 $T = (J, B)$ が放射性を満すための必要十分条件は，次の (i)，(ii) を満たすことである.

(i)(放射性) 各ジョイント $j \in J$ と部分木 $T(j)$ 内の点 $p \in T(j)$ に対して,

$$p \in \breve{C}(|oj|).$$

(ii)(単純性) 任意の 2 バー $b, b' \in B$ が交差しない.

証明 放射状の木の定義より明らか. □

ここで, 条件 (i) は, 次の条件とも等価である.

(i)′(放射性) 任意のジョイント $j \in J$ とその子ジョイント $j' \in N(j)$ に対して,

$$j' \in w_j[\theta(j) - \frac{\pi}{2}, \theta(j) + \frac{\pi}{2}].$$

証明 j と部分木上の点 $p \in T(j)$ を結ぶ道 $T[j,p]$ を $j = j_1 j_2 \cdots j'' p$ とする.

(十分性) $j_2 \in w_{j_1}[\theta(j) - \frac{\pi}{2}, \theta(j) + \frac{\pi}{2}] \subset \breve{C}(|oj_1|)$ であり, $|oj_1| \leqq |oj_2|$ である．同様にして, $|oj_1| \leqq |oj_2| \leqq \cdots \leqq |o_l|$ であり, $\breve{C}(|oj_1|) \supset \breve{C}(|oj_2|) \supset \cdots \supset \breve{C}(|oj_l|)$ である．よって,$j'' \in \breve{C}(|oj|)$ である.

(必要性)

$$j'' \notin w_j[\theta(j) - \frac{\pi}{2}, \theta(j) + \frac{\pi}{2}]$$

と仮定する．このとき,$\varepsilon > 0$ が存在し, $C(|oj| - \varepsilon)$ が親バー $\bar{j} = (j_0, j_1)$ および道 $P[j_1, j'']$ と交差し, 放射単調性に矛盾する. □

この補題を基にして，放射状の木を生成するアルゴリズムを与える．アルゴリズム設計のもう一つのアイディアは，平面走査法を用いて根から順にバーを生成するということである．平面走査法は幾何アルゴリズムの設計技法の一つであり，走査線と呼ばれる仮想的な線を用いて平面上の幾何データを走査することで問題を解く．平面走査法を用いれば様々な幾何問題を効率よく解くことができることが知られている [13]．平面走査法では，走査線の (進行方向側の) 先にある幾何情報は陽に必要ではない．この性質を利用することにより，走査線に交差するような生成過程のバーだけを管理すれば放射状の木を生成できる．すなわち, バーの生成, 走査, 交差除去を相互に繰り返すことにより, 適切に木を生成できる．また, 従来の平面走査法では, 走査線として直線を用いるものが多いが, 本稿で与えるアルゴリズムでは, 走査線として円を用いる．すなわち, 根から同心円状に平面を走査しながら木を生成する．根を中心として同心円状に平面を走査する円を, **走査円**と呼ぶ．また, 走査円 C と交差するバーを**被走査バー**と呼ぶ.

3.2 アルゴリズム

以下に放射状の木の生成アルゴリズムの概略を示す.

本アルゴリズムでは，2 つのデータ構造を用いる．1 つは，走査円 C と交差する被走査バーを蓄えるためのデータ構造 SB である．ただし，被走査バー集合 SB の各要素 $b_i \in SB$ は，走査円との交点 $c(b_i, C)$ によって環順に順序づけられているものとする．もう一つは，走査円 C の移動を制御し，C が停止したときの処理の種類を定めるためのデータ構造で，以下に定めるような 2 種類のイベントを蓄えるイベント計画 ES である.

{ ジョイントイベント $JE(j)$} ジョイントイベント $JE(j)$ は被走査バー $b = (j', j)$ が接続するジョイント j と一対一に対応する．ジョイントイベント $JE(j)$ が実行されると, ジョイント j の子供となるいくつかの被走査バーを生成する．$JE(j)$ は, 走査円 C が $r(j') \leqq r(C) \leqq r(j)$ である間 ES 中に保持されて, 走査円 C が $r(j)$ に到達したときに実行される.

{ 交点イベント $CE(b, b')$} 交点イベント $CE(b, b')$ は, 走査円 C 上で連続する 2 本の被走査バー $b = (j_s, j_t), b_j =$

$(j'_s, j'_t) \in SB$ の交点 $c(b,b')$ と一対一に対応する．$CE(b,b')$ が実行されると，バー b とバー b' のどちらかが切断され，切断されたバーが B に挿入される．交点イベント $CE(b,b')$ は，$max\{r(j_s), r(j'_s)\} \leqq r(C) \leqq r(c(b,b'))$ である間 ES 中に保持されて，走査円 C が $r(b,b')$ に到達したときに実行される．

ジョイントイベントと交点イベントを総称し，**イベント**E と呼ぶ．各イベント E は極座標 r の情報を含んでいることに注意する．イベント E の保持する r 座標を $r(E)$ と書く．

これらの r 座標にしたがって，走査円 C の移動管理を適切行なうことができる．すなわち，$r(E)$ の小さい順に，$r(C)$ を更新していけば良い．後から生成されたイベントであっても，先に処理されることもある．本アルゴリズムは，生成ジョイント数 n，最大次数 Δ をパラメータとする確率的アルゴリズムである．このアルゴリズムによって，次数 $d(j)$ が高々Δ であるような木が生成される．

(Generate Radial Tree)

アルゴリズム GRT(n,Δ)

$B := \emptyset$, $J := \emptyset$, $SB := \emptyset$, $ES := \emptyset$, $r(C) := 0$ とする．

Step1: o に対し，$1 \leqq d(o) \leqq \Delta$ の範囲でランダムに $d(o) \in \mathbb{N}$ を選ぶ．さらに，ランダムに $j_1, j_2, \cdots, j_{d(o)} \in \mathbb{R}^2$ を生成し，$JE(j_1), JE(j_2), \cdots, JE(j_{d(o)})$ を ES に挿入する．

Step2: イベント計画 ES から r 座標が最小のイベント E を取り出し，ES から E を削除する．

Step3: 走査円 C を $r(E)$ まで進める．すなわち，$r(C) := r(E)$ とする．

Step4: イベント E の種類にしたがって (後述する) 処理を行なう．

Step5: 生成されたジョイント数が n に達していなければ，(すなわち $|J| < n$ ならば，) **Step2** へ戻る．

($JE(j)$ の処理) ($j \neq o$ ならば) j に内側から接続しているバー $b = (j', j)$ が SB に蓄えられている．C 上で b と時計廻り，反時計廻りに連続しているバーをそれぞれ b_c,b_{cc} とする．まず，$b = (j', j)$ を SB から削除し，B に $b = (j', j)$ を挿入し J に j を挿入する．次に，$ES = \phi$ なら $1 \leqq d(j) \leqq \Delta$ の範囲で，$ES \neq \phi$ なら $0 \leqq d(j) \leqq \Delta$ の範囲で，次数 $d(j)$ をランダムに選び以下の処理を行なう．

$\{d(j) = 0$ の場合 $\}$ b_c と b_{cc} の交差判定をし，b_c と b_{cc} が交差するときには $CE(b_c, b_{cc})$ を ES に挿入する．

$\{d(j) \geqq 1$ の場合 $\}$ 以下の処理を $d(j)$ 回繰り返す．半平面 $y \geqq 0$ 中からランダムに 1 点 $p = (p_x, p_y), p_y \geqq 0$ を選び，子ジョイント j'' を次式のように設定する．

$$x(j'') = x(j) + |op| \cos(\theta(j) + \theta(p) - \frac{\pi}{2})$$

$$y(j'') = y(j) + |op| \sin(\theta(j) + \theta(p) - \frac{\pi}{2})$$

バー $b'' = (j, j'')$ を生成する．(補題 1 (i)′ および, 図 5 参照)．この時点では B は更新されない．$JE(j'')$ を ES に挿入する．b'' が b_c と交差するときには $CE(b'', b_c)$ を ES に挿入し，b_{cc} と b'' が交差するときには $CE(b_{cc}, b'')$ を ES に挿入する．

バー生成の様子を図 5 に，ジョイントイベント $JE(j)$ の様子を図 6 に示す．図 6 において，$JE(j)$ によって B に挿入されるバー $b = (j', j)$ は太線で，$JE(j)$ により新たに生成されるバー $b'' = (j, j'')$ は点線で，被走査バーは破線で，B 中のバーは実線で，走査円 S は一点破線でそれぞれ描かれている．

($CE(b,b')$ の処理) $b = (j_s, j_t)$,$b' = (j'_s, j'_t)$ とし，$c = c(b,b')$ とする．C において，b と時計廻りに連続しているバーを b_c とし，反時計廻りに連続しているバーを b_{cc} とする．このとき，b' は b_c か b_{cc} のどちらかである．一般性を失うことなく，$b' = b_c$ とする．

b, b' のどちらかを確率的に選び，選ばれたバーには子孫

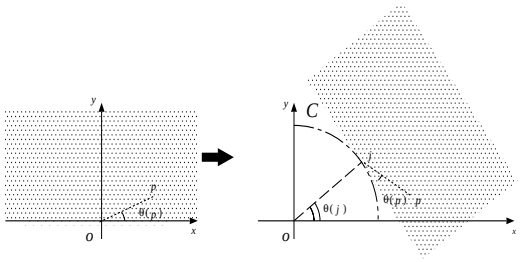

図 5 バーの生成
Fig. 5 Generating a bar.

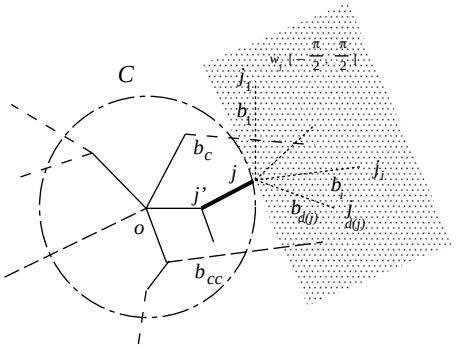

図 6 ジョイントイベント $JE(j)$
Fig. 6 Joint Event$JE(j)$.

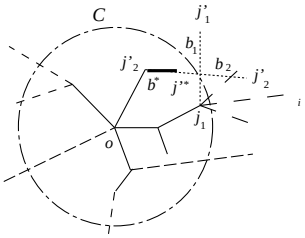

図 7　交点イベント $CE(b,b')$
Fig. 7　Crossing point Event$CE(b,b')$.

を持たせない．子孫を持たないように選択されたバーを**切断バー**と呼ぶ．b を切断バーと仮定する．このとき，SB から b を削除し，ES から $JE(j_t)$ を削除する．さらに，切断バー b と他のバー b'' との交点イベント $CE(b,b'')$ が E に残っていれば，それを削除する．次に，倍率 $s \in (0,1)$ を確率的に選ぶ．$|(j_s,j^*)| = s \times |(j_s,c)|$ となるようにバー $b^* = (j_s,j^*)$ とジョイント j^* 求め，b^* を B に j^* を J に挿入する．この一連の処理を b の**切断**と呼ぶ．あるバーを切断しても，B 内のバー間には必ず交差が生じない (補題 1 (ii) 参照)．残ったバー b' が b_{cc} と交差するときには, それらの交点イベント $CE(b_c,b_{cc})$ を ES に挿入する．

切断バーを b とする交点イベント $CE(b,b')$ の様子を図 7 に示す．図 7 において，バー b,b' は点線で，B に挿入されるバー $b^* = (j_s,j^*)$ は太線で描かれている．

3.3　アルゴリズムの解析

ここでは，**アルゴリズム GRT** (以下では，**GRT** と略す．) を解析する．

GRT(n,Δ) により木 $T = (J,B)$ が生成されるとする．このとき，$|J| = n$ であり，$|B| = n-1$ である．T の内点からなる集合を $J_i \subseteqq J$ とし，葉からなる集合を $J_l \subseteqq J$ とする．各内点 $j_i \in J_i$ はジョイントイベントの実行により生成され，各葉 $j_l \in J_l$ は次の (a)，(b)，(c) のいずれかで生成される．

(a)　ジョイントイベントで次数 $d(j_l) = 0$ が選ばれて生成される．

(b)　交点イベントの切断により生成される．

(c)　アルゴリズム終了時に ES に残っているジョイントイベントが削除され，その結果として親ジョイントが葉となる．

$p \in \{a,b,c\}$ に対して，上の (p) で生成される葉の集合をそれぞれ $J_{l_p} \subseteqq J_l$ と表わす．

このとき次式が成り立つ．

$$\begin{aligned} |J| &= |J_i| + |J_l| \\ &= |J_i| + |J_{l_a}| + |J_{l_b}| + |J_{l_c}| \\ &= n. \end{aligned} \tag{1}$$

GRT(n,Δ) の実行中に生成されるイベントの集合を $\mathcal{E}$ とする．$\mathcal{E}$ には，実行されるもの，切断により実行されずに削除されるもの，アルゴリズム終了時に ES に残っているものがある．実行されるイベントの集合を $\mathcal{E}^{\mathrm{e}}$，切断により削除されるイベントの集合を $\mathcal{E}^{\mathrm{c}}$，アルゴリズム終了時に残っているイベントの集合を $\mathcal{E}^{\mathrm{r}}$ とする．一方，生成されるジョイントイベントの集合を $\mathcal{JE}$ とし，交点イベントの集合を $\mathcal{CE}$ とする．このとき，次式が成り立つ．

$$\begin{aligned} |\mathcal{E}| &= |\mathcal{E}^{\mathrm{e}}| + |\mathcal{E}^{\mathrm{c}}| + |\mathcal{E}^{\mathrm{r}}| \\ &= |\mathcal{JE}| + |\mathcal{CE}|. \end{aligned} \tag{2}$$

イベント数に関して，次の補題 2-補題 4 が成り立つ．(証明は, 本稿では省略する.)

［補題 2］　**GRT**(n,Δ) により生成されるジョイントイベント数 $|\mathcal{JE}|$ は $O(n)$ である．

［補題 3］　**GRT**(n,Δ) により生成される交点イベント数 $|\mathcal{CE}|$ は $O(n)$ である．

［補題 4］　**GRT**(n,Δ) により生成されるイベントの総数 $|\mathcal{E}|$ は，$O(n)$ である．

以上より, 次の定理が成り立つ．

［定理 1］　**GRT**(n,Δ) は，n 個のジョイントを持つ単調な木を，$O(n)$ の記憶量を用いて $O(n\log n)$ 時間で生成する．

証明　補題 1 より，**GRT** の正当性は満たされる．

次に計算量を考察する．

イベント計画 ES には, 極座標の r で順序づけられた優先待ち行列 (ヒープ) を用いる．補題 4 より，**GRT**(n,Δ) で生成されるイベントの総数は $O(n)$ 個である．ES に対する, 挿入や削除等の各処理は $O(\log n)$ 時間で実行できる．また, 被走査バー集合 SB には, 走査円との交点によって, 環状順序づけられた平衡木 (2 色木,AVL 木等) を用いて実現する．被走査バーは明かに $O(n)$ 本であるので, SB に対する挿入や削除等の各処理は $O(\log n)$ 時間で実行できる．

Step1 は ES および CS への処理が $O(d(o))$ 回生じるだけであるので, $O(\log n)$ 時間で行なえる．**Step2**〜**Step4** は，明かに高々イベント数だけ繰り返されるので, $O(n\log n)$ 時間で行なえる．よって, アルゴリズム全体に必要な時間計算量は，$O(n\log n)$ である．

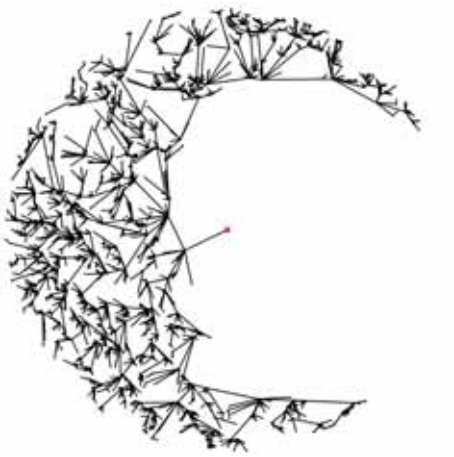

図 8　**GRT**(5000,15) の出力例
Fig. 8　An example of output obtained by **GRT**(5000,15).

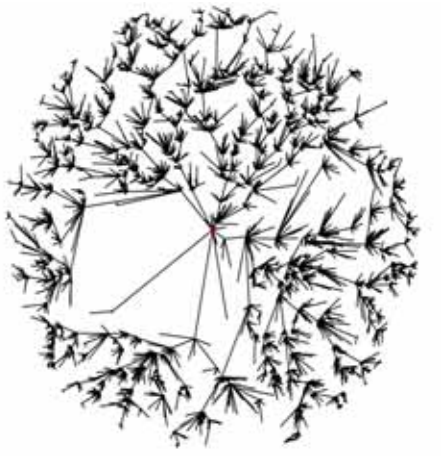

図 9　**GRT**(100000,30) の出力例
Fig. 9　An example of output obtained by **GRT**(10000,30).

また，イベント計画 E と被走査バー集合 S を実現する平衡木には $O(n)$ の記憶量しか必要でなく，アルゴリズムに用いる記憶量は $O(n)$ である．　□

なお，任意の放射状の木は，**GRT** により，生成可能であることも容易に確かめられる．すなわち，ジョイント数 n で最大次数 Δ であるようなどんな放射状の木も，**GRT**(n,Δ) で生成されるような乱数系列が存在する．

4. 実　　装

本節では，前節のアルゴリズムを実装して得られた放射状の木を示す．

Java 言語を用いて提案アルゴリズム **GRT** の実装を行なった．各パラメータを，$n = 5000, \Delta = 15$ のように設定して得られた単調な木の生成例図 8 に示し，$n = 10000$,$\Delta = 30$ のように設定して得られた単調な木の生成例図 9 に示す．

生成時間を計測したところ，図 10 のような結果が得られた．

図 10 において横軸は次数であり，縦軸は生成時間で単位はミリ秒である．　各系列はそれぞれ,$n = 5000, 10000, 20000, 30000$ を表わしている．また，各値は放射状の木を 10 回生成したときの平均時間である．提案アルゴリズムを用いれば，大規模な放射状の木も数分で生成できることが読みとれる．また，同じジョイント数であっても，次数が大きくなる程生成に時間がかかる傾向にある．これは，次数の増加にしたがって，走査円上に蓄えられる被走査バーの数が増えるためであると考えられる．

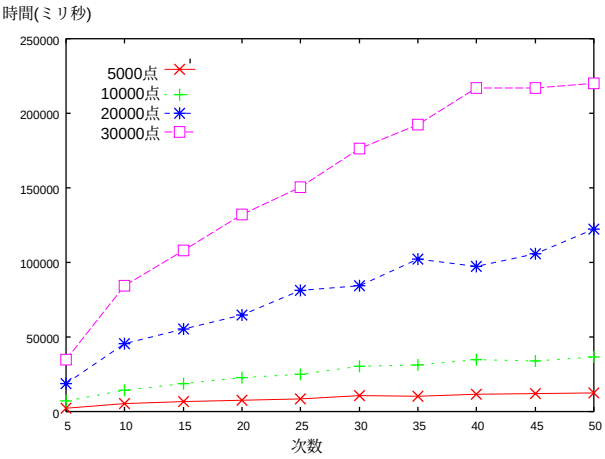

図 10　次数と生成時間
Fig. 10　A relationship of degree and executing time.

5. む　す　び

本稿では，放射状の木の自動生成アルゴリズム **GRT** を提案し，実装を行なった．本インスタンス生成アルゴリズムは，走査円による平面走査に基づいており，n 個のジョイントを持つ放射状の木を $O(n)$ の記憶量を用いて，$O(n \log n)$ 時間で生成する．実装においては，大規模な放射状の木でも数分で生成されることを確認している．

“放射状の木” には直線化不可能な木が存在するが，これらの自動生成で得られた放射状の木ではほとんどのものが直線化可能であるように思われる．本アルゴリズムで生成される放射状の木が直線化可能性にどのように関係しているのかを明かにすることが今後の課題である．

文　献

[1] A.V. Aho, J. E. Hopcroft, J. D. Ullman, “Data Structures and Algorithms,” Addison-Wesley Publishing, 1983.

[2] H. Alt,C. Knauer,G. Rote,and S. Whitesides, “The Complexity of (Un)folding,” Proc. 19th ACM Symp. Computational Geometry, pp.164–170,2003.

[3] T. Biedl,E. Demaine,M. Demaine,S. Lazard, A. Lubiw,J. O’Rourke, S. Robbins,I. Streinu, G. Toussaint,and S. Whitesides, “A note on reconfiguring tree linkages: Trees can lock,” Discrete Applied Mathematics, vol.254,pp. 19–32,2002.

[4] R. Connelly,E. Demaine, and G. Rote, “Straightening polygonal arcs and convexifying polygonal cycles,” Proc. IEEE Symp. Foundations of Computer Science, pp. 432–442,2000.

[5] J. Culberson, “Sokoban is PSPACE-complete,” TechnicalREport TR97-02, Department of Computer Science, The University of Alverta,1997.

[6] R. Connelly, E. Demaine, and G. Rote, “Infinitesimally Locked Self-Touching Linkages with Applications to Locked Trees,” Physical Knots: Knotting, Linking, and Folding Geometric Objects in R^3, American Mathematical Society, pp. 287–311,2002.

[7] R. Diestel,Graph Theory,Springer-Verlag,1997.

[8] Y. Kusakari, M. Sato, and T. Nishizeki, “Planar Reconfiguration of Monotone Trees,” in IEICE Trans. Fund., Vol.E85-A, No.5, pp.938–943,2002.

[9] Y. Kusakari, “On Reconfiguring Radial Trees,” Proc. JCDCG2002, Lecture Notes in Computer Science, vol.2866, Springer-Verlag, pp.182-191,2003.

[10] 草苅良至, 新里善美, 能登谷淳一, 笠井雅夫, “単調な木を自動生成するアルゴリズム,” 2004 年度冬の LA シンポジウム pp.28-1 –28-8, 2005.

[11] 水野一徳, 西原清一, “極小非可解構造に基づく 3COL インスタンスの組織的生成,” 信学論 (D-I),Vol.J87-D-I,No.11,pp.1012-1019,Nov,2004.

[12] 村瀬芳生, 松原仁, 平賀譲, “「倉庫番」の問題の自動生成,” 情報処理学会論文誌,39(3),1998.

[13] F.P. Preparata, M.I. Shamos, Computational Geometry:An Introduction, Springer-Verlag, 1985.

[14] I. Streinu, “A combinatorial approach to planar non-colliding robot arm motion planning,” Proc. IEEE Symp. Foundations of Computer Science, pp. 443-453,2000.