地図情報システムのためのBD木を用いた空間探索

2N−8

横川完治　鈴鹿豊明
日立ソフトウェアエンジニアリング(株)

1 はじめに

近年、地図情報のデジタル化が進み、大量の地図図形データが出現した。そのため、地図情報の様々な処理がコンピュータ化されてきた。中でも、空間探索は基本的な処理であり、その高速化が地図情報システムの重要な課題の1つとなっている。空間探索とは地図上のある範囲の領域を指定し、その領域と交わりのある地図図形データを取り出す処理である。

2 空間探索の方式

空間探索の処理を実現するための最も単純なデータストラクチャは図形データの集合に適当な順序関係を導入してできる1次元配列である。この方式は探索範囲とすべての図形データとの交わりの判定を必要とするため、計算コストが高いという欠点がある。

従来の空間探索の高速化の手段は外接長方形を用いることと、ブロック分割を行なうことである。前者は個々の図形データにそれを囲む最小の長方形を前もって生成しておき、探索範囲とこの長方形が交わる図形データのみ、実際の交わりの判定を行なうものである。後者は空間を座標軸ごとに等間隔に分割することによってできる複数のブロックを導入し、このブロックにかためて管理する方法である。この方法では交わりの判定がブロック中の図形データの数のオーダになる。このため図形データの分布が偏る場合はかならずしも効率的ではない。

ブロック法の欠点は図形データの分布に依存した階層的分割を導入することで改良される。すなわち、対象空間を図形データの分布が密な部分では分割を細かくし、疎な部分では粗くすることにより、含まれる図形データの数がほぼ一定の小空間に領域分割するのである。階層的分割の方式に対応していくつかの木構造が考えられている[1, 2]。すなわち、点データの集合を対象とした領域2等分割木、4分木、k-d木、BD木などが挙げられる。

Spatial Search Using a BD Tree for GIS
Kanji Yokokawa, Toyoaki Suzuka
Hitachi Software Engineering Co. Ltd. 6-81, Onoe-Machi, Naka-ku, Yokohama 231, Japan

3 BD木

BD木は坂内、大沢によって提案された[1]。この木構造は本来点データの集合のついて定義されている。まず、領域2等分割木について説明する。この方法は対象空間のx座標軸とy座標軸について交互に空間を2等分割し、すべての小空間に含まれる点データの数を0または1とするものである。領域2等分割木には片側の部分木にしか点データを含んでいない無駄なノードが生成される。そこで無駄なノードを削除し、代わりに領域式と呼ばれる領域2等分割木の階層性を表現するための情報を各ノードに付加してできる2分木がBD木である。あるノードの領域式は領域2等分割木のルートからそのノードへのパスに対して左ノードをたどる場合を0、右ノードをたどる場合を1に対応させて生成される2進数として定義される。

根ノードrを持つBD木はノードpを投入する関数insertを繰り返し使って構築される。

```
根ノード insert(ノードp, 根ノードr)
{
    ノード t, u, create();
    int i;

    if (r == NULL) {
        r = p;
        return(r);
    } else if (r が葉ノードである) {
        u = r;
        r = create();
        r の領域式の長さ = 0;
        link(r, u);
    }
    t = r;
    while (t != NULL && t の領域式が
        p の領域式を包含する) {
        u = t;
        i = t の領域式の長さ;
        t = (p の領域式の i ビット目が == 0) ?
            t の左子ノード : t の右子ノード;
    }
    if (t == NULL)
        link(u, p);
    else {
```

```
        u = create();
        u の領域式 =
            p の領域式と t の領域式の共通部分;
        link(t の親ノード、 u);
        link(u, p);
        link(u, t);
    }
    return(r);
}
```

なお、関数 create は 1 個のノードを生成するものである。また、関数 link(t, u) はノード u がノード t の子ノードになるようにポインタを結合する。u が t の左子ノードになるか右ノードになるかは u と t の領域式の情報によって決定される。

BD 木の優れた特徴として、点データの分布に偏りのある場合でもバランスのある木構造であること、そして点データの投入順序によらない安定した木構造であることが挙げられる。これらの特徴のため、BD 木は k-d 木や領域 2 等分割木などと比べて、管理木としてより適していると言われている。

4 図形データのための BD 木

次に図形データの集合の空間探索のための BD 木の構造を述べる。ここで図形データとは連結した複数の線分の始終点座標の列である。各図形データについてそれを囲む外接長方形の中心点を求める。この中心点全体の集合について点データの場合と同様に BD 木を生成する。BD 木の各ノードは次の要素から構成されている。

1. 親ノード、左右子ノードへのポインタ
2. 葉ノードであるか否かを示す情報
3. 外接長方形の中心座標
4. 領域式とその長さ
5. 外接長方形の情報がセットされているか否かを示す情報
6. 外接長方形の左下隅と右上隅の座標
7. 図形データの大きさと図形データの先頭へのポインタ（葉ノードのみ）

葉ノードでないノードの外接長方形は、そのノードより下位にあるすべての図形データを含む最小の長方形として定義される。また、図形データの実体は BD 木とは別の場所にある 1 次元配列の中に置かれており、上記の 7 により BD 木の葉ノードと関連付けられている。

5 BD 木を用いた空間探索

BD 木と外接長方形を導入した空間探索の関数 search は次のように極めて単純である。

```
search(ノード p, 探索範囲 a)
{
    if (p の外接長方形が a と交わらない)
        ;
    else if (p が葉ノードである) {
        if (intersect(p, a) == YES)
            p の図形データを出力する;
    } else {
        search(p の左子ノード, a);
        search(p の右子ノード, a);
    }
}
```

但し、上記の関数 intersect(p, a) は葉ノード p の図形データと探索範囲 a との交わりを実際に判定するもので、一般に計算コストが高い。この関数の呼び出しをなるべく避けるために、外接長方形による交わりの判定と木構造の枝刈りが用いられている。

6 おわりに

坂内, 大沢は BD 木の各葉ノードに 1 個の線分を結び付けるアプローチを取っている [1]。図形データ単位の処理は別に 1 次元配列やリスト構造を用意して、連結する線分を関連付けて行なう。

これに対して我々の空間探索の特徴は BD 木の葉ノードに図形データを直接結び付けたことである。その利点は

1. 葉ノードの数が少なくてすみ、メモリの節約や処理の高速化に役立つ。
2. 1 個の図形データが 1 個のオブジェクトに対応している場合が多く、処理の単位として好ましい。

が挙げられる。現在、我々は処理対象として住宅地図を扱っているが、1 個の地図図形データは 1 個の住宅を表現することが多い。したがって、このようなアプリケーションで有効である。

参考文献

[1] 坂内、大沢: 画像データベース, 昭晃堂, 1987.

[2] H. Samet: The Design and Analysis of Spatial Data Structure, Addison-Wesley, 1990.