ISS-SP-164

高速幾何計算のための空間分割アルゴリズムの並列化について

石河　孝太†　山本　修身††

† 名城大学理工学研究科情報工学専攻　　†† 名城大学理工学部情報工学科

1　入力点の組み合わせの削除

本稿では，計算幾何学 [1] で与えられるいくつかの問題に対して空間分割を用いた解法を考える．ここで扱う代表的な問題として，凸包問題，点ボロノイ図などがある（図 1 参照）．本稿で提案するアルゴリズムでは，いくつかの部分問題を構成する空間分割によって並列計算に適した構造を作り，幾何計算の高速化を図る．[2] では，同様の考え方に基づき点ボロノイ図を計算しようとした．

本稿で取り扱う問題は，点や線分などの図形を入力し，意味のある入力点の組み合わせを列挙するものである．例えば 3 次元凸包問題では，3 つの実数値の組み合わせとして 3 次元空間上の点が入力され，凸包を構成する単体の頂点の組み合わせを列挙する．このような問題では，入力点の全ての組み合わせについて目的の条件を満たすか否かを調べれば良いが，入力点の数が多いと多大な計算時間が必要になる．したがって，いちいち調べる必要のない入力点の組み合わせを削除することが，幾何計算において重要となる．

2　空間分割アルゴリズムの並列計算について

空間に配置されている図形を対象にした問題は多くの場合局所性を持ち，空間を部分的に見れば意味のない入力点の組み合わせをある程度知ることができる．そこで，空間をいくつかの部分空間に分割し，各部分空間を独立に調べることを考える．ただし，いくつもの点が限りなく近くに配置される場合，それだけ細かい空間が必要になると考えられるから，入力点に解像度の存在を仮定する．しかしながら，計算機で表現可能な数値の精度には限界がある．例えば，平面上の異なる 2 点には一定の距離がある．

並列計算では，異なる場所について複数のプロセッサが独立にほぼ同一の処理を行う．例えば，要素数が n 個の配列 A が与えられたとき，$A[i]$ $(i = 1, 2, \ldots, n)$ に対して i 番目のプロセッサ P_i が何らかの処理を行う．ここでは，部分空間との関係について不要な入力点を削除することで，調べる入力点の組み合わせを少なくするが，入力点 d_i とその d_i に関係のある部分空間 S_i を i 番目要素として持つ配列 A を用いる（図 2 参照）．そこで空間分割を行う際，一度 A を膨張させ，その後不要な入力点（すなわち不要な A の要素）の削除を行う．階層的に各部分空間に対して空間分割を行えば，より不要な入力点が削除される．また，同じ部分空間を持つ配列要素は連続して配置されるとする．この構造は各部分空間に関係のある入力点の個数を把握することに適している．

3　GPU を用いた計算機実験および今後の課題

ここでは，n 個の入力点に対する平面上の点ボロノイ図をとりあげる．図 3 は，ランダムに発生させた入力点の数に対する

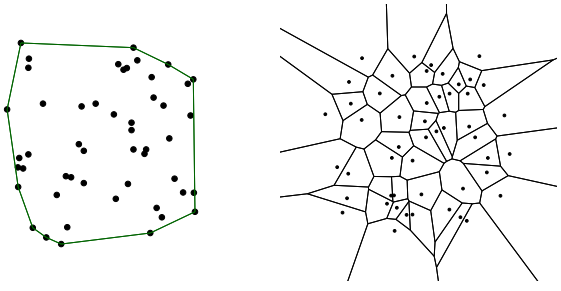

図 1　50 個の点に対する凸包（左）と点ボロノイ図（右）．

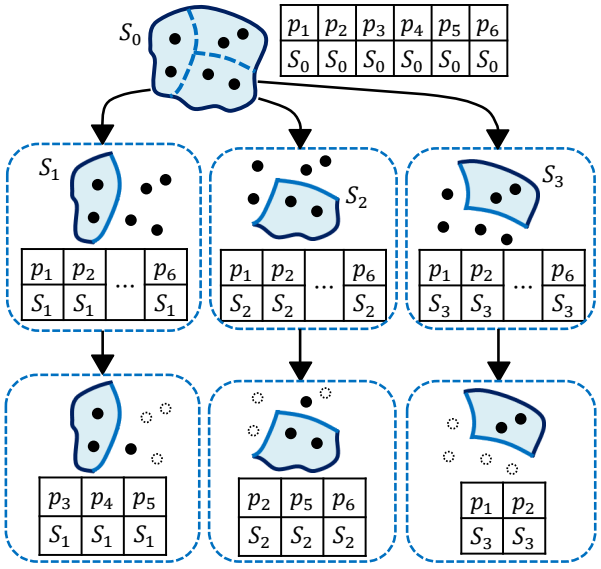

図 2　空間分割アルゴリズムの動作．p_i は入力点を表す．

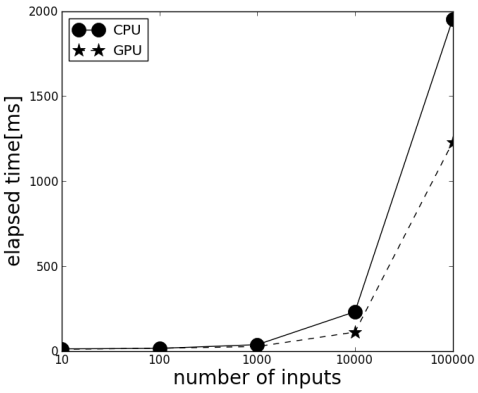

図 3　CPU（Triangle プログラム）または GPU（提案アルゴリズム）を用いて測定した入力点の数に対する点ボロノイ図生成の実行時間．

点ボロノイ図生成の実行時間である．CPU[1] には Triangle[2] プログラムを，GPU である CUDA[3] には本稿での空間分割アルゴリズムを適用した．図 3 より，n が大きいとき，CPU よりも GPU を用いた場合の方が実時間の面で速く動作している．

しかしながら，CUDA には共有メモリなどの高速化するための機能が備わっており，本実験ではそれを用いていない．そこで，より高速に動作するように，CUDA の性能を最大限に活かせるプログラムに改良する必要がある．また，本稿での考え方に基づき，より多くの問題の解法について考える予定である．

参考文献

[1] Mark de Berg, Otfried Cheong, Marc van Kreveld, Mark Overmars: *Computational Geometry:Algorithms and Applications*. Third Edition, Springer (2008)

[2] 河野勇人：GPGPU によるボロノイ図計算アルゴリズムの効率化に関する研究，名城大学大学院理工学研究科情報工学専攻修士論文 (2013)

[1] Intel(R) Xeon(R) CPU E31245 @ 3.30GHz

[2] https://www.cs.cmu.edu/~quake/triangle.html

[3] NVIDIA Tesla K20c