プログラミング 18－14
(1998. 3. 23)

固定サイズの再構成メッシュ上で凸包を求めるアルゴリズム

笹田良治 松前進 都倉信樹

大阪大学大学院基礎工学研究科情報数理系専攻
〒560-8531 大阪府豊中市待兼山町 1-3
電子メール:{sasada, matsumae, tokura}@ics.es.osaka-u.ac.jp

あらまし

再構成バス（reconfigurable bus system）とは，形状を動的に変化させることのできるバスである．格子状に並べたプロセッサを再構成バスで接続したプロセッサアレイを再構成メッシュ（reconfigurable mesh）という．
本稿では，平面上の N 個の点の凸包を，大きさ $M \times M$ の再構成メッシュを用いて，$O(\frac{N}{M} + \frac{N}{M^2} \log \frac{N}{M^2})$ 時間で求めるアルゴリズムを示す（$N \geq M$）．入力が x 座標でソートされているときは，$O(\frac{N}{M})$ 時間で求められる．

キーワード 並列アルゴリズム，再構成メッシュ，凸包

A Convex Hull Algorithm on a Fixed Size Reconfigurable Mesh

Ryoji Sasada, Susumu Matsumae, Nobuki Tokura

Graduate School of Engineering Science, Osaka University
1-3 Machikane-yama, Toyonaka, Osaka 560-8531, Japan
E-mail:{sasada, matsumae, tokura}@ics.es.osaka-u.ac.jp

Abstract

A reconfigurable bus system is a bus system whose configuration can be dynamically changed. A reconfigurable mesh is a processor array that consists of processors arranged to a 2-dimensional grid with a reconfigurable bus system.
In this paper, we present an algorithm for computing the convex hull of N points in the plane in $O(\frac{N}{M} + \frac{N}{M^2} \log \frac{N}{M^2})$ time on a reconfigurable mesh of size $M \times M$ ($N \geq M$). In case the input is sorted by x-coordinate, the convex hull can be computed in $O(\frac{N}{M})$ time.

key words parallel algorithm, reconfigurable mesh, convex hull

1 はじめに

並列アルゴリズムの計算モデルのひとつとして，再構成バス（reconfigurable bus system）を用いて，複数のプロセッサを接続したプロセッサアレイが提案されている．再構成バスとは，形状を動的に変化させることのできるバスである．格子状に並べたプロセッサを再構成バスで接続したプロセッサアレイを再構成メッシュ（reconfigurable mesh，以下 RM）という．また，RM を制限したモデルとして，線形再構成メッシュ（linear reconfigurable mesh，以下 LRM）などが考えられている．LRM 上でのアルゴリズムの例として以下のものが挙げられる．

- N 個の値のソートを $N \times N$ LRM 上で $O(1)$ 時間で行なう [1].
- 平面上の N 点が与えられたとき，その凸包を $N \times N$ LRM 上で $O(1)$ 時間で求める [2].
- $M^{1+\epsilon} \leq N \leq M^2$（$\epsilon$：任意の正定数）の条件のもとで，平面上の N 点が与えられたとき，その凸包を $M \times M$ LRM 上で $O(\frac{N}{M})$ 時間で求める [3].

ただし，上記の文献 [3] のアルゴリズムは，LRM よりも能力の低い計算モデルで提案されたアルゴリズムである．

上で挙げたアルゴリズムをはじめとして，これまで提案されている RM のアルゴリズムの多くは，問題の大きさ N の関数で表されるような，N に応じた大きさの RM が利用できることを仮定している．しかし，このようなアルゴリズムを実装するとき，必要とされる大きさの RM が利用できるとは限らない．そこで，小さい RM を用いて大きい RM の動作をシミュレートするアルゴリズム（セルフシミュレーションアルゴリズム，self-simulation algorithm）が提案されている．ここで，$N > M$ とする．

- $N \times N$ LRM の 1 ステップを $M \times M$ LRM を用いて $\Theta((\frac{N}{M})^2)$ ステップでシミュレートするアルゴリズム [4].
- $N \times N$ RM の 1 ステップを $M \times M$ RM を用いて $\Theta((\frac{N}{M})^2 \log N \log \frac{N}{M})$ ステップでシミュレートするアルゴリズム [4].

これらのセルフシミュレーションアルゴリズムは，アルゴリズムを設計する際に実際に利用可能な RM の大きさに注意を払う必要がなくなるため，非常に有用である．しかし，実際に利用可能な RM の大きさを考慮してアルゴリズムを設計することにより，セルフシミュレーションアルゴリズムを用いるよりも効率良く問題を解くことができる場合がある．例えば，文献 [5] では以下のアルゴリズムが提案されている．ここで，$N \geq M$ とする．

- N 個の値のソートを $M \times M$ LRM 上で $O(\frac{N}{M} + \frac{N}{M^2} \log \frac{N}{M^2})$ 時間で行なうアルゴリズム [5].

本稿では，実際に利用可能な RM の大きさを考慮したアルゴリズムとして，以下を示す．ここで，$N \geq M$ とする．

- 平面上の N 点が与えられたとき，その凸包を $M \times M$ LRM 上で $O(\frac{N}{M} + \frac{N}{M^2} \log \frac{N}{M^2})$ 時間で求めるアルゴリズム．入力が x 座標でソートされている場合には，$O(\frac{N}{M})$ 時間で求められる．

このアルゴリズムは，$N = M$ または $M^{1+\epsilon} \leq N \leq M^2$（$\epsilon$：任意の正定数）のとき以外では，文献 [2] や [3] のアルゴリズムをセルフシミュレーションアルゴリズムを用いて実行するよりも効率が良い．

本稿の構成は以下のとおりである．まず，2 節で本稿で扱うモデルを定義する．3 節で凸包アルゴリズムで利用する置換アルゴリズムを示し，4 節で凸包アルゴリズムを示す．最後に，5 節で本稿の結果をまとめる．

2 モデル

2.1 再構成メッシュ

$M \times N$ RM は，2 次元格子状に配置された $M \times N$ 個のプロセッサからなる SIMD（single instruction stream multiple data stream）型の並列計算モデルである．すべてのプロセッサは同じプログラムを実行し，同期して動作する．また，各プロセッサは局所メモリを持ち，共有メモリは存在しない．2 次元格子の $[i,j](0 \leq i < M, 0 \leq j < N)$，すなわち，$i$ 行 j 列に位置するプロセッサを PE$[i,j]$ と表す．PE$[0,j](0 \leq j < N)$ は RM の最上行に位置するプロセッサであり，PE$[i,0](0 \leq i < M)$ は RM の最左列に位置するプロセッサである（図 1）．各 PE$[i,j]((0 \leq i < M, 0 \leq j < N)$ は，自分の位置，すなわち，i,j を知っていると仮定する．

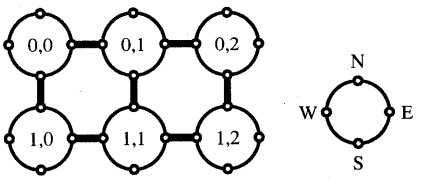


図 1: 2 × 3 RM とプロセッサ

各プロセッサは，4 つのポート N, S, E, W を備えており，隣接するプロセッサの互いに向かい合うポートは静的な外部バスで接続されている（図 1）．すなわち，各 $i,j(0 \leq i < M, 0 \leq j < N-1)$ について，PE$[i,j]$ のポート E は PE$[i,j+1]$ のポート W と静的な外部バスで接続されている．同様に，各 $i,j(0 \leq i < M-1, 0 \leq j < N)$ について，PE$[i,j]$ のポート S は PE$[i+1,j]$ のポート N と静的な外部バスで接続されている．また，各プロセッサは各ポートを内部バスによって接続することができる（図 2）．内部バスの形状の表現は，例えば，ポート N と E，ポート S と W を接続したときは {NE,SW} と表す．ポート間を接続する内部バスの形状は，アルゴリズムの実行の過程で動的に変更することができる．

各プロセッサは単位時間に以下の 4 つの動作を順に行なう．

1. 内部バスの形状を変更する．
2. 各ポートにデータを送信する．ポートに送信されたデータは，プロセッサの内部バスとプロセッサ間の静的な外部バスを通じて転送される．

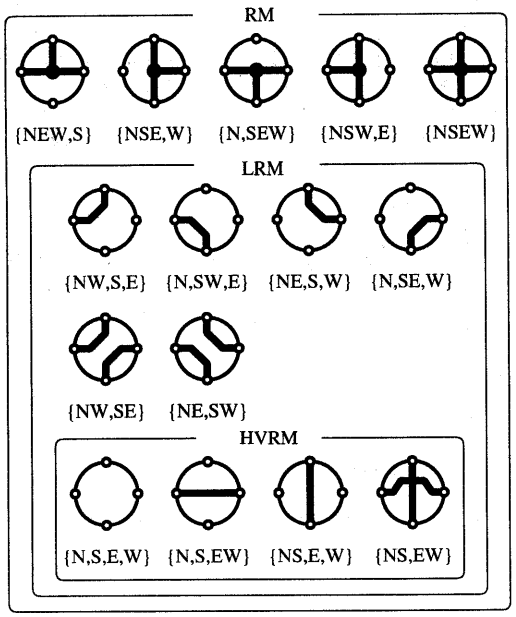


図 2: 内部バスの形状

3. 各ポートからデータを受信する．ポートにデータが流れてないときは，特別な値 ϕ を受信する．

4. RAM (random access machine) の命令を定数個実行する．

上記の4つの動作を1ステップとし，アルゴリズムの実行時間はステップ数で評価する．バス上のデータ伝送遅延時間はバスの長さに関わらず定数時間であると仮定し，1ステップは定数時間で実行できると考える．また，同一バスに同時にデータを送信するプロセッサは高々ひとつであると仮定する．初期状態では，各プロセッサの入力が与えられる変数以外の全ての変数は，特別な値 ϕ に初期化されているものとする．

取り得る内部バスの形状を制限することにより，以下のモデルが考えられている（図 2）．

水平垂直再構成メッシュ（Horizontal-Vertical RM） 各プロセッサの取り得る内部バスの形状が {N,S,E,W}, {N,S,EW}, {NS,E,W}, {NS,EW} の 4 種類に制限されたもの．以下，HVRM と書く．

線形再構成メッシュ（Linear RM） 各プロセッサの取り得る内部バスの形状が HVRM で取り得る内部バスの形状に加えて，{NW,S,E}, {N,SW,E}, {NE,S,W}, {N,SE,W}, {NW,SE}, {NE,SW} の 10 種類に制限されたもの．以下，LRM と書く．

アルゴリズムの記述には，以下の表記を用いる．

- PE$[i,j]$ が内部バスの形状を，例えば，{NE,S,W} にすることを次のように表す．
  PE$[i,j]$: {NE,S,W}
- PE$[i,j]$ がポートにデータを送信する，例えば，ポート S に局所変数 c の値を送信することを次のように表す．
  PE$[i,j]$: S $\leftarrow c$
- PE$[i,j]$ がポートからデータを受信する，例えば，ポート S からデータを受信し，局所変数 c に格納することを次のように表す．
  PE$[i,j]$: $c \leftarrow$ S

2.2 基本的性質

以下の結果が知られている．

定理 2.1 [4] $N \times N$ HVRM の1ステップを $M \times M$ HVRM によって，$5(\frac{N}{M})^2 + O(\frac{N}{M})$ ステップで，シミュレートできる． □

定理 2.2 [4] $N \times N$ LRM の1ステップを $M \times M$ LRM によって，$\Theta((\frac{N}{M})^2)$ ステップで，シミュレートできる． □

定理 2.3 [4] $N \times N$ RM の1ステップを $M \times M$ RM によって，$\Theta((\frac{N}{M})^2 \log N \log \frac{N}{M})$ ステップで，シミュレートできる． □

3 置換アルゴリズム

3.1 置換問題

定義 3.1 (置換問題) $M \times M$ HVRM 上での N 個の値の置換問題は，次のように定義される($N \geq M$)．ここで，配列変数 $\alpha[j], \beta[j]$ の値は，それぞれ，PE$[0, j \bmod M]$ の局所変数 $a[j \operatorname{div} M], b[j \operatorname{div} M]$ に記憶される（$0 \leq j < N$）．

入力 値 p_j, 置換先 q_j が，それぞれ，$\alpha[j], \beta[j]$ に与えられる（$0 \leq j < N$）．各 q_j は $0 \leq q_j < N$，または，$q_j = \phi$ であり，ϕ でない各 q_j は互いに異なる．

出力 $q_j \neq \phi$ ならば，$\alpha[q_j]$ に値 p_j が格納される（$0 \leq j < N$）． □

一般に定義される置換問題は上記の定義と違い，置換先として ϕ を許さない．ここでは，4 節の凸包アルゴリズムのために，置換問題の定義を拡張している．

以下，置換問題を解くアルゴリズムを示す．

3.2 PERMUTATION-HVRM

ここでは，$M \times M$ HVRM 上での N 個の値の置換問題を $O(\frac{N}{M})$ 時間で解くアルゴリズム PERMUTATION-HVRM を示す（$N \geq M$）．簡単のため，$N \bmod M = 0$ とする．

図 3 に，$M \times M$ HVRM 上での N 個の値の置換問題を解くアルゴリズム PERMUTATION-HVRM を示す．アルゴリズム中の $c, d[k] (0 \leq k < \frac{N}{M})$ は各プロセッサが持つ局所変数であり，初期状態では，特別な値 ϕ に初期化されている．

定理 3.1 $M \times M$ HVRM 上での N 個の値の置換問題は，$O(\frac{N}{M})$ 時間で解くことができる．

ステージ1

for $k \leftarrow 0$ **to** $\frac{N}{M} - 1$

PE$[i,j]$ $(0 \leq i,j < M)$: {NS,E,W}
PE$[0,j]$ $(0 \leq j < M)$: S $\leftarrow a[k]$
PE$[j,j]$ $(0 \leq j < M)$: $a[k] \leftarrow$ N
PE$[0,j]$ $(0 \leq j < M)$: S $\leftarrow b[k]$
PE$[j,j]$ $(0 \leq j < M)$: $b[k] \leftarrow$ N

ステージ2

for $k \leftarrow 0$ **to** $\frac{N}{M} - 1$

PE$[i,j]$ $(0 \leq i,j < M)$: {N,S,EW}
PE$[j,j]$ $(0 \leq j < M)$: E $\leftarrow b[k]$
PE$[i,j]$ $(0 \leq i,j < M)$: $c \leftarrow$ E
PE$[j,j]$ $(0 \leq j < M)$: E $\leftarrow a[k]$
PE$[i,j]$ $(0 \leq i,j < M)$:
  if $(c \neq \phi) \wedge (j = c \bmod M)$
  then $d[c \operatorname{div} M] \leftarrow$ E

ステージ3

for $k \leftarrow 0$ **to** $\frac{N}{M} - 1$

PE$[i,j]$ $(0 \leq i,j < M)$: {NS,E,W}
PE$[i,j]$ $(0 \leq i,j < M)$: **if** $d[k] \neq \phi$ **then** N $\leftarrow d[k]$
PE$[0,j]$ $(0 \leq j < M)$: $a[k] \leftarrow$ S

図 3: PERMUTATION-HVRM

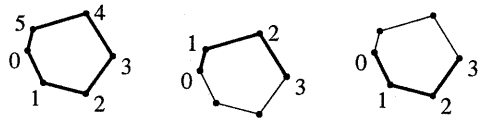
証明 PERMUTATION-HVRM のステージ1で，値 p_j, q_j は PE$[j \bmod M, j \bmod M]$ の局所変数 $a[j \operatorname{div} M], b[j \operatorname{div} M]$ に格納される．ステージ2で，$q_j \neq \phi$ ならば，値 p_j は PE$[j \bmod M, q_j \bmod M]$ の局所変数 $d[q_j \operatorname{div} M]$ に格納される．ステージ3で，値 p_j は PE$[0, q_j \bmod M]$ の局所変数 $a[q_j \operatorname{div} M]$ に格納される．すなわち，値 p_j は変数 $\alpha[q_j]$ に格納される．各ステージの繰り返しを1回実行するのにかかる時間は $O(1)$ 時間である．各ステージの繰り返しの回数は $\frac{N}{M}$ 回なので，PERMUTATION-HVRM は $O(\frac{N}{M})$ 時間で実行できる． □

4 凸包アルゴリズム

4.1 凸包

S を二次元平面上の N 点集合とするとき，S の凸包 $CH(S)$ (convex hull) とは，S を包含する最小の凸多角形である．点 p が凸包 $CH(S)$ の頂点であることを $p \in CH(S)$ と表す．凸包 $CH(S)$ の頂点の個数を $n_{CH(S)}$ と表す．最左頂点と最右頂点を通る直線によって，凸包を分割したとき，上部境界を上部凸包 $UH(S)$ (upper hull)，下部境界を下部凸包 $LH(S)$ (lower hull) という．

$CH(S)$ の頂点の $CH(S)$ におけるランクとは，最左頂点を 0 として反時計順につけられた頂点の番号である．点 $p \notin CH(S)$ のとき，点 p の $CH(S)$ におけるランクは ϕ とする (図 4 (a))．同様に，$UH(S)$ の頂点の $UH(S)$ におけるランクは，最左頂点を 0 として時計順につけられた頂点の番号であり (図 4 (b))，$LH(S)$ の頂点の $LH(S)$ におけるランクは，最左頂点を 0 として反時計順につけられた頂点の番号である (図 4 (c))．頂点 p の $CH(S), UH(S), LH(S)$ におけるランクを，それぞれ，$Rank^p_{CH(S)}, Rank^p_{UH(S)}, Rank^p_{LH(S)}$ と表す．



(a) 凸包におけるランク　(b) 上部凸包におけるランク　(c) 下部凸包におけるランク

図 4: ランク

4.2 凸包問題

定義 4.1 (凸包問題) $M \times M$ LRM 上での N 個の点の凸包問題は，次のように定義される ($N \geq M$)．ここで，配列変数 $\alpha[j]$ の値は，PE$[0, j \bmod M]$ の局所変数 $a[j \operatorname{div} M]$ に記憶される $(0 \leq j < N)$．

入力 N 点集合 $S = \{p_0, p_1, \ldots, p_{N-1}\}$ の各点 p_j が，$\alpha[j]$ に与えられる $(0 \leq j < N)$．

出力 $CH(S)$ の頂点が，$CH(S)$ の任意の頂点から始めて反時計順で，$\alpha[j](0 \leq j < n_{CH(S)})$ に格納される．また，各プロセッサは $n_{CH(S)}$ を知る． □

以下，まず，入力が x 座標でソートされているときに，$M \times M$ LRM 上での N 個の点の凸包問題を解くアルゴリズムを示し，その後，一般の入力の場合のアルゴリズムを示す．

簡単のため，入力として与えられる点は，一般の位置にあると仮定する．すなわち，任意の二点は同じ x 座標をもたず，任意の三点は同一直線上にないと仮定する．

4.3 SORTHULL-LRM

ここでは，入力が x 座標でソートされているときに，$M \times M$ LRM 上での N 個の点の凸包問題を $O(\frac{N}{M})$ 時間で解くアルゴリズム SORTHULL-LRM を示す ($N \geq M$)．p_j の x 座標 $< p_{j+1}$ の x 座標 $(0 \leq j < N-1)$ が成り立つ．簡単のため，$N \bmod M = 0$ とする．SORTHULL-LRM は，図 5 の3つのステージからなる．

以下，各ステージについて，詳しく述べる．

4.3.1 ステージ1

ここでは，SORTHULL-LRM のステージ1を $M \times M$ LRM 上でいかに実現するかを示す．SORTHULL-LRM のステージ1では，S を $\frac{N}{M}$ 個の M 点集合 $S_i (0 \leq i < \frac{N}{M})$ に分割し，各 $CH(S_i)$ を求める $(0 \leq i < \frac{N}{M})$．また，ステージ2の前処理として，各点 $p \in S_i$ について，$Rank^p_{UH(S_i)}, Rank^p_{LH(S_i)}$ を求める $(0 \leq i < \frac{N}{M})$．図 6 に，SORTHULL-LRM のステージ1を行なうアルゴリズムを示す．ここで，$l, r, subrank_u[i] (0 \leq i <$

ステージ1

次の条件を満たすように，S を $\frac{N}{M}$ 個の M 点集合 $S_i(0 \leq i < \frac{N}{M})$ に分割する．

- S_i の任意の点の x 座標 $< S_{i+1}$ の任意の点の x 座標 $(0 \leq i < \frac{N}{M} - 1)$

各 $CH(S_i)$ を求める $(0 \leq i < \frac{N}{M})$．

ステージ2

$CH(S_i)(0 \leq i < \frac{N}{M})$ をもとに，$UH(S)$ を求める．

$CH(S_i)(0 \leq i < \frac{N}{M})$ をもとに，$LH(S)$ を求める．

ステージ3

$UH(S)$ と $LH(S)$ を合わせて，$CH(S)$ を求める．

図 5: SORTHULL-LRM

$\frac{N}{M}), subrank_l[i](0 \leq i < \frac{N}{M})$ は，各プロセッサの持つ局所変数である．また，各 $\alpha[j](0 \leq j < N)$ には，点の x 座標，y 座標，識別子の三つ組が格納されるものとする．すなわち，点 p_i の (x, y) 座標が (x_i, y_i) であり，$\alpha[j]$ に点 p_i が保持されているとき，$\alpha[j]$ には (x_i, y_i, i) が格納されている．それぞれ，$\alpha[j].x, \alpha[j].y, \alpha[j].index$ で参照する．

S は x 座標でソートされているので，$S_i = \{p_{iM}, p_{iM+1}, \ldots, p_{iM+(M-1)}\}(0 \leq i < \frac{N}{M})$ である．S の各 $S_i(0 \leq i < \frac{N}{M})$ への分割は，各 S_i の点を $\alpha[iM + j](0 \leq j < M)$ に格納することによって行なう $(0 \leq i < \frac{N}{M})$．しかし，これは初期状態で満たされるので，分割のために特に何もする必要はない．また，各 $S_i(0 \leq i < \frac{N}{M})$ は $M \times M$ LRM 上での M 個の点の凸包問題の入力と同じ形式で記憶されている．したがって，各 $S_i(0 \leq i < \frac{N}{M})$ の凸包は文献 [2] のアルゴリズムを用いて求められる．フェーズ1で，各 $CH(S_i)(0 \leq i < \frac{N}{M})$ の頂点が，任意の頂点から始めて反時計順に，$\alpha[iM + j](0 \leq j < n_{CH(S_i)})$ に格納され，各プロセッサは，$n_{CH(S_i)}$ を知る．すなわち，各 $CH(S_i)(0 \leq i < \frac{N}{M})$ が求まる．

フェーズ2の i 回目の繰り返しでは，まず 1, 2 で，$CH(S_i)$ の最左頂点 p_{iM}，および，最右頂点 $p_{iM+(M-1)}$ を保持しているプロセッサが何列目に位置するかを各 PE$[0, j](0 \leq j < M)$ の変数 l, r に格納する．次に 3 で，各 PE$[0, j](0 \leq j < M)$ は，$a[i]$ に保持している点 p について，j, l, r をもとに $Rank^p_{UH(S_i)}, Rank^p_{LH(S_i)}$ を計算し，それぞれを $subrank_u[i], subrank_l[i]$ に格納する．

フェーズ1の繰り返しを1回実行するのにかかる時間は，文献 [2] より，$O(1)$ 時間である．フェーズ2の繰り返しを1回実行するのにかかる時間も $O(1)$ 時間である．各フェーズの繰り返しの回数は $\frac{N}{M}$ 回なので，SORTHULL-LRM のステージ1全体にかかる時間は $O(\frac{N}{M})$ 時間である．

4.3.2 ステージ2

ここでは，SORTHULL-LRM のステージ2を $M \times M$ LRM 上でいかに実現するかを示す．ステージ2では，$UH(S), LH(S)$ を求める．$LH(S)$ は $UH(S)$ と同

フェーズ1

for $i \leftarrow 0$ **to** $\frac{N}{M} - 1$

文献 [2] のアルゴリズムを用いて，$\alpha[iM + j](0 \leq j < M)$ の凸包を求める．

フェーズ2

for $i \leftarrow 0$ **to** $\frac{N}{M} - 1$

1. PE$[0, j](0 \leq j < M)$: {N,S,EW}
   PE$[0, j](0 \leq j < n_{CH(S_i)})$:
   if $a[i].index = iM$ **then** E $\leftarrow j$
   PE$[0, j](0 \leq j < M)$: $l \leftarrow$ W
2. PE$[0, j](0 \leq j < n_{CH(S_i)})$:
   if $a[i].index = iM + (M - 1)$ **then** E $\leftarrow j$
   PE$[0, j](0 \leq j < M)$: $r \leftarrow$ W
3. PE$[0, j](0 \leq j < M)$: $subrank_u[i] \leftarrow f_u(j, l, r)$
   PE$[0, j](0 \leq j < M)$: $subrank_l[i] \leftarrow f_l(j, l, r)$
   - $f_u(j, l, r), f_l(j, l, r)$ は次のように定義される．

$$f_u(j,l,r) = \begin{cases} l - j \\ \quad (l \geq j \geq r, \text{ または, } r > l \geq j) \\ n_{CH(S_i)} - j + l \\ \quad (n_{CH(S_i)} > j \geq r > l) \\ \phi \quad (\text{それ以外}) \end{cases}$$

$$f_l(j,l,r) = \begin{cases} j - l \\ \quad (r \geq j \geq l, \text{ または,} \\ \quad\quad n_{CH(S_i)} > j \geq l > r) \\ j + n_{CH(S_i)} - l \quad (l > r \geq j) \\ \phi \quad (\text{それ以外}) \end{cases}$$

図 6: SORTHULL-LRM のステージ1

様にして求めることができるので，ここでは，$UH(S)$ の求め方のみ示す．

$UH(S_0 \cup S_1), UH(S_0 \cup S_1 \cup S_2), \ldots$ の順に求めていき，最終的に $UH(S)$ を求める．$UH(S)$ を求めるアルゴリズムを図7に示す．ここで，$pred[k](1 \leq k < \frac{N}{M})$ は整数値をとる変数，uh は上部凸包を表すために使われる点の列を値としてとる変数である．これらの変数を LRM 上でいかに表現するか，また，このアルゴリズムを LRM 上でいかに実現するかは後で述べる．

以下，図7のアルゴリズムを説明する．まず，必要な記号を定義する．m_i を次のように定義する．

$$m_i = |\ \{S_j(0 \leq j \leq i)\ |\ \exists p(p \in S_j \wedge p \in UH(\cup_{l=0}^{i} S_l))\}\ |$$

すなわち，m_i は $UH(\cup_{l=0}^{i} S_l)$ の頂点を少なくとも一つ含む部分集合 $S_j(0 \leq j \leq i)$ の個数である．また，$k_0^i, k_1^i, \ldots, k_{m_i-1}^i$ を次を満たす列とする．

$$l' \in \{k_0^i, k_1^i, \ldots, k_{m_i-1}^i\} \Leftrightarrow \exists p(p \in S_{l'} \wedge p \in UH(\cup_{l=0}^{i} S_l))$$

$$k_0^i < k_1^i < \ldots < k_{m_i-1}^i$$

すなわち，$k_0^i, k_1^i, \ldots, k_{m_i-1}^i$ は，$UH(\cup_{l=0}^{i} S_l)$ の頂点を少なくとも一つ含む部分集合 $S_j(0 \leq j \leq i)$ の識別子（S_j の識別子は j）を，小さい順に並べた列である．明らかに，$k_0^i = 0, k_{m_i-1}^i = i$ である．例えば，図9の

1. $uh \leftarrow UH(S_0)$

for $i \leftarrow 1$ **to** $\frac{N}{M} - 1$

2. $pred[i] \leftarrow i - 1$

repeat

3. $CH(S_{pred[i]})$ と $CH(S_i)$ の上側共通接線を求める.
   - $CH(S_{pred[i]}), CH(S_i)$ 上の接点を，それぞれ，q_2, q_1 とする.
4. $pred[i] > 0$ ならば，$CH(S_{pred[pred[i]]})$ と $CH(S_{pred[i]})$ の上側共通接線を求める.
   - $CH(S_{pred[pred[i]]}), CH(S_{pred[i]})$ 上の接点を，それぞれ，q_4, q_3 とする.
5. 図 8 のどのケースに当てはまるか調べる.
6. **if** （ケース 4 か 5） **then**

   uh から，$S_{pred[i]}$ の点を取り除く.
   $pred[i] \leftarrow pred[pred[i]]$

until （ケース 1 か 2 か 3 に当てはまるまで）

7. uh から $CH(S_{pred[i]})$ の q_2 の右隣の頂点から時計順で最右頂点までの頂点を取り除く.
   uh に $CH(S_i)$ の q_1 から時計順で最右頂点までの頂点を加える.

図 7: $UH(S)$ を求めるアルゴリズム

場合，$m_7 = 4, k_0^7 = 0, k_1^7 = 3, k_2^7 = 5, k_3^7 = 7$ であり，$m_8 = 3, k_0^8 = 0, k_1^8 = 3, k_2^8 = 8$ である.

図 7 のアルゴリズムでは，i 回目の **for** ループで，uh に $UH(\cup_{l=0}^{i} S_l)$ を求める（$1 \leq i < \frac{N}{M}$）. $UH(\cup_{l=0}^{i-1} S_l)$ が求まっているとき，次のようにして，$UH(\cup_{l=0}^{i} S_l)$ を求めている.

$k = k_{m_{i-1}-1}^{i-1}, k_{m_{i-1}-2}^{i-1}, k_{m_{i-1}-3}^{i-1}, \ldots$ の順に，$CH(S_k)$ と $CH(S_i)$ の上側共通接線を求めていき，$UH(\cup_{l=0}^{i-1} S_l)$ と $CH(S_i)$ の上側共通接線を見つける. 例えば，図 9 の場合，$UH(\cup_{l=0}^{7} S_l)$ が求まっているとき，$k = 7, 5, 3$ の順に，$CH(S_k)$ と $CH(S_8)$ の上側共通接線を求めていく. $CH(S_3)$ と $CH(S_8)$ の上側共通接線が $UH(\cup_{l=0}^{7} S_l)$ と $CH(S_8)$ の上側共通接線である.

$UH(\cup_{l=0}^{i-1} S_l)$ と $CH(S_i)$ の上側共通接線の $UH(\cup_{l=0}^{i-1} S_l), CH(S_i)$ 上の接点を，それぞれ，q_2, q_1 とする. このとき，$UH(\cup_{l=0}^{i} S_l)$ は，$UH(\cup_{l=0}^{i-1} S_l)$ の最左頂点から時計順で q_2 までの頂点，および，$CH(S_i)$ の q_1 から時計順で最右頂点までの頂点である.

i 回目の **for** ループでは uh に $UH(\cup_{l=0}^{i} S_l)$ を求めることに加え，$UH(\cup_{l=0}^{i} S_l)$ の頂点を少なくとも一つ含む各 $CH(S_j)$ を，右から順にたどれるように，$pred[k] (1 \leq k < \frac{N}{M})$ にリストとして保持する. 以下が成り立つ.

補題 4.1 図 7 のアルゴリズム中の $i (1 \leq i < \frac{N}{M})$ 回目の **for** ループ実行後，次が成り立つ.

(条件 1) $uh = UH(\cup_{l=0}^{i} S_l)$.

(条件 2) 各 $j (0 < j < m_i)$ について，$pred[k_j^i] = k_{j-1}^i$.

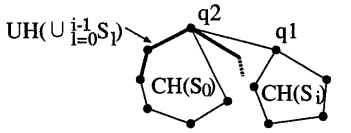
ケース 1 $pred[i] = 0$ （図 8 (a)）.

ケース 2 $pred[i] > 0$，かつ，q_3 より q_2 が右にある（図 8 (b)）.

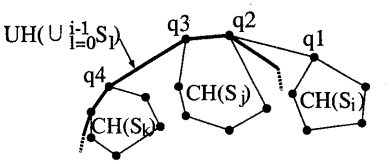
ケース 3 $pred[i] > 0$，かつ，$q_2 = q_3$，かつ，$q_1 q_2 q_4$ の下側の角度が 180 度未満（図 8 (c)）.

ケース 4 $pred[i] > 0$，かつ，q_3 より q_2 が左にある（図 8 (d)）.

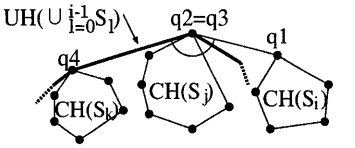
ケース 5 $pred[i] > 0$，かつ，$q_2 = q_3$，かつ，$q_1 q_2 q_4$ の下側の角度が 180 度以上（図 8 (e)）.



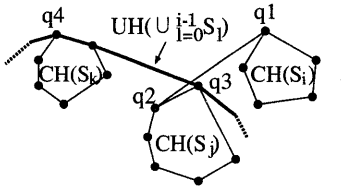
(a) ケース 1



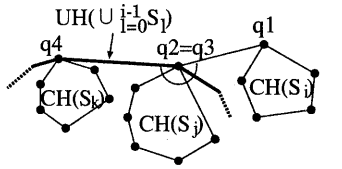
(b) ケース 2



(c) ケース 3



(d) ケース 4



(e) ケース 5

- 図中では，$j = pred[i], k = pred[j]$.

図 8: 各ケース

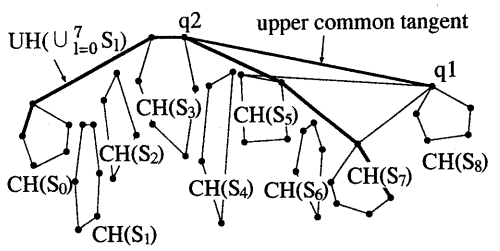


図 9: $UH(\cup_{l=0}^{7} S_l)$ と $CH(S_8)$ の上側共通接線

証明 i に関する帰納法で証明できる．詳細は省略する（文献 [6] 参照）． □

次に，図 7 のアルゴリズムの時間計算量を評価するために，以下を示す．

補題 4.2 図 7 のアルゴリズム中の **repeat** ループの実行回数は，アルゴリズム全体で $O(\frac{N}{M})$ 回である．

証明 i 回目の **for** ループでの **repeat** ループの実行回数を r_i 回とする．$r_1 = 1$，$r_i = m_{i-1} - m_i + 2 (1 < i < \frac{N}{M})$ が成り立つ（文献 [6] 参照）．アルゴリズム全体での **repeat** ループの実行回数は，次のようになる．

$$
\begin{aligned}
\sum_{i=1}^{\frac{N}{M}-1} r_i = & \quad 1 \\
& + \quad 1 \quad - \quad \not{m}_2 \quad + 2 \\
& + \quad \not{m}_2 \quad - \quad \not{m}_3 \quad + 2 \\
& \qquad \vdots \\
& + \not{m}_{\frac{N}{M}-2} - m_{\frac{N}{M}-1} + 2 \\
= & \ 2(\tfrac{N}{M} - 1) - m_{\frac{N}{M}-1} \\
\leq & \ 2(\tfrac{N}{M} - 1)
\end{aligned}
$$

したがって，補題 4.2 が成り立つ． □

補題 4.1 より，図 7 のアルゴリズム実行後，uh に $UH(S)$ が求まることがいえる．以下，図 7 のアルゴリズムを $M \times M$ LRM 上でいかに実現するかを示す．

まず，$pred[k] (1 \leq k \leq \frac{N}{M})$，および，$uh$ を LRM 上でいかに表現するかを述べる．$pred[k]$ の値は全てのプロセッサの局所変数 $p[k]$ に同じ値が記憶されるものとする．また，uh の表現には，PE$[0,j] (0 \leq j < M)$ の局所変数 $rank_u[k] (0 \leq k < \frac{N}{M})$ を用いる．i 回目の **for** ループの実行では，各 PE$[0,j] (0 \leq j < M)$ は，$a[k] (0 \leq k \leq i)$ に保持している点 p について，$Rank^p_{uh}$ を $rank_u[k]$ に格納することによって，uh を表現する．すなわち，i 回目の **for** ループ実行後，$Rank^p_{UH(\cup_{l=0}^{i} S_l)}$ が $rank_u[k]$ に格納される．

以下，図 7 の $UH(S)$ を求めるアルゴリズムの 1 ～ 7 を $M \times M$ LRM 上でいかに実現するかを示す．

1. $UH(S_0)$ の各頂点のランクは，ステージ 1 で，$subrank_u[0]$ に求まっているので，以下を行なう．

   PE$[0,j] (0 \leq j < M)$: $rank_u[0] \leftarrow subrank_u[0]$

2. $pred[i]$ の値は全てのプロセッサの局所変数 $p[i]$ に記憶されるので，以下を行なう．

   PE$[k,j] (0 \leq k,j < M)$: $p[i] \leftarrow i - 1$

3. $CH(S_{pred[i]}), CH(S_i)$ の頂点は，それぞれ，$\alpha[pred[i]M + j] (0 \leq j < n_{CH(S_{pred[i]})}), \alpha[iM + j] (0 \leq j < n_{CH(S_i)})$ に格納されている．$n_{CH(S_{pred[i]})} \leq M, n_{CH(S_i)} \leq M$ であることから，文献 [2] 中で使用されているアルゴリズムを用いて，$O(1)$ 時間で，$CH(S_{pred[i]})$ と $CH(S_i)$ の上部共通接線を求めることができる．各プロセッサは q_1, q_2 を知る．

4. 3 と同様にして，各プロセッサは q_3, q_4 を知る．

5. 各プロセッサは，q_1, q_2, q_3, q_4 を知っている．それらの位置関係から，各プロセッサは図 8 のいずれのケースに当てはまるか判定できる．

6. ケース 4 か 5 なら，$S_{pred[i]}$ の点のランクを ϕ に更新し，$pred[i]$ をリストから取り除く．以下を行なう．

   PE$[0,j] (0 \leq j < M)$: $rank_u[p[i]] \leftarrow \phi$
   PE$[k,j] (0 \leq k,j < M)$: $p[i] \leftarrow p[p[i]]$

7. $CH(S_{pred[i]})$ の q_2 の右隣の頂点から最右頂点までの頂点のランクをϕ に更新する．また，$CH(S_i)$ の q_1 から最右頂点までの頂点のランクを求める．以下を行なう．ここで，c, d は，各プロセッサのもつ局所変数である．

   PE$[0,j] (0 \leq j < M)$: {N,S,EW}
   PE$[0,j] (0 \leq j < n_{CH(S_{p[i]})})$:
   if $a[p[i]] = q_2$ **then** E $\leftarrow rank_u[p[i]]$
   PE$[0,j] (0 \leq j < M)$: $c \leftarrow$ W
   PE$[0,j] (0 \leq j < n_{CH(S_i)})$:
   if $a[i] = q_1$ **then** E $\leftarrow subrank_u[i]$
   PE$[0,j] (0 \leq j < M)$: $d \leftarrow$ W
   PE$[0,j] (0 \leq j < M)$:
   if $c < rank_u[p[i]]$ **then** $rank_u[p[i]] \leftarrow \phi$
   PE$[0,j] (0 \leq j < M)$:
   if $d \leq subrank_u[i]$
   then $rank_u[i] \leftarrow subrank_u[i] + c - d + 1$

補題 4.2 より，**repeat** ループの実行回数は，全体で $O(\frac{N}{M})$ 回である．図 7 のアルゴリズムの 1 ～ 7 はいずれも $O(1)$ 時間で実行出来るので，SORTHULL-LRM のステージ 2 で，$UH(S)$ を求めるのにかかる時間は $O(\frac{N}{M})$ 時間である．

$LH(S)$ も $UH(S)$ と同様に，$O(\frac{N}{M})$ 時間で求められる．ステージ 2 実行後，各 PE$[0,j] (0 \leq j < M)$ の $a[k] (0 \leq k < \frac{N}{M})$ に格納されている点 p について，$Rank^p_{UH(S)}$ が $rank_u[k]$ に格納される．同様に，$Rank^p_{LH(S)}$ が $rank_l[k]$ に格納される．

4.3.3 ステージ3

ここでは，SORTHULL-LRM のステージ3を $M \times M$ LRM 上でいかに実現するかを示す．ステージ3では，各頂点の $UH(S), LH(S)$ におけるランクから，$CH(S)$ におけるランクを求め，頂点が反時計順に $\alpha[j](0 \leq j < n_{CH(S)})$ に格納されるように，置換を行なう．また，各プロセッサは $n_{CH(S)}$ を知る．図 10 に，SORTHULL-LRM のステージ3を行なうアルゴリズムを示す．ここで，$\beta[j]$ は，PE$[0, j \bmod M]$ の局所変数 $rank_l[j \operatorname{div} M]$ に値が記憶される配列変数とする $(0 \leq j < N)$．また，n, nu, nl は，各プロセッサのもつ局所変数である．

フェーズ1

1. PE$[0,j]$ $(0 \leq j < M)$: {N,S,EW}
   PE$[0,j]$ $(0 \leq j < M)$:
   if $a[\frac{N}{M}-1].index = N-1$
   then E $\leftarrow rank_u[\frac{N}{M}-1]$
   PE$[0,j]$ $(0 \leq j < M)$: $nu \leftarrow$ W
   PE$[0,j]$ $(0 \leq j < M)$:
   if $a[\frac{N}{M}-1].index = N-1$ **then** E $\leftarrow rank_l[\frac{N}{M}-1]$
   PE$[0,j]$ $(0 \leq j < M)$: $nl \leftarrow$ W
2. PE$[i,j]$ $(0 \leq i,j < M)$: {NS,E,W}
   PE$[0,j]$ $(0 \leq j < M)$: S $\leftarrow nu + nl$
   PE$[i,j]$ $(0 \leq i,j < M)$: $n \leftarrow$ N
3. **for** $i \leftarrow 0$ bf to $\frac{N}{M}-1$
   PE$[0,j]$ $(0 \leq j < M)$:
   if $(rank_u[i] \neq \phi) \wedge (rank_u[i] \neq 0)$
   then $rank_l[i] \leftarrow nl + nu - rank_u[i]$

フェーズ2

PERMUTATION-HVRM を用いて，$\alpha[k]$ の値を $\alpha[\beta[k]]$ に置換する $(0 \leq k < N)$．

図 10: SORTHULL-LRM のステージ3

フェーズ1では，$n_{CH(S)}$，および，各頂点の $CH(S)$ におけるランクを求めている．1では，最右頂点 p_{N-1} の $UH(S), LH(S)$ におけるランクを最上行の各プロセッサに知らせている．2では，それをもとに，各プロセッサの局所変数 n に $n_{CH(S)}$ を求めている．3では，各 PE$[0,j](0 \leq j < M)$ は，$a[k](0 \leq k < \frac{N}{M})$ に格納されている点 p について，$Rank^p_{CH(S)}$ を求めている．$p \in LH(S)$ ならば $Rank^p_{CH(S)} = Rank^p_{LH(S)}$ なので，$UH(S)$ の頂点のみランクの計算を行なう．フェーズ2では，$CH(S)$ におけるランクを置換先として，各頂点を置換し，頂点が反時計順に $\alpha[j](0 \leq j < n_{CH(S)})$ に格納される．

フェーズ1の1,2にかかる時間は $O(1)$ 時間，3にかかる時間は $O(\frac{N}{M})$ 時間である．定理 3.1 より，フェーズ2にかかる時間は $O(\frac{N}{M})$ 時間である．したがって，SORTHULL-LRM のステージ3全体にかかる時間は $O(\frac{N}{M})$ 時間である．

4.3.4 時間計算量

SORTHULL-LRM の各ステージは，いずれも $O(\frac{N}{M})$ 時間で実行できる．したがって，以下の定理を得る．

定理 4.1 入力が x 座標でソートされているときに，$M \times M$ LRM 上での N 個の点の凸包問題は，$O(\frac{N}{M})$ 時間で解くことができる． □

4.4 CONVEXHULL-LRM

ここでは，$M \times M$ LRM 上での N 個の点の凸包問題を $O(\frac{N}{M} + \frac{N}{M^2} \log \frac{N}{M^2})$ 時間で解くアルゴリズム CONVEXHULL-LRM を示す $(N \geq M)$．文献 [5] で，N 個の値のソートを $M \times M$ LRM 上で $O(\frac{N}{M} + \frac{N}{M^2} \log \frac{N}{M^2})$ 時間で行なうアルゴリズムが提案されている $(N \geq M)$．したがって，文献 [5] のアルゴリズムを用いて，$O(\frac{N}{M} + \frac{N}{M^2} \log \frac{N}{M^2})$ 時間で与えられた点集合を x 座標でソートしたあと，SORTHULL-LRM を用いて $O(\frac{N}{M})$ 時間で凸包を求めることができる．以下を得る．

定理 4.2 $M \times M$ LRM 上での N 個の点の凸包問題は，$O(\frac{N}{M} + \frac{N}{M^2} \log \frac{N}{M^2})$ 時間で解くことができる． □

5 まとめ

本稿では，平面上の N 個の点の凸包を，$M \times M$ LRM を用いて，$O(\frac{N}{M} + \frac{N}{M^2} \log \frac{N}{M^2})$ 時間で求めるアルゴリズムを示した $(N \geq M)$．入力が x 座標でソートされているときは，$O(\frac{N}{M})$ 時間で求められる．このアルゴリズムは，$N = M$ または $M^{1+\epsilon} \leq N \leq M^2$ (ϵ：任意の正定数) のとき以外では，文献 [2] や [3] のアルゴリズムをセルフシミュレーションアルゴリズムを用いて実行するよりも効率が良い．

参考文献

[1] M. Nigam and S. Sahni. "Sorting n Numbers on $n \times n$ Reconfigurable Meshes with Buses". *J. of Parallel Distributed Computing*, 23(1):37–48, October 1994.

[2] J. Jang, M. Nigam, V. K. Prasanna, and S. Sahni. "Constant Time Algorithms for Computational Geometry on the Reconfigurable Mesh". *IEEE Trans. Parallel and Distributed Systems*, 8(1):1–12, January 1997.

[3] V. Bokka, H. Gurla, S. Olariu, and J. L. Schwing. "Podality-Based Time-Optimal Computations on Enhanced Meshes". *IEEE Trans. Parallel and Distributed Systems*, 8(10):1019–1035, October 1997.

[4] Y. Ben-Asher, D. Gordon, and A. Schuster. "Efficient Self-Simulation Algorithms for Reconfigurable Arrays". *J. of Parallel Distributed Computing*, 30(1):1–22, October 1995.

[5] 松前, 笹田, 都倉. "問題のサイズより小さい再構成メッシュ上でのソーティングアルゴリズム". 電子情報通信学会コンピューテーション研究会（発表予定）, March 1998.

[6] 笹田. "固定サイズの再構成メッシュ上で凸包を求めるアルゴリズム". 修士学位論文, 大阪大学, February 1998.