NAIST-IS-DT9761002

博士論文

BSP モデル上の並列アルゴリズムに関する研究

石水 隆

2000 年 3 月 13 日

奈良先端科学技術大学院大学
情報科学研究科 情報処理学専攻

本論文は奈良先端科学技術大学院大学情報科学研究科に
博士 (工学) 授与の要件として提出した博士論文である。

石水 隆

審査委員：　藤原 秀雄 教授
福田　晃 教授
伊藤　実 教授
増澤 利光 助教授

BSP モデル上の並列アルゴリズムに関する研究*

石水 隆

内容梗概

高速計算や大規模データ処理の必要性から, 近年並列処理の必要性が高まってきている. しかしながら, 既存の逐次処理と異なり, 並列処理ではプロセッサ間の通信やデータの分割など, 並列処理独自の処理に対する考慮が必要である. このため, 並列処理に対しては並列処理用の並列アルゴリズムが必要である.

本論文では, BSP(Bulk-Synchronous Parallel) モデルおよび BSP*モデル上で, 選択問題を解く並列アルゴリズムおよび全最近点問題を解く並列アルゴリズムを提案する. BSP モデルおよび BSP*モデルはバリア同期という緩い同期に基づいた並列計算のためのモデルであり，非同期式並列計算モデルと呼ばれる．これらのモデルでは，具体的な通信の特性などは同期周期 L, 通信命令実行時間 g，通信パケットサイズ B といったパラメタにより抽象化されている．

第 3 章では，データ数 n の選択問題に対し，p 個のプロセッサを用いて BSP モデル上で任意の整数 $d(1 \leq d \leq \log n)$ に対し内部計算時間 $O(\frac{n}{p} + d\log p \log\log n + L\frac{\log p \log\log n}{\log(d+1)})$，通信時間 $O(g\frac{n}{p} + (gd+L)\frac{\log p \log\log n}{\log(d+1)})$，また，BSP*モデル上で内部計算時間 $O(\frac{n}{p} + d\log p\log\log n + L\frac{\log p\log\log n}{\log(d+1)})$，通信時間 $O(g(\frac{n}{pB} + (\frac{n}{p})^{\frac{1}{7}}(\log p)^{\frac{6}{7}}) + (gd+L)\frac{\log p\log\log n}{\log(d+1)})$ の並列アルゴリズを提案する．

また，第 4 章では, 2 値画像に対して全最近点を求める BSP モデル上の並列アルゴリズムを，重み付き距離および L_p距離それぞれに対して提案する．これらのアルゴリズムは共にプロセッサ数 p が $1 \leq p \leq n$ の場合, 内部計算時間 $O(\frac{n^2}{p} + L)$，通信時間 $O(g\frac{n}{\sqrt{p}} + L)$ で全最近点を求める．また，$n < p \leq n^2$の場

*奈良先端科学技術大学院大学 情報科学研究科 情報処理学専攻 博士論文, NAIST-IS-DT9761002, 2000 年 3 月 13 日.

合，任意の整数 d $(1 \leq d \leq \frac{p}{n})$ に対し内部計算時間 $O(\frac{n^2}{p} + (d+L)\ \frac{\log \frac{p}{n}}{\log(d+1)})$, 通信時間 $O(g\frac{n}{\sqrt{p}} + (gd+L)\frac{\log \frac{p}{n}}{\log(d+1)})$ で全最近点を求める．

キーワード

並列アルゴリズム，BSP モデル，選択問題，全最近点問題

Studies on Parallel Algorithms on the BSP Model*

Takashi Ishimizu

Abstract

Due to the requirement of high speed computation for large number of data, parallel processing has received great interests in recent years. In the parallel processing, we must consider communications and task alocations to processors. Therefore, parallel processing needs especial parallel algorithms.

In this thesis, we present parallel algorithms for the selection problem and the all nearest neighbors problem on the BSP(Bulk-Synchronous Parallel) model and the BSP* model. The BSP model and the BSP* model are parallel computation models with barrier synchronization and are called asynchronous parallel computing models, which communication features are abstracted by three parameters L, g and B: L denotes synchronization periodicity, g denotes a time to excute one communication operation and B denotes a message packet size.

In Section 3, we present two algorithms for selection problem: one for n elements using p processors for the BSP model and the other for the BSP* model. The BSP algorithm runs in $O(\frac{n}{p} + d \log p \log\log n + L\frac{\log p \log\log n}{\log(d+1)})$ computation time and in $O(g\frac{n}{p} + (gd+L)\frac{\log p \log\log n}{\log(d+1)})$ communication time. The parameter d is a design parameter and can be set to any integer d $(1 \leq d \leq \log n)$. The BSP* algorithm runs in $O(\frac{n}{p} + d \log p \log\log n + L\frac{\log p \log\log n}{\log(d+1)})$ computation time and in $O(g(\frac{n}{pB} + (\frac{n}{p})^{\frac{1}{7}}(\log p)^{\frac{6}{7}}) + (gd+L)\frac{\log p \log\log n}{\log(d+1)})$ communication time.

In Section 4, we present two parallel algorithms on the BSP model for the all nearest neighbor problem of $n \times n$ binary images: one for weighted distance and the other for L_p distance. Both two algorithms run in $O(\frac{n^2}{p}+L)$ computation time

*Doctor's Thesis, Department of Information Processing, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DT9761002, March 13, 2000.

and in $O(g\frac{n}{\sqrt{p}}+L)$ communication time using p processors if $1 \leq p \leq n$, and they runs in $O(\frac{n^2}{p}+(d+L)\frac{\log \frac{p}{n}}{\log(d+1)})$ computation time and in $O(g\frac{n}{\sqrt{p}}+(gd+L)\frac{\log \frac{p}{n}}{\log(d+1)})$ communication time for any integer d $(1 \leq d \leq \frac{p}{n})$ using p processors if p $(n < p \leq n^2)$.

Keywords:

parallel algorithm, BSP model, selection, all nearest neighbors

関連発表一覧

- 学術論文誌

  1. 石水隆，藤原暁宏，井上美智子，増澤利光，藤原秀雄,“ 選択問題を解く BSP モデルおよび BSP*モデル上の並列アルゴリズム ”, 電子情報通信学会論文誌 (DI)，Vol. J82 – D – 1 No. 4, pp.533 – 542, April 1999.
  2. T.Ishimizu, A.Fujiwara, M.Inoue, T.Masuzawa and H.Fujiwara, “Parallel algorithms for all nearest neighbors of binary images on the BSP model”, Transaction of IEICE (D), Vol. E83 – D, No. 2, pp.151 – 158, February 2000.

- 国際会議 (査読付き)

  1. T.Ishimizu, A.Fujiwara, M.Inoue, T.Masuzawa and H.Fujiwara, “Parallel algorithms for all nearest neighbors of binary images on the BSP model”, Proceedings of Joint Symposium on Parallel Processing (JSPP'99), pp.253-260, June 1999
  2. A.Fujiwara, T.Ishimizu, M.Inoue, T.Masuzawa and H.Fujiwara, “Parallel selection algorithms for CGM and BSP with application to sorting ”, Proceedingsp of Joint Symposium on Parallel Processing (JSPP'99), pp.261-268, June 1999 .
  3. T.Ishimizu, A.Fujiwara, M.Inoue, T.Masuzawa and H.Fujiwara “Parallel algorithms for all nearest neighbors of binary images on the BSP model”, Proceedings of International Symposium on Parallel Architectures Algorithms, and Networks (I-SPAN'99), pp.394-399, June 1999 .

- 研究会報告

1. 石水隆，藤原暁宏，井上美智子，増澤利光，藤原秀雄,“ 選択問題を解く BSP モデルおよび BSP*モデル上の並列アルゴリズム ”, 信学技報, Vol. 97, No. 375 (COMP97-60), pp.1–8, November, 1997.
2. A.Fujiwara, T.Ishimizu, M.Inoue, T.Masuzawa and H.Fujiwara, “Parallel selection algorithms for CGM and BSP with application to sorting”, Technical Report of IEICE, Vol. 97, No. 627 (COMP97-103), pp.129–136, March, 1998.
3. T.Ishimizu, A.Fujiwara, M.Inoue, M.Toshimitsu and H.Fujiwara, “Parallel algorithms for finding all nearest neighbors of binary images on the BSP model”, Technical Report of IEICE, Vol. 98, No. 380 (COMP98-42), pp.33–40, October, 1998.

目 次

図 目 次

表 目 次

第 1 章

序論

高速計算や大規模データ処理の必要性から, 近年並列処理の研究が盛んに行われている. 並列処理とは複数のプロセッサを用いて 1 つの問題解決を行う処理であり, 負荷を分散させることにより処理のスピードアップが達成可能となる. 既に, 数千プロセッサを持つ並列計算機も実用化されており, 今後, 並列処理に関する研究は, ますます重要になるものと思われる. しかしながら, 既存の逐次処理とは異なり, 並列処理では通信のオーバーヘッドやデータの分割など並列処理独自の処理に対する考慮が必要である. このため, 並列処理に対しては, 今までの逐次処理における逐次アルゴリズムとは異なる並列処理用の並列アルゴリズムが必要である.

並列アルゴリズムは, 一般に並列計算機を抽象化した並列計算モデル上でその設計や解析が行われる. これまで並列計算モデルとしては, 共有メモリ型の PRAM(Parallel Random Access Machine)[12] や分散メモリ型のネットワークモデル [12](メッシュ接続やハイパーキューブ接続などによるもの) が主に用いられてきた. これらの並列計算モデルのほとんどは, すべてのプロセッサが 1 ステップごと完全に同期して動作する同期式並列計算モデルである. 同期式並列計算モデルは, 細粒度 SIMD(Single Instruction stream Multiple Data stream) 型の並列計算機に対応している. 初期の並列計算機の主流が細粒度 SIMD 型並列計算機であったことから, 従来同期式並列計算モデルを用いた並列アルゴリズムの設計や解析が盛んに行われてきた. しかし, 近年, 並列計算機の主流は, 細粒度 SIMD 型から中粒度 MIMD (Multiple Instruction stream Multiple Data stream) 型に変わりつつある. また, ネットワークに接続された多数のワークステーションを 1 つの仮想的な並列計算機として動作させることを可能にする PVM(Parallel Virtual Machine)[8] や MPI(Message-Passing Interface)[10] などによる並列処理も盛んである. これらの MIMD 型並列計算機や, ワークステーション群による仮想並列計

算機では, SIMD 型の処理とは異なり, 各プロセッサは各自の処理の多くを独立に行い, 必要に応じて他のプロセッサと通信により同期を取る. このような同期の緩い並列処理のための並列アルゴリズムの設計や解析には, 同期式並列計算モデルは不適当である. そこで, 近年, 同期の緩い並列処理に対応したモデルとして, BSP(Bulk-Synchronous Parallel) モデル [17], LogP モデル [3], CGM(Coarse Grained Multicomputer)[4] 等の非同期式並列計算モデルが提案されている. しかし, 非同期式並列計算モデル上での並列アルゴリズムの研究はまだ始まったばかりであり, 多くの基本的な問題に対して, 並列アルゴリズムがまだ提案されていないのが現状である.

本論文では上記の要求に対応した並列計算モデルである BSP(Bulk-Synchronous Parallel) モデル [17], およびその拡張モデルである BSP*モデル [1] を使用してアルゴリズムの提案を行う. BSP モデルは Valiant により提案された並列計算モデルであり, L, gという通信コストを同期周期及び通信命令実行時間を表す 2 つのパラメタにより表すことが可能になっている. また同期機構については, バリア同期を仮定することにより, 非常に緩い同期の処理に対応したモデルである. BSP* モデルでは, 通信パケットサイズを表すパラメタ Bを導入することにより, より実際に即したアルゴリズムの計算量の検証を可能にしている.

本論文では, これらのモデル上で選択問題を解く並列アルゴリズムおよび 2 値画像上の全最近点問題を解く並列アルゴリズムの提案を行う. 選択問題とは, 全順序関係を持つ n 個のデータの集合 Sと自然数 k $(1 \leq k \leq n)$ が与えられたときに, Sの中から k番目に小さい要素を求める問題であり, 多くのアプリケーションにおいて, 部分問題として利用されている基本問題である. 2 値画像上の全最近点問題とは, $n \times n$ の 2 値画像 Iが与えられたときに, Iの各黒画素に対して, 最も近い黒画素を求める問題であり, 画像処理, コンピュータグラフィックス, パターン認識等といった分野で利用される問題である.

本論文では, 並列アルゴリズムの尺度として以下の評価基準を用いる. 並列アルゴリズムのプロセッサ数と時間計算量の積が最速の逐次アルゴリズムの時間計算量と漸近的に等しいとき, その並列アルゴリズムは最適加速な並列アルゴリズムであるという. つまり, 最適加速な並列アルゴリズムとは, 漸近的にプロセッサ台数分のスピードアップを実現したアルゴリズムである,

3章では先に述べた選択問題についての並列アルゴリズムを考察する. 選択問題に対しては, $O(n)$ 時間の最適逐次アルゴリズム [16] が知られている. また, 選択問題に対する並列アルゴリズムとしては, 以下のものが知られている. Cole[2] は EREW PRAM 上で $O(\log n \log^* n)$ 時間[1], $\frac{n}{\log n \log^* n}$プロセッサ, CRCW PRAM 上で $O(\frac{\log n \log^* n}{\log \log n})$ 時間, $\frac{n \log \log n}{\log n \log^* n}$プロセッサで選択問題を解く最適加速な並列アル

[1] $\log^* n = \min\{i | \log^{(i)} n \leq 2\}$. ここで, $\log^{(i)} n = \log(\log^{(i-1)} n), \log^{(1)} n = \log n$ である.

ゴリズムを示した．BSP モデル上では Gerbessiotis ら [7] が内部計算時間 $O(\frac{n}{p} + L\log p)$，通信時間 $O(g\frac{n^{\frac{2}{3}+\delta}}{p} + L\log p)$ で停止し，確率 $1 - O(n^{1-\rho})$ で解を出力する p $(1 \leq p \leq n^{\frac{2}{3}} + \zeta)$ プロセッサの確率的並列アルゴリズムを示した．ρ, δ, ζ は $0 < \zeta < \delta < \frac{1}{3}$, $\rho > 1$ となる任意の定数である．BSP*モデル上では Bäumker ら [1] がプロセッサ数 $p = O(\frac{n}{\log^4 n})$，$c$ が任意の定数のとき $B \leq \sqrt{\frac{n}{p}}$に対して内部計算時間 $O(\frac{n}{p} + L\log p)$，通信時間 $O(\frac{g}{B}\sqrt{\frac{n}{p}} + (L+g)\log p)$ となる確率 $1 - \frac{1}{n^c}$の確率的並列アルゴリズムを示した．以上の様に，BSP モデル，および BSP*モデルでは，選択問題を解く確率的なアルゴリズムは提案されているが，決定性アルゴリズムは提案されていなかった．

本論文では選択問題を解く以下の 2 つの決定性並列アルゴリズムを示す．

(1) BSP モデル上で内部計算時間 $O(\frac{n}{p} + d\log p\log\log n + L\frac{\log p\log\log n}{\log(d+1)})$，通信時間 $O(g\frac{n}{p} + (gd+L)\frac{\log p\log\log n}{\log(d+1)})$ のアルゴリズム．

(2) BSP*モデル上で内部計算時間 $O(\frac{n}{p} + d\log p\log\log n + L\frac{\log p\log\log n}{\log(d+1)})$，通信時間 $O(g(\frac{n}{pB} + (\frac{n}{p})^{\frac{1}{7}}(\log p)^{\frac{6}{7}}) + (gd+L)\frac{\log p\log\log n}{\log(d+1)})$ のアルゴリズム．

ただし d は $1 \leq d \leq \log n$ を満たす任意の整数であり，かつプロセッサ数 p は $1 \leq p \leq \frac{n}{\log n}$である．(1) のアルゴリズムの計算量は，$g = d = O(1)$ のとき内部計算時間，通信時間が共に $O(\frac{n}{p} + L\log p\log\log n)$ となり，プロセッサ数が小さい場合は，最適加速なアルゴリズムとなる．(2) のアルゴリズムも，$d = O(1)$, $g \leq B = O((\frac{n}{p\log p})^{\frac{6}{7}})$ の場合，内部計算時間，通信時間が共に $O(\frac{n}{p} + L\log p\log\log n)$ となり，プロセッサ数が小さい場合は，最適加速なアルゴリズムとなる．

次に，4章では，前述の 2 値画像の全最近点問題について考察する．2 値画像上の全最近点問題に対しては，$O(n^2)$ 時間の最適逐次アルゴリズム [11] が知られている．また，2 次元実数座標上の全最近点問題については，CGM[4] 上で $O(\frac{n\log n}{p} + T_{sort}(n,p))$ 時間で解く並列アルゴリズム [4] が知られている．ここで，$T_{sort}(n,p)$ は，CGM 上でソートを解く時間である．

本論文では 2 値画像上の全最近点問題を解く 2 つの並列アルゴリズムを示す．1 つは重み付き距離に対する並列アルゴリズムであり，もう 1 つは L_p距離に対する並列アルゴリズムである．この 2 つのアルゴリズムの計算量は，共に BSP モデル上で p $(1 \leq p \leq n)$ プロセッサに対して，内部計算時間 $O(\frac{n^2}{p} + L)$，通信時間 $O(g\frac{n}{\sqrt{p}} + L)$ であり，また p $(n < p \leq n^2)$ プロセッサに対して，内部計算時間 $O(\frac{n^2}{p} + (d+L)\frac{\log\frac{p}{n}}{\log(d+1)})$，通信時間 $O(g\frac{n}{\sqrt{p}} + (gd+L)\frac{\log\frac{p}{n}}{\log(d+1)})$ となる．ただし，d は $1 \leq d \leq \frac{p}{n}$を満たす任意の整数である．このアルゴリズムの計算量は，$g = O(\frac{n}{\sqrt{p}})$ のとき通信時間が内部計算時間に等しい $O(\frac{n^2}{p} + L)$ となり，プロセッサ数が小さい場合は，最適加速なアルゴリズムとなる．

第 2 章

準備

2.1. BSP モデルおよび BSP*モデル

2.1.1 定義

BSP(Bulk-Synchronous Parallel) モデル [17] は Valiant によって提案された非同期式並列計算モデルであり，以下の構成要素から成る．

- 局所メモリを持つ複数のプロセッサ (本論文中ではプロセッサ数を p とし，各プロセッサを P_i $(0 \leq i \leq p-1)$ または $P_{i,j}$ $(0 \leq i \leq \sqrt{p}-1, 0 \leq j \leq \sqrt{p}-1)$ で表す．)
- プロセッサ間の 1 対 1 メッセージ通信を行う完全結合網
- プロセッサ間の同期を実現するための同期機構

BSP モデル上での並列アルゴリズムは，各プロセッサが実行するプログラムにより表される．各プロセッサが実行するプログラムは**スーパーステップ**の列からなる．各スーパーステップにおいて，各プロセッサは内部計算命令の列および送信命令，受信命令の列を割り当てられ，各プロセッサは各命令を非同期に実行する．全てのプロセッサが割り当てられた命令を実行後，全プロセッサ間でバリア同期[1]を取り，そのスーパーステップを終了する．メッセージの受信については，

[1]バリア同期とは，協調して動作する多数のプロセッサの歩調を合わせることを目的とした同期プリミティブである．バリア同期を実行して同期を取る場合，全てのプロセッサがバリアに到達するまでどのプロサッサも実行を継続できず，動作を封鎖される．

各スーパーステップ中ので送信されたメッセージは同一のスーパーステップの通信で受信されるが，その受信メッセージは次のスーパーステップ以降でしか利用できないと仮定する．

以下では簡単のために，各スーパーステップは内部計算命令のみ, あるいは送信命令および受信命令のみからなるとし，内部計算命令のみからなるスーパーステップを内部計算スーパーステップ，送信命令および受信命令のみからなるスーパーステップを通信スーパーステップと呼び，その実行時間をそれぞれ，内部計算時間，通信時間と呼ぶ．内部計算スーパープテップと通信スーパーステップを分けることにより，アルゴリズムの設計および評価が容易となる．一方，両者を合わせてもアルゴリズムの実行時間は，漸近的には変わらない．

BSP モデルは以下の 2 つのパラメタにより，具体的なネットワーク構造やメッセージ配送の仕組みを抽象化している．

- L:バリア同期周期
- g ($\leq L$):1 個の送信命令または受信命令の実行に必要な時間

BSP モデル上の並列アルゴリズムの基本命令の実行時間について，以下のように仮定されている．

- 各プロセッサは 1 単位時間に 1 内部計算命令を局所メモリにのみ基づいて実行する．
- メッセージ 1 個の送信命令または受信命令の実行は g単位時間で行なわれる．ただし，1 メッセージは 1 語からなるものとする．
- あるスーパーステップにおいて，全てのプロセッサで命令の実行を終了してから L 時間以内にバリア同期が取られ，次のスーパーステップの実行に移る．よって，ある内部計算スーパーステップにおいて，各プロセッサが高々w個の内部計算命令割り当てられた場合，そのスーパーステップの実行には $O(w + L)$ 時間，ある通信スーパーステップにおいて，各プロセッサに高々h 個の送信命令または受信命令を割り当てられた場合，そのスーパーステップの実行には $O(gh + L)$ 時間かかる．

  図 2.1にスーパーステップの動作例を示す.

BSP*モデル [1] は Bäumker らによって提案された BSP モデルの拡張モデルであり, BSP モデルのパラメタに加えて以下の様な通信パケットのサイズを表すパラメタを持つ．

- B (≥ 1):通信パケットの最小サイズ

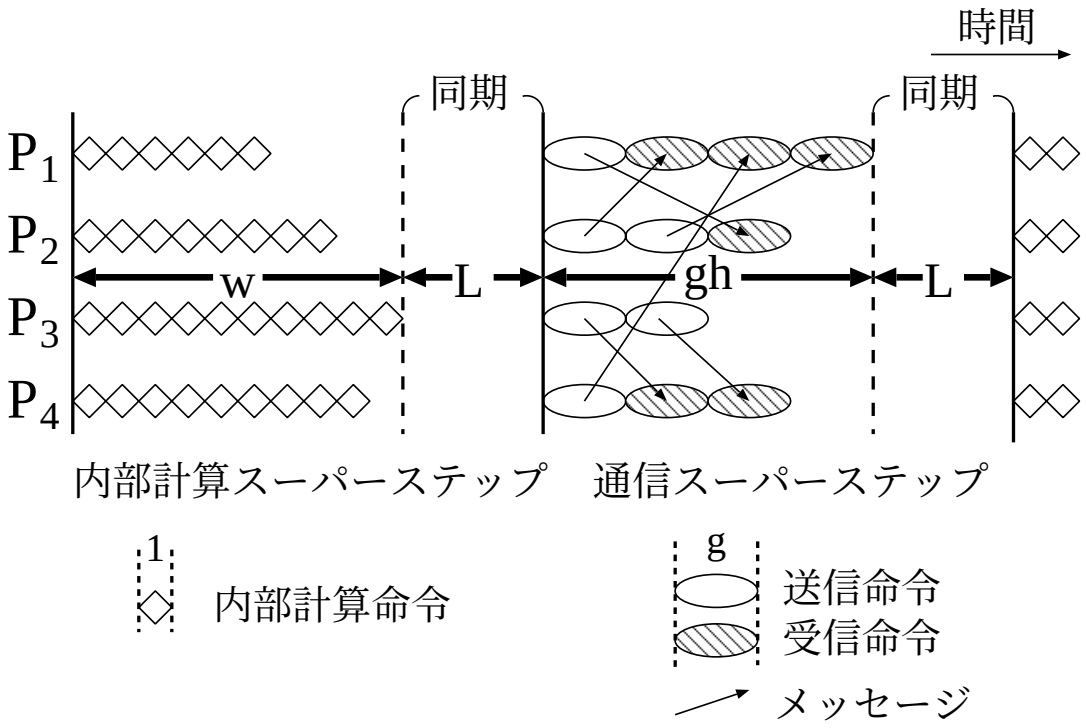

図 2.1 スーパーステップの動作例

この拡張は，多くの並列計算機において，メッセージがある特定のサイズの通信パケットとして伝達されるという事実に基づいている．

また，この拡張による BSP モデルからの変更点は以下の通りである．

- 同じプロセッサに対する s 語のメッセージをサイズ s のメッセージとして送信または受信できる．
- サイズ s のメッセージの送信命令または受信命令の実行は $g\lceil \frac{s}{B} \rceil$ 単位時間で行われる．

上記の変更点より，各プロセッサが高々h 個のプロセッサにサイズ $s_1, s_2, ..., s_h$であるメッセージの送信命令または受信命令からなる通信スーパーステップの時間計算量は，$O(g \sum_{i=1}^{h} \lceil \frac{s_i}{B} \rceil + L)$ 時間となる．

2.1.2 BSP モデル及び BSP*モデル上の基本アルゴリズム

本論文中で提案する並列アルゴリズムでは，ブロードキャスト操作，接頭部演算操作，ソーティング操作を行なう BSP モデル上の並列アルゴリズムを利用する．各操作は BSP モデルおよび BSP*モデル上では以下の様に定義される．以下の定義では各要素 a，a_i，b_i等はいずれも 1 語であるとする．

定義 1 (ブロードキャスト操作) ブロードキャスト操作とは，ある 1 プロセッサが保持する要素を全てのプロセッサに送信する操作である．

入力: 値 a（プロセッサ P_0が保持する．）

出力: 全てのプロセッサが値 a を保持する． □

定義 2 (接頭部演算操作) $\circ$を結合則を満たす *2* 項演算子とする．接頭部演算操作とは，各プロセッサが *1* 要素ずつ保持する要素列において，列の先頭からその要素までの要素に対して *2* 項演算子を施した値を出力する操作である．

入力: n 個の要素列 $(a_0, a_1, ..., a_{n-1})$．各 P_i $(0 \leq i \leq p-1)$ が $(a_{\lceil i\frac{n}{p} \rceil}, a_{\lceil i\frac{n}{p} \rceil+1}, ..., a_{\lceil (i+1)\frac{n}{p} \rceil-1})$ を保持する．

出力: 各 j $(0 \leq j \leq n-1)$ について, $b_j = a_0 \circ a_1 \circ \cdots \circ a_j$を満たす要素列 $(b_0, b_1, \cdots, b_{n-1})$．各 P_i $(0 \leq i \leq p-1)$ が $(b_{\lceil i\frac{n}{p} \rceil}, b_{\lceil i\frac{n}{p} \rceil+1}, ..., b_{\lceil (i+1)\frac{n}{p} \rceil-1})$ を保持する．

□

表 2.1 BSP モデル上の基本操作に対する計算量

操作	時間計算量	プロセッサ数	文献
ブロードキャスト	$T_C : O((gd+L)\frac{\log p}{\log(d+1)})$	$1 \leq p \leq n$	[17]
接頭部演算	$T_I : O((d+L)\frac{\log p}{\log(d+1)} + \frac{n}{p})$ $T_C : O((gd+L)\frac{\log p}{\log(d+1)})$	$1 \leq p \leq n$	[17]
ソーティング	$T_I : O(\frac{n \log n}{p} + L\frac{\log n}{\log \frac{n}{p}})$ $T_C : O((g\frac{n}{p} + L)\frac{\log n}{\log \frac{n}{p}})$	$1 \leq p \leq n$	[9]

T_I:内部計算時間，T_C:通信時間，d :$1 \leq d \leq p$ の任意の定数

定義 3 (ソーティング操作) ソーティング操作とは以下の様に要素を昇順に並べ替える操作である．

入力: 全順序関係を持つ n 個の要素集合 $A = \{a_0, a_1, ..., a_{n-1}\}$．各 P_i $(0 \leq i \leq p-1)$ が $\{a_{\lceil i\frac{n}{p} \rceil}, a_{\lceil i\frac{n}{p} \rceil+1}, ..., a_{\lceil (i+1)\frac{n}{p} \rceil-1}\}$ を保持する．

出力: A のソート列 $(b_0, b_1, ..., b_{n-1})$．各 P_i $(0 \leq i \leq p-1)$ が $(b_{\lceil i\frac{n}{p} \rceil}, b_{\lceil i\frac{n}{p} \rceil+1}, ..., b_{\lceil (i+1)\frac{n}{p} \rceil})$ を保持する．

□

表 2.1にブロードキャスト操作，接頭部演算操作，ソーティング操作に関する BSP モデル上の既知の結果を示す．

BSP モデルで計算量が $f(n, p, g, L)$ であるアルゴリズムは，BSP*モデル上で同じ計算量 $f(n, p, g, L)$ で動作することが可能である，すなわち，BSP*モデルにも表 2.1の結果が適用できる．

また，表 2.1中の Goodrich のソーティングアルゴリズム [9] は，以下の様な特性により，BSP*モデル上では計算量が改善される．

Goodrich のソーティングアルゴリズム [9] は，各プロセッサは 1 スーパーステップで $O(\frac{n}{p})$ 個の要素の送信，および受信を行い，また，$O(\frac{\log n}{\log \frac{n}{p}})$ スーパーステップで終了するので，BSP モデル上では，通信時間が $O((g\frac{n}{p} + L)\frac{\log n}{\log \frac{n}{p}})$ となっている．しかしながら，このアルゴリズムでは，各プロセッサが 1 スーパーステップで送信するメッセージの送信先，および受信するメッセージの送信元のプロセッサ数は高々$2(\frac{n}{p})^{\frac{1}{7}}$である．BSP*モデルでは，1 スーパーステップにおいて，k個の

プロセッサに s_i個 $(1 \leq i \leq k)$ ずつ送信するためには，$O(g\sum_{i=1}^{k}\lceil\frac{s_i}{B}\rceil + L)$ の通信時間しか必要としない．したがって各プロセッサの 1 スーパーステップの通信時間は $O(\sum_{i=1}^{(\frac{n}{p})^{\frac{1}{7}}}\lceil\frac{s_i}{B}\rceil + L)$ となる．また，各プロセッサは，1 スーパーステップで $O(\frac{n}{p})$ 個の要素の送受信を行うので，$\sum_{i=1}^{2(\frac{n}{p})^{\frac{1}{7}}} s_i = O(\frac{n}{p})$ である．したがって，このソーティングアルゴリズムの 1 スーパーステップの通信時間は，

$$
\begin{aligned}
& O(g(\sum_{i=1}^{2(\frac{n}{p})^{\frac{1}{7}}}\lceil\frac{s_i}{B}\rceil) + L) \\
= & O(g(\sum_{i=1}^{2(\frac{n}{p})^{\frac{1}{7}}}(\frac{s_i}{B}+1)) + L) \\
= & O(g(\frac{n}{pB} + (\frac{n}{p})^{\frac{1}{7}}) + L)
\end{aligned}
$$

となる．

したがって, 以下の補題が得られる．

補題 1 ソーティング操作は，BSP^*モデル上で内部計算時間 $O(\frac{n\log n}{p} + L\frac{\log n}{\log\frac{n}{p}})$, 通信時間 $O((g(\frac{n}{pB} + (\frac{n}{p})^{\frac{1}{7}}) + L)\frac{\log n}{\log\frac{n}{p}})$ で実行できる． □

第 3 章

選択問題を解く並列アルゴリズム

3.1. 定義

全順序関係を持つ要素集合 $A = \{a_0, a_1, ..., a_{n-1}\}$ に対して，関数 $rank(a_i, A)\ (1 \leq i \leq n)$ を

$$rank(a_i, A) = |\{a \in A | a < a_i\}|$$

と定義する．なお，簡単のために任意の $i, j\ (0 \leq i < j \leq n-1)$ について $a_i \neq a_j$ とする．このとき，選択問題は以下の様に定義される．

定義 4 (選択問題) 選択問題は全順序関係を持つ n 個の要素の集合 A と整数 $k\ (1 \leq k \leq n)$ が与えられたときに，以下の様に A の中で k番目に小さい要素を求める問題である．

入力: 全順序関係を持つ n 個の要素集合 $A = \{a_0, a_1, ..., a_{n-1}\}$，及び，整数 $k\ (1 \leq k \leq n)$．各 $P_i\ (0 \leq i \leq p-1)$ が $\{a_{\lceil i\frac{n}{p} \rceil}, a_{\lceil i\frac{n}{p} \rceil + 1}, ..., a_{\lceil (i+1)\frac{n}{p} \rceil - 1}\}$ を，P_{p-1}が kを保持する．

出力: ある 1 つのプロセッサが $rank(a, A) = k$を満たす要素 $a \in A$ を出力する．

□

3.2. BSP モデル上のアルゴリズム

3.2.1 アルゴリズムの概要

選択問題を解くアルゴリズムは Vishkin によって提案された EREW PRAM 上の並列アルゴリズム [18] を基にしている．

選択問題はソーティング操作を用いて解くことが出来るのは明らかである．しかし，一般にソーティグ操作の計算量は，選択問題を解く計算量よりも大きくなる．BSP モデル上の並列アルゴリズムの場合も，表 2.1のソーティングアルゴリズム [9] により, 内部計算時間 $O(\frac{n\log n}{p}+L\frac{\log n}{\log \frac{n}{p}})$, 通信時間 $O((g\frac{n}{p}+L)\frac{\log n}{\log \frac{n}{p}})$ でソートを行い, 選択問題を解くことができるが, この計算量では選択問題に対しては最適加速なアルゴリズムとはならない．そこで本アルゴリズムでは，Viskin[18] のアルゴリズムの方針を用いて, k番目では有り得ない要素を以下に述べる操作により取り除き，対象要素数を$\frac{n}{\log n}$まで減少させた後にソーティング操作を行う．このことにより，ソーティングのための計算量を減らすことができ, 最適加速なアルゴリズムとなる.

要素数を n から$\frac{n}{\log n}$に減らすための操作は，以下のフェーズを反復することによって行なう．まず，対象要素の中から適当な要素 m を選び，対象要素集合を m よりも小さいの要素からなる集合，および大きい要素からなる集合に分割する．これらの 2 集合について，どちらに k番目に小さい要素が含まれているかを計算し，k番目の要素が含まれていない方の集合を対象要素から除外する．また，m が k番目の要素であればそれを出力して停止する．本アルゴリズムでは，各プロセッサが保持する要素の中央値を求め，その得られた中央値集合の更に中央値を求め，上記の要素 m として用いる．ここで，要素集合 $A=\{a_0,a_1,...,a_{n-1}\}$ の中央値とは，$rank(a,A)=\lceil\frac{n}{2}\rceil$ を満たす要素 $a\in A$ である．中央値の中央値を分割の基準として利用することにより, 対象の要素数を 1 回のフェーズで $\frac{1}{c}$ (c は $c>1$ を満たす定数) 以下にすることができるので, 後述の通り $O(\log\log n)$ のフェーズの反復により, 対象要素数は $\frac{n}{\log n}$以下となる.

3.2.2 分配操作

BSP モデルは分散メモリ型の並列計算モデルであるので, 各プロセッサがどの要素を保持するかを工夫する必要がある．本アルゴリズムでは，各プロセッサが保持する要素数を均等にするために, 以下の分配操作を使用する．

定義 5 (分配操作) n 個の要素が各 $P_i(0\le i\le p-1)$ に n_i個ずつ保持されているとする．ただし，$n=\sum_{j=0}^{p-1}n_j$である．分配操作とは, 以下の様に 各プロセッサが

高々$\lceil \frac{n}{p} \rceil$ 個，少なくとも $\lfloor \frac{n}{p} \rfloor$ 個の要素を保持する様に要素を分配する操作である．

入力: n 要素からなる集合 A．各 P_i $(0 \leq i \leq p-1)$ は互いに素である集合 A の部分集合 $A_i(|A_i| = n_i, A_0 \cup A_1 \cup ... \cup A_{p-1} = A)$ を保持する．

出力: n 要素からなる集合 A．各 P_i $(0 \leq i \leq p-1)$ は互いに素である A の部分集合 $A'_i(A'_0 \cup A'_1 \cup ... \cup A'_{p-1} = A)$ を保持する．ただし，各 A_iは $\lfloor \frac{n}{p} \rfloor \leq |A'_i| \leq \lceil \frac{n}{p} \rceil$ を満たすものとする．

□

以下に分配操作のアルゴルズムを示す．

分配操作アルゴリズム

ステップ 1 接頭部演算操作を用いて各 i $(0 \leq i \leq p-1)$ に対し，$s_i = \sum_{j=0}^{i} n_j$を計算する．

ステップ 2 各 P_i $(0 \leq i \leq p-1)$ が保持する n_i個の要素からなる要素集合を $A_i = \{a_{s_i - n_i}, a_{s_i - n_i}, ..., a_{s_i - 1}\}$ とする．各 P_i は保持する各要素 a_j $(s_i - n_i \leq j \leq s_i - 1)$ を $\lceil i' \frac{n}{p} \rceil \leq j \leq \lceil (i'+1) \frac{n}{p} \rceil - 1$ を満たす $P_{i'}$に送信する．

ステップ 3 各 P_i $(0 \leq i \leq p-1)$ において *(2)* で受信した要素の集合を A'_iとする．

補題 2 $n_{max} = \max\{n_0, n_1, ..., n_{p-1}\}$ とする．任意の定数 d $(1 \leq d \leq p)$ に対して，先の分配操作は *BSP* モデル上で，
内部計算時間 $O(n_{max} + (d+L)\frac{\log p}{\log(d+1)})$
通信時間 $O(gn_{max} + (gd+L)\frac{\log p}{\log(d+1)})$
で実行できる．

(証明) p 要素の接頭部和演算操作は, 表 2.1のアルゴリズム [9] を用いて,
内部計算時間 $O((d+L)\frac{\log p}{\log(d+1)})$，通信時間 $O((gd+L)\frac{\log p}{\log(d+1)})$
で実行できる．また, ステップ 2 において, 各プロセッサは高々n_{max}個の要素を送信し，高々$\lceil \frac{n}{p} \rceil (\leq n_{max})$ 個の要素を受信する．よって通信時間は $O(gn_{max} + L)$ である．また要素の送信先の決定に要する内部計算時間は, $O(n_{max} + L)$ である. □

3.2.3 アルゴリズム *Selection*

ここでは BSP モデル上で選択問題を解く p $(1 \leq p \leq \frac{n}{\log n})$ プロセッサのアルゴリズム *Selection* を示す．

アルゴリズム *Selection*

入力: 全順序関係を持つ n 個の要素集合 $A = \{a_0, a_1, ..., a_{n-1}\}$，及び，整数 k $(1 \leq k \leq n)$．各 P_i $(0 \leq i \leq p-1)$ が $\{a_{\lceil i\frac{n}{p}\rceil}, a_{\lceil i\frac{n}{p}\rceil+1}, ..., a_{\lceil (i+1)\frac{n}{p}\rceil-1}\}$ を，P_{p-1}が kを保持する．

出力: ある 1 つのプロセッサが $rank(a, A) = k$を満たす要素 $a \in A$ を出力する．

ステップ 1 各 $P_i(0 \leq i \leq p-1)$ において，$s := n$, $k' := k$とする．(s はアルゴリズム中の対象要素数を，k'は見つけ出す要素のランクを表す．)

ステップ 2 $s > \frac{n}{\log n}$ ならば，以下のフェーズ *(1)*∼*(6)* を繰り返す．

(1) 各 P_i $(0 \leq i \leq p-1)$ 上において, A_iの中央値 m_iを求める．

(2) プロセッサ全体により, 中央値集合 $\{m_0, m_1, ..., m_{p-1}\}$ をソートし, 中央値集合の中央値 (m とする) を計算する．m を保持するプロセッサは, m をすべてのプロセッサにブロードキャストする．

(3) 各 P_i $(0 \leq i \leq p-1)$ 上において, A_iの要素を以下のような 2 つの部分集合 A_i^1, A_i^2に分割する．
$A_i^1 = \{x \in A_i | x < m\}, A_i^2 = \{x \in A_i | x > m\}$

(4) 各 P_i $(0 \leq i \leq p-1)$ 上において, A_i^1のサイズ $|A_i^1|$ を計算する．次に, プロセッサ全体により, その和 $s^1 = \sum_{j=0}^{p-1} |A_j^1|$ を計算し，s^1をすべてのプロセッサにブロードキャストする．

(5) 各 P_i $(0 \leq i \leq p-1)$ 上において, 以下を実行する．

- ($k' < s^1 + 1$ の場合) $A_i := A_i^1$, $s := s^1$とする．
- ($k' > s^1 + 1$ の場合) $k' := k' - (s^1 + 1)$ とし, $A_i := A_i^2$，$s := s - (s^1 + 1)$ とする．
- ($k' = s^1 + 1$ の場合) P_0は m を出力し, アルゴリズムを停止する．

(6) 各プロセッサが保持する要素 A_i $(0 \leq i \leq p-1)$ に対し，分配操作を行う．分配操作後の各プロセッサが保持する要素を A_iとする．

ステップ 3 全てのプロセッサを用いて, 要素集合 $A_0 \cup A_1 \cup ... \cup A_{p-1}$をソートし, k'番目の要素を保持するプロセッサがその要素を出力する．

3.2.4 正当性の証明

アルゴリズムが停止すれば，その出力が選択問題の解であることはアルゴリズムより明らかであるので, ここではアルゴリズムの停止性のみを示す．このために，ステップ 2 の繰り返し回数が高々$O(\log\log n)$ であることを示す．

以下ではステップ 2 の (1) から (6) までの 1 回の実行を 1 反復フェーズと呼ぶ．

補題 3 *Selection* の各反復フェーズ中の *(4)* 終了時，$\frac{1}{6}s - 1 < s^1 < \frac{5}{6}s$ が成り立つ．

(証明) A をステップ 2 の各反復フェーズの開始時点の要素集合 $A = A_1 \cup A_2 \cup ... \cup A_p$ とする．このとき，$s = |A|$ である．

各反復フェーズの (4) 終了時の s^1に対して，$s^1 = rank(m, A) - 1$ であるので，補題を示すには$\frac{1}{6}s < rank(m, A) < \frac{5}{6}s + 1$ を示せばよい．

s の値により場合分けを行なう．

(i) $s < 3p$ のとき

m は $\{m_0, m_1, ..., m_{p-1}\}$ の中央値であるので，A の要素のうち m 以下の要素は少なくとも $\lceil \frac{p}{2} \rceil$ 個存在する．$s < 3p$ より，$rank(m, A) \geq \lceil \frac{p}{2} \rceil > \frac{s}{6}$が成り立つ．

同様に，m より大きい要素は少なくとも $\lfloor \frac{p}{2} \rfloor$ 個存在する．したがって $s - rank(m, A) \geq \lfloor \frac{p}{2} \rfloor > \frac{s}{6} - 1$が成り立ち，$rank(m, A) \leq \frac{5}{6}s$ となる．

(ii) $s \geq 3p$ のとき

m は $\{m_0, m_1, ..., m_{p-1}\}$ の中央値であるので，$\{m_0, m_1, ..., m_{p-1}\}$ の要素のうち m 以下の要素は $\lceil \frac{p}{2} \rceil$ 個存在する．また，各 m_i $(0 \leq i \leq p-1)$ は A_iの中央値であるので A_iの要素のうち m_i以下の要素は $\lceil \frac{1}{2} \lfloor \frac{s}{p} \rfloor \rceil$ 個存在する．したがって A の要素のうち m 以下の要素は少なくとも $\lceil \frac{p}{2} \rceil \lceil \frac{1}{2} \lfloor \frac{s}{p} \rfloor \rceil$ 個存在する．ここで $s \geq 3p$ より $\lceil \frac{p}{2} \rceil \lceil \frac{1}{2} \lfloor \frac{s}{p} \rfloor \rceil > \frac{p}{2}(\frac{s}{2p} - \frac{1}{2}) \geq \frac{p}{2}(\frac{s}{2p} - \frac{s}{6p}) = \frac{s}{6}$ が成り立つので，$rank(m, A) > \frac{s}{6}$が成り立つ．

同様に，$\{m_0, m_1, ..., m_{p-1}\}$ の要素のうち m 以上の要素は $\lfloor \frac{p}{2} \rfloor + 1$ 個存在すること，及び，B_i $(0 \leq i \leq p-1)$ の要素のうち m_i以上の要素は $\lfloor \frac{1}{2} \lfloor \frac{s}{p} \rfloor \rfloor + 1$ 個存在することから $s - rank(m, A) > \frac{s}{6} - 1$が成り立ち，$rank(m, A) \leq \frac{5}{6}s$ となる．

以上より，$\frac{s}{6} < rank(m, A) < \frac{5s}{6} + 1$ となる． □

補題 4 アルゴリズム *Selection* のステップ *2* の反復フェーズ数は $O(\log\log n)$ である．

(証明) 補題 3より，第 j番目の反復フェーズ開始時点の要素集合 A の要素数 s は $s < n(\frac{5}{6})^{j-1}$となる．よって $O(\log\log n)$ 回の反復により $s \leq \frac{n}{\log n}$となる． □

3.2.5 計算量

定理 1 任意の整数 d $(1 \leq d \leq \log n)$ に対して，アルゴリズム *Selection* は *BSP* モデル上で,

内部計算時間 $O(\frac{n}{p} + d\log p\log\log n + L(\frac{\log p\log\log n}{\log(d+1)}))$

通信時間 $O(g\frac{n}{p} + (gd+L)\frac{\log p\log\log n}{\log(d+1)d})$

で選択問題を解く.

(証明) アルゴリズムの各ステップの計算量を評価する.

ステップ 1 は各 P_iにおいて定数個の内部計算であるので，内部計算時間 $O(L)$ で実行できる．また，ステップ 3 は高々$\frac{n}{\log n}$個の要素のソーティング操作であり, 表 2.1のソーティングアルゴリズム [9] を用いると,

$$\text{内部計算時間 } O\left(\frac{\frac{n}{\log n}\log\frac{n}{\log n}}{p} + L\frac{\log\frac{n}{\log n}}{\log\frac{\frac{n}{\log n}}{p}}\right)$$
$$= O\left(\frac{n}{p} + L\frac{\log\frac{n}{p\log n}+\log p}{\log\frac{n}{p\log n}}\right) = O(\frac{n}{p} + L\log p)$$
$$\text{通信時間 } O\left(\left(g\frac{\frac{n}{\log n}}{p} + L\right) \times \frac{\log\frac{n}{\log n}}{\log\frac{\frac{n}{\log n}}{p}}\right) = O(g\frac{n}{p} + L\log p)$$

で実行できる.

したがって以下ではステップ 2 の計算量のみについて検証する．まず 1 反復フェーズの計算量を評価する．(s は各反復フェーズ開始時点の要素数である)

(1) の各プロセッサ上の中央値の計算は, 既知の逐次アルゴリズム [16] を用いて内部計算時間 $O(\frac{s}{p} + L)$ で求められる．また, (2) のソーティング操作を除いたその他のステップは, 前述のブロードキャスト操作, 接頭部和演算, 分配操作, 及び, $O(\frac{s}{p} + L)$ 時間の内部計算により実現されているので,

内部計算時間 $O(\frac{s}{p} + (d+L)\frac{\log p}{\log(d+1)}))$，通信時間 $O(g\frac{s}{p} + (gd+L)\frac{\log p}{\log(d+1)})$

で実行できる.

(2) のソーティング操作は, 以下のように実現する. 各プロセッサ 1 つずつの要素をすべてのプロセッサを使って表 2.1のアルゴリズム [9] によりソートすると, $O(L\log p)$ の通信時間がかかることになり, 効率が悪い. そこで, 最初に, 要素を $\lceil\frac{p}{d}\rceil$ 個のプロセッサ[1] に集め, 少ないプロセッサ数によりソートを行うことにより, 通信時間を減らす. 具体的には, 以下のような操作を行う.

[1] d はブロードキャスト演算等の d と同じ値を用いる.

(2-1) 各 P_i $(0 \leq i \leq p-1)$ は m_iを $P_{\lfloor \frac{i}{d} \rfloor}$に送信する.

(2-2) P_i $(0 \leq i \leq \frac{p}{d}-1)$ を用いて，$\{m_0, m_1, ..., m_{p-1}\}$ をソートする.

(2-3) $\{m_0, m_1, ..., m_{p-1}\}$ の中央値を保持するプロセッサが, その中央値をブロードキャストする.

以上の操作により, (2) の計算量は, 表 2.1のソーティングアルゴリズム [9] を, 要素数 p, プロセッサ数 $\lceil \frac{p}{d} \rceil$ で実行した計算量となるので,

内部計算時間 $O\left(\frac{p \log p}{\lceil \frac{p}{d} \rceil} + L \times \frac{\log p}{\log \frac{p}{\lceil \frac{p}{d} \rceil}}\right) = O(d \log p + L\frac{\log p}{\log(d+1)})$

通信時間 $O\left(\left(g\frac{p}{\lceil \frac{p}{d} \rceil} + L\right) \times \frac{\log p}{\log \frac{p}{\lceil \frac{p}{d} \rceil}}\right) = O((gd+L)\frac{\log p}{\log(d+1)})$

となる.

以上よりステップ 2 の 1 反復フェーズは

内部計算時間 $O(\frac{s}{p} + d \log p + L\frac{\log p}{\log(d+1)})$，通信時間 $O(g\frac{s}{p} + (gd+L)\frac{\log p}{\log(d+1)})$

である.

以下で, ステップ 2 全体の計算量を考える. ステップ 2 の j回目の反復フェーズの開始時点の A の要素数 s を $s^{(j)}$とする. 補題 3より，$s^{(j)} < (\frac{5}{6})^{j-1}n$ であり, また, 補題 4より，ステップ 2 は $O(\log \log n)$ フェーズ繰り返されるので，ステップ 2 の計算量は,

内部計算時間 $O(\sum_{j=1}^{\log \log n}(\frac{s^{(j)}}{p} + d \log p + L\frac{\log p}{\log(d+1)}))$

$= O(\sum_{j=1}^{\log \log n}((\frac{5}{6})^{j-1}\frac{n}{p} + d \log p + L\frac{\log p}{\log(d+1)}))$

$= O(\frac{n}{p} + d \log p \log \log n + L\frac{\log p \log \log n}{\log(d+1)})$

通信時間 $O(\sum_{j=1}^{\log \log n}(g\frac{s^{(j)}}{p} + (gd+L)\frac{\log p}{\log(d+1)}))$

$= O(\sum_{j=1}^{\log \log n}(g(\frac{5}{6})^{j-1}\frac{n}{p} + (gd+L)\frac{\log p}{\log(d+1)}))$

$= O(g\frac{n}{p} + (gd+L)\frac{\log p \log \log n}{\log(d+1)})$

となる.

仮定する $d \leq \log n$ の範囲[2]においては，$\log \log n \geq \log d$ であり，ステップ 3 のソートの計算量よりステップ 2 の計算量の方が漸近的に大きいので，アルゴリズム全体の計算量も上記の計算量となる. □

定理 1 より，$selection$ は $p \log p \leq \frac{n \log d}{L \log \log n}$，$g = O(1)$ のとき内部計算時間，通信時間ともに $O(\frac{n}{p})$ となり，最適加速となることが示される.

[2]プロセッサ数が多い場合でも，内部計算時間を n の対数以下にするために仮定.

3.3. BSP*モデル上のアルゴリズム

3.3.1 アルゴリズムの概要

BSP*モデル上で選択問題を解く並列アルゴリズム *Selection**は前述の *Selection* における分配操作を BSP*用に改良したアルゴリズムである．

BSP*モデルでは異なるプロセッサにそれぞれサイズ 1 のメッセージを s 個送信するのに gs 単位時間要するが，同一のプロセッサにサイズ s のメッセージ 1 個の送信は $g\lceil\frac{s}{B}\rceil$ 時間で実行できる．すなわち，BSP*モデルではサイズの小さいメッセージを多数のプロセッサに対し送信または受信することは非効率的である．*Selection* で用いられる分配操作では各プロセッサは高々サイズ 1 のメッセージを $\lceil\frac{n}{p}\rceil$ 個のプロセッサから受信することがある．

*Selection**では分配操作の代りに各プロセッサの保持する要素数をある程度均等にする疑似分配操作を用いることにより通信時間を改善する．

3.3.2 疑似分配操作

定義 6 (疑似分配操作) n 個の要素が各 $P_i(0 \leq i \leq p-1)$ に n_i個ずつ保持されているとする．ただし，$n = \sum_{j=0}^{p-1} n_j$である．疑似分配操作とは，以下の様に各プロセッサが高々$\frac{1}{2}\frac{n}{p} + \lceil\frac{n}{p}\rceil$ 個, 少なくとも 1 個の要素を保持する様に要素を分配する操作である．

入力: n 要素からなる集合 A．各 P_i $(0 \leq i \leq p-1)$ は互いに素である集合 A の部分集合 $A_i(|A_i| = n_i, A_0 \cup A_1 \cup ... \cup A_{p-1} = A)$ を保持する．

出力: n 要素からなる集合 A．各 P_i $(0 \leq i \leq p-1)$ は互いに素である A の部分集合 $A'_i(A'_0 \cup A'_1 \cup ... \cup A'_{p-1} = A)$ を保持する．ただし，各 A'_iは $1 \leq |A'_i| \leq \frac{1}{2}\frac{n}{p} + \lceil\frac{n}{p}\rceil$ を満たすものとする．

□

以下に BSP*モデル上での疑似分配操作のアルゴルズムを示す．

疑似分配操作アルゴリズム

ステップ 1 接頭部演算操作を用いて各 i $(0 \leq i \leq p-1)$ に対し，$s_i = \sum_{j=1}^{i} n_j$を計算する．

ステップ 2 各 P_i $(0 \leq i \leq p-1)$ が保持する n_i 個の要素を $\{a_{s_i-n_i}, a_{s_i-n_i}, ..., a_{s_i-1}\}$ とする．各 P_i において以下の操作を行う．

- $(n_i \geq \frac{1}{2}\frac{n}{p}$ の場合) 各 P_i は保持する各要素 a_j $(s_i - n_i \leq j \leq s_i - 1)$ を $\lceil i'\frac{n}{p} \rceil \leq j \leq \lceil (i'+1)\frac{n}{p} \rceil - 1$ を満たす $a_{i'}$ に送信する．
- $(n_i < \frac{1}{2}\frac{n}{p}$ の場合) 各 P_i $(0 \leq i \leq p-1)$ は保持する各要素 a_j $(s_i - n_i \leq j \leq s_i - 1)$ のうち, ある i' に対し $j = \lceil i'\frac{n}{p} \rceil$ となる a_j を $P_{i'}$ に送信する．

ステップ 3 各 P_i において，(2) で受信した要素と未送信の要素からなる集合を A'_i とする．

補題 5 疑似分配操作アルゴリズム実行後，各 P_i $(0 \leq i \leq p-1)$ において $1 \leq |A'_i| \leq \frac{1}{2}\frac{n}{p} + \lceil \frac{n}{p} \rceil$ が成り立つ．

(証明) 疑似分配操作において，各 P_i は $\{a_{\lceil i\frac{n}{p} \rceil}, a_{\lceil i\frac{n}{p} \rceil + 1}, ..., a_{\lceil (i+1)\frac{n}{p} \rceil - 1}\}$ の部分集合を受信し，このうち $a_{\lceil i\frac{n}{p} \rceil}$ は必ず受信する．また，未送信の要素数は $\frac{1}{2}\frac{n}{p}$ より少ない．したがって，$1 \leq |A'_i| \leq \frac{1}{2}\frac{n}{p} + \lceil \frac{n}{p} \rceil$ が成り立つ． □

補題 6 $n_{max} = \max\{n_0, n_1, ..., n_{p-1}\}$ とする．任意の定数 d $(1 \leq d \leq p)$ に対して，*BSP**モデル上での疑似分配操作の計算量は

内部計算時間 $O(n_{max} + (d+L)\frac{\log p}{\log(d+1)})$

通信時間 $O(g\frac{n_{max}}{B} + (gd+L)\frac{\log p}{\log(d+1)})$

である．

(証明) ステップ 1，ステップ 3 は BSP モデル上の分配操作と同じで, 内部計算時間 $O(n_{max} + (d+L)\frac{\log p}{\log(d+1)})$，通信時間 $O((gd+L)\frac{\log p}{\log(d+1)})$ である．

ステップ 2 において，各 P_i $(0 \leq i \leq p-1)$ は連続したプロセッサに対して要素を送信する．P_i が要素を送信する連続した j 個のプロセッサを $P_{i'}, P_{i'+1}, ..., P_{i'+j-1}$ とする．P_i は高々 n_{max} 個の要素を保持し，$P_{i'+1}$, $P_{i'+2}, ..., P_{i'+j-2}$ へは少なくとも $\lfloor \frac{n}{p} \rfloor$ 個の要素を送信する．よって，各 P_i は高々 n_{max} 個の要素を 1 つのプロセッサに $\lceil \frac{n}{p} \rceil$ 個ずつ，高々 $\frac{n_{max}}{\lfloor \frac{n}{p} \rfloor} + 2$ プロセッサに対して送信する．また，各 P_i は高々 $\lceil \frac{n}{p} \rceil$ 個の要素を高々 $\frac{\lceil \frac{n}{p} \rceil}{\frac{1}{2}\frac{n}{p}} + 2$ プロセッサから受信する．

よって

内部計算時間 $O(n_{max} + L)$

通信時間 $O(g(\frac{n_{max}}{\lfloor \frac{n}{p} \rfloor} + 2 + \frac{\lceil \frac{n}{p} \rceil}{\frac{1}{2}\frac{n}{p}} + 2)\lceil \frac{n}{pB} \rceil + L) = O(g(\frac{n_{max}}{B} + 1) + L)$

である．

□

3.3.3 アルゴリズム $Selection^*$

アルゴリズム $Selection$ のステップ 2 の (6) を以下の (6)' に変更する. また, ステップ 3 でソートが行われる前には要素は均等に分配されてなければならないので, 要素を均等にするために無限大の値を持つ要素をダミー要素として加える. このために 3 の直前に以下のステップ 3(0) を挿入する.

ステップ 2(6)' 全てのプロセッサが保持する要素に対して疑似分配操作を行う.

ステップ 3(0) 各 P_iにおいて, 保持する要素に$\frac{3}{2}\frac{n}{p\log n} - |A_i|$ 個のダミー要素を加える.

3.3.4 正当性の証明

アルゴリズム $Selection^*$には, アルゴリズム $Selection$ の正当性の証明と同様にして以下の補題 7,8が成り立つ.

補題 7 $Selection^*$の各反復フェーズ中の *(4)* 終了時, $\frac{1}{12}s + 1 < s^1 < \frac{11}{12}s$ が成り立つ. □

補題 8 $Selection^*$のステップ *2* の反復フェーズ数は $O(\log\log n)$ である. □

3.3.5 計算量

定理 2 任意の整数 d $(1 \leq d \leq \log n)$ に対して, アルゴリズム $Selection^*$は,BSP* モデル上で

内部計算時間 $O(\frac{n}{p} + d\log p\log\log n + L(\frac{\log p\log\log n}{\log(d+1)}))$

通信時間 $O(g(\frac{n}{pB} + (\frac{n}{p})^{\frac{1}{7}}(\log p)^{\frac{6}{7}}) + (gd + L)\frac{\log p\log\log n}{\log(d+1)})$

で選択問題を解く.

(証明) s をステップ *2* の各反復フェーズの開始時点の要素数とする.

以下では BSP モデルの場合と計算量の異なる部分について検証する.

(2) における $\lceil\frac{p}{d}\rceil$ プロセッサによる p 要素のソートは補題 1より,

内部計算時間 $O(d\log p + L\frac{\log p}{\log(d+1)})$, 通信時間 $O((g(\frac{d}{B} + d^{\frac{1}{7}}) + L)\frac{\log p}{\log(d+1)})$

で実行できる.

また, *(6)'* の疑似分配操作については, $\max\{|A_1|, |A_2|, ..., |A_p|\} = O(\frac{s}{p})$ なので, 補題 6より,

内部計算時間 $O(\frac{s}{p} + (d + L)\frac{\log p}{\log(d+1)})$, 通信時間 $O(g\frac{s}{pB} + (gd + L)\frac{\log p}{\log(d+1)})$

で実行できる.

以上より, ステップ *2* の 1 反復フェーズは

内部計算時間 $O(\frac{s}{p} + d\log p + L\frac{\log p}{\log(d+1)})$，通信時間 $O(g\frac{s}{pB} + (gd+L)\frac{\log p}{\log(d+1)})$ で実行できるので, *(2)* 全体の計算量は

内部計算時間 $O(\frac{n}{p} + d\log p\log\log n + L\frac{\log p\log\log n}{\log(d+1)})$

通信時間　　$O(g\frac{n}{pB} + (gd+L)\frac{\log p\log\log n}{\log(d+1)})$

となる.

最後のステップ *3* については, ステップ *3* の *(0)* には, 高々$\frac{3}{2}\frac{n}{p\log n}$個のダミー要素の挿入であるから, 内部計算時間 $O(\frac{n}{p\log n} + L)$ であり，また残りのステップは, $\frac{n}{\log n}$要素のソーティング操作であるので, 補題 1より *BSP* 上のソーティング操作と同様に計算すると,

内部計算時間 $O(\frac{n}{p} + L\log p)$，通信時間 $O(g(\frac{n}{pB} + (\frac{n}{p})^{\frac{1}{7}}(\log p)^{\frac{6}{7}}) + L\log p)$

となる. □

定理 2 より，$Selection^*$は $p\log p \leq \frac{n\log d}{L\log\log n}, g = O(B), B \leq (\frac{n}{p\log p})^{\frac{6}{7}}$のとき，内部計算時間，通信時間とも最適加速となることが示される.

第 4 章

2 値画像上の全最近点問題を解く並列アルゴリズム

4.1. 2 値画像上の全最近点問題の定義

$n \times n$ の 2 値画像 Iが与えられたときに，$I[x,y] \in \{0,1\}$ を画素 (x,y) の値とする．ただし，x,yはそれぞれ行，列のインデックスであり，$(0,0)$ を左上の画素とする．$I[x,y] = 1$ のとき，(x,y) は黒画素であるといい，$I[x,y] = 0$ のとき，(x,y) は白画素であるという．$Black, White$ をそれぞれ Iの黒画素，白画素の集合とする．

簡単のために，ある自然数 kに対し，$n = k\sqrt{p}$であるとする．

定義 7 (2 値画像上の全最近点問題) 2 値画像上の全最近点問題は，$n \times n$ の 2 値画像 Iが与えられたとき，以下の様に Iの各黒画素に対して，最も近い黒画素を求める問題である．

入力: $n \times n$ の 2 値画像 I. Iは大きさ $\frac{n}{\sqrt{p}} \times \frac{n}{\sqrt{p}}$の $\sqrt{p} \times \sqrt{p}$個の部分画像 $I_{i,j}$ $(0 \leq i \leq \sqrt{p}-1, 0 \leq j \leq \sqrt{p}-1)$ に分割され，各 $P_{i,j}$ $(0 \leq i \leq \sqrt{p}-1, 0 \leq j \leq \sqrt{p}-1)$ に部分画像 $I_{i,j}$が保持されている．

出力: 各 $(x,y)(0 \leq x \leq n-1, 0 \leq y \leq n-1)$ に対し，以下の式で定義される $NN[x,y]$ を出力する．

$NN[x,y]$

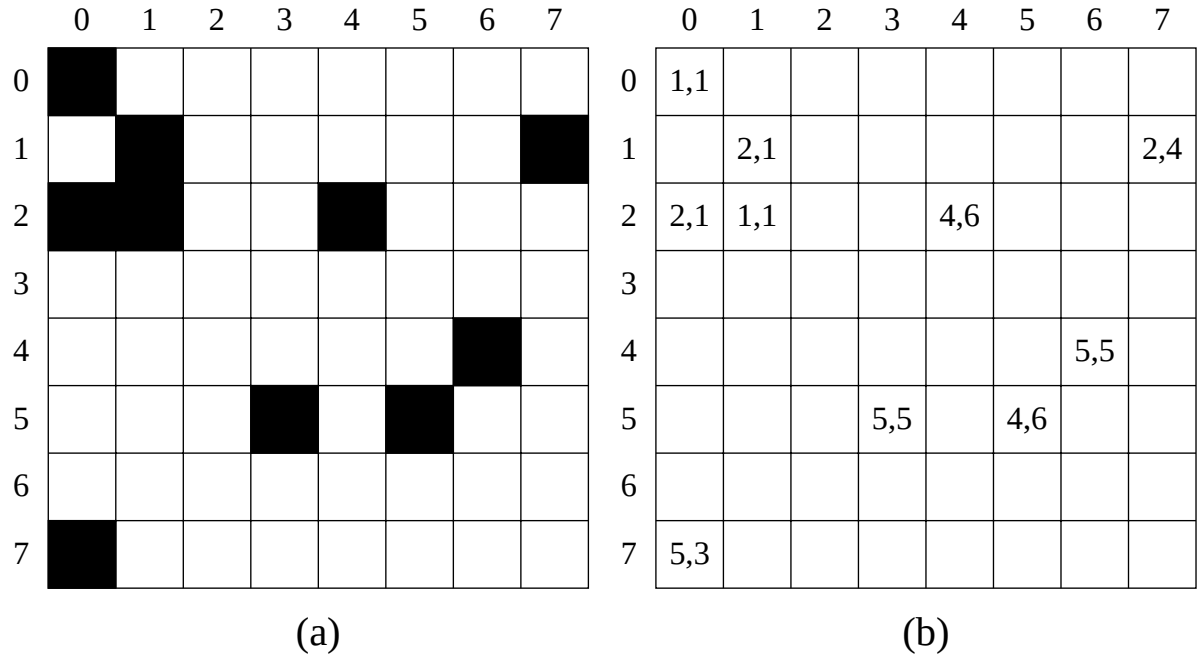

図 4.1 全最近点問題の例:(a) 入力画像 I (b)Iの L_2距離最近点

$$= \begin{cases} (v,w) \text{ s.t. } ((v,w) \in Black - \{(x,y)\}) \\ \quad \wedge (\forall (v',w') \in Black - \{(x,y)\}, d((x,y),(v,w)) \leq d((x,y),(v',w')) \\ \qquad\qquad ((x,y) \in Black \text{ の場合}) \\ undefined \qquad\qquad (\text{それ以外の場合}) \end{cases}$$

ただし， $d((x_1,y_1),(x_2,y_2))$ は 2 画素 (x_1,y_1) と (x_2,y_2) の間の距離を表す.

□

4.2. 距離

本論文では，重み付き距離 [5] および L_p[15] 距離の 2 種類の距離を用いる.

2 画素 $p_1 = (x_1,y_1)$, $p_2 = (x_2,y_2)$ 間の重み付き距離 $d_w(p_1,p_2)$ は以下の様に定義される.

$$d_w(p_1,p_2) = \begin{cases} w_0|x_1 - x_2| + w_1|y_1 - y_2| & (|x_1 - x_2| \geq |y_1 - y_2| \text{ のとき}) \\ w_1|x_1 - x_2| + w_0|y_1 - y_2| & (\text{それ以外のとき}) \end{cases}$$

ただし，w_0, w_1は非負定数である．

重み付き距離は，以下の距離を一般化したものである．

- 4 近傍距離 ($(w_0, w_1) = (1, 1)$ のとき)
- 8 近傍距離 ($(w_0, w_1) = (1, 0)$ のとき)
- 疑似ユークリッド距離 ($(w_0, w_1) = (1, \sqrt{2} - 1)$ のとき)

次に，2 画素 $p_1 = (x_1,\ y_1)$,$p_2 = (x_2,\ y_2)$ 間の L_p距離は下の様にに定義される．

$d_{L_p}(p_1, p_2) = (|x_1 - x_2|^p + |y_1 - y_2|^p)^{\frac{1}{p}}$

プロセッサ数を表す p との混乱を避けるため，以降では L_p距離を L_q距離と記述する．

L_q距離は，下の距離を一般化したものである．

- ユークリッド距離 ($q = 2$ のとき)
- 4 近傍距離 ($q = 1$ のとき)
- 8 近傍距離 ($q \to \infty$ のとき)

以降では，距離として重み付き距離を用いる全最近点問題を，重み付き距離全最近点問題，L_q距離を用いる全最近点問題を，L_q距離全最近点問題と呼ぶ．

4.3. 2 次元接頭部演算操作

本節では，2 つの基本操作，垂直 2 次元接頭部演算操作および対角 2 次元接頭部演算操作を示す．記号$\circ$を結合則を満たす 2 項演算子とする．垂直 2 次元接頭部演算操作および対角 2 次元接頭部演算操作は以下のように定義される．

定義 8 (垂直 2 次元接頭部演算操作)

垂直 2 次元接頭部演算操作とは，$n \times n$ の 2 次元配列 $A[x, y]$ $(0 \leq x \leq n-1, 0 \leq y \leq n-1)$ が与えられたときに，各 (x, y) に対して以下の配列 $B[x, y]$ を求める演算である．

$B[x, y] = A[0, y] \circ A[1, y] \circ \ldots \circ A[x, y]$ □

定義 9 (対角 2 次元接頭部演算操作)

対角 2 次元接頭部演算操作は，$n \times n$ の 2 次元配列 $A[x, y]$ $(0 \leq x \leq n-1, 0 \leq y \leq n-1)$ が与えられたときに，各 (x, y) に対して以下の配列 $B[x, y]$ を求める演算である．

$$B[x, y] = \begin{cases} A[0, y-x] \circ A[1, y-x+1] \circ \ldots \circ A[x, y] & (x \leq y\text{の場合}) \\ A[x-y, 0] \circ A[x-y+1, 1] \circ \ldots \circ A[x, y] & (\text{それ以外の場合}) \end{cases}$$

□

入力状態において，各プロセッサ $P_{i,j}$ $(0 \leq i \leq \sqrt{p}-1, 0 \leq j \leq \sqrt{p}-1)$ は，A のサイズ $\frac{n}{\sqrt{p}} \times \frac{n}{\sqrt{p}}$ の部分配列部分配列 $A_{i,j}$ を保持しているとする．

上記の 2 つの 2 次元接頭部演算操作に対して，次の補題が成立する．

補題 9 垂直及び対角 *2* 次元接頭部演算操作は，共に *BSP* モデル上で，p $(1 \leq p \leq n^2)$ プロセッサを用いて

内部計算時間 $O(\frac{n^2}{p}+L)$, 通信時間 $O(g\frac{n}{\sqrt{p}}+L)$ $(1 < p \leq n$ の場合$)$

内部計算時間 $O(\frac{n^2}{p}+L\frac{\log \frac{p}{n}}{\log(d+1)})$，通信時間 $O(g\frac{n}{\sqrt{p}}+(gd+L)\frac{\log \frac{p}{n}}{\log(d+1)})$ $(n < p \leq n^2$の場合$)$

で求められる．ただし，d は $1 \leq d \leq \frac{p}{n}$を満たす任意の整数である．

(証明)

(垂直 2 次元接頭部演算操作)

Juurlink と Wijshoff[13] は $p \times k$配列に対する 2 次元接頭部演算操作のアルゴリズムを提案した．

彼らのアルゴリズムは，入力として，各プロセッサが各行を保持する $p \times k$ 配列が与えられたときに，各列の接頭部を求めるアルゴリズムであり，BSP モデル上で p $(1 \leq p \leq k)$ プロセッサを用いて内部計算時間 $O(k+L)$，通信時間 $O(gk+L)$，p $(k < p)$ プロセッサを用いて任意の d $(1 \leq d \leq \frac{p}{k})$ に対し内部計算時間 $O(k+(d+L)\frac{\log \frac{p}{k}}{\log(d+1)})$，通信時間 $O(gk+(gd+L)\frac{\log \frac{p}{k}}{\log(d+1)})$ で 2 次元接頭部を求める．

プロセッサ集合 $P_{0,j}, P_{1,j}, \ldots P_{\sqrt{p}-1,j}$上の列についての垂直 2 次元接頭部演算操作を考える． この操作は，以下の様に行う．

1. 各プロセッサ $P_{i,j}$ の各列 h $(0 \leq h \leq \frac{n}{\sqrt{p}}-1)$ に対して，$A_{i,j}[0,h] \circ A_{i,j}[1,h] \circ \ldots \circ A_{i,j}[\frac{n}{\sqrt{p}}-1,h]$ を求め，その結果を $A'_{i,j}[h]$ に保持する．

2. 2 次元接頭部演算操作のアルゴリズム [13] 用いて，各列 h $(0 \leq h \leq \frac{n}{\sqrt{p}}-1)$ に対して，$A'_{0,j}[h], A'_{1,j}[h], \ldots, A'_{\sqrt{p}-1,j}[h]$ の接頭部を求め，その結果を $A''_{i,j}[h]$ に保持する．

3. 各プロセッサ $P_{i,j}$は，$P_{i+1,j}$に $A''_{i,j}$ を送信，各プロセッサ $P_{i,j}$は，各列 h $(0 \leq h \leq \frac{n}{\sqrt{p}}-1)$ に対して，$B[(i-1)\frac{n}{\sqrt{p}}+g,(j-1)\frac{n}{\sqrt{p}}+h] = A''_{i-1,j}[h] \circ A_{i,j}[0,h] \circ A_{i,j}[1,h] \circ \ldots, A_{i,j}[g,h]$ を求める．

1. 及び 3. の操作は内部計算時間 $(\frac{n^2}{p}+L)$，通信時間 $O(g\frac{n}{\sqrt{p}}+L)$ で行うことができる．また，2 の操作は，Juurlink と Wijshoff のアルゴリズム [13] において，$k = \frac{n}{\sqrt{p}}$とすることにより行うことができる．従って，補題が成立する．

(対角 2 次元接頭部演算操作)

対角 2 次元接頭部演算操作は以下の様に行う．第 1 要素が $P_{0,0}, P_{1,0}, \ldots P_{\sqrt{p}-1,0}$ 上にある対角列の集合を $DSL_0, DSL_1, \ldots DSL_{\sqrt{p}-1}$，第 1 要素が $P_{0,1}, P_{0,2} \ldots P_{0,\sqrt{p}-1}$, 上にある対角列の集合を $DSU_1, DSU_2, \ldots DSU_{\sqrt{p}-1}$ とする．

まず，対角列集合 DSL_0, DSL_2, DSL_4, $\ldots$ $DSL_{2\lfloor\frac{\sqrt{p}-1}{2}\rfloor}$ と DSU_2, DSU_4, $\ldots$ $DSU_{2\lfloor\frac{\sqrt{p}-1}{2}\rfloor}$ に対する対角 2 次元接頭部演算操作を求める．この操作は，以下の理由により，垂直 2 次元接頭部演算操作と同様の計算量で求めることができる．

1. 各対角列に対して，各プロセッサが保持する要素数は高々$\frac{n}{\sqrt{p}}$であり，プロセッサ数は高々$2\frac{n}{\sqrt{p}}$である．
2. 各対角列の集合は，相異なるプロセッサ上にあり各対角列集合で独立に計算できる．

同様に，残りの対角列集合 DSL_1, DSL_3, $\ldots$ $DSL_{2\lfloor\frac{\sqrt{p}}{2}\rfloor-1}$, DSU_1, DSU_3, $\ldots$ $DSU_{2\lfloor\frac{\sqrt{p}}{2}\rfloor-1}$ も計算できる．

よって，対角 2 次元接頭部演算操作も垂直 2 次元接頭部演算操作と同様に行うことができる． □

以降では，サイズ $n \times n$ の 2 次元配列に対する p プロセッサによる 2 次元接頭部演算操作の内部計算時間，通信時間をそれぞれ $T^{comp}_{prefix}(n,n,p)$,$T^{comm}_{prefix}(n,n,p)$ と記述する．

4.4. 重み付き距離全最近点問題を解くアルゴリズム

本節では，重み付き距離全最近点問題を解くアルゴリズムを提案し，その計算量を解析する．

藤原ら [5] は PRAM 上で重み付き特徴点変換を求める並列アルゴリズムを提案した．このアルゴリズムは，4.4.1 節に示す通り，BSP モデル上に適用できる．

以下では，重み付き特徴点変換アルゴリズムを用いて，重み付き距離全最近点を求められることを示す．

4.4.1 重み付き特徴点変換を解く並列アルゴリズム

この章では，重み付き特徴点変換アルゴリズム [5] を紹介し，BSP 上での計算量について解析する．

特徴点変換操作は，各画素 (x,y) に対して，(x,y) 自身を含む最も近い黒画素 $NBP[x,y]$ を求める操作であり，以下の様に定義される．

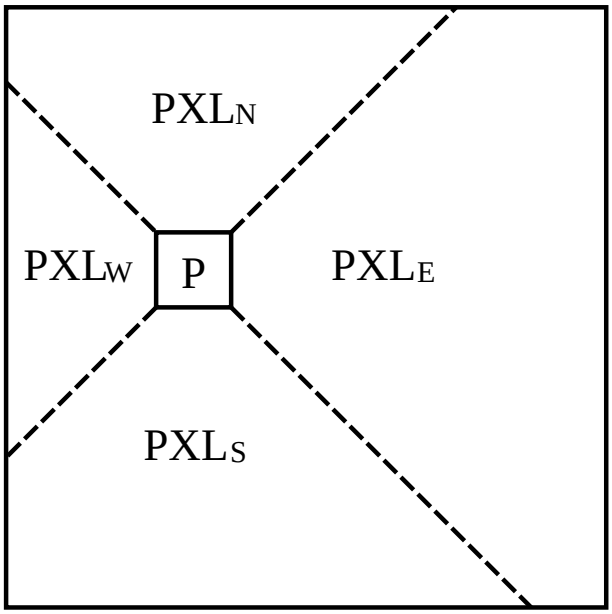

図 4.2　画素 $p = (x, y)$ と画素集合 $PXL_N(x, y), PXL_S(x, y), PXL_E(x, y), PXL_W(x, y)$.

$NBP[x, y] = (v, w)$ s.t. $((v, w) \in Black)$
$\wedge\, (\forall (v', w') \in Black,\ d((x, y), (v, w)) \leq d((x, y), (v', w')))$

ただし，$d((x_1, y_1), (x_2, y_2))$ は 2 画素 (x_1, y_1),(x_2, y_2) 間の距離とする．

各画素 (x, y) に対して，4 つの画素集合 $PXL_N(x, y)$, $PXL_S(x, y)$, $PXL_E(x, y)$, $PXL_W(x, y)$ を以下の様に定義する．(図 4.2参照)

$$PXL_N(x, y) = \{(x - x_1, y + y_1) | \\ 1 \leq x_1 \leq x, \max\{-y, x_1\} \leq y_1 \leq \min\{n - 1 - y, x_1\}\}.$$

$$PXL_S(x, y) = \{(x + x_1, y + y_1) | \\ 1 \leq x_1 \leq n - 1 - x, \max\{-y, x_1\} \leq y_1 \leq \min\{n - 1 - y, x_1\}\}$$

$$PXL_W(x, y) = \{(x + x_1, y - y_1) | \\ \max\{-x, -y_1\} \leq x_1 \leq \min\{n - 1 - x, y_1\}, 1 \leq y_1 \leq y\}$$

$$PXL_E(x, y) = \{(x + x_1, y + y_1) | \\ \max\{-x, -y_1\} \leq x_1 \leq \min\{n - 1 - x, y_1\}, 1 \leq y_1 \leq n - 1 - y\}\}$$

重み付き特徴点変換アルゴリズムは次の 5 ステップから成る．

ステップ 1 各画素 (x, y) に対して，$PXL_N(x, y)$ 内の黒画素中から，最近の黒画素を求める．

ステップ 2 同様に，$PXL_S(x,y)$ 内の黒画素中から，最近の黒画素を求める．

ステップ 3 同様に，$PXL_E(x,y)$ 内の黒画素中から，最近の黒画素を求める．

ステップ 4 同様に，$PXL_W(x,y)$ 内の黒画素中から，最近の黒画素を求める．

ステップ 5 各画素 (x,y) に対して，ステップ 1〜4 で求めた中から最近の黒画素を選択する．

各画素 (x,y) に対して，$PXL_S(x,y), PXL_E(x,y), PXL_W(x,y)$ 内の最近黒画素は，$PXL_N(x,y)$ 内の最近黒画素を求める手法と同様の手法で求められる．よって以降は，各画素 (x,y) に対して，$PXL_N(x,y)$ 内の最近黒画素を求める手法について述べる．

重み付き距離については，藤原ら [5] により，以下の補題が示されている． この補題を用いて，最近黒画素を接頭部演算操作等の基本操作のみで求めることができる．

補題 10 [5] 画素 $q=(x,y)$, $q_1, q_2 \in PXL_N(x,y)$ を $d_w(q,q_1) \leq d_w(q,q_2)$ を満たす 3 画素とする．このとき，任意の画素 $q'=(k,y)(x \leq k \leq n-1)$ に対して，$d_w(q',q_1) \leq d_w(q',q_2)$ が成立する． □

画素 (x,y) に対して，$PXL_N(x,y)$ 内の最近黒画素を $NBP_N[x,y]$ と記述する．また，下記の画素集合 DI_{NE}, DI_{NW}を定義する．

$DI_{NW}(x,y) = \{(x-i, y-i) | 0 \leq i \leq \min\{x,y\}\}$

$DI_{NE}(x,y) = \{(x-i, y+i) | 0 \leq i \leq \min\{x, n-1-y\}\}$

($DI_{NW}(x,y) \cup DI_{NE}(x,y) = PXL_N(x,y) - PXL_N(x-1,y)$ である．)

画素 (x,y) に対して，$DI_{NW}(x,y)$, $DI_{NE}(x,y)$ 内の最近黒画素を $NBP_{NW}[x,y]$, $NBP_{NE}[x,y]$ と記述する．補題 10 より，$NN_N[x,y]$ は $NN_N[x-1,y], NN_{NW}[x,y]$, $NN_{NE}[x,y]$. の中の一つである．

まず，各 (x,y) $(0 \leq x \leq n-1, 0 \leq y \leq n-1)$ に対して，$NBP_{NW}[x,y]$, $NBP_{NE}[x,y]$ を求める．以下の補題により，$NBP_{NW}[x,y]$, $NBP_{NE}[x,y]$ は接頭部演算で求めることができる．

補題 11 [5] 画素 $q=(x,y)$, $q_1=(x_1,y_1)$, $q_2=(x_2,y_2)$ を $q_1, q_2 \in DI_{NW}(x,y) \cup DI_{NE}(x,y)$, $d_w(q,q_1) \leq d_w(q,q_2)$ を満たす 3 画素とする．このとき，$x_1 \geq x_2$ が成立する． □

補題 11 より，$DI_{NW}(x,y) \cup DI_{NE}(x,y)$ 内の (x,y) の最近黒画素は，$DI_{NW}(x,y) \cup DI_{NE}(x,y)$ 内の黒画素中，最大の列番号を持つ黒画素である．従って，NBP_{NW},

NBP_{NE} を $DI_{NW}(x,y), DI_{NE}(x,y)$ に対する対角 2 次元接頭部演算操作で求めることができる．

次に，各 (x,y) に対し，最近の黒画素 $NBP_N[x,y]$ を求める．$NBP_{NW}[x,y]$ と $NBP_{NE}[x,y]$ のうち，近い方を $NBP_{DI}[x,y]$ と記述する．$NBP_N[x,y]$ を求めるために，以下の値 $F(x,y)$ を計算する．

$F(x,y) = -w_0 g + w_1|y-h|$ ただし $NBP_{DI}[x,y] = (g,h)$.

以下の理由により，各列に対して，$F(x,y)$ $(1 \leq k \leq n-1)$ の接頭部最小値を計算することにより，$NBP_N[x,y]$ を計算できる．

$NBP_{DI}[x_1,y] = (g_1,h_1)$, $NBP_{DI}[x_2,y] = (g_2,h_2)$, とし，$x \geq x_1, x_2, d_w((x,y),$ $NBP_{DI}[x_1,y]) \leq d_w((x,y), NBP_{DI}[x_2,y])$. と仮定する．このとき，次が成り立つ．

$$
\begin{aligned}
F(x_1,y) &\leq F(x_2,y) \\
-w_0 g_1 + w_1|y-h_1| &\leq -w_0 g_2 + w_1|y-h_2| \\
w_0(x-g_1) + w_1|y-h_1| &\leq w_0(x-g_2) + w_1|y-h_2| \\
d_w((x,y), NN_{DI}[x_1,y]) &\leq d_w((x,y), NN_{DI}[x_2,y])
\end{aligned}
$$

従って，各列に対して，上から下に $F(k,y)(0 \leq k \leq n-1)$ の接頭部最小値を計算することにより， $NBP_N[x,y]$ を計算できる．

$NBP_N[x,y]$ を求める計算のうち，接頭部演算操作に関しては，2章で述べた 2 次元接頭部演算操作を使用することができる．また，接頭部演算操作以外の部分では，$O(\frac{n^2}{p})$ 時間の内部計算と，$O(\frac{n}{\sqrt{p}})$ 回の通信で行うことができる．よって以下の補題が成り立つ．

補題 12 各画素 (x,y) に対し, $PXL_N(x,y)$, $DI_{NW}(x,y)$, $DI_{NE}(x,y)$ 内の最近黒画素 $NBP_N[x,y]$, $NBP_{NW}[x,y]$, NBP_{NE} $[x,y]$ は BSP モデル上で，p $(1 \leq p \leq n^2)$ プロセッサを用いて，

内部計算時間 $O(\frac{n^2}{p} + T^{comp}_{prefix}(n,n,p))$，通信時間 $O(g\frac{n}{\sqrt{p}} + T^{comm}_{prefix}(n,n,p))$ で求めることができる． □

各画素 (x,y) に対して，重み付き特徴点変換は $NBP_N[x,y]$, $NBP_S[x,y]$, $NBP_E[x,y]$,$NBP_W[x,y]$. の中の最近黒画素を求めればよい．よって，以下の補題が成り立つ．

補題 13 サイズ $n \times n$ の 2 値画像上での重み付き特徴点変換は，は BSP モデル上で，p $(1 \leq p \leq n^2)$ プロセッサを用いて，

内部計算時間 $O(\frac{n^2}{p} + T^{comp}_{prefix}(n,n,p))$，通信時間 $O(g\frac{n}{\sqrt{p}} + T^{comm}_{prefix}(n,n,p))$ で求めることができる． □

4.4.2 全最近点問題への変換

本節では，重み付き特徴点変換アルゴリズムを重み付き距離全最近点アルゴリズムに変換する方法を示す．

上記 2 つの問題の相違点は，全最近点問題が，各黒画素からその黒画素自身を除く最近の黒画素を求めるのに対し，特徴点変換問題は，各黒画素および各白画素からその画素自身を含む最近の黒画素を求めることである．

この相違点については，以下の操作を行うことにより，特徴点変換問題から全最近点問題に変換できる．

各黒画素 (x, y) に対して，$PXL_N(x, y)$ 内の最近の黒画素 ((x, y) 自身を除く) は $DI_{NW}(x-1, y-1)$, $DI_{NE}(x-1, y+1)$, $PXL_N(x-1, y)$ 内の最近の黒画素である．従って，$NBP_{NW}[x-1, y-1]$, $NBP_{NE}[x-1, y+1]$, $NBP_N[x-1, y]$ を用いて最近点を求めることができる．

$PXL_N(x, y)$ 内の最近点を求めるアルゴリズムは次の 2 ステップから成る．

ステップ 1 各黒画素 (x, y) に対して，$DI_{NW}(x-1, y-1)$, $DI_{NE}(x-1, y+1)$, $PXL_N(x-1, y)$ 内の最近の黒画素 $NBP_{NW}[x-1, y-1]$, $NBP_{NE}[x-1, y+1]$,$NBP_N[x-1, y]$ を計算する．

ステップ 2 各黒画素 (x, y) に対して，$NBP_{NW}[x-1, y-1]$, $NBP_{NE}[x-1, y+1]$,$NBP_N[x-1, y]$ から最近の黒画素を選択する．

補題 12より，3 つの黒画素 $NBP_{NW}[x-1, y-1]$, $NBP_{NE}[x-1, y+1]$,$NBP_N[x-1, y]$ は BSP モデル上で，p $(1 \leq p \leq n^2)$ プロセッサを用いて内部計算時間 $O(\frac{n^2}{p} + T^{comp}_{prefix}(n, n, p))$，通信時間 $O(g\frac{n}{\sqrt{p}} + T^{comm}_{prefix}(n, n, p))$ で計算できる．従って以下の定理が成り立つ．

定理 3 サイズ $n \times n$ の 2 値画像上の重み付き距離全最近点問題は，BSP モデル上で，p $(1 \leq p \leq n^2)$ プロセッサを用いて

内部計算時間 $O(\frac{n^2}{p} + T^{comp}_{prefix}(n, n, p))$，通信時間 $O(g\frac{n}{\sqrt{p}} + T^{comm}_{prefix}(n, n, p))$

で計算できる． □

補題 9 と定理 3 より，以下の系が成り立つ．

系 1 サイズ $n \times n$ の 2 値画像上の重み付き距離全最近点問題は，BSP モデル上で，p $(1 \leq p \leq n^2)$ プロセッサを用いて

内部計算時間 $O(\frac{n^2}{p} + L)$, 通信時間 $O(g\frac{n}{\sqrt{p}} + L)$ $(n < p \leq n^2$の場合)

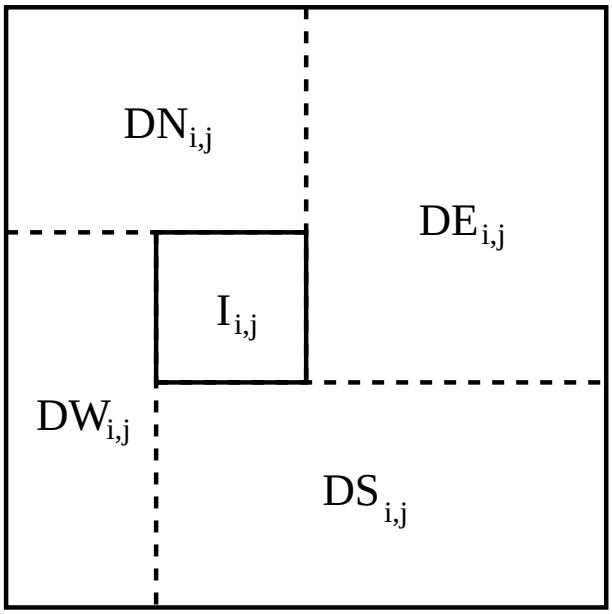

図 4.3 部分画像 $I_{i,j}$に対する 4 つの部分画像

内部計算時間 $O(\frac{n^2}{p} + L\frac{\log \frac{p}{n}}{\log(d+1)})$，通信時間 $O(g\frac{n}{\sqrt{p}} + (gd+L)\frac{\log \frac{p}{n}}{\log(d+1)})$
$(n < p \leq n^2$の場合)
で求められる．ただし，d は $1 \leq d \leq \frac{p}{n}$を満たす任意の整数である． □

4.5. L_q距離全最近点問題を解くアルゴリズム

本節では，L_q距離全最近点問題を解くアルゴリズムを提案し，その計算量を解析する．

まず，各プロセッサ $P_{i,j}$が保持する各部分画像 $I_{i,j}$ に対し，次の 4 つの部分画像を定義する．(図 4.3参照)

$DN_{i,j} = \{(x,y)|0 \leq x \leq i\frac{n}{\sqrt{p}} - 1, 0 \leq y \leq (j+1)\frac{n}{\sqrt{p}} - 1\}$
$DS_{i,j} = \{(x,y)|(i+1)\frac{n}{\sqrt{p}} \leq x \leq n-1, j\frac{n}{\sqrt{p}} \leq y \leq n-1\}$
$DW_{i,j} = \{(x,y)|i\frac{n}{\sqrt{p}} \leq x \leq n-1, 0 \leq y \leq j\frac{n}{\sqrt{p}} - 1\}$
$DE_{i,j} = \{(x,y)|0 \leq x \leq (i+1)\frac{n}{\sqrt{p}} - 1, (j+1)\frac{n}{\sqrt{p}} \leq y \leq n-1\}$

上記の 4 つの部分画像に対して，部分画像 $I_{i,j}$内の黒画素の最近点を求める．本論文では，計算量の減少のために，基底画素，および候補画素という画素を以下

の様に定義する．部分画像 A(A は上記 4 つの部分画像のうちの一つ) に対して，その最近点が A に含まれる可能性がある $I_{i,j}$の黒画素を A に対する $I_{i,j}$の基底画素と呼ぶ．また，$I_{i,j}$の黒画素の最近点となる可能性のある A の黒画素を，$I_{i,j}$に対する A の候補画素と呼ぶ．A に含まれる $I_{i,j}$の各黒画素の最近点を計算するためには，各基底画素から，最近の候補画素を求めれば充分である．

$DS_{i,j}, DE_{i,j}, DW_{i,j}$内の最近黒画素は，$DN_{i,j}$に対して求める方法と同様の手法で求めることができる．また，$I_{i,j}$の黒画素は，既知の逐次アルゴリズム [11] を用いて求めることができる．よって，以下では，$I_{i,j}$の各基底画素に対して，$DN_{i,j}$の最近黒画素を並列に求める手法について示す．

4.5.1 $DN_{i,j}$内の最近黒画素の計算

まず，$I_{i,j}$と $DN_{i,j}$において，基底画素および候補画素の選び方を示す．

まず，基底画素としては，以下の定義の様に，$I_{i,j}$の各列に対し，最小の行番号を持つ黒画素を基底画素として選択する．

定義 10 (基底画素) $DN_{i,j}$に対する $I_{i,j}$の基底画素集合 $Base^N_{i,j}$は，以下を満たす黒画素 (x, y) の集合である．

$$x = min\{x' | (x', y) \in Black \wedge (x', y) \in I_{i,j}\}$$

□

$I_{i,j}$の列の数は $\frac{n}{\sqrt{p}}$であるので，$DN_{i,j}$に対する $I_{i,j}$の基底画素は高々 $\frac{n}{\sqrt{p}}$個である．最近点が $DN_{i,j}$内に存在する $I_{i,j}$内の黒画素は，基底画素であることは明らかである．

次に，候補画素の決定方法を説明する．$I_{i,j}$内の任意の黒画素に対して，$DN_{i,j}$の各列において最大の行番号を持つ黒画素は同じ列の他の黒画素よりも近いことは明らかである．さらに，次の 2 つの補題を用いると，候補画素の数を減らすことができる．

補題 14 $I_{i,j}$内に *2* つ以上黒画素が存在する場合，$DN_{i,j-3}$内の黒画素は，$I_{i,j}$内の黒画素の最近点となることはない．

(証明) $p_0 = (x_0, y_0)$,$p_1 = (x_1, y_1)$ を $I_{i,j}$内の 2 つの黒画素，$p_2 = (x_2, y_2)$ を $DN_{i,j-3}$内の黒画素とする．このとき，$|x_0 - x_1| < \frac{n}{\sqrt{p}}$, $|y_0 - y_1| < \frac{n}{\sqrt{p}}$, $y_0 - y_2 > 2\frac{n}{\sqrt{p}}$が

満たされ，

$$\begin{aligned} d_{L_q}(p_0,p_1) &= (|x_0-x_1|^q+(y_0-y_1)^q)^{\frac{1}{q}} \\ &< (2(\frac{n}{\sqrt{p}})^q)^{\frac{1}{q}} \\ &\leq 2\frac{n}{\sqrt{p}} \ < \ y_0-y_2 \\ d_{L_q}(p_0,p_2) &= ((x_0-x_2)^q+(y_0-y_2)^q)^{\frac{1}{q}} \ > \ y-y_2 \end{aligned}$$

よって，$d_{L_q}(p_0,p_1) < d_{L_q}(p_0,p_2)$ が満たされ，また同様に $d_{L_q}(p_0,p_1) < d_{L_q}(p_1,p_2)$ が導かれる．したがって，p_2は p_0及び p_1の最近点ではない． □

補題 15 $I_{i,k},(0 \leq k < j)$ に黒画素が含まれていると仮定する．このとき，$DN_{i,k-2}$ 内の黒画素は，$I_{i,j}$内の黒画素の最近点となることはない．

(証明) $p=(x,y)$ を $I_{i,j}$内の黒画素，$p_1=(x_1,y_1)$ を $I_{i,k}$内の黒画素 $p_2=(x_2,y_2)$ を $DN_{i,k-2}$内の黒画素とする．このとき，$|x-x_1| < \frac{n}{\sqrt{p}}$，$y-y_2 > (y-y_1)+\frac{n}{\sqrt{p}}$ が満たされ，

$$\begin{aligned} d_{L_q}(p,p_1) &= (|x-x_1|^q+(y-y_1)^q)^{\frac{1}{q}} \\ &< ((\frac{n}{\sqrt{p}})^q+(y-y_1)^q)^{\frac{1}{q}} \\ &\leq \frac{n}{\sqrt{p}}+y-y_1 \ < \ y-y_2 \\ d_{L_q}(p,p_2) &= ((x-x_2)^q+(y-y_2)^q)^{\frac{1}{q}} \ > \ y-y_2 \end{aligned}$$

よって，$d_{L_q}(p,p_1) < d_{L_q}(p,p_2)$ が満たされ，p_2は p の最近点ではない． □

上記の 2 つの補題により，候補画素は以下の様に定義される．

定義 11 (候補画素)

1. $I_{i,j}$内に，2つ以上黒画素が存在する場合
   $I_{i,j}$に対する $DN_{i,j}$内の候補画素集合 $Candidate^N_{i,j}$は，以下を満たす黒画素の集合である．

$$x = max\{x'|(x',y) \in Black \wedge (DN_{i,j}-DN_{i,j-3})\}$$

2. $I_{i,j}$内に，ただ 1 つ黒画素が存在する場合
$I_{i,j}$に対する $DN_{i,j}$内の候補画素集合 $Candidate^N_{i,j}$は以下を満たす黒画素の集合である．

$$x = max\{x' | (x', y) \in Black \wedge (DN_{i,j} - DN_{i,k-2})\}$$

ただし $I_{i,k}$は，$\{I_{i,0}, I_{i,1}, ..., I_{i,j-1}\}$ 中で，その内部に少なくとも 1 つ黒画素が存在する最も右の部分画像である． □

候補画素および基底画素は，各列に対する黒画素の座標に対する接頭部演算で計算することができる．

以下にアルゴリズムの詳細を記述する．

$DN_{i,j}$に対する最近点を求めるアルゴリズム

ステップ 1: (候補画素の計算)

(1) 各プロセッサ $P_{i,j}$ $(0 \leq i \leq \sqrt{p}-1, 0 \leq j \leq \sqrt{p}-1)$ において，各画素 (x, y) に対して，画素 (x, y) が黒画素であれば $A[x, y] = (x, y)$，白画素であれば $A[x, y] = (-\infty, -\infty)$ とする．

(2) 各 j $(0 \leq j \leq \sqrt{p}-1)$ に対して，プロセッサ $P_{0,j}, P_{1,j}, ..., P_{\sqrt{p}-1,j}$を用いて A の各列に対して，上から下に A の第 1 要素を比較して接頭部最小値 A'を計算する．各プロセッサ $P_{i,j}$は，$A'[i\frac{n}{\sqrt{p}}-1, j\frac{n}{\sqrt{p}}+h]$ $(0 \leq h \leq \frac{n}{\sqrt{p}}-1)$ をプロセッサ $P_{i+1,j}$に送信する．各プロセッサ $P_{i,j}$は，候補画素集合 $Candidate^N_{i,j} - Candidate^N_{i,j-1}$ を受信する．

ステップ 2: (基底画素の計算)

(1) 各プロセッサ $P_{i,j}$ $(0 \leq i \leq \sqrt{p}-1, 0 \leq j \leq \sqrt{p}-1)$ において，画素 (x, y) に対して，画素 (x, y) が黒画素であれば $B[x, y] = (x, y)$，白画素であれば $B[x, y] = (+\infty, +\infty)$ とする．

(2) 各プロセッサ $P_{i,j}$ $(0 \leq i \leq \sqrt{p}-1, 0 \leq j \leq \sqrt{p}-1)$ において，Bの各列に対して，下から上に Bの第 1 要素を比較して接頭部最小値 B'を計算する．各プロセッサ $P_{i,j}$ $(0 \leq i \leq \sqrt{p}-1, 0 \leq j \leq \sqrt{p}-1)$ において，$B'[i\frac{n}{\sqrt{p}}, j\frac{n}{\sqrt{p}}+h]$ $(0 \leq h \leq \frac{n}{\sqrt{p}}-1)$ は基底画素集合 $Base^N_{i,j}$の画素である．

ステップ 3: ($I_{i,j}$内に黒画素が 2 つ以上ある場合の計算)

(1) 各プロセッサ $P_{i,j}$に対して，プロセッサ $P_{i,j-2}$, $P_{i,j-1}$は候補画素集合 $Candidate^N_{i,j-1} - Candidate^N_{i,j-3}$ をプロセッサ $P_{i,j}$に送信する．

(2) $I_{i,j}$内に黒画素が 2 つ以上あるならば，各プロセッサ $P_{i,j}$は，基底画素集合 $Base_{i,j}^{N}$内の各黒画素に対して，基底画素集合 $Candidate_{i,j}^{N} - Candidate_{i,j-3}^{N}$ 内の最近の黒画素を選択する．

ステップ 4: ($I_{i,j}$内に黒画素がただ 1 つある場合の計算)

(1) 各プロセッサ $P_{i,j}$はただ 1 つの基底画素をプロセッサ $P_{i,k-1}$, $P_{i,k}$, ..., $P_{i,j}$ に送信する．ただし，$I_{i,k}$は部分画像 $\{I_{i,0}, I_{i,1}, ..., I_{i,j-1}\}$ の中で，その内部に少なくとも 1 つ黒画素を持ち最も右に存在する部分画像である．

これは以下の手順により行われる．

(1-1) 各プロセッサ $P_{i,j}$ $(0 \leq i \leq \sqrt{p}-1, 0 \leq j \leq \sqrt{p}-1)$ において，$I_{i,j}$内にただひとつ黒画素が存在するならば $A[i,j] = (j, b_{i,j})$，2 つ以上黒画素が存在するならば $A[i,j] = (j, \emptyset)$，存在しないならば $A[i,j] = (+\infty, \emptyset)$ とする．ただし，$b_{i,j}$は $I_{i,j}$の黒画素である．

(1-2) 各 i に対し，プロセッサ $P_{i,0}, P_{i,1}, ..., P_{i,\sqrt{p}-1}$を用いて右から左へ $A[i,0], A[i,1]$, ..., $A[i, \sqrt{p}-1]$ の第 1 要素を比較して接頭部最小値を A'を計算する．各プロセッサ $P_{i,j}$において，$A'[i,j]$ の第 2 要素が黒画素であれば，$b_{i,j}^{0}$をその黒画素とする．

(1-3) 各プロセッサ $P_{i,j}$において，$I_{i,j-1}$内に黒画素があるならば，$P_{i,j-1}$,$P_{i,j-2}$ に黒画素 $b_{i,j}^{0}$を送信する．$P_{i,j}$が $P_{i,j+1}$から黒画素を受信したならば，その黒画素を $b_{i,j}^{1}$とする．同様に，$P_{i,j+2}$から受信したならば，$b_{i,j}^{2}$とする．

(2) 各プロセッサ $P_{i,j}$ $(0 \leq i \leq \sqrt{p}-1, 0 \leq j \leq \sqrt{p}-1)$ において，

各黒画素 $b_{i,j}^{0}$, $b_{i,j}^{1}$及び $b_{i,j}^{2}$に対して，各プロセッサ上の候補画素集合 $Candidate_{i,j}^{N} - Candidate_{i,j-1}^{N}$ 内の黒画素の中から最近点 $NN_{i,j}(b_{i,j}^{0})$, $NN_{i,j}(b_{i,j}^{1})$,$NN_{i,j}(b_{i,j}^{2})$ を計算する．

(3) プロセッサ $P_{i,k-1}, P_{i,k}, ..., P_{i,j}$ を用いて，部分画像 $I_{i,j}$の基底画素の最近点を計算し，結果を $P_{i,j}$に送信する．ただし，部分画像 $I_{i,k}$は，部分画像 $\{I_{i,0}, I_{i,1}, ...,$ $I_{i,j-1}\}$. の中で，その内部に少なくとも 1 つ黒画素を持つ最も右に存在する部分画像である．

これは以下の手順により行われる．

(3-1) 各プロセッサ $P_{i,j}$において，(1-3) で黒画素 $b_{i,j}^{1}$ を受信したならば，最近点 $NN_{i,j}(b_{i,j}^{1})$ をプロセッサ $P_{i,j+1}$に送信する．同様に，最近点 $NN_{i,j}(b_{i,j}^{2})$ をプロセッサ $P_{i,j+2}$に送信する．

(3-2) 各プロセッサ $P_{i,j}$において，(3-1)で最近点 $NN_{i,j-1}(b^1_{i,j-1})$, $NN_{i,j-2}$ $(b^2_{i,j-2})$ を受信したならば，最近点 $NN_{i,j}(b^0_{i,j})$ を $NN_{i,j-2}(b^2_{i,j-2})$, $NN_{i,j-1}(b^1_{i,j-1})$, $NN_{i,j}(b^0_{i,j})$ の中の最も近いものに交換する．

(3-3) 各プロセッサ $P_{i,j}$において，$B[i,j] = (-j', d_{L_q}(b^0_{i,j}, NN_{i,j}(b^0_{i,j})), NN_{i,j}(b^0_{i,j}))$ とする．ただし，j'は $A'[i,j]$ の第 1 要素である．

(3-4) 各 i に対して，プロセッサ $P_{i,0}, P_{i,1}, ..., P_{i,\sqrt{p}-1}$ を用いて，左から右に，B の第 1 要素，第 2 要素を辞書式順で比較して接頭部最小値 B'を計算する．このとき，黒画素 $b_{i,j}$の最近点は，$B'[i,j]$ の第 3 要素となる．

上記のアルゴリズムにおいて，各プロセッサは，接頭部演算および，$O(\frac{n^2}{p})$ 回の内部計算と，$O(\frac{n}{\sqrt{p}})$ 回の通信を行う．
従って，以下の補題が成り立つ．

補題 16 各基底画素 $Base^N_{i,j}$に対して，候補画素 $Candidate^N_{i,j}$内の画素中の最近点は，BSP モデル上で p $(1 \leq p \leq n^2)$ プロセッサを用いて
内部計算時間 $O(\frac{n^2}{p} + T^{comp}_{prefix}(n,n,p))$，通信時間 $O(g\frac{n}{\sqrt{p}} + T^{comm}_{prefix}(n,n,p))$
で計算できる． □

補題 16より，以下の定理および系が示される．

定理 4 サイズ $n \times n$ の 2 値画像上の L_q距離全最近点問題は，BSP モデル上で，p $(1 \leq p \leq n^2)$ プロセッサを用いて
内部計算時間 $O(\frac{n^2}{p} + T^{comp}_{prefix}(n,n,p))$，通信時間 $O(g\frac{n}{\sqrt{p}} + T^{comm}_{prefix}(n,n,p))$
で計算できる． □

系 2 サイズ $n \times n$ の 2 値画像上の L_q距離全最近点問題は，BSP モデル上で，p $(1 \leq p \leq n^2)$ プロセッサを用いて
内部計算時間 $O(\frac{n^2}{p} + L)$, 通信時間 $O(g\frac{n}{\sqrt{p}} + L)$ $(n < p \leq n^2$の場合)
内部計算時間 $O(\frac{n^2}{p} + L\frac{\log\frac{p}{n}}{\log(d+1)})$，通信時間 $O(g\frac{n}{\sqrt{p}} + (gd+L)\frac{\log\frac{p}{n}}{\log(d+1)})$
$(n < p \leq n^2$の場合)
で求められる．ただし，d は $1 \leq d \leq \frac{p}{n}$を満たす任意の整数である． □

第 5 章

結論

本論文では，BSP モデルおよび BSP*モデルを用いて，選択問題および 2 値画像上の全最近点問題を解く最適加速な並列アルゴリズムを提案した．

第 3 章では，選択問題を解く以下の 2 つの決定性並列アルゴリズムを示した．

(1) BSP モデル上で内部計算時間 $O(\frac{n}{p} + d\log p\log\log n + L\frac{\log p\log\log n}{\log(d+1)})$，通信時間 $O(g\frac{n}{p} + (gd+L)\frac{\log p\log\log n}{\log(d+1)})$ のアルゴリズム．

(2) BSP*モデル上で内部計算時間 $O(\frac{n}{p} + d\log p\log\log n + L\frac{\log p\log\log n}{\log(d+1)})$，通信時間 $O(g(\frac{n}{pB} + (\frac{n}{p})^{\frac{1}{7}}(\log p)^{\frac{6}{7}}) + (gd+L)\frac{\log p\log\log n}{\log(d+1)})$ のアルゴリズム．

ただし d は $1 \le d \le \log n$ を満たす任意の整数であり，かつプロセッサ数 p は $1 \le p \le \frac{n}{\log n}$である．(1) のアルゴリズムの計算量は，$g = d = O(1)$ のとき内部計算時間，通信時間が共に $O(\frac{n}{p} + L\log p\log\log n)$ となり，プロセッサ数が小さい場合は，最適加速なアルゴリズムとなる．(2) のアルゴリズムも，$d = O(1),\ g \le B = O((\frac{n}{p\log p})^{\frac{6}{7}})$ の場合，内部計算時間，通信時間が共に $O(\frac{n}{p} + L\log p\log\log n)$ となり，プロセッサ数が小さい場合は，最適加速なアルゴリズムとなる．

第 4 章では，2 つの距離，重み付き距離および L_q距離に対して，全最近点問題を解く 2 つの決定性並列アルゴリムを示した．2 つのアルゴリズムの計算量は，共に BSP モデル上で p $(1 \le p \le n)$ プロセッサに対して，内部計算時間 $O(\frac{n^2}{p} + L)$，通信時間 $O(g\frac{n}{\sqrt{p}} + L)$ であり，また p $(n < p \le n^2)$ プロセッサに対して，内部計算時間 $O(\frac{n^2}{p} + (d+L)\frac{\log\frac{p}{n}}{\log(d+1)})$，通信時間 $O(g\frac{n}{\sqrt{p}} + (gd+L)\frac{\log\frac{p}{n}}{\log(d+1)})$ となる．ただし，d は $1 \le d \le \frac{p}{n}$を満たす任意の整数である．このアルゴリズムの計算量は，$g = O(\frac{n}{\sqrt{p}})$ のとき通信時間が内部計算時間に等しい $O(\frac{n^2}{p} + L)$ となり，プロセッサ数が小さい場合は，最適加速なアルゴリズムとなる．

BSP モデルを始めとする非同期式並列計算モデル上での並列アルゴリズムの研究は，まだ始まったばかりであり，多くの基本的な問題に対して，並列アルゴリズムがまだ提案されていない．よって，それら未解決の問題に対する効率の良い並列アルゴリズムを提案していくことが，今後の課題であると考えられる．

謝辞

本研究の機会を与えて下さるとともに，本研究の全過程を通じて，方針や問題点などのあらゆる面において，懇切丁寧な御指導と御助言を賜わりました藤原秀雄教授に心から感謝の意を表します.

本研究を進めるにあたり，適切な御指導を頂きました福田晃教授に感謝の意を表します.

本研究を進めるにあたり，貴重な御助言を頂きました伊藤実教授に感謝の意を表します.

本研究の全過程を通じて，研究方針を始め懇切丁寧な御討論，御助言を頂き，懇切丁寧に直接的な御指導を頂きました増澤利光助教授に深く感謝致します.

本研究の全過程を通じて，貴重な御討論，御助言を頂き，懇切丁寧に直接的な御指導を頂きました井上美智子助手に深く感謝致します.

本研究をまとめるに際し，様々な面でお世話になり, 御討論, 御協力頂きました九州工業大学藤原暁宏講師に深く感謝致します.

平素から多岐にわたって御助言を頂きました大竹哲史助手に深く感謝致します.

最後に，有益な御討論を頂きました藤原研究室の皆様に感謝いたします.

参考文献

[1] Bäumker, W.Dittrich, F.M.Heide and I.Rieping, "Realistic Parallel Algorithms:Priority Queue Operations and Selection for BSP* Model", *Proc. Euro-Par'96*, Vol.II, pp.369–376, 1996.

[2] R.J.Cole, "An Optimally Efficient Selection Algorithm", *Information Processing Letters*, Vol.26, No.6, pp.295–299, January 1988.

[3] D.Culler, R.Karp, D.Patterson, A.Sahay, K.E.Schauser, E.Santos, R.Subromanian and T. Von Eicken, "LogP:Towards a Realistic Model of Parallel Computation", *In Proceeding of Forth ACM SIGPLAN Symposium on Principles and Paractice of Parallel Programming,* pp.1–12, May 1993.

[4] F.Dehne, A.Fabri and A.Rau-Chaplin, "Scalable parallel computational geometry for coarse grained multicomputers", *Proc. ACM Symposium on Computational Geometry*, pp.298–307, 1998.

[5] A.Fujiwara, M.Inoue, T.Masuzawa and H.Fujiwara, "A cost optimal parallel algorithm for weighted distance transforms", *Parallel computing*, Vol.25, No.4, pp.405-416, 1999.

[6] A.V.Gerbessiotis and L.G.Valiant, "Direct Bulk-Syncronous Parallel Algrithms", *Journal of Parallel and Distributed Computing* Vol. 22, pp.251–267, 1994.

[7] A.V.Grebessioits and C.J.Siniolakis "Selection on the Bulk-Synchronous Parallel Model with Applications to Priority Queues", *In Proceedings of the 1996 International Conference on Parallel and Distributed Processing Techniques and Applications,* August 1996.

[8] A.Geist, A.Beguelin, J.Dongarra, W.Jiang, R.Mancheck and V.Sunderam. "PVM: Parallel Virtual Machine – A User's Guide and Tutorial for Networked Parallel Computing", *MIT Press*, 1994.

[9] M.T. Goodrich, "Communication-Efficient Parallel Sorting", *In Proceedings of the 28th ACM Symposium on Theory of Computing(STOC)*, 1996.

[10] W.Gropp, E.Lusk and A.Skjellum, "USING MPI:Portable Parallel Programming with the Message-Passing Interface", *MIT Press*, 1994.

[11] T.Hirata, "A unified linear-time algorithm for computing distance maps, " *Information Processing Letters* Vol. 58, pp.129–133, 1996.

[12] J. JáJá, "An Introduction to Parallel Algorithms", *Addison-Wesley Publishing Company,* 1992.

[13] B.H.H.Juurlink and H.A.G.Wijshoff, "Communication primitives for BSP computers, " *Information Processing Letters* 58, pp.303–310, 1996.

[14] D.W.Paglieroni, "Distance transforms: Properties and machine vision applications, " *CVGIP: Graphical Models and Image Processing* Vol.54, pp.56–74, 1992.

[15] F.P.Preparata and M.I.Shamos, "Computational Geometry: An Introduction," *Springer-Verlag*, New York, 1985.

[16] A.Schönhage, M.Peterson and N.Pippenger, "Finding the median", *Journal of Computer and System Science*, pp.184–199, 1976.

[17] L.G.Valiant, "A Bridging Model for Parallel Computation", *Communications of the ACM*, Vol.33, No.8, pp.103–111, August 1990.

[18] U.Vishkin. "An optimal parallel algorithm for selection", *Advances in Computiong Reserch, JAI Press Inc., Greenwich,* CT, 1987.