論　文

選択問題を解くBSPモデル及びBSP* モデル上の並列アルゴリズム

石水　隆[†]　藤原　暁宏[††]　井上美智子[†]　増澤　利光[†]
藤原　秀雄[†]

Parallel Algorithms for Selection on the BSP Model and the BSP* Model

Takashi ISHIMIZU[†], Akihiro FUJIWARA[††], Michiko INOUE[†],
Toshimitsu MASUZAWA[†], and Hideo FUJIWARA[†]

あらまし　本論文では，BSP（Bulk-Synchronous Parallel）モデル及びBSP* モデル上で選択問題を解く並列アルゴリズムを提案する．BSPモデル及びBSP* モデルは，近年提案された並列計算モデルであり，最近の並列計算において重要とされている通信コストを，同期周期 L，通信路帯域幅の逆数 g，パケットサイズ B といったパラメータにより表すことを可能にしたモデルである．本論文では，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 $(1 \leqq d \leqq \log n)$ に対し内部計算時間 $O(\frac{n}{p}+d\log p\log\log n+L\frac{\log p\log\log n}{\log d})$，通信時間 $O(g\frac{n}{p}+(gd+L)\frac{\log p\log\log n}{\log d})$，また，BSP* モデル上で内部計算時間 $O(\frac{n}{p}+d\log p\log\log n+L\frac{\log p\log\log n}{\log d})$，通信時間 $O(g(\frac{n}{pB}+(\frac{n}{p})^{\frac{1}{7}}(\log p)^{\frac{6}{7}})+(gd+L)\frac{\log p\log\log n}{\log d})$ の並列アルゴリズムを提案する．

キーワード　並列アルゴリズム，BSPモデル，選択問題，計算量

1. まえがき

従来の並列アルゴリズムに関する研究は，共有メモリ型並列計算モデルではPRAM（Parallel Random Access Machine），分散メモリ型並列計算モデルではメッシュモデル，ハイパキューブモデル等の特定のネットワーク構造をもつ並列計算モデルに関する研究が主流であった．初期の並列計算機の多くが，低ビットの並列処理専用に開発されたプロセッサを使って，短い周期で同期をとりながら処理を行うものであったため，これらの並列計算モデルでも1命令ごとの同期が仮定されることが多かった．また，初期の並列計算機では，プロセッサの演算能力が低かったこともあり，プロセッサ内部の演算に比べて，プロセッサ間の通信はそれほど考慮が必要とされず，上記の並列計算モデルにおいても，通信コストの表現には重点がおかれていなかった．

しかしながら，プロセッサ能力の向上に伴い，プロセッサ間の通信コストがプロセッサ内部の演算コストとともに，並列計算のコストにおける重要な要素となってきた．また，同時に多くのプロセッサが大部分の処理を他のプロセッサと同期せずに処理を行う非同期処理も主流となってきた．これらの特徴をもつ最近の並列計算機に対しては，PRAMを代表とする従来の並列計算モデルでは，アルゴリズムの評価を正確に行うことが困難であり，これらの特徴に対応した新しい並列計算モデルが望まれていた．

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[9]，及びその拡張モデルであるBSP* モデル[2]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，通信コストを同期周期，通信命令実行時間を表す L，g という二つのパラメータにより表すことが可能になっている．また同期機構を仮定することにより，非常に緩い同期の処理に対応可能なモデルである．BSP* モデルでは，通信パケットサイズを表すパラメータ 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

[†] 奈良先端科学技術大学院大学情報科学研究科，生駒市
Graduate School of Information Science Nara Institute of Science and Technology, 8916–5 Takayama, Ikoma-shi, 630–0101 Japan
[††] 九州工業大学情報工学部電子情報工学科，飯塚市
Department of Computer Science and Electronics, Kyushu Institute of Technology, 680–4 Kawazu, Iizuka-shi, 820–8502 Japan

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の提案を行う．選択問題とは，全順序関係をもつ n 個のデータの集合 S と自然数 k $(1 \leqq k \leqq n)$ 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．この選択問題に対しては，$O(n)$ 時間の最適逐次アルゴリズム [8] が知られている．

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 [3] は EREW PRAM 上で $O(\log n \log^* n)$ 時間(注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(注2)な並列アルゴリズムを示した．BSP モデル上では Gerbessiotis ら [5] 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 $(1 \leqq p \leqq n^{2/3+\zeta})$ 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 $0 < \zeta < \delta < 1/3$, $\rho > 1$ 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 $B \leqq \sqrt{\frac{n}{p}}$ 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．以上のように，BSP モデル，及び 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二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 $O(\frac{n}{p} + d \log p \log \log n + L\frac{\log p \log \log n}{\log d})$，通信時間 $O(g\frac{n}{p} + (gd + L)\frac{\log p \log \log n}{\log d})$ のアルゴリズム．

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

ただし d は $1 \leqq d \leqq \log n$ を満たす任意の整数であり，かつプロセッサ数 p は $1 \leqq p \leqq \frac{n}{\log n}$ 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，通信時間がともに $O(\frac{n}{p} + L \log p \log \log n)$ 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，$d = O(1)$, $g \leqq B = O((\frac{n}{p \log p})^{\frac{6}{7}})$ の場合，内部計算時間，通信時間がともに $O(\frac{n}{p} + L \log p \log \log n)$ 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. 準　　備

2.1 BSP モデル及び BSP* モデル

2.1.1 定　　義

BSP（Bulk-Synchronous Parallel）モデル [9] 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なる．

- 局所メモリをも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 $(1 \leqq i \leqq p)$ 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ステップの列からなる．各スーパステップは内部計算命令の列からなる内部計算フェーズと，送信命令，受信命令の列からなる通信フェーズで構成されており，各プロセッサはスーパステップの命令を非同期に実行する．また，スーパステップの命令を終了後，プロセッサ間でバリヤ同期(注3)をとり，次のスーパステップの実行に移る．メッセージの受信については，各スーパステップ中の通信フェーズで送信されたメッセージは同一のスーパステップの通信で受信されるが，そのメッセージはその次のスーパステップ以降でしか利用できないと仮定する．

BSP モデルは以下の二つのパラメータにより，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．

(注1)：$\log^* n = \min\{i | \log^{(i)} n \leqq 2\}$．ここで，$\log^{(i)} n = \log(\log^{(i-1)} n)$, $\log^{(1)} n = \log n$ である．

(注2)：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．

(注3)：バリヤ同期とは，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．バリヤ同期を実行して同期をとる場合，すべてのプロセッサがバリヤに到達するまでどのプロサッサも実行を継続できず，封鎖される．

- L：バリヤ同期周期
- g ($\leqq L$)：1 個の送信命令又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的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 個の送信命令又は受信命令の実行は g 単位時間で行われる．ただし，1 メッセージは 1 語からなるものとし，サイズ 1 のメッセージと呼ぶ．
- あるスーパステップにおいて，すべてのプロセッサで命令の実行を終了してから L 時間以内にバリヤ同期がとられ，次のスーパステップの実行に移る．よって，あるスーパステップにおいて，各プロセッサがたかだか w 個の内部計算命令，たかだか h 個の送信命令又は受信命令を割り当てられた場合，そのスーパステップの実行には $O(w+gh+L)$ 時間かかる．

以降では簡単のために，各スーパステップは内部計算命令のみ，あるいは送信命令及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ステップの実行時間を内部計算時間，送信命令及び受信命令のみからなるスーパステップの実行時間を通信時間と呼ぶ．

BSP* モデル [2] 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 モデルのパラメータに加えて以下のような通信パケットのサイズを表すパラメータをもつ．

- B ($\geqq 1$)：通信パケットの最小サイズ

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とおりである．

- 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又は受信できる．
- サイズ s のメッセージの送信命令又は受信命令の実行は $g\lceil\frac{s}{B}\rceil$ 単位時間で行われる．よって各プロセッサがたかだか w 個の内部計算命令，各サイズ $s_1, s_2, \ldots, s_h$ であるたかだか h 個のメッセージの送信命令又は受信命令からなるスーパステップの実行には，$O(w+g\sum_{i=1}^{h}\lceil\frac{s_i}{B}\rceil+L)$ 時間かかる．

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選択問題を解く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 モデル及び BSP* モデル上では以下のように定義される．以下の定義では要素 a，各 a_i，各 b_i 等はいずれもサイズ 1 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すべ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_1 が保持する．）

出力：すべ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） 入力：n 個の要素列 $(a_1, a_2, \ldots, a_n)$．各 P_i $(1 \leqq i \leqq p)$ が $(a_{\lceil(i-1)\frac{n}{p}\rceil+1}, a_{\lceil(i-1)\frac{n}{p}\rceil+2}, \ldots, a_{\lceil i\frac{n}{p}\rceil})$ を保持する．

出力：各 j $(1 \leqq j \leqq n)$ について，$b_j = a_1 \circ a_2 \circ \cdots \circ a_j$ を満たす要素列 $(b_1, b_2, \cdots, b_n)$．各 P_i $(1 \leqq i \leqq p)$ が $(b_{\lceil(i-1)\frac{n}{p}\rceil+1}, b_{\lceil(i-1)\frac{n}{p}\rceil+2}, \ldots, b_{\lceil i\frac{n}{p}\rceil})$ を保持する．

［定義 3］（ソーティング操作） ソーティング操作とは要素を昇順に並べ替える操作である．

入力：全順序関係をもつ n 個の要素集合 $A = \{a_1, a_2, \ldots, a_n\}$．各 P_i $(1 \leqq i \leqq p)$ が $(a_{\lceil(i-1)\frac{n}{p}\rceil+1}, a_{\lceil(i-1)\frac{n}{p}\rceil+2}, \ldots, a_{\lceil i\frac{n}{p}\rceil})$ を保持する．

出力：A のソート列 $(b_1, b_2, \ldots, b_n)$．各 P_i $(1 \leqq i \leqq p)$ が $(b_{\lceil(i-1)\frac{n}{p}\rceil+1}, b_{\lceil(i-1)\frac{n}{p}\rceil+2}, \ldots, b_{\lceil i\frac{n}{p}\rceil})$ を保持する．

表 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既知の結果を示す．以下では内部計算時間，通信時間をそれぞれ T_I，T_C と表す．

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 $f(n,p,g,L)$

表 1　BSP モデル上の基本操作に対する計算量
Table 1　The complexities for basic operations on the BSP model.

操作	時間計算量	プロセッサ数	文献
ブロードキャスト	$T_C : O((gd+L)\frac{\log p}{\log d})$	p	[9]
接頭部演算	$T_I : O((d+L)\frac{\log p}{\log d}+\frac{n}{p})$ $T_C : O((gd+L)\frac{\log p}{\log d})$	p $1 \leqq p \leqq n$	[9]
ソーティング	$T_I : O(\frac{n\log n}{p}+L\frac{\log n}{\log \frac{n}{p}})$ $T_C : O((g\frac{n}{p}+L)\frac{\log n}{\log \frac{n}{p}})$	p $1 \leqq p \leqq n$	[6]

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

で動作することが可能である，すなわち，BSP* モデルにも表 1 の結果が適用できる．

また，表 1 中の Goodrich のソーティングアルゴリズム [6] は，以下のような特性により，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 [6] は，各プロセッサは 1 スーパステップで $O(\frac{n}{p})$ 個の要素の送信，及び受信を行い，また，$O(\frac{\log n}{\log \frac{n}{p}})$ スーパステップで終了するので，BSP モデル上では，通信の計算量が $O((g\frac{n}{p}+L)\frac{\log n}{\log \frac{n}{p}})$ 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 スーパステップで送信するメッセージの送信先，及び受信するメッセージの送信元のプロセッサ数はたかだか $2(\frac{n}{p})^{\frac{1}{7}}$ である．BSP* モデルでは，1 スーパステップにおいて，d 個のプロセッサに s_i 個 ($1 \leqq i \leqq d$) ずつ送信するためには，$O(g\sum_{i=1}^{d}\lceil\frac{s_i}{B}\rceil+L)$ の通信時間しか必要としない．また，各プロセッサは，1 スーパステップで $O(\frac{n}{p})$ 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 スーパステップの通信時間は，

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

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

$$T_I : O\left(\frac{n\log n}{p}+L\frac{\log n}{\log\frac{n}{p}}\right)$$

$$T_C : O\left(\left(g\left(\frac{n}{pB}+\left(\frac{n}{p}\right)^{\frac{1}{7}}\right)+L\right)\frac{\log n}{\log\frac{n}{p}}\right)$$

で実行できる．

2.2 選択問題

全順序関係をもつ要素集合 $A=\{a_1,a_2,\ldots,a_n\}$ に対して，関数 $rank(a_i,A)$ $(1 \leqq i \leqq n)$ を

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

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

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

入力：全順序関係をもつ n 個の要素集合 $A=\{a_1, a_2,\ldots,a_n\}$，及び，整数 k $(1 \leqq k \leqq n)$．各 P_i $(1 \leqq i \leqq p)$ が $\{a_{\lceil(i-1)\frac{n}{p}\rceil+1}, a_{\lceil(i-1)\frac{n}{p}\rceil+2},\ldots,a_{\lceil i\frac{n}{p}\rceil}\}$ を，P_p が k を保持する．

出力：ある一つのプロセッサが $rank(a,A)=k$ を満たす要素 $a \in A$ を出力する．

3. 選択問題を解くアルゴリズム

3.1 BSP モデル上のアルゴリズム

3.1.1 アルゴリズムの概要

本アルゴリズムは Vishkin によって提案された EREW PRAM 上の並列アルゴリズム [10] をもとにしている．

選択問題はソーティング操作を用いて解くことができるのは明らかである．しかし，一般にソーティング操作の計算量は，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 1 のソーティングアルゴリズム [6] 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 [10] のアルゴリズムの方針を用いて，k 番目ではあ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 よりも小さいのものからなる集合，及び大きいもの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各要素の各中央値を求め，その各中央値の中央値を上記の要素 m として用いる．ここで，要素集合 $A=\{a_1,a_2,\ldots,a_n\}$ 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，後述のとおり $O(\log\log n)$ 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．

3.1.2 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考慮する必要がある．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 $(1\leqq i\leqq p)$ に n_i 個ずつ保持されているとする．ただし，$n=\sum_{j=1}^{p}n_j$ である．分配操作とは，各プロセッサがたかだか $\lceil\frac{n}{p}\rceil$ 個，少なくとも $\lfloor\frac{n}{p}\rfloor$ 個の要素を保持するよう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 $(1\leqq i\leqq p)$ は互いに素である集合 A の部分集合 A_i $(|A_i|=n_i, A_1\cup A_2\cup\ldots\cup A_p=A)$ を保持する．

出力：n 要素からなる集合 A．各 P_i $(1\leqq i\leqq p)$ は互いに素である A の部分集合 A'_i $(A'_1\cup A'_2\cup\ldots\cup A'_p=A)$ を保持する．ただし，各 A_i は $\lfloor\frac{n}{p}\rfloor\leqq|A'_i|\leqq\lceil\frac{n}{p}\rceil$ 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］

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

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

（3） 各 P_i $(1\leqq i\leqq p)$ において（2）で受信した要素の集合を A'_i とする．

［補題 2］ $n_{max}=\max\{n_1,n_2,\ldots,n_p\}$ とする．任意の定数 d $(1\leqq d\leqq p)$ に対して，先の分配操作は BSP モデル上で，

$$T_I: O\left(n_{max}+(d+L)\frac{\log p}{\log d}\right)$$

$$T_C: O\left(gn_{max}+(gd+L)\frac{\log p}{\log d}\right)$$

で実行できる．

（証明） p 要素の接頭部和演算操作は，表 1 のアルゴリズム [6] を用いて，

$$T_I: O\left((d+L)\frac{\log p}{\log d}\right),\ T_C: O\left((gd+L)\frac{\log p}{\log d}\right)$$

で実行できる．また，(2) において，各プロセッサはたかだか n_{max} 個の要素を送信し，たかだか $\lceil\frac{n}{p}\rceil$ $(\leqq n_{max})$ 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． □

3.1.3 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 $(1\leqq p\leqq\frac{n}{\log n})$ プロセッサのアルゴリズム *Selection* を示す．

［アルゴリズム *Selection*］

（1） 各 P_i $(1\leqq i\leqq p)$ 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）．

（2） $s>\frac{n}{\log n}$ ならば，$s\leqq\frac{n}{\log n}$ 以下のフェーズ (2.1)〜(2.6) を繰り返す．

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

（2.2） プロセッサ全体により，中央値集合 $\{m_1,m_2,\ldots,m_p\}$ 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．

（2.3） 各 P_i $(1\leqq i\leqq p)$ 上において，A_i の要素を以下のような二つの部分集合 A_i^1，A_i^2 に分割する．

$$A_i^1=\{x\in A_i|x<m\},\quad A_i^2=\{x\in A_i|x>m\}$$

（2.4） 各 P_i $(1\leqq i\leqq p)$ 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 $s^1=\sum_{j=1}^{p}|A_j^1|$ 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．

（2.5） 各 P_i $(1\leqq i\leqq p)$ 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_1 は m を出力し，アル

ゴリズムを停止する．

（2.6） 各プロセッサが保持する要素 A_i $(1 \leqq i \leqq p)$ 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．

（3） すべてのプロセッサを用いて，要素集合 $A_1 \cup A_2 \cup \ldots \cup A_p$ 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．

3.1.4 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，(2) の繰返し回数がたかだか $O(\log\log n)$ であることを示す．

以下では（2.1）から（2.6）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（2.4）終了時，$\frac{1}{6}s - 1 < s^1 < \frac{5}{6}s$ が成り立つ．

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

各反復フェーズの（2.4）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う．

（i） $s < 3p$ のとき

m は $\{m_1, m_2, \ldots, m_p\}$ 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，$rank(m, A) \geqq \lceil \frac{p}{2} \rceil > \frac{s}{6}$ 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 $s - rank(m, A) \geqq \lfloor \frac{p}{2} \rfloor > \frac{s}{6} - 1$ が成り立つ．

（ii） $s \geqq 3p$ のとき

m は $\{m_1, m_2, \ldots, m_p\}$ の中央値であるので，$\{m_1, m_2, \ldots, m_p\}$ 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 $(1 \leqq i \leqq p)$ 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 $s \leqq 3p$ より $\lceil \frac{p}{2} \rceil \lceil \frac{1}{2} \lfloor \frac{s}{p} \rfloor \rceil > \frac{p}{2}(\frac{s}{2p} - \frac{1}{2}) \geqq \frac{p}{2}(\frac{s}{2p} - \frac{s}{6p}) = \frac{s}{6}$ が成り立つので， $rank(m, A) > \frac{s}{6}$ が成り立つ．

同様に，$\{m_1, m_2, \ldots, m_p\}$ 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 $(1 \leqq i \leqq p)$ 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 $s - rank(m, A) > \frac{s}{6} - 1$ が成り立つ．

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（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 $s \leqq \frac{n}{\log n}$ となる． □

3.1.5 計 算 量

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

$$T_I : O\left(\frac{n}{p} + d\log p \log\log n + L\left(\frac{\log p \log\log n}{\log d}\right)\right)$$

$$T_C : O\left(g\frac{n}{p} + (gd + L)\frac{\log p \log\log n}{\log d}\right)$$

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．

（1）は各 P_i において定数個の内部計算であるので，$T_I : O(L)$ で実行できる．また，(3) はたかだか $\frac{n}{\log n}$ 個の要素のソーティング操作であり，表 1 のソーティングアルゴリズム [6] を用いると，

$$\begin{aligned} T_I &: O\left(\frac{\frac{n}{\log n}\log\frac{n}{\log n}}{p} + L\frac{\log\frac{n}{\log n}}{\log\frac{\frac{n}{\log n}}{p}}\right) \\ &= O\left(\frac{n}{p} + L\frac{\log\frac{n}{p\log n} + \log p}{\log\frac{n}{p\log n}}\right) \\ &= O\left(\frac{n}{p} + L\log p\right) \\ T_C &: O\left(\left(g\frac{\frac{n}{\log n}}{p} + L\right) \times \frac{\log\frac{n}{\log n}}{\log\frac{\frac{n}{\log n}}{p}}\right) \\ &= O\left(g\frac{n}{p} + L\log p\right) \end{aligned}$$

で実行できる．

したがって以下では（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）．

（2.1）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 [8] を用いて内部計算時間 $O(\frac{s}{p} + L)$ で求められる．また，(2.3) 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，

$$T_I : O\left(\frac{s}{p}+(d+L)\frac{\log p}{\log d}\right),$$
$$T_C : O\left(g\frac{s}{p}+(gd+L)\frac{\log p}{\log d}\right)$$

で実行できる．

（2.2）のソーティング操作は，以下のように実現する．各プロセッサ一つずつの要素をすべてのプロセッサを使って表 1 のアルゴリズム [6] 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 $\lceil\frac{p}{d}\rceil$[注4]個のプロセッサに集め，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．

（2.3.1） 各 P_i $(1 \leqq i \leqq p)$ は m_i を $P_{\lceil\frac{i}{d}\rceil}$ に送信する．

（2.3.2） P_i $(1 \leqq i \leqq \frac{p}{d})$ を用いて，$\{m_1, m_2, \ldots, m_p\}$ をソートする．

（2.3.3） $\{m_1, m_2, \ldots, m_p\}$ 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，(2.3) の計算量は，表 1 のソーティングアルゴリズム [6] 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，

$$T_I : O\left(\frac{p\log p}{\lceil\frac{p}{d}\rceil}+L\times\frac{\log p}{\log\frac{p}{\lceil\frac{p}{d}\rceil}}\right)$$
$$= O\left(d\log p+L\frac{\log p}{\log d}\right)$$
$$T_C : O\left(\left(g\frac{p}{\lceil\frac{p}{d}\rceil}+L\right)\times\frac{\log p}{\log\frac{p}{\lceil\frac{p}{d}\rceil}}\right)$$
$$= O\left((gd+L)\frac{\log p}{\log d}\right)$$

となる．

以上より（2）の 1 反復フェーズは

$$T_I : O\left(\frac{s}{p}+d\log p+L\frac{\log p}{\log d}\right),$$
$$T_C : O\left(g\frac{s}{p}+(gd+L)\frac{\log p}{\log d}\right)$$

である．

以下で，(2) 全体の計算量を考える．(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，(2) は $O(\log\log n)$ フェーズ繰り返されるので，(2) の計算量は，

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

となる．

仮定する $d \leqq \log n$ の範囲[注5]においては，$\log\log n \geqq \log d$ であり，(3) のソートの計算量より（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 $p\log p \leqq \frac{n\log d}{L\log\log n}$，$g = O(1)$ 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．

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

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

(注4)：d はブロードキャスト演算等の d と同じ値を用いる．
(注5)：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．

セッサはたかだかサイズ 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* では分配操作の代わ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擬似分配操作を用いることにより通信時間を改善する．

3.2.2 擬似分配操作

［定義 6］（擬似分配操作） n 個の要素が各 P_i $(1 \leqq i \leqq p)$ に n_i 個ずつ保持されているとする．ただし，$n = \sum_{j=1}^{p} n_j$ である．擬似分配操作とは，各プロセッサがたかだか $\frac{1}{2}\frac{n}{p} + \lceil \frac{n}{p} \rceil$ 個，少なくとも 1 個の要素を保持するよう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 $(1 \leqq i \leqq p)$ は互いに素である集合 A の部分集合 A_i $(|A_i| = n_i, A_1 \cup A_2 \cup \ldots \cup A_p = A)$ を保持する．

出力：n 要素からなる集合 A．各 P_i $(1 \leqq i \leqq p)$ は互いに素である A の部分集合 A'_i $(A'_1 \cup A'_2 \cup \ldots \cup A'_p = A)$ を保持する．ただし，各 A'_i は $1 \leqq |A'_i| \leqq \frac{1}{2}\frac{n}{p} + \lceil \frac{n}{p} \rceil$ を満たすものとする．

以下に BSP* モデル上での擬似分配操作のアルゴルズムを示す．

［擬似分配操作アルゴリズム］

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

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

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

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

（証明） 擬似分配操作において，各 P_i は $\{a_{\lceil (i-1)\frac{n}{p} \rceil + 1}, a_{\lceil (i-1)\frac{n}{p} \rceil + 2}, \ldots, a_{\lceil i\frac{n}{p} \rceil}\}$ の部分集合を受信し，このうち $a_{\lceil (i-1)\frac{n}{p} \rceil + 1}$ 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，$1 \leqq |A'_i| \leqq \frac{1}{2}\frac{n}{p} + \lceil \frac{n}{p} \rceil$ が成り立つ． □

［補題 6］ $n_{max} = \max\{n_1, n_2, \ldots, n_p\}$ とする．任意の定数 d $(1 \leqq d \leqq p)$ に対して，BSP* モデル上での擬似分配操作の計算量は

$$T_I : O\left(n_{max} + (d+L)\frac{\log p}{\log d}\right)$$

$$T_C : O\left(g\frac{n_{max}}{B} + (gd+L)\frac{\log p}{\log d}\right)$$

である．

（証明）（1），（3）は BSP モデル上の分配操作と同じで，$T_I : O(n_{max} + (d+L)\frac{\log p}{\log d})$，$T_C : O((gd+L)\frac{\log p}{\log d})$ である．

（2）において，各 P_i $(1 \leqq i \leqq p)$ 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 $P_{i'}, P_{i'+1}, \ldots, P_{i'+j-1}$ とする．P_i はたかだか n_{max} 個の要素を保持し，$P_{i'+1}, P_{i'+2}, \ldots, P_{i'+j-2}$ 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 はたかだか n_{max} 個の要素を一つのプロセッサに $\lceil \frac{n}{p} \rceil$ 個ずつ，たかだか $\frac{n_{max}}{\lfloor \frac{n}{p} \rfloor} + 2$ プロセッサに対して送信する．また，各 P_i はたかだか $\lceil \frac{n}{p} \rceil$ 個の要素をたかだか $\frac{\lceil \frac{n}{p} \rceil}{\frac{1}{2}\frac{n}{p}} + 2$ プロセッサから受信する．

よって

$$T_I : O(n_{max} + L)$$

$$T_C : O\left(g\left(\frac{n_{max}}{\lfloor \frac{n}{p} \rfloor} + 2 + \frac{\lceil \frac{n}{p} \rceil}{\frac{1}{2}\frac{n}{p}} + 2\right)\left\lceil \frac{n}{pB} \right\rceil + L\right)$$

$$= O\left(g\left(\frac{n_{max}}{B} + 1\right) + L\right)$$

である． □

3.2.3 アルゴリズム *Selection**

アルゴリズム *Selection* の（2.6）を以下の（2.6）′に変更する．また，（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，ダミー要素を加え要素を均等にするために，（3）の直前に以下の（3.0）を挿入する．

（2.6）′ すべてのプロセッサが保持する要素に対して擬似分配操作を行う．

（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．

3.2.4 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，補題 8 が成り立つ．

［補題 7］ *Selection** の各反復フェーズ中の（2.4）終了時，$\frac{1}{12}s+1<s^1<\frac{11}{12}s$ が成り立つ．

［補題 8］ *Selection** の（2）の反復フェーズ数は $O(\log\log n)$ である．

3.2.5 計　算　量

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

$$T_I : O\left(\frac{n}{p}+d\log p\log\log n+L\left(\frac{\log p\log\log n}{\log d}\right)\right)$$

$$T_C : O\left(g\left(\frac{n}{pB}+\left(\frac{n}{p}\right)^{\frac{1}{7}}(\log p)^{\frac{6}{7}}\right)+(gd+L)\frac{\log p\log\log n}{\log d}\right)$$

で選択問題を解く．

（証明） s を（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．

(2.3) 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，

$$T_I : O\left(d\log p+L\frac{\log p}{\log d}\right),$$

$$T_C : O\left(\left(g\left(\frac{d}{B}+d^{\frac{1}{7}}\right)+L\right)\frac{\log p}{\log d}\right)$$

で実行できる．

また，(2.6)′の擬似分配操作については，$\max\{|A_1|, |A_2|, \ldots, |A_p|\} = O(\frac{s}{p})$ なので，補題 6 より，

$$T_I : O\left(\frac{s}{p}+(d+L)\frac{\log p}{\log d}\right)$$

$$T_C : O\left(g\frac{s}{pB}+(gd+L)\frac{\log p}{\log d}\right)$$

で実行できる．

以上より，（2）の 1 反復フェーズは

$$T_I : O\left(\frac{s}{p}+d\log p+L\frac{\log p}{\log d}\right)$$

$$T_C : O\left(g\frac{s}{pB}+(gd+L)\frac{\log p}{\log d}\right)$$

で実行できるので，（2）全体の計算量は

$$T_I : O\left(\frac{n}{p}+d\log p\log\log n+L\frac{\log p\log\log n}{\log d}\right)$$

$$T_C : O\left(g\frac{n}{pB}+(gd+L)\frac{\log p\log\log n}{\log d}\right)$$

となる．

また，(3.0) については，たかだか $\frac{3}{2}\frac{n}{p\log n}$ 個のダミー要素の挿入であるから，$T_I : O(\frac{n}{p\log n}+L)$ である．

最後の（3）については，$\frac{n}{\log n}$ 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，

$$T_I : O\left(\frac{n}{p}+L\log p\right)$$

$$T_C : O\left(g\left(\frac{n}{pB}+\left(\frac{n}{p}\right)^{\frac{1}{7}}(\log p)^{\frac{6}{7}}\right)+L\log p\right)$$

となる． □

定理 2 より，*selection** は $p\log p \leqq \frac{n\log d}{L\log\log n}$，$g = O(B)$，$B \leqq (\frac{n}{p\log p})^{\frac{6}{7}}$ 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．

4. むすび

本論文では選択問題を解く以下の二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 $O(\frac{n}{p}+d\log p\log\log n+L\frac{\log p\log\log n}{\log d})$，通信時間 $O(g\frac{n}{p}+(gd+L)\frac{\log p\log\log n}{\log d})$ のアルゴリズム．

（2） BSP* モデル上で内部計算時間 $O(\frac{n}{p}+d\log p\log\log n+L\frac{\log p\log\log n}{\log d})$，通信時間 $O(g(\frac{n}{pB}+(\frac{n}{p})^{\frac{1}{7}}(\log p)^{\frac{6}{7}})+(gd+L)\frac{\log p\log\log n}{\log d})$ のアルゴリズム．ただし d は $1 \leqq d \leqq \log n$ を満たす任意の整数であり，かつプロセッサ数 p は $1 \leqq p \leqq \frac{n}{\log n}$ 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，通信時間がともに $O(\frac{n}{p}+L\log p\log\log n)$ 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，$d = O(1)$，$g \leqq B = O((\frac{n}{p\log p})^{\frac{6}{7}})$ の場合，内部計算時間，通信時間がともに $O(\frac{n}{p}+L\log p\log\log n)$ 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．

今後の課題としては p，g の制限を考慮し，より大きな p，g に対して最適加速なアルゴリズムを得ることが考えられる．

文　献

[1] A. 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] A. Bäumker, W. Dittrich, and F.M. Heide, "Truly efficient parallel algorithms: 1-optimal multisearch for an extension of the BSP model," Technical Report TR-RSFB-96-008, University Pederborn, Department of Mathematics and Computer Science, Jan. 1996.

[3] R.J. Cole, "An optimally efficient selection algorithm," Inf. Proc. Letters, vol.26, no.6, pp.295–299, Jan. 1988.

[4] A.V. Gerbessiotis and L.G. Valiant, "Direct bulk-synchronous parallel algorithms," Journal of Parallel and Distributed Computing, vol.22, pp.251–267, 1994.

[5] A.V. Gerbessioits and C.J. Siniolakis, "Selection on the bulk-synchronous parallel model with applications to priority queues," Proc. of the 1996 International Conference on Parallel and Distributed Processing Techniques and Applications, Aug. 1996.

[6] M.T. Goodrich, "Communication-efficient parallel sorting," Proc. of the 28th ACM Symposium on Theory of Computing (STOC), pp.247–256, May 1996.

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

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

[9] L.G. Valiant, "A bridging model for parallel computation," Commun. ACM, vol.33, no.8, pp.103–111, Aug. 1990.

[10] U. Vishkin, "An optimal parallel algorithm for selection," Advances in Computing Research, JAI Press Inc., Greenwich, CT, 1987.

（平成 10 年 5 月 14 日受付，10 月 21 日再受付）

石水　隆　（学生員）

平 9 奈良先端大大学院博士前期課程了．工修．現在同大大学院博士後期課程在学中．並列アルゴリズムの研究に従事．

藤原　暁宏　（正員）

平 5 阪大・基礎工・情報卒．平 9 奈良先端大大学院博士後期課程了．工博．同年九工大・情報工・電子情報講師，現在に至る並列分散アルゴリズム，並列計算モデル，計算機クラスタなどの研究に従事．IEEE，情報処理学会各会員．

井上美智子　（正員）

昭 62 阪大・基礎工・情報卒．平 1 同大大学院博士前期課程了．同年（株）富士通研究所入社．平 7 阪大大学院博士後期課程了．同年奈良先端科学技術大学院大学助手．現在に至る．分散アルゴリズム，並列アルゴリズム，グラフ理論，テスト容易化設計，高位合成の研究に従事．工博．IEEE，情報処理学会，人工知能学会各会員．

増澤　利光　（正員）

昭 57 阪大・基礎工・情報卒．昭 62 同大大学院博士後期課程了．同年同大情報処理教育センター助手．同大基礎工助教授を経て，平 6 奈良先端大情報科学助教授，現在に至る．平 5 コーネル大客員準教授（文部省在外研究員）．分散アルゴリズム，並列アルゴリズム，テスト容易化設計，テスト容易化高位合成に関する研究に従事．ACM，IEEE，EATCS，情報処理学会各会員．工博．

藤原　秀雄　（正員）

昭 44 阪大・工・電子卒．昭 46 同大大学院博士後期課程了．阪大工学部助手，明治大理工学部教授を経て，現在奈良先端大情報科学教授．昭 56 ウォータールー大客員助教授．昭 59 マッギル大客員準教授．論理設計，高信頼設計，設計自動化，テスト容易化設計，テスト生成，並列処理，計算複雑度に関する研究に従事．著書 "Logic Testing and Design for Testability"（The MIT Press）など．大川出版賞．IEEE，情報処理学会各会員．工博．IEEE Fellow．