アルゴリズム 21－1
（1991. 5. 29）

行および列のソートのみを用いたメッシュ計算機上での確率的並列ソーティングアルゴリズム

長谷川 進　今井 浩　博田 浩司

東京大学理学部情報科学科

アブストラクト

$n \times n$ のメッシュ計算機上で、行と列を交互にソートすることによってソーティングを行なう単純で実用的なアルゴリズムについて考える。このアルゴリズムは、行と列のソートを $O(\log\log n)$ 回行なうことによりソートを完成させる。したがって全体では $O(n\log\log n)$ 並列時間でソートを行なう。$O(n)$ 並列時間でソートを行なうアルゴリズムは他に提案されているが、これらは、行と列の複雑な操作を必要とする。一方、$O(n\log\log n)$ 並列時間で行や列の他にシフト操作を使ってソートを行なうアルゴリズムも提案されている。これらのアルゴリズムにくらべ、この論文で扱うアルゴリズムはより単純で実用的である。また、我々の得た結果は最悪 $O(n\log n)$ でソートを完成させる shear ソートと呼ばれる deterministic アルゴリズムの確率的解析にもなっている。

Analysis of Randomized Sorting Algorithms on the Mesh Computer Using Only Row and Column Sorting

Susumu Hasegawa, Hiroshi Imai and Koji Hakata

Department of Information Science, University of Tokyo
Hongo, Tokyo 113, Japan

Abstract

Simple and practical randomized sorting algorithms on the $n \times n$ mesh computer, which alternately sort its rows and columns, are investigated. It is shown that these algorithms require $O(\log\log n)$ iterations of row and column sorting operations, and $O(n\log\log n)$ parallel time. Although there exist several $O(n)$ parallel time algorithms (e.g., [5,6]), such algorithms require non-uniform operations for rows and columns and are complicated. There is an $O(n\log\log n)$ parallel time algorithm which uses row and column sorting as well as shifting operations only [5]. Compared with it, the randomized algorithms given here are simpler and more practical. Furthermore, our results can be viewed as a probabilistic analysis of an $O(n\log n)$ deterministic time sorting algorithm, called shear sort [5], which is shown to run faster on the average.

1. Introduction

Sorting is the most fundamental and useful problem in computer science, and there have been developed many efficient sequential and parallel algorithms for sorting. In this paper, we are interested in parallel sorting algorithms which are simple enough and useful in practice. Parallel algorithms depends highly upon parallel computation models. PRAM is the most powerful model, and is widely used as a standard model to discuss the inherent parallel complexity. In 1980's, optimal parallel sorting algorithms for N numbers are given which run in $O(\log N)$ time using $O(N)$ processors on PRAM (e.g., [1]). However, these algorithms are fairly complicated, and have large constant factors in the time complexity function. Also, PRAM is too powerful to realize as it is.

This paper aims at developing simple sorting algorithms on a much simpler parallel computation model, the two-dimensional mesh connected computer. This model is often slower than other machines such as the hypercube, but is ideally suited to VLSI implementation. Considering VLSI implementation, the simplicity of the algorithm is quite important.

For the two-dimensional mesh computer, there have been proposed many sorting algorithms. Thompson and Kung [6] give the first $O(n)$ parallel time algorithm on the $n \times n$ mesh machine. Later, several $O(n)$ time algorithms are proposed, among which Schnorr and Shamir [5] present an optimal algorithm with $3n + o(n)$ parallel steps. However, the former algorithm is based on the divide-and-conquer principle, and is rather complicated. The latter algorithm is non-recursive, but its low order term is relatively large for moderate values of n.

Schnorr and Shamir [5] also present two simple algorithms which use only row and column operations. Roughly, these algorithms alternately sort the rows and columns. Since sorting rows (columns) can be performed in a completely synchronized manner and there are only sorting operation as well as its slight variants for each row and column, the algorithms are very simple. Among the two algorithms, one is called shear sort, and requires $\lceil \log n \rceil$ steps of alternately sorting rows and columns. The other is called revsort, and requires $\lceil \log\log n \rceil$ steps plus 3 iterations of a step of the shear sort. The revsort additionally uses shift operations for rows and columns. Iwama, Miyano and Kambayashi [2] describes a randomized sorting algorithm, which may be viewed as a randomized version of the shear sort, without any analysis on it. They give a complicated randomized algorithm by extending the algorithm, and analyze it by approximating the probabilistic distribution, to be explained in section 2, by a simple binomial distribution.

This paper considers randomized sorting algorithms on the mesh machine, and analyzes them. A randomized algorithm of alternately sorting (or randomizing) rows and sorting columns is first introduced. This algorithm together with the randomized algorithm in [2] are then investigated in detail. It is shown that these algorithms require $O(\log\log n)$ iterations of row and column sorting operations. Compared with the revsort, the randomized algorithms given here do not need shift operations but use randomization. Our results can be regarded as a probabilistic analysis of the shear sort. While the shear sort requires $O(\log n)$ deterministic steps, these randomized algorithms only need $O(\log\log n)$ steps. It is also observed by the analysis as well as computational experiments that, if an initial configuration of n^2 numbers on the mesh may be considered to be distributed randomly, the shear sort performs quite better in practice.

2. Algorithms and Preliminaries

We now describe the sorting algorithms on the mesh machine. Initially, n^2 numbers are given, one at each processor of the $n \times n$ mesh computer. We consider sorting these numbers in the row-major snake-like order as shown in Figure 2.1(e). Rows (columns) can be sorted simultaneously in $O(n)$ parallel time. In the sequel, we suppose that sorting rows (columns) can be performed in a unit time, and will be concerned with the number of times row and column operations are executed.

The shear sort [5], to be called Algorithm S, can be described as follows:

Algorithm S: [5]

```
1. Shear the rows (sort odd-numbered rows to the right and sort even-numbered rows to the left);
   Check if sorting is done; if yes, halt;
2. Sort each column from the top to the bottom;
   Return to 1;
```

An example is given in Figure 2.1. Within $\lceil \log n \rceil$ iterations, Algorithm S completes sorting [5], where throughout the paper log and ln denote the logarithm of base 2 and e, respectively.

A new randomized sorting algorithm, Algorithm R, is given below. Here, by randomizing rows, it is meant to rearrange numbers in each row at random. This may be done by producing a random number at each element, and rearrange numbers in a sorted order of these random numbers in each row. This randomized row operation may thus be regarded as a kind of sorting operation.

Algorithm R:
After repeating 1 and 2 $\lceil \log\log n \rceil$ times, switch to Algorithm S.

$$\text{(randomize rows)} \Longrightarrow \underset{\text{(a)}}{\begin{matrix} 3 & 16 & 12 & 4 \\ 1 & 9 & 5 & 14 \\ 11 & 8 & 2 & 7 \\ 10 & 6 & 13 & 15 \end{matrix}} \;\text{(column sort)} \Longrightarrow \underset{\text{(b)}}{\begin{matrix} 1 & 6 & 2 & 4 \\ 3 & 8 & 5 & 7 \\ 10 & 9 & 12 & 14 \\ 11 & 16 & 13 & 15 \end{matrix}} \;\text{shear} \Longrightarrow$$

$$\underset{\text{(c)}}{\begin{matrix} 1 & 2 & 4 & 6 \\ 8 & 7 & 5 & 3 \\ 9 & 10 & 12 & 14 \\ 16 & 15 & 13 & 11 \end{matrix}} \;\text{column sort} \Longrightarrow \underset{\text{(d)}}{\begin{matrix} 1 & 2 & 4 & 3 \\ 8 & 7 & 5 & 6 \\ 9 & 10 & 12 & 11 \\ 16 & 15 & 13 & 14 \end{matrix}} \;\text{shear} \Longrightarrow \underset{\text{(e)}}{\begin{matrix} 1 & 2 & 3 & 4 \\ 8 & 7 & 6 & 5 \\ 9 & 10 & 11 & 12 \\ 16 & 15 & 14 & 13 \end{matrix}}$$

Figure 2.1. How Algorithms S and R proceed (the last three operations are for S and the first two for R which switches to S after 1 iteration)

1. Randomize each row;
2. Sort each column from the top to the bottom;

The following is the randomized algorithm given in [2].

Algorithm RS: [2]

```
1. Randomize each row;
2. Sort each column from the top to the bottom;
3. Shear rows;
   Check if sorting is done; if yes, halt;
4. Sort each column from the top to the bottom;
   Return to 1;
```

We will analyze the complexity of these two randomized algorithms. Using the zero-one principle in Knuth [3] as usual, we will reduce the analysis of sorting n^2 general numbers to that of sorting n^2 binary numbers as follows, where its weaker statement is made in [2].

Lemma 2.1. If n^2 binary numbers (0 or 1) can be sorted on the mesh machine using row sorting in the row-major snake-like order in $t(n)$ time with probability $1 - p(n)$, the general sorting problem can be solved in $t(n)$ time with probability $1 - np(n)$.

Proof: Let a_i be the maximum among numbers in row i after sorting is done. If each a_i $(i = 1, \ldots, n)$ is in its sorted position, sorting each row finishes the whole sorting. Define a function $f_{a_i}(x)$ by

$$f_{a_i}(x) = \begin{cases} 0 & x \le a_i \\ 1 & x > a_i \end{cases}$$

Then, a_i is in the correct position if $f_{a_i}(x)$ for all n^2 numbers x are sorted correctly. For example,

$$\begin{matrix} 1 & 2 & 3 & 4 \\ 8 & 7 & 6 & 5 \\ 9 & 10 & 11 & 12 \\ 16 & 15 & 14 & 13 \end{matrix} \quad \underset{\Longrightarrow}{f_8(x) = \begin{cases} 0 & x \le 8 \\ 1 & x > 8 \end{cases}} \quad \begin{matrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{matrix}$$

We thus have the lemma, where about the probability we use Lemma 2.2 below. □

Below are summarized fundamentals on probability to be used in the paper. First, for an event A in a probabilistic space, $\Pr(A)$ denotes the probability of A. For a random variable X, $\mathrm{E}[X]$ and $\mathrm{V}[X]$ denote the expectation and variance, respectively, of X. The following lemma will often be used implicitly.

Lemma 2.2. For events $A_1, A_2, \ldots, A_n$ in a probabilistic space (not necessarily disjoint),

$$\Pr(\bigcup_{j=1}^{n} A_j) \le \sum_{j=1}^{n} \Pr(A_j). \quad \square$$

Lemma 2.3. (Chebyshev inequality)

$$\Pr(|X - \mathrm{E}[X]| > t\sqrt{\mathrm{V}[X]}) < \frac{1}{t^2}. \quad \square$$

Lemma 2.4. (Chernoff bound; e.g., see [4]) Let $X_1, X_2, \ldots, X_n$ be independent Bernoulli trials with $\Pr(X_i = 1) = p_i$, $\Pr(X_i = 0) = 1 - p_i$, $0 < p_i < 1$. Let $X = \sum_{i=1}^{n} X_i$ and $\mu = \sum_{i=1}^{n} p_i$. Then

$$\Pr(X > \mu + \delta\mu) < \frac{\exp(\delta\mu)}{(1+\delta)^{(1+\delta)\mu}} < \begin{cases} \exp(-\mu\delta^2/4) & 0 < \delta \le 4.1 \\ 2^{-(1+\delta)\mu} & \delta > 2e - 1 \end{cases}$$

$$\Pr(X < \mu - \delta\mu) < \exp(-\frac{\mu\delta^2}{2}). \quad \square$$

3. Analysis on Algorithm R

Based on Lemma 2.1, we will investigate the problem of sorting n^2 binary numbers, 0 and 1, on the $n \times n$ mesh machine. We call a row *dirty* if there are both 0 and 1 in the row. A row which is not dirty is called *clean*. If the total number of dirty rows becomes at most one, sorting is done.

By the algorithms considered here, once a row becomes clean, the row will never become dirty again. Also, regarding all the rows in the beginning as dirty, all dirty rows are consecutive at any time due to column sorting. Lower and upper rows become clean first, and then middle rows become clean. Hence, at each iteration, it suffices to evaluate how the number of dirty rows will be reduced. We extract a submesh consisting of dirty rows at each stage, and consider the sorting problem for the submesh. Let m be the number of dirty rows at some stage, i.e., the number of rows in the submesh. Initially, m is n.

Consider the $m \times n$ submesh computer. Rows are re-indexed from 1 to m. In the beginning of the stage, let n_i be the number of 1's in row i, and let $p_i = n_i/n$. By randomizing rows, the binary numbers are distributed at each row and events on different rows are independent.

The probabilistic behavior of these algorithms can be modeled as follows. Let X_{ij} be a random variable, taking 0 or 1 as its value, at the (i,j)-element of the mesh computer. Then, in row i, for any subset S_i of n_i elements of $\{1, 2, \ldots, n\}$,

$$\Pr\left(X_{ij} = \begin{cases} 1 & \text{for } j \in S_i \\ 0 & \text{for } j \notin S_i \end{cases}\right) = \frac{1}{\binom{n}{n_i}}.$$

It should be noted that X_{ij} $(j = 1, \ldots, n)$ for fixed i are not independent to one another, while X_{ij} $(i = 1, \ldots, m)$ for fixed j are independent. Defining Y_j by

$$Y_j = \sum_{i=1}^{m} X_{ij},$$

Y_j is the sum of independent Bernoulli trials $X_1, \ldots, X_n$ with $\Pr(X_i = 1) = p_i = n_i/n$. Define μ and p by

$$p = \frac{1}{m}\sum_{i=1}^{m} p_i, \qquad \mu = mp = \sum_{i=1}^{m} p_i.$$

Then,

$$E[Y_j] = \mu = mp.$$

Without loss of generality, we can assume $0 < p \le 1/2$. Let $Y_{\max}$, Y_{med} and $Y_{\min}$ be the maximum, median and minimum, respectively, among Y_j $(j = 1, \ldots, n)$. Then, after an iteration of Algorithm R, the number of dirty rows is $Y_{\max} - Y_{\min} + 1$. We will estimate $Y_{\max}$ and $Y_{\min}$ by using the Chernoff bound.

Lemma 3.1. For any positive constant c,

$$\Pr(Y_j < \mu - c\sqrt{\mu \ln n}) < n^{-c^2/2}.$$

Proof: Apply the Chernoff bound with $\delta = c\sqrt{\ln n/\mu}$. $\square$

Let $\overline{\mu} = m - \mu = m(1-p)$. Applying Lemma 3.1 for a random variable $\overline{Y}_j = 1 - Y_j$ with expectation $\overline{\mu}$, we have the following.

Lemma 3.2. For any positive constant c,

$$\Pr(Y_j > \mu + c\sqrt{\overline{\mu} \ln n}) < n^{-c^2/2}. \quad \square$$

Lemma 3.3. An iteration of randomizing rows and sorting columns in Algorithm R reduces the number m of dirty rows to $\sqrt{(8+4\epsilon)m \ln n}$ with probability at least $1 - 2n^{-1-\epsilon}$.

Proof: Setting $c = \sqrt{4+2\epsilon}$ in Lemmas 3.1 and 3.2, and applying Lemma 2.2 with considering that here the union of $2n$ events are taken, we obtain the following for any positive constant ϵ:

$$\begin{aligned} &\Pr((Y_{\min} < \mu - \sqrt{(4+2\epsilon)\mu \ln n}) \cup (Y_{\max} > \mu + \sqrt{(4+2\epsilon)\overline{\mu} \ln n})) \\ &= \Pr(\bigcup_{j=1}^{n}[(Y_j < \mu - \sqrt{(4+2\epsilon)\mu \ln n}) \cup (Y_j > \mu + \sqrt{(4+2\epsilon)\overline{\mu} \ln n})]) \le 2n^{-1-\epsilon}. \end{aligned}$$

This implies that, by an iteration of Algorithm R, m is reduced to

$$\sqrt{(4+2\epsilon)m \ln n}(\sqrt{p} + \sqrt{1-p}) \le \sqrt{(8+4\epsilon)m \ln n}$$

with probability at least $1 - 2n^{-1-\epsilon}$, where the last inequality follows from Lemma 3.4 below. $\square$

Lemma 3.4. For $0 < a \le b$ and $0 \le \alpha \le \beta \le 1$,

$$\max\{\sqrt{ax} + \sqrt{b(1-x)} \mid \alpha \le x \le \beta\} = \begin{cases} \sqrt{a\alpha} + \sqrt{b(1-\alpha)} & a/(a+b) < \alpha \\ \sqrt{a+b} & \alpha \le a/(a+b) \le \beta \\ \sqrt{a\beta} + \sqrt{b(1-\beta)} & a/(a+b) > \beta. \end{cases} \quad \square$$

Theorem 3.1. For $0 < \epsilon < 0.8$, after $2\lceil \log\log n \rceil + 4$ total iterations of alternately randomizing or sorting rows and sorting columns (including iterations after switching to Algorithm S), Algorithm R completes sorting n^2 binary numbers with probability at least $1 - 2\lceil \log\log n \rceil n^{-1-\epsilon}$.

Proof: Let m_i be the number of dirty rows after i iterations in Algorithm R, where $m_0 \equiv n$. Then, by Lemma 3.3,

$$\begin{aligned} m_i &\le \sqrt{(8+4\epsilon)m_{i-1}\ln n} \qquad (i \ge 1) \\ &\le ((8+4\epsilon)\ln n)^{1/2+1/2^2+\cdots+1/2^i} n^{1/2^i} \le (8+4\epsilon) n^{1/2^i} \ln n. \end{aligned}$$

For $i = \lceil \log\log n \rceil$,

$$m_i \le 2(8+4\epsilon)\ln n.$$

After switching Algorithm S, from Lemma 4.1 below, it takes $\log((16+8\epsilon)\ln n) < 4 + \log\log n$ iterations for $\epsilon < 0.8$. $\square$

Applying Lemma 2.1 to this theorem, we have the following.

Corollary 3.1. To sort n^2 general numbers, Algorithm R requires at most $2\lceil \log\log n \rceil + 4$ iterations of sorting (randomizing) rows and columns with probability at least $n^{-\epsilon}$ for $0 < \epsilon < 0.8$. $\square$

4. Analysis of Algorithm S

The following lemma for Algorithm S is mentioned in [5].

Lemma 4.1. [5] By shearing rows and sorting columns, the number m of dirty rows is reduced to $\lceil m/2 \rceil$. $\square$

The proof of this lemma may be obtained from the following by conceptually dividing the shearing step into two: first, sort rows to the right, and then reverse even-numbered rows.

Lemma 4.2. For a nondecreasing real-valued function f on $I \equiv [-a, a]$, define g on I by

$$g(x) = \frac{f(x) + f(-x)}{2}.$$

Then,

$$(\max_{x \in I} g(x)) - (\min_{x \in I} g(x)) \le (f(a) - f(-a))/2. \quad \square$$

In our problem, $f(a)$ and $f(-a)$ correspond to $Y_{\max}$ and $Y_{\min}$, respectively, and the left hand side $(\max g(x)) - (\min g(x))$ corresponds to the number of dirty rows after shearing rows and sorting columns. We can sharpen this lemma to the following.

Lemma 4.3.

$$(\max_{x \in I} g(x)) - (\min_{x \in I} g(x)) \le \max\{f(a) - f(0),\ f(0) - f(-a)\}/2.$$

Proof: Let $\alpha \in I$ minimize g on I. As $g(x) = g(-x)$ holds for all $x \in I$, we can assume $\alpha \ge 0$. Let

$$\begin{aligned} J_1 &\equiv [-a, -\alpha] \cup [\alpha, a] \\ J_2 &\equiv [-\alpha, \alpha]. \end{aligned}$$

If $x \in J_1$,

$$g(x) - g(\alpha) = \frac{f(x) + f(-x)}{2} - g(\alpha) \le \frac{f(a) + f(-\alpha)}{2} - g(\alpha).$$

If $x \in J_2$,

$$g(x) - g(\alpha) = \frac{f(x) + f(-x)}{2} - g(\alpha) \le \frac{f(\alpha) + f(0)}{2} - g(\alpha).$$

Therefore,

$$\begin{aligned} \max_{x \in I} g(x) - \min_{x \in I} g(x) &\le \max\{\frac{f(a) + f(-\alpha)}{2} - g(\alpha),\ \frac{f(\alpha) + f(0)}{2} - g(\alpha)\} \\ &= \max\{\frac{f(a) - f(\alpha)}{2},\ \frac{f(0) - f(-\alpha)}{2}\} \\ &\le \max\{\frac{f(a) - f(0)}{2},\ \frac{f(0) - f(-a)}{2}\}. \quad \square \end{aligned}$$

That is, if $f(0) = (f(a)+f(-a))/2$, $(\max g(x))-(\min g(x)) \leq (f(a)-f(-a))/4$. Here, $f(0)$ corresponds to the median $Y_{\rm med}$ among Y_j. Therefore, if the median $Y_{\rm med}$ is shown to be near the middle between $Y_{\max}$ and $Y_{\min}$, the reduction factor may be estimated more nicely. This will be done in the next section.

5. Analysis on Algorithm RS

Before estimating the median $Y_{\rm med}$ among Y_j, we first analyze Algorithm RS only using the results in sections 3 and 4. ¿From Lemmas 3.3 and 4.1, we have the following.

Lemma 5.1. An iteration of $1 \sim 4$ in Algorithm RS reduces m to $\sqrt{(2+\epsilon)m\ln n}$ with probability at least $1-2n^{-1-\epsilon}$. □

We will improve this lemma by giving a bound on the median $Y_{\rm med}$. This elucidates the probabilistic behavior of the shear sort, Algorithm S. Recall that $Y_{\max}$, $Y_{\rm med}$ and $Y_{\min}$ are the maximum, median and minimum, respectively, among Y_j $(j=1,\ldots,n)$. We first analyze $Y_{\rm med}$.

For column j and a constant k with $0 \leq k < \mu$, define a random variable $Z_{k,j}$ by

$$Z_{k,j} = \begin{cases} 1 & Y_j \leq k \\ 0 & Y_j > k \end{cases}$$

Define a random variable Z_k by

$$Z_k = \sum_{j=1}^{n} Z_{k,j}.$$

If $\Pr(Z_k \geq n/2)$ is small, $Y_{\rm med}$ is greater than k with high probability. Let $q_k = \Pr(Y_j \leq k)$ which is independent of j.

Lemma 5.2. $\mathrm{E}[Z_k] = nq_k$, and, for $k = \mu - \delta\mu$ for $0 < \delta < 1$,

$$\mathrm{E}[Z_k] < n\exp(-\mu\delta^2/2), \qquad \mathrm{V}[Z_k] < n\exp(-\mu\delta^2/2) + n^2\exp(-\mu\delta^2).$$

Proof:

$$\mathrm{E}[Z_k] = \mathrm{E}[\sum_{j=1}^{n} Z_{k,j}] = \sum_{j=1}^{n}\mathrm{E}[Z_{k,j}] = nq_k.$$

By the Chernoff bound,

$$q_k = \Pr(Y_j \leq \mu - \delta\mu) \leq \exp(-\mu\delta^2/2)$$

and we obtain the bound for $\mathrm{E}[Z_k]$. For $\mathrm{V}[Z_k]$, we have

$$\begin{aligned}\mathrm{V}[Z_k] &= \mathrm{E}[(\sum_{j=1}^{n} Z_{k,j} - \mathrm{E}[Z_k])^2] \\ &= n\mathrm{E}[Z_{k,j}^2] + n(n-1)\mathrm{E}[Z_{k,j}Z_{k,j'}] - \mathrm{E}[Z_k]^2 \quad \text{for distinct } j \text{ and } j' \\ &= nq_k + n(n-1)\mathrm{E}[Z_{k,j}Z_{k,j'}] - (nq_k)^2.\end{aligned}$$

By the definition of $Z_{k,j}$,

$$\mathrm{E}[Z_{k,j}Z_{k,j'}] = \Pr((Y_j \leq k)\cap(Y_{j'} \leq k)) = \Pr(Y_{j'} \leq k \mid Y_j \leq k)q_k,$$

where $\Pr(\cdot \mid \cdot)$ is the conditional probability. Let $\mu' = \mathrm{E}[Y_{j'} \mid Y_j \leq k]$, where $\mathrm{E}[\cdot \mid \cdot]$ is the conditional expectation. Since $k < \mu$, we have $\mu' > \mu$ and $\mu' - \delta\mu' > \mu - \delta\mu = k$ for $0 < \delta < 1$. Under the condition $Y_j \leq k$, $Y_{j'}$ is again the sum of independent Bernoulli trials whose expectation is μ', so that we have

$$\Pr(Y_{j'} \leq k \mid Y_j \leq k) \leq \Pr(Y_{j'} < \mu' - \delta\mu' \mid Y_j \leq k) < \exp(-\mu'\delta^2/2) < \exp(-\mu\delta^2/2).$$

Using the bound for q_k, $\mathrm{E}[Z_{k,j}Z_{k,j'}] < \exp(-\mu\delta^2)$, and we finally have the following.

$$\mathrm{V}[Z_k] < n\exp(-\mu\delta^2/2) + n^2\exp(-\mu\delta^2). \quad \square$$

Thus, bounds for the expectation and variance of Z_k are derived, and the Chebyshev inequality may be used to estimate $\Pr(Z_k \geq n/2)$.

Lemma 5.3. For $0 < \epsilon < 1$ and $n \geq 16$, the median $Y_{\rm med}$ among Y_j $(j=1,\ldots,n)$ is greater than $\mu - \sqrt{(1+\epsilon)\mu\ln n}$ with probability at least $1 - 32n^{-1-\epsilon}$.

Proof: For $k \equiv \mu - \sqrt{(1+\epsilon)\mu\ln n}$, $q_k < n^{-(1+\epsilon)/2}$, and, from Lemma 5.2,

$$\mathrm{V}[Z_k] < n^{1-(1+\epsilon)/2} + n^{2-(1+\epsilon)} < 2n^{1-\epsilon},$$

where the last inequality follows since $0 < \epsilon < 1$. Using the Chebyshev bound (Lemma 2.3), we have

$$\Pr(Z_k > n^{1-(1+\epsilon)/2} + t\sqrt{2}n^{(1-\epsilon)/2}) < \frac{1}{t^2}.$$

Since $0 < \epsilon < 1$ and $n \geq 16$, setting $t = n^{(1+\epsilon)/2}/(4\sqrt{2})$,

$$n^{1-(1+\epsilon)/2} + t\sqrt{2}n^{(1-\epsilon)/2} \leq \frac{n}{4} + \frac{n}{4} = \frac{n}{2}.$$

Hence,

$$\Pr(Z_k \geq n/2) < \frac{1}{t^2} = 32n^{-1-\epsilon}.$$

If $Z_k \geq n/2$, the median $Y_{\rm med}$ among Y_j is less than or equal to k. We thus obtain the lemma. □

We have thus given a bound from below to $Y_{\rm med}$. To obtain a better bound for $Y_{\max} - Y_{\rm med}$, we need to re-examine $\Pr(Y_j > \mu + \delta\mu)$ in more detail, which will be done in the following two lemmas.

Lemma 5.4. If $\mu < (\log n)/(\mathrm{e} - 1/2)$, for any positive constant ϵ, $Y_{\max}$ is at most $(5/2 + \epsilon)\log n$ with probability at least $1 - n^{-1-\epsilon}$.

Proof: Applying the Chernoff bound with

$$\delta = \frac{(2+\epsilon)\log n}{\mu} > 2\mathrm{e} - 1,$$

we have

$$\Pr(Y_j > \mu + (2+\epsilon)\log n) < n^{-2-\epsilon}.$$

Since $\mu + (2+\epsilon)\log n < (5/2 + \epsilon)\log n$, we obtain the lemma. □

Lemma 5.5. If $\mu \geq (\log n)/(\mathrm{e} - 1/2)$, for $0 < \epsilon < 0.7$,

$$Y_{\max} \leq \begin{cases} \mu + \sqrt{(4+2\epsilon)(1-p)m\ln n} & \frac{1}{3} \leq p \leq \frac{1}{2} \\ \mu + \sqrt{(8+4\epsilon)pm\ln n} & 0 < p < \frac{1}{3} \end{cases}$$

with probability at least $1 - n^{-1-\epsilon}$.

Proof: Applying the Chernoff bound with

$$\delta = \sqrt{(8+4\epsilon)\ln n/\mu} \leq \sqrt{(8+4\epsilon)(\mathrm{e}-1/2)\ln n/\log n} \leq 4.1,$$

for $0 < \epsilon < 0.7$, we have

$$\Pr(Y_j > \mu + \sqrt{(8+4\epsilon)\mu\ln n}) < n^{-2-\epsilon}.$$

This with Lemma 3.2 implies

$$Y_{\max} \leq \min\{\,\mu + \sqrt{(4+2\epsilon)\overline{\mu}\ln n},\ \mu + \sqrt{(8+4\epsilon)\mu\ln n}\,\}$$

with probability at least $1 - n^{-1-\epsilon}$. Considering the minimum in the right hand side yields the lemma. □

We now give a bound on $Y_{\max} - Y_{\rm med}$.

Lemma 5.6. Suppose $\mu \geq (\log n)/(\mathrm{e} - 1/2)$ and $0 < \epsilon < 0.7$. Then, $Y_{\max} - Y_{\rm med} < \sqrt{(5+3\epsilon)m\ln n}$ with probability at least $1 - 33n^{-1-\epsilon}$.

Proof: Combining Lemmas 5.3 and 5.5, with probability at least $1 - 33n^{-1-\epsilon}$, when $1/3 \leq p \leq 1/2$,

$$\begin{aligned} Y_{\max} - Y_{\rm med} &\leq (\sqrt{(4+2\epsilon)(1-p)} + \sqrt{(1+\epsilon)p})\sqrt{m\ln n} \\ &\leq (\sqrt{(8+4\epsilon)/3} + \sqrt{(1+\epsilon)/3})\sqrt{m\ln n} < \sqrt{(5+3\epsilon)m\ln n}, \end{aligned}$$

and, when $0 < p < 1/3$,

$$\begin{aligned} Y_{\max} - Y_{\rm med} &\leq (\sqrt{(8+4\epsilon)p} + \sqrt{(1+\epsilon)p})\sqrt{m\ln n} \\ &< (\sqrt{(8+4\epsilon)/3} + \sqrt{(1+\epsilon)/3})\sqrt{m\ln n} < \sqrt{(5+3\epsilon)m\ln n}. \end{aligned}$$ □

We can evaluate $Y_{\rm med} - Y_{\min}$ in a similar way (in fact, this case is easier), and have the following.

Lemma 5.7. Suppose $\mu \geq (\log n)/(\mathrm{e} - 1/2)$ and $0 < \epsilon < 0.7$. Then, $Y_{\rm med} - Y_{\min} < \sqrt{(5+3\epsilon)m\ln n}$ with probability at least $1 - 33n^{-1-\epsilon}$.

Proof: Due to the space limitation, we omit the proof in this extended abstract. □

Theorem 5.1. Algorithm RS sorts n^2 binary numbers within $2\lceil\log\log n\rceil + 1$ iterations of $1 \sim 4$ in the algorithm with probability at least $1 - 33\lceil\log\log n\rceil n^{-1-\epsilon}$ for $0 < \epsilon < 1/4$.

Proof: We first show that, after $\lceil\log\log n\rceil - 1$ iterations, the number of dirty rows becomes at most $(5+3\epsilon)\ln n$. From Lemmas 4.3, 5.6 and 5.7, in an iteration, the number m of dirty rows is reduced to $\sqrt{(5+3\epsilon)m\ln n}/2$ with probability at least $1 - 33n^{-1-\epsilon}$ for $0 < \epsilon < 1/4 < 0.7$ if μ at that stage is greater than or equal to $(\log n)/(\mathrm{e}-1/2)$. Hence, if $\mu \geq (\log n)/(\mathrm{e}-1/2)$ in each of the first $\lceil\log\log n\rceil - 1$ iterations, by similar arguments in the proof of Theorem 3.1, the number of dirty rows is reduced to

$$\frac{5+3\epsilon}{4} \cdot 2^2 \ln n = (5+3\epsilon)\ln n$$

with the probability mentioned in the theorem. If $\mu < (\log n)/(e - 1/2)$ for some stage in the iterations, by Lemma 5.4, the number of dirty rows becomes at most $(5/2+\epsilon)\log n < (5+3\epsilon)\ln n$ at this stage with probability at least $1 - n^{-1-\epsilon}$. Thus the number of dirty rows becomes sufficiently small at the stage.

Then, by the effect of shearing in the algorithm, the number of dirty rows is reduced by a factor of 1/2 (Lemma 4.1), so that additional $\lceil \log\log n \rceil + 2$ iterations finish sorting. □

Corollary 5.1. Algorithm RS sorts n^2 general numbers within $2\lceil \log\log n \rceil + 1$ iterations of $1 \sim 4$ in the algorithm with probability at least $1 - 33\lceil \log\log n \rceil n^{-\epsilon}$ for $0 < \epsilon < 1/4$. □

6. Concluding Remarks

Computational testing has been done for the algorithms described in the paper. For sorting n^2 general numbers with $n = 2^i$ $(i = 6, \ldots, 11)$, Algorithm RS requires at most 4 iterations for 50 trials for each n. Corollary 5.1 states that the number of iterations is at most 9 for this range of n with high probability. Therefore, the bound in Corollary 5.1 is fair, and moreover it is interesting to observe that Algorithm RS only requires 4 iterations even for sorting four million numbers. We also show a table to see why the algorithms run fast in the experiments. Table 6.1 shows the number of dirty rows after executing an iteration of Algorithm RS. As is seen from the table, even for $n = 2048$, the number of dirty rows is reduced to at most 12 by one iteration. Although the bounds given in sections 3 through 5 are not so tight for it, these still explain why these algorithms are fast in practice. Since the reduction in the number of dirty rows at the first iteration is drastic, it can be said that, if an initial configuration of n^2 numbers on the mesh is random, the shear sort does perform quite well in practice.

Table 6.1. The maximum $m_{\max}$, average m_{av} and minimum $m_{\min}$ numbers of dirty rows after one iteration of Algorithm RS for sorting 50 different test sets of n^2 binary numbers with half 0's and half 1's

n	64	128	256	512	1024	2048
$m_{\max}$	5	5	6	9	12	12
m_{av}	2.46	2.62	3.26	4.30	5.16	6.40
$m_{\min}$	2	2	2	2	2	3

Summarizing the algorithms and their analyses, we have presented the randomized sorting algorithms on the mesh computer. These algorithms use only row and column sorting operations, and therefore are quite simple and useful in practice. Also, our analysis can be viewed as a probabilistic analysis of the shear sort algorithm. In the analysis, we have evaluated the median value of random variables Y_j $(j = 1, \ldots, n)$, and utilized this to improve the analysis on the effect of shearing, which would be of interest in itself.

Acknowledgment

This work was partially supported by the Grant-in-Aid of the Ministry of Education, Science and Culture of Japan.

References

[1] R. Cole: Parallel Merge Sort. *SIAM Journal on Computing*, Vol.17, No.4 (1988), pp.770–785.

[2] K. Iwama, E. Miyano and Y. Kambayashi: A Parallel Sorting Algorithm on the Mesh-Bus Machine. *Technical Report SIGAL 18-2*, IPSJ, November 1990.

[3] D. Knuth: *The Art of Computer Programming: Vol.3: Sorting and Searching.* Addison-Wesley, 1973.

[4] P. Raghavan: Lecture Notes on Randomized Algorithms. *IBM Research Report RC 15340*, IBM Research Division, 1990.

[5] C. P. Schnorr and A. Shamir: An Optimal Sorting Algorithm for Mesh Connected Computers. *Proceedings of the 18th Annual ACM Symposium on Theory of Computing*, 1986, pp.255–263.

[6] C. Thompson and H. Kung: Sorting on a Mesh-Connected Parallel Computer. *Communications of the ACM*, Vol.20 (1977), pp.263–271.