PAPER

Parallel Algorithms for the All Nearest Neighbors of Binary Image on the BSP Model

Takashi ISHIMIZU[†], ***Student Member***, **Akihiro FUJIWARA**[††], **Michiko INOUE**[†], **Toshimitsu MASUZAWA**[†], ***and*** **Hideo FUJIWARA**[†], ***Members***

SUMMARY In this paper, we present two parallel algorithms for computing the all nearest neighbors of an $n \times n$ binary image on the Bulk-Synchronous Parallel(BSP) model. The BSP model is an asynchronous parallel computing model, where its communication features are abstracted by two parameters L and g: L denotes synchronization periodicity and g denotes a reciprocal of communication bandwidth. We propose two parallel algorithms for the all nearest neighbor problems based on two distance metrics. The first algorithm is for L_p distance, and the second algorithm is for weighted distance. Both two algorithms run in $O(\frac{n^2}{p} + L)$ computation time and in $O(g\frac{n}{\sqrt{p}} + L)$ communication time using p $(1 \leq p \leq n)$ processors and 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 using p $(n < p \leq n^2)$ processors on the BSP model, for any integer d $(1 \leq d \leq \frac{p}{n})$.

key words: *parallel algorithm, BSP model, all nearest neighbors*

1. Introduction

The all nearest neighbors problem (ANNP) for a black and white binary image is a problem to compute for each black pixel, the coordinates of the nearest black pixel. Computation of the all nearest neighbor for a binary image is an important operation because of its applications in various areas such as image processing, computer graphics, pattern recognition and so on.

An $O(n^2)$ time sequential algorithm [2] was proposed for ANNP of an $n \times n$ binary image. The sequential algorithm is obviously time optimal because the size of an input image is n^2. In this paper, we consider algorithms for ANNP on the Bulk-Synchronous Parallel(BSP) model. The BSP model was proposed by Valiant [6] as a parallel computation model with features of currently realized parallel machines, and has great interests in these years.

In this paper, we propose two parallel algorithms for ANNP of an $n \times n$ binary image, one is for *L_p distance* and the other is for *weighted distance*. Both of the distances are generalizations of many distances, for example city block, chessboard, chamfer and Euclidean distances. Therefore these two distance metrics contain almost all kinds of distances used in image processing.

Both of our two algorithms run in $O(\frac{n^2}{p} + T^{comp}_{prefix}(n,n,p))$ computation time and in $O(g\frac{n}{\sqrt{p}} + T^{comm}_{prefix}(n,n,p))$ communication time using p $(1 \leq p \leq n^2)$ processors on the BSP model, where $T^{comp}_{prefix}(n,n,p)$ and $T^{comm}_{prefix}(n,n,p)$ denote computation time and communication time for 2D prefix operations of an $n \times n$ array using p processors, respectively.

We also showed $T^{comp}_{prefix}(n,n,p) = O(\frac{n^2}{p} + L)$ and $T^{comm}_{prefix}(n,n,p) = O(g\frac{n}{\sqrt{p}} + L)$ if $1 \leq p \leq n$, and $T^{comp}_{prefix}(n,n,p) = O(\frac{n^2}{p} + (d+L)\frac{\log \frac{p}{n}}{\log(d+1)})$ and $T^{comm}_{prefix}(n,n,p) = O(g\frac{n}{\sqrt{p}} + (gd+L)\frac{\log \frac{p}{n}}{\log(d+1)})$ if $n < p \leq n^2$ for any integer d $(1 \leq d \leq \frac{p}{n})$. Therefore our all nearest neighbor algorithms are cost optimal* for wide range of the number of processors. It is worth while noticing that the proposed 2D prefix algorithms can be applied to various problems since the prefix operation is one of the basic operations for parallel processing.

2. Preliminaries

2.1 The All Nearest Neighbors of a Binary Image

Given an $n \times n$ image I, let $I[x,y] \in \{0,1\}$ denote a value for a pixel (x,y) of $I (0 \leq x \leq n-1, 0 \leq y \leq n-1)$, where x(resp. y) stands for row (resp. column) index. We assume a pixel $(0,0)$ is on the top left corner of the image. We call a pixel (x,y) a *black pixel* if $I[x,y] = 1$, otherwise we call the pixel a *white pixel*, and *Black* (resp. *White*) denotes a set of all black pixels (resp. white pixels) in the input image.

In this paper, we use two distances, weighted distance [1] and L_p distance [5].

The *weighted distance* $d_w(p_1,p_2)$, between two pixels $p_1 = (x_1,y_1)$ and $p_2 = (x_2,y_2)$, is defined as follows.

$$d_w(p_1,p_2) = \begin{cases} w_0|x_1-x_2| + w_1|y_1-y_2| \\ \qquad\qquad (\text{if } |x_1-x_2| \geq |y_1-y_2|) \\ w_1|x_1-x_2| + w_0|y_1-y_2| \\ \qquad\qquad (\text{otherwise}) \end{cases}$$

Manuscript received February 19, 1999.

[†]The authors are with the Graduate School of Information Science, Nara Institute of Science and Technology, Ikoma-shi, 630–0101 Japan.

[††]The author is with the Department of Computer Science and Electronics, Kyushu Institute of Technology, Iizuka-shi, 820–8502 Japan.

*A parallel algorithm is called *cost optimal* if product of its running time and the number of processors is equal to lower bound of sequential time for the problem.

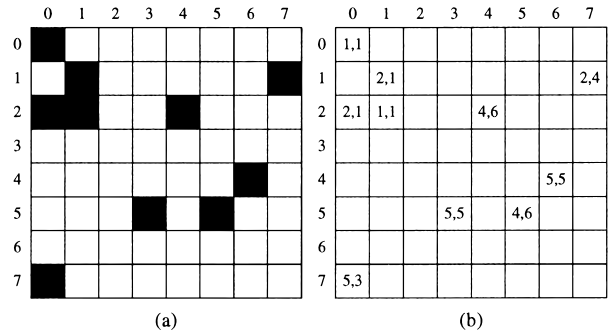


Fig. 1 An examples of the *all nearest neighbors*:(a) an input image I and (b) the L_2 distance nearest neighbors of I.

where w_0 and w_1 are nonnegative constants. The weighted distance is generalization of the following distances [4].

- city block distance (if $(w_0, w_1) = (1, 1)$)
- chessboard distance (if $(w_0, w_1) = (1, 0)$)
- quasi-Euclidean distance (if $(w_0, w_1) = (1, \sqrt{2} - 1)$)

The L_p *distance* d_{L_p} (p_1, p_2) $(1 \leq p)$, between two pixels $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$, is defined as follows.

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

To avoid confusion with the number of processors p, we use L_q distance instead of L_p distance hereafter.

The L_q distance is also generalization of the following distances.

- Euclidean distance (if $q = 2$)
- city block distance (if $q = 1$)
- chessboard distance (if $q \to \infty$)

The all nearest neighbors problem (ANNP) of a binary image is to find for each black pixel, the nearest black pixel except itself. We call the nearest black pixel *nearest neighbor* (See Fig. 1.). Formally, the ANNP requires to compute an array of coordinates $NN[x, y]$ such that

$$NN[x,y] = \begin{cases} (v,w) \text{ s.t. } ((v,w) \in Black - \{(x,y)\}) \\ \quad \wedge (\forall (v',w') \in Black - \{(x,y)\}, \\ \quad d((x,y),(v,w)) \leq d((x,y),(v',w'))) \\ \qquad\qquad (\text{if} (x,y) \in Black) \\ undefined \qquad\qquad (\text{otherwise}) \end{cases}$$

where $d((x_1, y_1), (x_2, y_2))$ is the distance between two pixels (x_1, y_1) and (x_2, y_2). If the distance is the weighted distance (resp. L_q distance), we call ANNP the *weighted distance all nearest neighbor problem* (resp. the L_q *distance all nearest neighbor problem*).

2.2 Bulk-Synchronous Parallel Model

In this paper, we use the Bulk-Synchronous Parallel (BSP) model. The BSP model was proposed by Valiant [6] as a parallel computation model with features of currently realized parallel machines.

The BSP model consists of the following three equipments: (1) a set of processors with local memory, (2) a router or a communication network that delivers messages among the processors in point-to-point fashion and (3) a facility for barrier synchronization among all of the processors.

A computation on the BSP model consists of a sequence of *supersteps*. For simplicity, we assume that, in each superstep, processors either perform local computation without any communication, or perform communication without local computation. We call a superstep for local computation (resp. for communication) *a computation superstep* (resp. *a communication superstep*).

The BSP model is characterized by the following parameters p, L and g.

- p: the number of processors.
- L: the synchronization periodicity, i.e. minimal time needed for synchronizing processors.
- g: a reciprocal of communication bandwidth, i.e. time which the router needs to deliver a message.

From the above definition, a computation superstep with at most w local operations on each processor can be executed in $O(w + L)$ time. A communication superstep, where each processor sends and receives at most h messages, can be executed in $O(gh + L)$ time. We call time to execute all computation supersteps (resp. all communication supersteps) *computation time* (resp. *communication time*).

2.3 Input Image

In this paper, each processor is denoted by $P_{i,j}$ $(0 \leq i, j \leq \sqrt{p} - 1)$. Let $I[x, y]$ $(0 \leq x, y \leq n - 1)$ be an input image of an $n \times n$ binary image. For simplicity, we assume that $n = k\sqrt{p}$ for some positive integer k. We also assume that the input image is partitioned into $\sqrt{p} \times \sqrt{p}$ sub-squares of size $\frac{n}{\sqrt{p}} \times \frac{n}{\sqrt{p}}$, and each processor $P_{i,j}$ initially has a sub square $I_{i,j}$ defined as follows.

$$I_{i,j}[g,h] = I\left[i\tfrac{n}{\sqrt{p}} + g, j\tfrac{n}{\sqrt{p}} + h\right] \left(0 \leq g \leq \tfrac{n}{\sqrt{p}} - 1,\ 0 \leq h \leq \tfrac{n}{\sqrt{p}} - 1\right)$$

2.4 Primitive Operations on BSP

In this section we introduce two fundamental operations, a *vertical 2D prefix* operation and a *diagonal 2D prefix* operation, used in this paper. In the following, $\oplus$ denotes a binary associative operator.

Definition 1: (vertical 2D prefix operation)
Given an $n \times n$ array $A[x, y]$ $(0 \leq x \leq n - 1, 0 \leq y \leq n - 1)$, the vertical 2D prefix operation computes the value

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

for each (x, y).

Definition 2: (diagonal 2D prefix operation)
Given an $n \times n$ array $A[x,y]$ $(0 \le x \le n-1, 0 \le y \le n-1)$, the diagonal 2D prefix operation computes the value

$$B[x,y] = \begin{cases} A[0,y-x] \oplus A[1,y-x+1] \oplus \ldots \oplus A[x,y] \\ \qquad\qquad (\text{if } x \le y) \\ A[x-y,0] \oplus A[x-y+1,1] \oplus \ldots \oplus A[x,y] \\ \qquad\qquad (\text{otherwise}) \end{cases}$$

for each (x,y).

Intuitively, the vertical 2D prefix operation computes the prefix for each column of a 2D array, and the diagonal 2D prefix operation computes the prefix for each diagonal sequence of a 2D array.

We assume that an input array of the prefix computation is partitioned and stored into each processor in the same fashion as an input image of ANNP. We assume that $A_{x,y}$ denotes an input sub-array stored in a processor $P_{x,y}$.

We can obtain the following lemma for the prefix operations on the BSP model.

Lemma 1: Both of vertical and diagonal 2D prefix operations can be executed in $O(\frac{n^2}{p} + L)$ computation time and $O(g\frac{n}{\sqrt{p}} + L)$ communication time using p $(1 \le p \le n)$ processors, and $O(\frac{n^2}{p} + (d+L)\frac{\log \frac{p}{n}}{\log(d+1)})$ computation time and $O(g\frac{n}{\sqrt{p}} + (gd+L)\frac{\log \frac{p}{n}}{\log(d+1)})$ communication time using p $(n < p \le n^2)$ processors, for any integer d $(1 \le d \le \frac{p}{n})$.

Proof:
(vertical 2D prefix operation)
Juurlink and Wijshoff [3] proposed an algorithm on the BSP model for a similar 2D prefix operation. An input of the algorithm is $p \times k$ array, where each processor holds one row of the array. An output of the operation is prefix computation for each column of the array. Using p processors, their algorithm runs in $O(k+L)$ computation time and $O(gk+L)$ communication time if $1 \le p \le k$, and in $O(k+(d+L)\frac{\log \frac{p}{k}}{\log(d+1)})$ computation time and $O(gk+(gd+L)\frac{\log \frac{p}{k}}{\log(d+1)})$ communication time if $k < p$ processors, for any integer d $(1 \le d \le \frac{p}{k})$.

We use the above algorithm to our vertical 2D prefix. We consider vertical 2D prefix operation for columns on a set of processors $P_{0,j}, P_{1,j}, \ldots P_{\sqrt{p}-1,j}$. First we compute $A_{i,j}[0,h] \oplus A_{i,j}[1,h] \oplus \ldots \oplus A_{i,j}[\frac{n}{\sqrt{p}} - 1, h]$ for each column h $(0 \le h \le \frac{n}{\sqrt{p}} - 1)$ on each processor $P_{i,j}$, and store the results in $A'_{i,j}[h]$. Second we compute the prefix of $A'_{0,j}[h], A'_{1,j}[h], \ldots, A'_{\sqrt{p}-1,j}[h]$ for each column h $(0 \le h \le \frac{n}{\sqrt{p}} - 1)$ by the 2D prefix algorithm [3] for $\sqrt{p} \times \frac{n}{\sqrt{p}}$ array using $\sqrt{p}$ processors, and store the result in $A''_{i,j}[h]$. Third, we send $A''_{i,j}$ to a processor $P_{i+1,j}$ for each processor $P_{i,j}$ if $P_{i+1,j}$ exists, and execute the prefix computation $A''_{i-1,j}[h] \oplus A_{i,j}[0,h] \oplus A_{i,j}[1,h] \oplus \ldots, A_{i,j}[f,h]$ for each row f $(0 \le f \le \frac{n}{\sqrt{p}} - 1)$ and each column h $(0 \le h \le \frac{n}{\sqrt{p}} - 1)$ on each processor. We can execute the first and the third steps in $O(\frac{n^2}{p} + L)$ computation time and $O(g\frac{n}{\sqrt{p}} + L)$ communication time, and we can execute the second step using the above prefix algorithm [3] by setting $k = \frac{n}{\sqrt{p}}$. Therefore the lemma holds for the vertical 2D prefix.
(diagonal 2D prefix operation)
Diagonal sequences have one remarkable difference from vertical sequences; each diagonal sequence is on diagonally arranged processors. For example, a diagonal sequence $A[0,1]$, $A[1,2]$, ... $A[n-2,n-1]$ is on $P_{0,0}$, $P_{0,1}$, $P_{1,1}$, ... $P_{\sqrt{p}-2,\sqrt{p}-2}$, $P_{\sqrt{p}-2,\sqrt{p}-1}$, $P_{\sqrt{p}-1,\sqrt{p}-1}$. We can compute the diagonal 2D prefix with the same idea as the vertical 2D prefix operation. Let $DSL_0, DSL_1, \ldots DSL_{\sqrt{p}-1}$ be sets of diagonal sequences whose left most element is on a processor $P_{0,0}, P_{1,0}, \ldots P_{\sqrt{p}-1,0}$, and let $DSU_1, DSU_2, \ldots DSU_{\sqrt{p}-1}$ be sets of diagonal sequences whose left most element is on a processor $P_{0,1}, P_{0,2} \ldots P_{0,\sqrt{p}-1}$, respectively. First, we compute the diagonal 2D prefix operation for DSL_0, DSL_2, DSL_4, ... $DSL_{2\lfloor \frac{\sqrt{p}-1}{2} \rfloor}$ and DSU_2, DSU_4, ... $DSU_{2\lfloor \frac{\sqrt{p}-1}{2} \rfloor}$ in parallel. This takes the same complexity as the vertical 2D prefix because of the following reasons. First both of size and length of each diagonal sequence on each processor are at most $\frac{n}{\sqrt{p}}$ and the number of processors used for each set are at most $2\sqrt{p}$. Second each set of diagonal sequences are on disjoint sets of processors. Thus we can process the diagonal 2D prefix for each set independently.

We can compute diagonal 2D prefix for the other diagonal sequences similarly. Therefore we obtain the lemma. □

Let $T^{comp}_{prefix}(n,n,p)$ (resp. $T^{comm}_{prefix}(n,n,p)$) denote computation time (resp. communication time) for the both of vertical and diagonal 2D prefix operations for an $n \times n$ array using p processors hereafter.

3. Algorithm for the Weighted Distance All Nearest Neighbors

3.1 Basic Idea

In this section, we present a parallel algorithm for weighted distance all nearest neighbors and analyze its complexity.

Fujiwara et al.[1] showed a parallel algorithm for weighted distance nearest feature transform on PRAM. This algorithm can be easily applied on the BSP model.

We show that weighted distance all nearest neighbors can be solved similar method as weighted distance nearest feature transform algorithm [1].

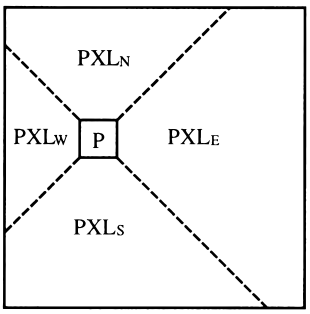


Fig. 2 A pixel $p = (x, y)$ and the four sets of pixels, $PXL_N(x, y), PXL_S(x, y), PXL_E(x, y)$ and $PXL_W(x, y)$.

3.1.1 Algorithm for the Weighted Distance Nearest Feature Transform

In this subsection, we briefly introduce the weighted distance nearest feature transform algorithm [1] and its complexity on the BSP model.

The nearest feature transform of a binary image is an operation which computes, for each pixels, the coordinates of the nearest black pixel.

Formally, the nearest feature transform requires to compute an array of coordinates $NBP[x, y]$ such that

$$\begin{aligned} NBP[x,y] = (v,w) \text{ s.t. } & ((v,w) \in Black) \\ & \wedge (\forall (v',w') \in Black, \\ & d((x,y),(v,w)) \le d((x,y),(v',w'))) \end{aligned}$$

where $d((x_1, y_1), (x_2, y_2))$ is the distance between two pixels (x_1, y_1) and (x_2, y_2).

First we define four sets of pixels $PXL_N(x, y)$, $PXL_S(x, y)$, $PXL_E(x, y)$ and $PXL_W(x, y)$ for a pixel (x, y) (see Fig. 2).

For example, $PXL_N(x, y)$ is defined as follows.

$$\begin{aligned} & PXL_N(x,y) \\ & = \{(x - x_1, y + y_1) | 1 \le x_1 \le x, \max \\ & \quad \{-y, -x_1\} \le y_1 \le \min\{n - 1 - y, x_1\}\}. \end{aligned}$$

Using the above four sets, we compute the weighted distance nearest feature transform in parallel in the following three step.

(1) Find the nearest black pixel in PXL_N for each pixel.
(2) Do the same procedure for each of PXL_S, PXL_E and PXL_W.
(3) Compute the nearest black pixel by selecting the nearest black pixel among the above four pixels for each pixel(x, y).

In the third step, each nearest black pixel can be easily computed independently on each processor. Thus we show how to find the nearest black pixel in PXL_N in the rest of this section.

For weighted distance, an important lemma was showed in [1]. Using the lemma, we can compute the nearest black pixel by prefix operations and some primitive operations only.

Lemma 2: [1] Let $q = (x, y)$, q_1 and q_2 be three pixels such that $q_1, q_2 \in PXL_N(x, y)$ and $d_w(q, q_1) \le d_w(q, q_2)$. Then $d_w(q', q_1) \le d_w(q', q_2)$ holds for any pixel $q' = (k, y)(x \le k \le n - 1)$. □

Let $NBP_N[x, y]$ denote the coordinate of the nearest black pixel of a pixel (x, y) in $PXL_N(x, y)$. In addition, we consider the following two sets of pixels $DI_{NW}(x, y)$ and $DI_{NE}(x, y)$.

$$\begin{aligned} & DI_{NW}(x,y) = \{(x - i, y - i) | 0 \le i \le \min\{x, y\}\} \\ & DI_{NE}(x,y) \\ & \quad = \{(x - i, y + i) | 0 \le i \le \min\{x, n - 1 - y\}\} \end{aligned}$$

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

Let $NBP_{NW}[x, y]$ and $NBP_{NE}[x, y]$ denote coordinates of the nearest black pixel for (x, y) in $DI_{NW}(x, y)$ and $DI_{NE}(x, y)$, respectively. From Lemma 2, $NN_N[x, y]$ is one of $NN_N[x - 1, y]$, $NN_{NW}[x, y]$ and $NN_{NE}[x, y]$.

First, we compute $NBP_{NW}[x, y]$ and $NBP_{NE}[x, y]$ for each (x, y) $(0 \le x \le n - 1, 0 \le y \le n - 1)$. We can compute $NBP_{NW}[x, y]$ and $NBP_{NE}[x, y]$ using prefix operations because of the following lemma.

Lemma 3: [1] Let $q = (x, y)$, $q_1 = (x_1, y_1)$ and $q_2 = (x_2, y_2)$ be three pixels such that $q_1, q_2 \in DI_{NW}(x, y) \cup DI_{NE}(x, y)$ and $d_w(q, q_1) \le d_w(q, q_2)$. Then $x_1 \ge x_2$ holds. □

This lemma implies that the nearest black pixel in $DI_{NW} \cup DI_{NE}$ to (x, y) has the largest row index among all black pixels in $DI_{NW}(x, y) \cup DI_{NE}(x, y)$. Therefore we can compute NBP_{NW}, NBP_{NE} by using two diagonal prefix maxima operations.

Next we compute $NBP_N[x, y]$ for each (x, y). Let $NBP_{DI}[x, y]$ denote coordinates of nearer one of $NBP_{NW}[x, y]$ and $NBP_{NE}[x, y]$. To compute the NBP_N $[x, y]$ by prefix operation, first we compute following value $F(x, y)$ for each (x, y),

$$F(x,y) = -w_0 g + w_1 |y - h|$$

where $NBP_{DI}[x, y] = (g, h)$.

We can compute $NBP_N[x, y]$ by comparing the value F because of the following reason. Let $NBP_{DI}[x_1, y] = (g_1, h_1)$, $NBP_{DI}[x_2, y] = (g_2, h_2)$, and assume $x \ge x_1, x_2$ and $d_w((x, y), NBP_{DI}[x_1, y])$ $\le d_w((x, y), NBP_{DI}[x_2, y])$. In this case, the followings hold.

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

Therefore we can compute $NBP_N[x, y]$ by computing

prefix minima of $F(k,y)(0 \le k \le n-1)$ for each column.

For the prefix operations, we can use the diagonal 2D prefix and the vertical 2D prefix described in Sect. 2, and it needs only $O(\frac{n^2}{p})$ local computation and $O(\frac{n}{\sqrt{p}})$ communication except these prefix operations. Thus the following lemma is also obtained.

Lemma 4: For each pixel (x,y), the nearest black pixels $NBP_N[x,y]$, $NBP_{NW}[x,y]$ and $NBP_{NE}[x,y]$, which are in $PXL_N(x,y)$, $DI_{NW}(x,y)$ and $DI_{NE}(x,y)$ respectively, can be computed in $O(\frac{n^2}{p} + T^{comp}_{prefix}(n,n,p))$ computation time and in $O(g\frac{n}{\sqrt{p}} + T^{comm}_{prefix}(n,n,p)$ communication time using p $(1 \le p \le n^2)$ processors on the BSP model.

For each pixel (x,y), the weighted distance nearest feature transform can be computed from the nearest black pixels $NBP_N[x,y]$, $NBP_S[x,y]$, $NBP_E[x,y]$ and $NBP_N[x,y]$. Thus, the following lemma is also obtained.

Lemma 5: The weighted distance nearest feature transform of an $n \times n$ binary image can be computed in $O(\frac{n^2}{p} + T^{comp}_{prefix}(n,n,p))$ computation time and in $O(g\frac{n}{\sqrt{p}} + T^{comm}_{prefix}(n,n,p)$ communication time using p $(1 \le p \le n^2)$ processors on the BSP model.

See [1] for details of the algorithm.

3.1.2 Modifications for ANNP

In this subsection, we show how the weighted distance nearest feature transform algorithm is modified for the weighted distance all nearest neighbor algorithm.

The difference between two problems is, for a black pixel, the nearest black pixel of the nearest feature transform is the black pixel itself, but the one of the nearest neighbor is another black pixel.

The difference is solved as follows. For black pixel (x,y), the nearest neighbor in $PXL_N(x,y)$ is one of the nearest black pixels among $DI_{NW}(x-1,y-1)$, $DI_{NE}(x-1,y+1)$ and $PXL_N(x-1,y)$. Thus, we can compute the nearest neighbor using $NBP_{NW}[x-1,y-1]$, $NBP_{NE}[x-1,y+1]$ and $NBP_N[x-1,y]$.

The algorithm consists of two steps. First, for black each pixel (x,y), we compute nearest black pixel $NBP_{NW}[x-1,y-1]$, $NBP_{NE}[x-1,y+1]$ and $NBP_N[x-1,y]$ in $DI_{NW}(x-1,y-1)$, $DI_{NE}(x-1,y+1)$ and $PXL_N(x-1,y)$, respectively. Next, for each black pixel (x,y), select one of the black pixel among $NBP_{NW}[x-1,y-1]$, $NBP_{NE}[x-1,y+1]$ and $NBP_N[x-1,y]$.

From Lemma4, these three black pixels can be computed in $O(\frac{n^2}{p} + T^{comp}_{prefix}(n,n,p))$ computation time and $O(g\frac{n}{\sqrt{p}} + T^{comm}_{prefix}(n,n,p)$ communication time using p $(1 \le p \le n^2)$ processors on the BSP model. Therefore, the following theorem is obtained.

Theorem 1: The weighted distance all nearest neighbors of an $n \times n$ binary image can be computed in $O(\frac{n^2}{p} + T^{comp}_{prefix}(n,n,p))$ computation time and in $O(g\frac{n}{\sqrt{p}} + T^{comm}_{prefix}(n,n,p))$ communication time using p $(1 \le p \le n^2)$ processors on the BSP model.

From Lemma 1 and Theorem 1, the following corollary is also obtained.

Corollary 1: The weighted distance all nearest neighbors of an $n \times n$ binary image can be computed in $O(\frac{n^2}{p} + L)$ computation time and in $O(g\frac{n}{\sqrt{p}} + L)$ communication time using p $(1 \le p \le n)$ processors and 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 using p $(n < p \le n^2)$ processors for any integer d $(1 \le d \le \frac{p}{n})$ on the BSP model.

4. Algorithm for L_q Distance Nearest Neighbors

4.1 Basic Idea

For each sub-image $I_{i,j}$ (possessed by processor $P_{i,j}$), we define four sub-images as in Fig. 3. For example, $DN_{i,j}$ is defined as follows.

$$DN_{i,j} = \Big\{(x,y)| \; 0 \le x \le i\frac{n}{\sqrt{p}} - 1, 0 \le y \le (j+1)\frac{n}{\sqrt{p}} - 1\Big\}$$

We compute nearest neighbors of black pixels in $I_{i,j}$ for each of the above four sub-images. As a key concept for reducing time complexity, we introduce *base pixels* and *candidate pixels*. For sub-image A (one of the above four sub-images), we call a pixel in $I_{i,j}$ a *base pixel for* A if it may have the nearest neighbor in A. On the other hand, we call a pixel in A *a candidate pixel for* $I_{i,j}$ if it may be the nearest neighbor of some black pixel in $I_{i,j}$. To compute the nearest neighbors in A for $I_{i,j}$, it is obviously sufficient to compute nearest pixels in the candidate pixels for each base pixel in $I_{i,j}$.

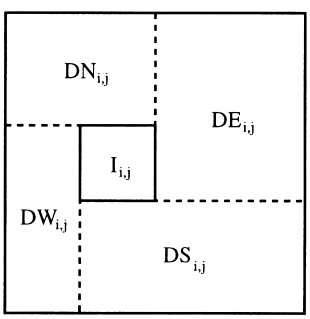


Fig. 3 Four sets for a processor $P_{i,j}$.

We show how the base pixels and the candidate pixels are determined and the number of the base pixels and the candidate pixels is small. This helps us to reduce time complexity for computing ANNP.

In the following, we show how to find the nearest black pixel in $DN_{i,j}$ for each pixel in $I_{i,j}$ in parallel. We can find the nearest black pixels in the other sub-images similarly, and we can find the nearest black pixels in $I_{i,j}$ using a known sequential algorithm [2].

4.2 Computing the Nearest Black Pixel in $DN_{i,j}$

We first show how we can determine a base pixels in $I_{i,j}$ for $DN_{i,j}$, and a candidate pixels in $DN_{i,j}$ for $I_{i,j}$. For each column in $I_{i,j}$, we choose, as a base pixel, one black pixel with the minimum row index from $I_{i,j}$ (if exists).

Definition 3: A set $Base^N_{i,j}$ of base pixels in $I_{i,j}$ for $DN_{i,j}$ is a set of black pixels (x, y) such that $x = min\{x'|(x', y) \in Black \wedge (x', y) \in I_{i,j}\}$

Since the number of columns in $I_{i,j}$ is $\frac{n}{\sqrt{p}}$, we choose at most $\frac{n}{\sqrt{p}}$ base pixels in $I_{i,j}$ for $DN_{i,j}$. It is obvious that a black pixel $(x, y) \in I_{i,j}$ has its nearest pixel in $DN_{i,j}$ only if (x, y) is one of the base pixels.

Next we show how we determine the candidate pixels in $DN_{i,j}$ for $I_{i,j}$. It is obvious that black pixels with the maximum row index for each column in $DN_{i,j}$ are nearer to all pixels in $I_{i,j}$ than any other pixels in the same column in $DN_{i,j}$. In addition, we show the following two lemmas to reduce the number of candidate pixels.

Lemma 6: If there exists at least two black pixels in $I_{i,j}$, none of black pixels in $DN_{i,j-3}$ is the nearest neighbor of any black pixel in $I_{i,j}$

Proof: Let $p_0 = (x_0, y_0)$ and $p_1 = (x_1, y_1)$ be black pixels in $I_{i,j}$, and $p_2 = (x_2, y_2)$ be a black pixel in $DN_{i,j-3}$. Then, $|x_0 - x_1| < \frac{n}{\sqrt{p}}$, $|y_0 - y_1| < \frac{n}{\sqrt{p}}$ and $y_0 - y_2 > 2\frac{n}{\sqrt{p}}$ hold.

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

Then, $d_{L_q}(p_0, p_1) < d_{L_q}(p_0, p_2)$ holds. Thus, p_2 is not the nearest neighbor of p_0. □

Lemma 7: If there exists a black pixels in $I_{i,k}$ for k $(0 \leq k < j)$, none of black pixels in $DN_{i,k-2}$ is the nearest neighbor of any black pixel in $I_{i,j}$.

Proof: Let $p = (x, y)$ be a black pixel in $I_{i,j}$, $p_1 = (x_1, y_1)$ be a black pixel in $I_{i,k}$ and $p_2 = (x_2, y_2)$ be a black pixel in $DN_{i,k-2}$. Then, $|x - x_1| < \frac{n}{\sqrt{p}}$ and $y - y_2 > (y - y_1) + \frac{n}{\sqrt{p}}$ hold.

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

Then, $d_{L_q}(p, p_1) < d_{L_q}(p, p_2)$ holds, and p_2 is not the nearest neighbor of p. □

From the above two lemmas, we define the candidate pixels as follows.

Definition 4:

1. If there are at least two black pixels in $I_{i,j}$, a set $Candidate^N_{i,j}$ of candidate pixels in $DN_{i,j}$ for $I_{i,j}$ is a set of black pixels (x, y) such that $x = max\{x'|(x', y) \in Black \wedge (DN_{i,j} - DN_{i,j-3})\}$
2. If there is only one black pixel in $I_{i,j}$, a set $Candidate^N_{i,j}$ of candidate pixels in $DN_{i,j}$ for $I_{i,j}$ is a set of black pixels (x, y) such that $x = max\{x'|(x', y) \in Black \wedge (DN_{i,j} - DN_{i,k-2})\}$, where $I_{i,k}$ is the right most sub-image which contains at least one black pixel in among $\{I_{i,0}, I_{i,1}, \ldots, I_{i,j-1}\}$

We can compute the candidate pixels and the base pixels by prefix operations for each column.

We describe details of our algorithm in the following.

[Algorithm for computing nearest neighbors for $DN_{i,j}$]

Step 1: (Computation of candidate pixels)

(1) For each pixel (x, y), set $A[x, y] = (x, y)$ if a pixel (x, y) is black, otherwise set $A[x, y] = (-\infty, -\infty)$.

(2) Compute prefix maxima A' of A for each column downward by comparing the first component. Each processor $P_{i,j}$, sends the results $A'[(i+1)\frac{n}{\sqrt{p}} - 1, j\frac{n}{\sqrt{p}} + h]$ $(0 \leq h \leq \frac{n}{\sqrt{p}} - 1)$ to the processor $P_{i+1,j}$ if exists. The received results are coordinates of candidate pixels in $Candidate^N_{i,j} - Candidate^N_{i-1,j}$.

Step 2: (Computation of base pixels)

(1) For each pixel (x, y), set $B[x, y] = (x, y)$ if a pixel (x, y) is black, otherwise set $B[x, y] = (+\infty, +\infty)$.

(2) Compute the prefix minima B' of B for each column upward by comparing the first component. The results $B'[i\frac{n}{\sqrt{p}}, j\frac{n}{\sqrt{p}} + h]$ $(0 \leq h \leq \frac{n}{\sqrt{p}} - 1)$ are coordinates of the base pixels in $Base^N_{i,j}$.

Step 3:

case 1: (There exist at least two black pixel in $I_{i,j}$)

(1) Send coordinates of each candidate pixels in $Candidate^N_{i,j-1} - Candidate^N_{i,j-3}$ on processors $P_{i,j-2}$ and $P_{i,j-1}$ to processor $P_{i,j}$.

(2) Compute nearest neighbors between base pixels in $Base^N_{i,j}$ and gathered candidate pixels in $Candidate^N_{i,j} - Candidate^N_{i,j-3}$ on processor $P_{i,j}$

case 2: (There exist only one black pixel in $I_{i,j}$)

(1) For each processor $P_{i,j}$, send coordinates of base pixel $b(i,j) \in Base^N_{i,j}$ among processors $P_{i,k-1}, P_{i,k}, \ldots, P_{i,j}$, where $I_{i,k}$ is the right most sub-image which contains at least one black pixel among $\{I_{i,0}, I_{i,1}, \ldots, I_{i,j-1}\}$. This can be done as follows.

(1-1) For each processor $P_{i,j}$, set $A[i,j] = (j, b(i,j))$ if there is exactly one pixel in $I_{i,j}$, set $A[i,j] = (j, \emptyset)$ if there are at least two black pixel in $I_{i,j}$, set $A[i,j] = (+\infty, \emptyset)$ otherwise.

(1-2) Compute prefix minima of $A[i,k]$ $(0 \leq k \leq \sqrt{p}-1)$ for each processor row from right to left by comparing the first component. We assume that the results for $P_{i,j}$ are stored in $A'[i,j]$.

(1-3) For each processor $P_{i,j}$, if there are no black pixel in $I_{i,j}$ and there is a black pixel in $I_{i,j-1}$, then send $A'[i,j]$ to processor $P_{i,j-2}$ and $P_{i,j-1}$.

(2) For each processor $P_{i,j}$, find the nearest neighbors $nn_{b_1}(i,j)$ and $nn_{b_2}(i,j)$ for base pixels $b_1(i,j)$ and $b_2(i,j)$ among candidate pixels on each processor, where $b_1(i,j)$ is a black pixel stored in $A'[i,j]$, and $b_2(i,j)$ is a black pixel received at (1-3), if $b_1(i,j)$ or $b_2(i,j)$ exists.

(3) Using processors $P_{i,k-1}, P_{i,k}, \ldots, P_{i,j}$, compute the nearest neighbor to the a pixel for $P_{i,j}$ among obtained nearest neighbors, and inform processor $P_{i,j}$ of the obtained nearest neighbor where $I_{i,k}$ is the right most sub-image which contains at least one black pixel in among $\{I_{i,0}, I_{i,1}, \ldots, I_{i,j-1}\}$. This can be done as follows.

(3-1) For each processor $P_{i,j}$, if $P_{i,j}$ received $b_2(i,j)$ at (1-3), then send $nn_{b_2}(i,j)$ to a processor that sent $b_2(i,j)$ to $P_{i,j}$

(3-2) For each processor $P_{i,j}$, if $P_{i,j}$ received nearest neighbors $nn_{b_2}(i,j-2)$ and $nn_{b_2}(i,j-1)$ at (3-1), then substitute $nn_{b_1}(i,j)$ for the nearest one among $nn_{b_2}(i,j-2)$, $nn_{b_2}(i,j-1)$ and $nn_{b_1}(i,j)$.

(3-3) For each processor $P_{i,j}$, set $B[i,j] = (-j', d_{L_q}(b_1(i,j), nn_{b_1}(i,j)), nn_{b_1}(i,j))$, where j' is the first component of $A'[i,j]$

(3-4) Compute the prefix minima $B'[i,k]$ of $B[i,k]$ $(0 \leq k \leq \sqrt{p}-1)$ for each row processor from left to right by comparing the first and second component by lexicographic order. Then, the nearest neighbor for base pixel $b(i,j)$ is the third component of $B'[i,j]$.

In the above algorithm, each processor requires $O(\frac{n^2}{p})$ computations and $O(\frac{n}{\sqrt{p}})$ data communications in addition to prefix operations.

Thus the following lemma is obtained.

Lemma 8: The nearest neighbors in $Candidate^N_{i,j}$ for pixels in $Base^N_{i,j}$ can be found in $O(\frac{n^2}{p} + T^{comp}_{prefix}(n,n,p))$ computation time and $O(g\frac{n}{\sqrt{p}} + T^{comm}_{prefix}(n,n,p))$ communication time.

From Lemma 8, the following theorem and corollary are obtained.

Theorem 2: The L_q distance all nearest neighbors of an $n \times n$ binary image can be computed in $O(\frac{n^2}{p} + T^{comp}_{prefix}(n,n,p))$ computation time and in $O(g\frac{n}{\sqrt{p}} + T^{comm}_{prefix}(n,n,p))$ communication time using p $(1 \leq p \leq n^2)$ processors on the BSP model.

Corollary 2: The L_q distance all nearest neighbors of an $n \times n$ binary image can be computed in $O(\frac{n^2}{p} + L)$ computation time and in $O(g\frac{n}{\sqrt{p}} + L)$ communication time using p $(1 \leq p \leq n)$ processors and 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 using p $(n < p \leq n^2)$ processors for any integer d $(1 \leq d \leq \frac{p}{n})$ on the BSP model.

5. Conclusion

In this paper, we have presented two parallel algorithms, for all nearest neighbors of an $n \times n$ binary image, one is for the weighted distance nearest neighbors and the other is for the L_p distance nearest neighbors, on the BSP model. Using an efficient parallel prefix algorithms which we also showed in this paper, the algorithms run in $O(\frac{n^2}{p} + L)$ computation time and in $O(g\frac{n}{\sqrt{p}} + L)$ communication time using p $(1 \leq p \leq n)$ processors and 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 using p $(n < p \leq n^2)$ processors for any integer d $(1 \leq d \leq \frac{p}{n})$ on the BSP model.

References

[1] A. Fujiwara, M. Inoue, T. Masuzawa, and H. Fujiwara, "A parallel algorithm for weighted distance transform," Proc. 11th International Parallel Processing Symposium, pp.407–412, April 1997.

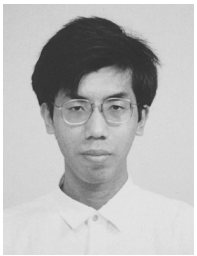
[2] T. Hirata, "A unified linear-time algorithm for computing distance maps," Inf. Process. Lett., vol.58, pp.129–133, 1996.

[3] B.H.H. Juurlink and H.A.G. Wijshoff, "Communication primitives for BSP computes," Inf. Process. Lett., vol.58, pp.303–310, 1996.

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

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

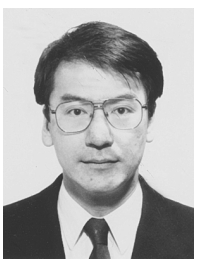
[6] L.G. Valiant, "A bridging model for parallel computation," Commun. ACM, vol.33, no.8, pp.103–111, Aug. 1990.



Takashi Ishimizu received the M.E. degree in Nara Institute of Science and Technology (NAIST) in 1997. He is now a student of Graduate School of Information Science, Nara Institute of Science and Technology (NAIST). His main research interests are parallel algorithms and parallel complexity theory.

Akihiro Fujiwara received the B.E. degree in Osaka University in 1993, and received the M.E. and Ph.D. degrees in Nara Institute of Science and Technology (NAIST) in 1995 and 1997, respectively. He is now a lecturer of Kyushu Institute of Technology. His main research interests are parallel algorithms, parallel complexity theory and cluster processing. He is a member of IEEE and IPSJ.

Michiko Inoue received her B.E., M.E, and Ph.D degrees from Osaka university in 1987, 1989, and 1995 respectively. She is an instructor of Graduate School of Information Science, Nara institute of Science and Technology (NAIST). Her research interests include distributed algorithms, parallel algorithms, graph theory and design and test of digital systems. She is a member of IEEE, the Information Processing Society of Japan, and Japanese Society for Artificial Intelligence.



Toshimitsu Masuzawa received the B.E., M.E. and D.E. degrees in computer science from Osaka University in 1982, 1984 and 1987. He had worked at Education Center for Information Processing, Osaka University between 1987–1990, and had worked at Faculty of Engineering Science, Osaka University between 1990–1994. He is now an associate professor of Graduate School of Information Science, Nara Institute of Science and Technology (NAIST). He was also a visiting associate professor of Department of Computer Science, Cornell University between 1993–1994. His research interests include distributed algorithms, parallel algorithms and graph theory. He is a member of ACM, IEEE, EATCS and the Information Processing Society of Japan.

Hideo Fujiwara received the B.E., M.E., and Ph.D. degrees in electronic engineering from Osaka University, Osaka, Japan, in 1969, 1971, and 1974, respectively. He was with Osaka University from 1974 to 1985 and Meiji University from 1985 to 1993, and joined Nara Institute of Science and Technology in 1993. In 1981 he was a Visiting Research Assistant Professor at the University of Waterloo, and in 1984 he was a Visiting Associate Professor at McGill University, Canada. Presently he is a Professor at the Graduate School of Information Science, Nara Institute of Science and Technology, Nara, Japan. His research interests are logic design, digital systems design and test, VLSI CAD and fault tolerant computing, including high-level/logic synthesis for testability, test synthesis, design for testability, built-in self-test, test pattern generation, parallel processing, and computational complexity. He is the author of Logic Testing and Design for Testability (MIT Press, 1985). He received the IECE Young Engineer Award in 1977, IEEE Computer Society Certificate of Appreciation Award in 1991, Okawa Prize for Publication in 1994, and IEEE Computer Society Meritorious Service Award in 1996. He is an advisory member of IEICE Trans. on Information and Systems and an editor of IEEE Trans. on Computers, J. Electronic Testing, J. Circuits, Systems and Computers, J. VLSI Design and others. Dr. Fujiwara is a fellow of the IEEE and a Golden Core member of the IEEE Computer Society as well as a member of the Information Processing Society of Japan.