1998 International Computer Symposium
Workshop on Algorithms
December 17-19, 1998, N.C.K.U., Tainan, Taiwan, R.O.C.

An Efficient Parallel Algorithm for Hierarchical Clustering on Arrays with Reconfigurable Optical Buses

Chin-Hsiung Wu† Ruey-Chang Tsai† Shi-Jinn Horng† Horng-Ren Tsai‡
†Department of Electrical Engineering, National Taiwan University of Science and Technology,
Taipei, Taiwan, R. O. C. E-mail:horng@mouse.ee.ntust.edu.tw
‡Department of Information Management, The Overseas Chinese College of Commerce,
Taichung, Taiwan, R. O. C. E-mail:hrt@rs2.occc.edu.tw

Abstract

Clustering is a basic operation in image processing and computer vision, and it plays an important role in unsupervised pattern recognition and image segmentation. This problem is intensive in computation and involve parallel processing on the patterns of an image. There are many methods for clustering, the single-link hierarchical clustering is one of the most popular techniques. In this paper, with the advantages of both optical transmission and electronic computation, we design efficient parallel hierarchical clustering algorithm on the arrays with reconfigurable optical buses. We first design two $O(1)$ and $O(\log N)$ time data operations for computing the matrix multiplication of two $N \times N$ matrices and finding the minimum spanning tree of a graph with N vertices using N^3 processors, respectively. Then based on these two data operations, an $O(\log N)$ time parallel hierarchical clustering algorithm is derived using N^3 processors. Note that this result improves the previously known algorithms on various parallel computation models.

1 Introduction

The array with a reconfigurable optical bus system is defined to be an array of processors connected to a reconfigurable optical bus system whose configuration can be dynamically changed by setting up the local switches of each processor, and message can be transmitted concurrently on a bus in a pipelined fashion. Recently, two related models have been proposed, namely the array with reconfigurable optical buses (AROB) [19] and linear array with a reconfigurable pipelined bus system (LARPBS) [17, 18]. A major difference between them lies in that the counting is not permitted in the LARPBS model during a bus cycle but it is allowed in the AROB model. The AROB model is a powerful computation model which incorporates some of the advantages and characteristics of reconfigurable meshes and meshes with optical buses [19]. Many algorithms have been proposed for applications on the AROB model [20, 21, 22].

Cluster analysis is the process of classifying objects into subsets that have meaning in the context of a particular problem [7]. Conventionally, the objects are characterized as patterns, and the patterns are numerical vectors in the pattern analysis. Assume that there are N patterns and each pattern contains M features. Clustering is a process of partitioning these N patterns in M-dimensional spaces into meaningful subsets or clusters. The clustering of such patterns is achieved by minimizing intracluster dissimilarity and maximizing intercluster dissimilarity. Clustering techniques are widely applied in many aspects such as life sciences, medical sciences, social sciences, earth sciences, image processing and so on, and the applications continue to grow [1, 7]. Clustering is especially useful when only a very little prior information about the problem is available. In image processing, clustering is used for pattern recognition, image segmentation, registration, compression and object detection. The detailed survey of the cluster analysis can be found in the literature [1, 4, 7, 8].

Many hierarchical clustering methods have been proposed using different distance metrics and techniques [7]. Because its simplicity and efficiency, the hierarchical clustering with the single-link method described by Johnson [9] is one of the popular clustering methods. The agglomerative approach starts with the disjoint partition, where each pattern is set to a distinct cluster initially. Then, two clusters are merged into together to form a new cluster from the current level to the next level, according to the dissimilarity among all of the remaining clus-

ters. Repeating this process to produce a sequence of nested partitions until a single cluster. Finally, a dendrogram is constructed.

Efficient sequential and parallel clustering algorithms have been studied extensively by many researchers [5, 10, 11, 12, 24]. Assume that there are N patterns each with M features. For the hierarchical clustering, the sequential algorithm can be computed in $O(N^2M + N^3)$ time in a straightforward manner. Kurita [10] proposed an $O(N^2 \log N)$ time sequential algorithm to solve this problem. Li and Fang [12] proposed an $O(N \log N)$ time parallel algorithm on the SIMD hypercube multiprocessors with MN processors. Li [11] also proposed an $O(N^2)$ time parallel algorithm on the SIMD shuffle-exchange networks using N processors. Gower and Ross [5] first specified that the hierarchical clustering with the single-link method can be derived from the minimum spanning tree (MST). Recently, based on the MST of proximity matrix, Tsai *et al.* [24] also proposed an $O(\log^2 N)$ time parallel algorithm on the processor array with a reconfigurable bus system (PARBS) using N^3 processors.

In this paper, we are interested in designing parallel clustering algorithm on the AROB. By integrating the advantages of both optical transmission and electronic computation, we first design two $O(1)$ and $O(\log N)$ time basic operations for matrix multiplication of two $N \times N$ matrices and finding the MST of a graph with N vertices using N^3 processors, respectively. Then, based on these two basic operations, an $O(\log N)$ time parallel hierarchical clustering algorithm is also derived using N^3 processors. This result is better than that of Tsai *et al.* [24] by $O(\log N)$ times and is better than the previously known algorithms.

The remainder of this paper is organized as follows. We give a brief introduction to the AROB computation model in Section 2. Section 3 designs two basic operations which will be used in the parallel clustering algorithm. Section 4 develops our parallel clustering algorithm. Finally, some concluding remarks are included in the last section.

2 The Computation Model and Basic Notations

In this section, we shall describe the architecture of the AROB model which is the computation model adopted in this paper. For the sake of convenience, we follow the notation which was proposed in the literature [19, 20].

The AROB model is essential a mesh using the basic structure of a classical reconfigurable network (RN) [2] and optical technology. A linear AROB (LAROB) of size N contains N processors connected to the optical bus with two couplers. One is used to write data on the upper (transmitting) segment of the bus and the other is used to read the data from the lower (receiving) segment of the bus. That is, it extends the capabilities of the linear arrays with pipelined optical buses (APPB) [6] by permitting each processor to connect to the bus through a pair of switches. Each processor with a local memory is identified by a unique index denoted as P_{i_0}, $0 \leq i_0 < N$, and each switch can be set to cross or straight by the local processor. As for the properties of the optical buses, the message propagates unidirectionally from right to left on the upper segment and from left to right on the lower segment. Each processor uses a set of control registers to store information needed to control the transmission and reception of messages by that processor.

There are two important properties of optical bus system: unidirectionally signal propagation and predictable delay of the signal per unit length. Due to these two properties, the optical buses enable synchronized concurrent access in a pipelined fashion.

A *2-dimensional* (2-D) AROB of size $M \times N$ contains $M \times N$ processors arranged in a 2-D grid. Each processor is identified by a unique 2-tuple index (i_1, i_0), $0 \leq i_1 < M$, $0 \leq i_0 < N$. The processor with index (i_1, i_0) is denoted by P_{i_1, i_0}. Each processor has four I/O ports, denoted by $-S_j$, $+S_j$, $0 \leq j < 2$, to be connected with a reconfigurable optical bus system. The interconnection among the four ports of a processor can be configured during the execution of algorithms. Thus, multiple arbitrary linear arrays like LAROB can be constructed in a 2-D AROB. Once the LAROB obtained by reconfiguration, we must specify the position of each processor and which orientation of the waveguides in the constructed LAROB for implementing the time division or coincident pulse techniques. The two *terminal* processors which are located in the end points of the constructed LAROB may specify as the *leader* processor. The related position of any processor on a bus to which it is connected, is its distance from the *leader* processor. For more details on the AROB, see [19].

A 2-D AROB model can be extended to a 3-D AROB model with two extra ports, $-S_2$, $+S_2$, for the communications of the neighboring processors in the third dimension. For a 3-D AROB of size $L \times M \times N$, each processor is identified by a unique

3-tuple index $(i_2,\ i_1,\ i_0)$, $0 \le i_2 < L$, $0 \le i_1 < M$, $0 \le i_0 < N$. The processor with index $(i_2,\ i_1,\ i_0)$ is denoted by $P_{i_2,\ i_1,\ i_0}$.

For a unit of time, assume each processor can either perform arithmetic and logic operations or communicate with others on a bus. We assume that a bus cycle length is compatible with the computation speed of an arithmetic (logic) operation. It allows multiple processors to broadcast data on the different buses or to broadcast the same data on the same bus simultaneously at a time unit, if there is no collision.

Let $var(k)$ denote the local variable *var* (memory or register) in a processor with index k. For example, $sum(0,\ 0,\ 1)$ is a local variable *sum* of processor $P_{0,\ 0,\ 1}$.

3 Basic Operations

In this section, we design two data operations, computing the matrix multiplication of two $N \times N$ matrices and finding the MST of a graph. These two data operations will be used for developing efficient parallel hierarchical algorithm in the next section. For the sake of completeness, several basic operations which have been proposed on the AROB are summarized in the following.

Lemma 1 *[21] Given N integer or normalized real numbers each of size $O(\log N)$-bit, the maximum (minimum) of these N integer numbers can be found in $O(1)$ time on a 1-D AROB if counting is allowed during a bus cycle.*

Lemma 2 *[14] Given N integer or normalized real numbers each of size $O(\log N)$-bit, these N numbers can be added by the bus split technique in $O(\log N)$ time on a 1-D N AROB.*

Lemma 3 *[20] Given a data array of size N, the ordered compaction problem is the problem of moving the n nonzero (or nonempty) data items to the first n consecutive locations of the array and remaining in the same order. The ordered compaction problem can be computed in $O(1)$ time on a 1-D N AROB.*

Lemma 4 *[21] Given a data array $a_{i,\ j}$, $0 \le a_{i,\ j} < N$, $0 \le i,\ j < N$, these N^2 number can be sorted in $O(1)$ time on a 2-D $N \times N$ AROB.*

Lemma 5 *[25] Given an undirected graph $G = (V, E)$ with $|V| = N$ and a vertex v, the connected component of G contains the vertex v, can be computed in $O(1)$ time on a 2-D $N \times N$ AROB.*

3.1 Computing the Matrix Multiplication

Let $A = a_{i_1,\ i_0}$ and $B = b_{i_1,\ i_0}$, $0 \le i_1,\ i_0 < N$ be two $N \times N$ matrices, and all elements of A and B be on the same domain D. The matrix multiplication $C = c_{i_1,\ i_0}$ is defined by

$$c_{i_1,\ i_0} = \oplus_{i_2=0}^{N-1}\{a_{i_1,\ i_2} \otimes b_{i_2,\ i_0}\},\quad 0 \le i_1,\ i_0 < N, \quad (1)$$

where $\oplus$ and $\otimes$ are two associative operators on the domain D.

By the reconfigurability and pipelined ability of the optical bus configuration, we can compute Eq. (1) efficiently on a 3-D AROB. Initially, $a_{i_1,\ i_0}$ and $b_{i_1,\ i_0}$ are stored in the local variables $a(0,\ i_1,\ i_0)$ and $b(0,\ i_1,\ i_0)$ of processor $P_{0,\ i_1,\ i_0}$, $0 \le i_1,\ i_0 < N$, one item per processor respectively. The result of $c_{i_1,\ i_0}$ is stored in the local variable $c(0,\ i_1,\ i_0)$ of processor $P_{0,\ i_1,\ i_0}$, $0 \le i_1,\ i_0 < N$. Like Chen *et al.* [3], Eq. (1) can be computed on a 3-D AROB by the following three steps.

1. Broadcast the elements of A and B over the $N \times N \times N$ processors through the optical buses so that $a(i_2,\ i_1,\ i_0) = a_{i_1,\ i_2}$ for $0 \le i_0 < N$ and $b(i_2,\ i_1,\ i_0) = b_{i_2,\ i_0}$ for $0 \le i_1 < N$.

2. Compute $e(i_2,\ i_1,\ i_0) = a(i_2,\ i_1,\ i_0) \otimes b(i_2,\ i_1,\ i_0)$.

3. Compute $c(0,\ i_1,\ i_0) = \oplus_{i_2=0}^{N-1} e(i_2,\ i_1,\ i_0)$.

The first two steps each takes $O(1)$ time. The time complexity of Step 3 is dependent on the operator $\oplus$. That is, by properly replacing the associative operator $\oplus$, the matrix multiplication can be computed efficiently. For example, if the associative operator $\oplus$ is replaced by the maximum/minimum operation, then Step 3 can be computed in $O(1)$ time by Lemma 1 on a 3-D $N \times N \times N$ AROB. Compared to the algorithm proposed by Chen *et al.* [3], our algorithm for computing the matrix multiplication with the maximum/minimum operation can be run in the same time complexity but reduce the number of processors by a factor N. Hence, this leads to the following lemma.

Lemma 6 *Given two $N \times N$ matrices A and B, if the operator $\oplus$ is the maximum (minimum) operator, then the matrix multiplication of A and B can be computed in $O(1)$ time on a 3-D $N \times N \times N$ AROB.* □

3.2 Finding the Minimum Spanning Tree

Given a graph G with N vertices, the minimum spanning tree (MST) problem of G is defined to find a tree with a minimum total weight. Assume that the adjacency matrix of G is given by

$$a_{i,j} = \begin{cases} w(e) & \text{if the edge } e \text{ is an edge from} \\ & \text{vertex } i \text{ to vertex } j, \\ \infty & \text{otherwise.} \end{cases}$$

Based on the approach specified by Maggs and Plotkin [13], the MST T of G can be represented by the following recursive formula:

$$\begin{cases} \text{Initially, } c^1_{i,j} = a_{i,j},\ 0 \le i,\ j < N. \\ \\ \text{For each iteration } l,\ 2 \le l \le N, \\ c^l_{i,j} = \min\limits_{0 \le k < N} \{\max\{c^{l/2}_{i,k},\ c^{l/2}_{k,j}\}\}, \\ 0 \le i, j < N. \\ \\ \text{Finally, Set } t_{i,j} = 1 \text{ if } c^N_{i,j} = a_{i,j}; \\ t_{i,j} = 0, \text{ otherwise.} \end{cases} \quad (2)$$

Like Chen *et al.* [3], $c^l_{i,j}$ of Eq. (2) at each iteration l can be computed using the matrix multiplication as mentioned in the previous subsection by replacing the operators $\oplus$ and $\otimes$ with minimum and maximum, respectively. That is, $c^l_{i,j}$ of Eq. (2) can be computed in $O(1)$ time at the iteration l by Lemma 6. Continuing this process at most $\log N$ iterations, $c^N_{i,j}$ can be obtained from $c^1_{i,j}$, $c^2_{i,j}$, $c^4_{i,j}$, ..., $c^{N/2}_{i,j}$. Finally, all edges of the MST T of G can be determined by setting $t_{i,j} = 1$ if $c^N_{i,j} = a_{i,j}$; $t_{i,j} = 0$, otherwise. Since each iteration l takes $O(1)$ time by Lemma 6 and the number of iterations is at most $\log N$, the total time complexity of the proposed algorithm is $O(\log N)$. Compared to the algorithm proposed by Chen *et al.* [3], our algorithm can be run in the same time complexity but reduce the number of processors by a factor N. Hence, this leads to the following lemma.

Lemma 7 *Given a graph G with N vertices, the minimum spanning tree of G can be solved in $O(\log N)$ time on a* 3-D $N \times N \times N$ *AROB.* □

4 Parallel Hierarchical Clustering Algorithm

Let $A = a_{i,j}$, $0 \le i < N$, $0 \le j < M$, be a pattern matrix of size $N \times M$ which consists of N patterns each with M features. In general, N is in the range of hundreds and $M < 30$. A hierarchical clustering is a sequence of partitions in which each partition is nested into the next partition in the sequence. A hierarchical clustering method is a procedure for transforming a proximity matrix into a sequence of nested partition [7]. The direct input to the hierarchical clustering is the proximity matrix D which is usually generated from a pattern matrix A. Each entry of the proximity matrix $D = d_{i,k}$, $0 \le i,\ k < N$, represent the pairwise indices of proximity according to the row and column of pattern matrix A. The proximity index is either a similarity or a dissimilarity. Because the Euclidean distance is the most common of Minkowski metrics, we use the Euclidean distance to measure the dissimilarity between patterns. That is,

$$d_{i,k} = \sqrt{\sum_{j=0}^{M-1} (a_{i,j} - a_{k,j})^2}, \quad 0 \le i,\ k < N. \quad (3)$$

The output of a hierarchical clustering algorithm can be represented by a *dendrogram* (*i.e.*, a level tree of nested partitions). Each level (denoted as l_i, $1 \le i < N$) consists of only one node (different to the regular tree), each representing a cluster. We can cut a dendrogram at any level to obtain a clustering.

Tsai *et al.* [24] have proposed an $O(\log^2 N)$ time parallel hierarchical clustering algorithm with the single-link method on a 3-D $N \times N \times N$ PARBS. The main idea of the algorithm proposed by Tsai *et al.* [24] can be described by the following three steps. First, transform the single-link clustering into building an MST of a graph derived from pattern matrix. Then, distinguish all $N-1$ clusters simultaneously by properly merging the edges of the MST. Finally, construct the level tree of the nested partitions of clustering. By the pipelined ability and reconfigurability of the optical buses, we also develop an efficient parallel clustering algorithm on a 3-D AROB in the following.

Let $G = (V, E, W)$ denote a weighted proximity graph derived from the proximity matrix D, where V is the set of vertices, E is the set of edges, $w(e_{i,j})$ is the associated weight of the edge $e_{i,j}$ and $e_{i,j} \in E$. Thus, the associated weight $w(e_{i,j})$ of edge $e_{i,j}$ is corresponding to the entry $d_{i,j}$ of D, where $e_{i,j} = (i, j)$ is the edge incident to vertices v_i and v_j. For the sake of convenience, we assume that no two edges in the MST have the same weight. Since D is a symmetric matrix and $d_{i,i} = 0$, $0 \le i < N$, only the upper triangular matrix of D is enough to specify the $N(N-1)/2$ edges of G. Hence, we can set $d_{i,j} = \infty$ for $0 \le j \le i < N$.

Let $T = (V, \{t_1, t_2, t_3, \cdots, t_{N-1}\})$ be the MST of G, where $w(t_1) < w(t_2) < w(t_3) < \cdots < w(t_{N-1})$. For agglomerative hierarchical clustering, the edges in T will be cut in the order of $t_1, t_2, t_3, \cdots, t_{N-1}$, and each cut corresponds to a level of the dendrogram, respectively. That is, cut the edge t_i in T will form the $(N-i)^{th}$ level of the nested partitions of the hierarchical clustering. At level l_{N-i}, two clusters containing the patterns corresponding to the two ending vertices of the cut edge t_i will be merged into together to form a new cluster. Since the newly generated cluster of each cut may contain many vertices, we set the vertex with the smallest vertex number among them as a supervertex (denoted as sv) to identify the newly generated cluster, and the other vertices in the same cluster can be discarded. In order to determine which clusters will be merged at level l_{N-i}, we may find the connected components containing the two ending vertices of the cut edge t_i respectively from the subtree which was made up of t_k, $1 \leq k < i$.

As for easily specified the information of each edge in MST and each nested partition, two arrays $T = t_{i,j}$ and $C = c_{i,j}$, $0 \leq i, j < N$, are used to store the edges of MST and the cluster, respectively. Each entry $t_{i,j}$ consists of three fields: tw, tl, tr, where tw represents the distance between vertices i and j, tl denotes the left ending vertex of edge (i, j) and tr denotes the right ending vertex of edge (i, j). Each entry $c_{i,j}$ consists of three fields: cn, cl, cr, where cn represents the cluster number, cl denotes the left child and cr denotes the right child.

Following the definition of proximity matrix D, proximity graph G and the MST T, the parallel algorithm for agglomerative hierarchical clustering can be described by the following three phases.

Phase 1. Compute the proximity matrix D from pattern matrix A.

Phase 2. Begin with the disjoint clustering and set each pattern as a cluster, then find an MST T of proximity graph G.

Phase 3. Cut the $N-1$ edges in the MST T in the order of weights simultaneously such that each cut forms a new clustering.

These three phases can be easily implemented in $O(\log N)$ time on a 3-D $N \times N \times N$ AROB. Initially, the pattern matrix $a_{i,j}$ is stored in the local variable $a(i, j, 0)$ of processor $P_{i,j,0}$, $0 \leq i < N$, $0 \leq j < M$. Finally, the dendrogram consisting of the level number, cluster number and two children are stored in the local variable $l(i, j, 0)$, $cn(i, j, 0)$, $cl(i, j, 0)$, and $cr(i, j, 0)$ of processor $P_{i,j,0}$, $0 \leq i \leq j < N$, respectively. The detailed single-link hierarchical clustering algorithm (**SLHCA**) is shown in the following.

Algorithm SLHCA(A, l, C);
/* A is an input variable. l and C are output variables. */

0: begin.

1: // Phase 1: Compute the proximity matrix D from pattern matrix A. //

1.1: // Distribute the value of $a_{i,j}$ over the $N \times M \times N$ processors, where $M < N$. Let features $a_{i,j}$ and $a_{k,j}$ of patterns i and k be stored in local variables $ai(i, j, k)$ and $ak(i, j, k)$, respectively. //
Copy $a(i, j, 0)$, $0 \leq i < N$, $0 \leq j < M$, to $ai(i, j, k)$, $0 \leq k < N$ through the i_0-dimension optical bus; then copy $ai(k, j, k)$, $0 \leq j < M$, $0 \leq k < N$, to $ak(i, j, k)$, $0 \leq i < N$ through the i_2-dimension optical bus. Thus, each processor $P_{i,j,k}$, $0 \leq i, k < N$, $0 \leq j < M$, holds two pattern features $a_{i,j}$ and $a_{k,j}$.

1.2: // Compute the Euclidean distance of the j^{th} feature of each pair of patterns i and k. //
Compute $d2(i, j, k) := (ai(i, j, k) - ak(i, j, k))^2$, $0 \leq i, k < N$, $0 \leq j < M$.

1.3: // Compute Eq. (3) //
Compute $d(i, 0, k) := (\sum_{j=0}^{M-1} d2(i, j, k))^{1/2}$, $0 \leq i, k < N$, through the i_1-dimension optical bus by Lemma 2.

1.4: // Copy $d(i, 0, k)$ to $d(i, k, 0)$. //
First copy $d(i, 0, k)$, $0 \leq i, k < N$, to $d'(i, k, k)$ through the i_1-dimension optical bus , then copy $d'(i, k, k)$, $0 \leq i, k < N$, to $d(i, k, 0)$ through the i_0-dimension optical bus. Finally, set $d(i, k, 0) := \infty$ for $0 \leq k \leq i < N$.

2: // Phase 2: Initial partition and find all edges of the MST T from G.//

2.1: // Find all edges of T from G. //
The proximity matrix D is a weighted matrix corresponding to proximity graph G. Find the MST T of G by Lemma 7. Let each edge of T be stored in the local variable $t(i, j, 0)$. That is, $tw(i, j, 0) = d(i, j, 0)$, $tl(i, j, 0) = i$ and $tr(i, j, 0) = j$ if the edge (i, j) is an edge

of T; $tw(i,\ j,\ 0) = \infty$, $tl(i,\ j,\ 0) = nil$ and $tr(i,\ j,\ 0) = nil$, otherwise.

2.2: // Sort the $N-1$ edges of T and put the sorted edges to diagonal processors. That is, t_i, $1 \leq i < N$, is stored in row i. //
Sort the edge $t(i,\ j,\ 0)$ according to its associated weight $tw(i,\ j,\ 0)$, $0 \leq i,\ j < N$, into nondecreasing order by Lemma 4. Assume the N^2 sorted data items are stored in the local variable $t(i,\ j,\ 0)$ of processor $P_{i,\ j,\ 0}$, $0 \leq i,\ j < N$, according to row-major order. As the result, the $N-1$ edges of T is moved to the first row of processor $P_{0,\ j,\ 0}$, $0 \leq j < N-1$. Then, copy $t(0,\ j-1,\ 0)$ to $t(0,\ j,\ 0)$ for $1 \leq j < N$. Finally, copy $t(0,\ j,\ 0)$ to $t(j,\ j,\ 0)$, for $1 \leq j < N$ through the i_2-dimension optical bus.

2.3: // Initial partition: set each pattern as a cluster and row i corresponding to level $N-i$. //
For $0 \leq i,\ j < N$, set $l(i,\ j,\ 0) := N - i$, $cn(0,\ j,\ 0) := j$, $cl(0,\ j,\ 0) := nil$, $cr(0,\ j,\ 0) := nil$, and $sv(0,\ j,\ 0) := 1$ (this means vertex j is a supervertex).

3: // Phase 3: Generate all clusters of each level in parallel. For the sake of convenience, a 3-D $N \times N \times N$ AROB can be partitioned into N 2-D $N \times N$ AROB which denoted as $2D-AROB_{i_0}$, $0 \leq i_0 < N$. A $2D-AROB_{i_0}$ contains processor $P_{i_2,\ i_1,\ i_0}$, $0 \leq i_2,\ i_1 < N$. //

3.1: // Identify the newly generated cluster of each level. //

3.1.1: // Construct the subtree of level l_i, $1 \leq i < N$. //
Copy $t(j,\ j,\ 0)$, $1 \leq j < N$, to $t(i,\ j,\ 0)$ for $j \leq i$ through the i_2-dimension optical bus. Thus, the subtree of l_{N-i} consists of edge $t(i,\ j,\ 0)$, $1 \leq j \leq i$.

3.1.2: // Label the vertices of each level which lie in the same connected component with vertices $tl(i,\ i,\ 0)$ and $tr(i,\ i,\ 0)$ by a new cluster number $N+i-1$. //
For those edges belonging to row i of $2D-AROB_0$, $1 \leq i < N$, find the connected component CC_i which contains vertices $tl(i,\ i,\ 0)$ and $tr(i,\ i,\ 0)$ from the subtree by Lemma 5 through the $i_1 \times i_0$-dimension processors. If a vertex j, $0 \leq j < N$, is belonged to the connected component CC_i found above, then set $cn(i,\ j,\ 0) := N+i-1$; $cn(i,\ j,\ 0) := nil$, otherwise.

3.1.3: // Find the supervertex of each cluster. //
For each row i of $2D-AROB_0$, $1 \leq i < N$, find the minimal index j of $cn(i,\ j,\ 0)$ with $cn(i,\ j,\ 0) \neq nil$, $0 \leq j < N$ by Lemma 1 through the i_1-dimension processors. Then, set $sv(i,\ j,\ 0) := 1$ if the vertex j is the minimal index; $sv(i,\ j,\ 0) := 0$, otherwise.

3.2: // Set the left and right children of each cluster. //

3.2.1: // Each l_i takes the latest cluster number which may be the child of the newly generated cluster, from l_k, $0 < i < k \leq N$. //
Processor $P_{i,\ j,\ 0}$ with $cn(i,\ j,\ 0) = nil$, $0 \leq i,\ j < N$, establishes the local connection $\{-S_{i_2},\ +S_{i_2}\}$; breaks this established local connection, otherwise.

3.2.2: // Find the children of each cluster. //
Processor $P_{i,\ j,\ 0}$ with $cn(i,\ j,\ 0) \neq nil$, $0 \leq i,\ j < N$, broadcasts $cn(i,\ j,\ 0)$ on the port $+S_{i_2}$ of the established optical bus. Processor $P_{i,\ j,\ 0}$ with $cn(i,\ j,\ 0) \neq nil$, $1 \leq i, j < N$, receives the broadcasted data from its port $-S_{i_2}$ on the established optical bus, and store it in $pre\_cn(i,\ j,\ 0)$.

3.2.3: // Set the cl and cr of each cluster. //
For each row i of $2D-AROB_0$, $1 \leq i < N$, set $cl(i,\ j,\ 0) := pre\_cn(i,\ tl(i,\ i,\ 0),\ 0)$ and $cr(i,\ j,\ 0) := pre\_cn(i,\ tr(i,\ i,\ 0),\ 0)$ if $sv(i,\ j,\ 0) = 1$ for $0 \leq j < N$ through the i_1-dimension optical bus; $cl(i,\ j,\ 0) := nil$ and $cr(i,\ j,\ 0) := nil$, otherwise.

3.3: // Broadcast the cluster of l_k, $0 < i < k \leq N$, which has not been merged to l_i. //

3.3.1: For each row i of $2D-AROB_0$, $1 \leq i < N$, processor $P_{i,\ j,\ 0}$ with

$cn(i, j, 0) = nil$, $0 \leq j < N$, establishes the local connection $\{-S_{i_2}, +S_{i_2}\}$; breaks this established local connection, otherwise.

3.3.2: // Broadcast $c(i, j, 0)$ and $sv(i, j, 0)$. //
For each row i of $2D - AROB_0$, $0 \leq i < N$, processor $P_{i,j,0}$ with $cn(i, j, 0) \neq nil$, $0 \leq j < N$, broadcasts $c(i, j, 0)$ and $sv(i, j, 0)$ on the port $+S_{i_2}$ of the established optical bus, respectively. Then, processor $P_{i,j,0}$ with $cn(i, j, 0) = nil$, $1 \leq i, j < N$, receives the broadcasted data from its port $-S_{i_2}$ on the established optical bus, and stores them in the corresponding local variables, respectively.

3.4: // Compact all clusters of each level into the left side. //
For each row i of $2D - AROB_0$, $0 \leq i < N$, processors $P_{i,j,0}$, $0 \leq j < N$, compacts $l(i, j, 0)$ and $c(i, j, 0)$ together through the i_1-dimension processors by Lemma 3 if the value of $sv(i, j, 0) = 1$.

end.

Theorem 1 *Given N patterns each with M features ($N \geq M$), the algorithm* SLHCA *can be computed in $O(\log N)$ time on a* 3-D $N \times N \times N$ AROB.

proof: The time complexity is analyzed as follows. Steps 1.1, 1.2 and 1.4 each takes $O(1)$ time. Step 1.3 takes $O(\log M)$ time by lemma 2. Hence, Step 1 takes $O(\log M)$ time. Step 2.1 takes $O(\log N)$ time using $N \times N \times N$ processors by Lemma 7. Step 2.2 takes $O(1)$ time by Lemma 4. Step 2.3 takes $O(1)$ time. Hence, Step 2 takes $O(\log N)$ time. Steps 3.1.1 and 3.1.2 each takes $O(1)$ time. Step 3.1.3 takes $O(1)$ time by Lemma 1. Hence, Step 3.1 takes $O(1)$ time. Steps 3.2 and 3.3 each takes $O(1)$ time. Step 3.4 takes $O(1)$ time by Lemma 3. Therefore, the total time complexity is $O(\log N)$ using N^3 processors. □

Compared to the algorithm proposed by Tsai *et al.* [24], the time complexity can be reduced from $O(\log^2 N)$ to $O(\log N)$ with the same number of processors.

5 Concluding Remarks

Clustering is a valuable tool for data analysis. In most applications such as pattern recognition in which the number of patterns may be very large. However, this problem can be overcome by using parallel processing algorithms running on multiprocessor computers. By integrating the advantages of both optical transmission and electronic computation, it makes the computation power of the parallel processing system enormously. To demonstrate the computation power of the AROB, we have designed an efficient parallel clustering algorithm on the AROB. The proposed algorithm is better and more elegant than the previous known results on various parallel computation models. The comparison result is summarized in Table 1.

References

[1] M. R. Anderberg, *Cluster Analysis for Applications*, Reading, Academic Press, New York, 1973.

[2] Y. Ben-Asher, D. Peleg, R. Ramaswami, and A. Schuster, "The power of reconfiguration," *Journal of Parallel and Distributed Computing*, vol. 13, pp. 139–153, 1991.

[3] G. H. Chen, B. F. Wang, and C. J. Lu, "On the parallel computation of the algebraic path problems," *IEEE Trans. on Parallel and Distributed Systems*, vol. 3, no. 2, pp. 251-256, 1992.

[4] R. C. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, Reading, John Wiley and Sons, New York, 1973.

[5] J. C. Gower and G. J. S. Ross, "Minimum spanning tree and single-linkage cluster analysis," *Applied Statistics*, vol. 18, pp. 54-64, 1969.

[6] Z. Guo, R. G. Melhem, R. W. Hall, D. M. Chiarulli, and S. P. Levitan, "Pipelined communications in optically interconnected arrays," *Journal of Parallel and Distributed Computing*, vol. 12, no. 3, pp. 269–282, 1991.

[7] A. Jain and R. Dubes, *Algorithms for Clustering Data*, Englewood Cliffs, NJ: Prentice-Hall, 1988.

[8] M. Jambu and M. O. Lebeaux, *Cluster Analysis and Data Analysis*, Reading, North-Holland Publishing Company, New York, 1983.

[9] S. Johnson, "Hierarchical clustering scheme," *Phychometrika*, vol. 23, pp. 241-254, 1967.

Table 1: Summary of comparison results for parallel hierarchical clustering.

Algorithm	Architecture	Processor	Time Complexity
Li and Fang [12]	hypercube	MN	$O(N \log N)$
Li [11]	shuffle-exchange networks	N	N^2
Tsai *et al.* [24]	PARBS	N^4	$O(1)$
		N^3	$O(\log^2 N)$
This paper	AROB	N^3	$O(\log N)$

[10] T. Kurita, "An efficient agglomerative clustering algorithm using a heap," *Pattern Recognition,* vol. 24, no.3, pp. 205-209, 1991.

[11] X. Li, "Parallel algorithms for hierarchical clustering and cluster validity," *IEEE Trans. on Patt. Analysis. and Machine Intellig.,* Vol. PAMI-12, no. 11, pp. 1088-1092, 1990.

[12] X. Li and Z. Fang, "Parallel clustering algorithms," *Parallel Computing,* Vol. 11, pp. 275-290, 1989.

[13] B. M. Maggs and S. A. Plotkin, "Minimum-cost spanning tree as a path-finding problem," *Information Processing Letters,* vol.26, 291-293, 1988.

[14] R. Miller, V. K. P. Kumar, D. Reisis, and Q. F. Stout, "Image computations on reconfigurable VLSI arrays," *Proc. of IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* pp. 925-930, 1988.

[15] F. Murtagh, *Multidimensional Clustering Algorithms,* Physica-Verlog, 1985.

[16] L. M. Ni and A. K. Jain, "A VLSI systolic architecture for pattern clustering," *IEEE Trans. on Patt. Analysis. and Machine Intellig.,* vol. PAMI-7, no.1, pp. 79-89, 1985.

[17] Y. Pan and M. Hamdi, "Quicksort on a linear arrays with a reconfigurable pipelined bus system," *Proc. of Int. Symposium on Parallel Architectures, Algorithms and Networks,* pp. 313-319, 1996.

[18] Y. Pan and K. Li, "Linear array with a reconfigurable pipelined bus system— concepts and applications," *Proc. of Int. Conf. on Parallel and Distributed Processing Techniques and Applications,* pp. 1431-1442, 1996.

[19] S. Pavel and S. G. Akl, "On the power of arrays with reconfigurable optical bus," *Proc. of Int. Conf. on Parallel and Distributed Processing Techniques and Applications,* pp. 1443-1454, 1996.

[20] S. Pavel and S. G. Akl, "Matrix operations using arrays with reconfigurable optical buses," *Parallel Algorithms and Applications,* vol. 8, pp. 223-242, 1996.

[21] S. Pavel and S. G. Akl, "Integer sorting and routing in arrays with reconfigurable optical buses," *Int. Journal of Foundations of computer science,* vol. 9, pp. 99-120, 1998.

[22] S. Rajasekaran and S. Sahni, "Sorting, selection and routing on the arrays with reconfigurable optical buses," *IEEE Transactions on Parallel and Distributed Systems,* vol. 8, no. 11, pp. 1123-1132, 1997.

[23] F. J. Rohlf, "Hierarchical clustering using the minimum spanning tree," *Computer Journal,* vol. 16, pp. 93-95, 1973.

[24] H. R. Tsai, S.J. Horng, S. S. Lee, S. S. Tsai, and T. W. Kao, "Parallel hierarchical clustering algorithms on processor arrays with a reconfigurable bus system," *Pattern Recognition,* vol. 30, no. 5, pp. 801-815, 1997.

[25] B. F. Wang and G. H. Chen, "Constant time algorithms for transitive closure and some related graph problems on processor arrays with reconfigurable bus systems," *IEEE Trans. on Parallel and Distributed Systems,* vol. 1, no. 10, pp. 500-507, 1990.