Journal of Computer and System Sciences **56**, 320–331 (1998)
Article No. SS981572

Lin Chen

FRL, P.O. Box 18345, Los Angeles, California 90018

Received January 24, 1996; revised July 28, 1997

We investigate some properties of minimal interval and circular arc representations and give several optimal sequential and parallel recognition and construction algorithms. We show that, among other things, given an $s \times t$ interval or circular arc representation matrix,

- deciding if the representation is minimal can be done in $O(\log s)$ time with $O(st/\log s)$ EREW PRAM processors, or in $O(1)$ time with $O(st)$ common CRCW PRAM processors;
- constructing an equivalent minimum interval representation can be done in $O(\log(st))$ time with $O(st/\log(st))$ EREW PRAM processors, or in $O(\log t/\log \log t)$ time with $O(st \log \log t/\log t)$ common CRCW PRAM processors, or in $O(1)$ time with $O(st)$ BSR processors;
- constructing an equivalent minimal circular arc representation can be done in $O(st)$ time. © 1998 Academic Press

1. INTRODUCTION

Circular arc graphs are well-known class of intersection graphs and properly contain interval graphs. Benzer [7] showed that overlap data involving fragments of a certain gene could be modeled by a set of intervals. This finding confirmed the hypothesis that DNA has a linear structure within genes and helped him win a Nobel prize. Circular arc graphs also find applications in some other areas such as register allocation. The best way to allocate registers corresponds to an optimal coloring of an interference graph which is often a circular arc graph or even an interval graph (see, e.g., [25]). Many algorithms on circular arc graphs in the literature work on circular arc representations (see, e.g., [4, 24, 29]) which can be constructed from circular arc graphs (see, e.g., [12, 15, 30]). Each circular arc representation can be represented by a (0, 1)-matrix. For each circular arc graph, there are infinitely many circular arc representations, among which minimized representations are the more efficient ones with no loss of information. In this paper, we study the properties of minimal interval and circular arc representations and present some efficient sequential and parallel recognition and construction algorithms.

In the next section, we will give some definitions and briefly review some prior work which helps in establishing the validity of our work. In Section 3, we investigate the properties of minimal interval and circular arc representations. Based on these properties, we give efficient sequential and parallel algorithms for deciding and constructing minimal interval and circular arc representations in Section 4. Finally, in Section 5, we conclude the paper with some discussion.

2. PRELIMINARIES

Given a finite family S of nonempty sets, the intersection graph G has vertices corresponding to the sets of S and two distinct vertices of S are adjacent if and only if the corresponding sets of S intersect. S is called an *intersection representation* for G. If S is a family of arcs on a circle, G is called a *circular arc graph*. If S is a family of arcs on a circle satisfying the Helly property (i.e., a family of arcs on a circle such that if several arcs mutually intersect, then the intersection of these arcs is nonempty), then G is called a *Helly circular arc graph* [23]. Helly circular arc graphs are also known as *Θ circular arc graphs* [20].

In this paper, we often use a pair of the locations of two endpoints of an arc in brackets to denote a *closed arc*, i.e., an arc that includes its two endpoints. If we move along an arc in the clockwise direction, the last point on the arc is called *clockwise endpoint*. The *counterclockwise endpoint* is defined analogously. If we use $[l_0, l_1]$ to denote an arc, l_0 indicates the location of the counterclockwise endpoint whereas l_1 indicates the location of the clockwise endpoint. Endpoints and the location of endpoints are sometimes used interchangeably if there is no confusion.

A graph is called an *interval graph* if it is an intersection graph on a family of intervals on a real line. If we embed a set of intervals on a circle which is large enough such that the intervals do not cover the entire circle, then we have a circular arc representation for a circular arc graph. It is known (see, e.g., [12]) that interval graphs are a proper subclass of Θ circular arc graphs, which are in turn a proper subclass of circular arc graphs.

* Portions of this paper appeared in preliminary form in ISCAS'89 Proceedings [9], ISAAC'94 Proceedings [13], and EuroPar'95 Proceedings [14].

The aforementioned classes of graphs can also be equivalently defined as intersection graphs on some finite sets. Take the circular arc graphs for example. Let D be a circularly ordered finite set (such as points on a circle). A circular arc of D is defined as any set of contiguous elements of D. Let S be a set of circular arcs on D. The intersection graph G of S is a circular arc graph and the pair (D, S) is a circular arc representation. Two intersection representations (D_1, S_1) and (D_2, S_2) are said to be *equivalent* if there exists a one-to-one onto function $f\colon S_1 \rightarrow S_2$ such that x and y in S_1 intersect if and only if $f(x)$ and $f(y)$ in S_2 intersect. (In other words, two intersection representations are equivalent if and only if the two corresponding intersection graphs are isomorphic.) An intersection representations (D, S) is said to be *minimal* if there does not exist an element d in D such that (D', S') is an equivalent intersection representation, where $D' = D - \{d\}$ and S' is the corresponding set. An intersection representation (D, S) is said to be *minimum* if, for any other equivalent intersection representation (D', S'), $|D'| \geqslant |D|$. We call $|D|$ the *size* of the intersection representation (D, S). An element, say d, in D is called an *intersection point* if there exist two elements (not necessarily distinct), say s_1 and s_2, in S such that $s_1 \cap s_2 = \{d\}$. An intersection representation, say (D, S), is often denoted by a $|S| \times |D|$ (0, 1)-matrix. A row, say R, of the matrix corresponds to an element in S. $R(i) = 1$ if and only if the ith element of D is contained in the element of S. We will simply refer to the matrix as an intersection representation if no confusion arises.

A (0, 1)-matrix is said to satisfy the *consecutive* 1'*s property* (for rows) if its columns can be permuted in such a way that the resulting matrix has consecutive 1's in each of its rows. A (0, 1)-matrix is said to satisfy the *circular* 1'*s property* (for rows) if its columns can be permuted in such a way that the resulting matrix has circularly consecutive 1's in each of its rows. By definition, an interval representation matrix satisfies the consecutive 1's property and a circular arc representation matrix satisfies the circular 1's property.

Suppose $D_1 = \{\clubsuit, \diamondsuit, \heartsuit, \spadesuit\}$, and $S_1 = \{\{\clubsuit, \diamondsuit\}, \{\diamondsuit, \heartsuit\}, \{\heartsuit, \spadesuit\}, \{\spadesuit\}\}$. The corresponding matrix is

$$M_1 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Let $D_2 = \{\text{Jack, Queen, King, Ace}\}$, and $S_1 = \{\{\text{Jack}\}, \{\text{Jack, Queen, King}\}, \{\text{Queen, King, Ace}\}, \{\text{Ace}\}\}$. Then the corresponding matrix is

$$M_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

It is easy to verify that these two interval representations are equivalent. An equivalent minimal interval representation is

$$M_3 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix},$$

which can be obtained by deleting the first column of M_1, or by deleting the second or the third column of M_2. Let us consider a circular arc representation

$$M_4 = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}.$$

Deleting the second or the third column yields an equivalent minimal circular arc representation

$$M_5 = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}.$$

The minimum circular arc representation is

$$M_6 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}.$$

In fact, the intersection graph is also an interval graph and M_6 is the minimum interval representation.

The computation models employed in this paper are more or less standard. One model used is the well known parallel random access machine (PRAM) (see, e.g., [3]). Some of our algorithms are implemented on exclusive read exclusive write (EREW) PRAM, in which case no concurrent access is allowed. Some other algorithms are designed for the common concurrent read concurrent write (CRCW) PRAM, for which concurrent access is allowed but the processors must write the same value into a memory location in the event of concurrent writes. Also mentioned is a stronger submodel of CRCW PRAM called priority CRCW PRAM, for which the processor with the highest priority succeeds in writing in case of concurrent writes.

It is straightforward to show that if a problem can be solved in $O(T)$ time with $O(P)$ processors on a PRAM, then the problem can also be solved in $O(PT)$ time on a RAM. A PRAM algorithm is said to be *work-optimal* if its work bound matches the lower time bound of the sequential algorithm. We say an algorithm, whether sequential or parallel, is *time-optimal* if its time bound matches the lower bound on the corresponding model.

Another model used in this paper is a relatively new one called broadcasting with selective reduction (BSR) introduced in Akl and Guenther [5]. It can be viewed as a CRCW PRAM with one extension: the BROADCAST instruction, which allows all processors to gain access to all memory locations simultaneously for the purpose of writing. The BROADCAST instruction is denoted by $x_j := \mathscr{R}_{t_i \sigma l_h} d_i$, for $1 \leqslant j \leqslant m$ and $1 \leqslant i \leqslant n$. The ranges of the variables i and j are sometimes omitted if they are understood. In the instruction, d_i is the datum broadcast by processor p_i, t_i is the associated tag, σ is one of the section operations in $\{<, \leqslant, =, \geqslant, >, \neq\}$, and $\mathscr{R}$ is one of the binary associative reduction operations in $\{\sum, \prod, \wedge, \vee, \oplus, \cap, \cup\}$, denoting, respectively, Sum, Product, And, Or, Exclusive Or, Maximum, and Minimum.

The BROADCAST instruction is carried out as follows. For each memory location x_j (with an associated limit value l_j), the proposition $(t_i \sigma l_j)$ is tested over all broadcast pairs (t_i, d_i). In every case for which t_i satisfies the proposition, d_i is accepted by location x_j. The set of all data accepted by x_j is reduced to one value based on the specified binary associative operation $\mathscr{R}$, and stored in x_j. If no data are accepted by a memory location, then there is no change in the value of the corresponding variable.

In designing PRAM algorithms, we often use the following result, usually attributed to Brent [8], to obtain the best time and processor bounds.

THEOREM 1. *If a problem can be solved in $O(T)$ time with $O(W)$ work on a PRAM, then the problem can also be solved in $O(T + W/P)$ time with P processors on the same PRAM.*

Cook, Dwork, and Reischuk [17] established the following lower bound.

THEOREM 2. *Computing the OR of n bits requires at least $\Omega(\log n)$ time on machines without simultaneous writes.*

However, on CRCW PRAM, the OR of n bits can be trivially obtained in $O(1)$ time with n processors. The following lower bound on CRCW PRAM has been given in Beame and Hastad [6].

THEOREM 3. *Checking parity for n bits requires at least $\Omega(\log n/\log\log n)$ time on priority CRCW PRAM if a polynomially bounded number of processors are used.*

We say a problem is in NC if there is an algorithm for it that runs in polylogarithmic time with a polynomially bounded number of processors. Such an algorithm is called an *NC algorithm*. So Theorem 3 tells us that any NC algorithm for the parity checking problem requires at least $\Omega(\log n/\log\log n)$ time on a priority CRCW PRAM. In this paper, we restrict our discussion of parallel algorithms to NC algorithms. So any parallel algorithm mentioned in the paper is meant to be an NC algorithm.

As is known, checking parity for n bits is actually computing the XOR (Exclusive Or) of n bits. On the BSR model, it can be done in one step by a single BROADCAST instruction; $p_j := \bigoplus_{1=1} a_i$ for $0 < i \leqslant n$ and $j = 1$. Therefore, BSR is in a sense strictly more powerful than priority CRCW PRAM.

One very useful procedure in parallel computing is the prefix sum computation. It is not hard to see that the above PRAM lower bounds apply to the prefix sum computation. It is known that all prefix sums for an array of n elements can be obtained in $O(\log n)$ time using $O(n/\log n)$ EREW PRAM processors [27], or in $O(\log n/\log\log n)$ time using $O(n \log\log n/\log n)$ common CRCW PRAM processors [16], or in $O(1)$ time using $O(n)$ BSR processors [28].

One problem mentioned in Chen [11] is the subarray computation. Given an array, say $a[1:n]$, composed of two types of elements, the problem is to obtain a subarray $b[1:k]$ of $a[1:n]$ such that $b[j]$ is the jth element in a of type 1, for $0 < j \leqslant k$, where k is the number of elements in a of type 1. The problem can be solved using the procedure for computing the prefix sums. It is now easy to conclude the following.

THEOREM 4. *The subarray computation can be done in $O(\log n)$ time with $O(n/\log n)$ EREW PRAM processors, or in $O(\log n/\log\log n)$ time with $O(n \log\log n/\log n)$ common CRCW PRAM processors. Both procedures are work-optimal. On the BSR model, the problem can be solved in $O(1)$ time with $O(n)$ processors. The algorithms are all time-optimal.*

These results are used frequently in designing other parallel algorithms. We may not make explicit reference to these results every time we use them later in this paper. In this paper, the minimal intersection representation matrices are obtained through column deletion. So the lower bound for subarray computation also applies to computing minimal representations, just as the $\Omega(n \log n)$ sorting lower bound is valid if the sorting is done by comparison (see, e.g., [2]).

3. PROPERTIES

In this section, we investigate some properties of minimal interval and circular arc representations.

LEMMA 1. *Suppose (D, I) is an interval representation. An element $d \in D$ is an intersection point if and only if there exist intervals I_i and $I_j \in I$ (I_i and I_j may be the same interval) such that d is the left endpoint of I_i and the right endpoint of I_j.*

Proof. ($\Rightarrow$) By definition.

($\Leftarrow$) If d is the left endpoint of I_i and also the right endpoint of I_j, then the intersection of I_i and I_j is $\{d\}$. It follows from the definition that d is an intersection point. ▮

LEMMA 2. *Suppose (D, I) is an interval representation, and M is the corresponding matrix. An element of D is an intersection point if and only if the corresponding column of M is not contained in any other column.*

Proof. Without loss of generality, assume $|D| \geqslant 2$.

($\Rightarrow$) Suppose an element d in D is an intersection point. Then there exist two intervals I_i and I_j such that their intersection is $\{d\}$. Consider column Y_s of M corresponding to the element d. Let Y_t be an arbitrary column of M other than Y_s. Since d is an intersection point, one of the two entries $Y_t(i)$ and $Y_t(j)$ must be 0. So Y_s is not contained in Y_t.

($\Leftarrow$) Suppose column Y_s is not contained in any other column of M, and suppose d is the corresponding element in D. Let us consider three cases one by one.

Case 1. Y_s is the first column. Then there exists an i such that $Y_1(i)=1$ and $Y_2(i)=0$. Recall that the 1's in each row (including row i) of M are consecutive. It follows that the ith row of M begins with a 1 followed by a sequence of 0's. So the intersection of I_i and I_i equals $\{d\}$. Therefore, d is an intersection point.

Case 2. Y_s is the last column. Analogous to Case 1.

Case 3. Y_s is neither the first column nor the last column. Since Y_s is not contained in any other column, there exists an integer i such that $Y_s(i)=1$ and $Y_{s-1}(i)=0$. Likewise, there exists an integer j such that $Y_s(j)=1$ and $Y_{s+1}(j)=0$. Then the intersection of I_i and I_j equals $\{d\}$. So d is an intersection point. ▮

THEOREM 5. *Suppose (D, I) is an interval representation and M is the corresponding matrix. The following four assertions are equivalent*:

1. *(D, I) is a minimum interval representation.*
2. *(D, I) is a minimal interval representation.*
3. *Every element of D is an intersection point.*
4. *No column of M contains another.*

Proof. ($1 \Rightarrow 2$) Any minimum interval representation is a minimal one, by definition.

($2 \Rightarrow 3$) Suppose an element d in D is not an intersection point. Let Y_k be the corresponding column of M. Delete Y_k from M. Denote by M' the resulting matrix. It is easy to see that any two rows, say the ith and the jth rows, of M intersect if and only if the ith and the jth rows of M' intersect. So M' is an interval representation matrix for the same graph as M, which contradicts the assumption that (D, I) is a minimal interval representation. It follows that every element of D is an intersection point.

($3 \Rightarrow 4$) Immediate from Lemma 2.

($4 \Rightarrow 1$) Suppose G is the interval graph corresponding to M. Note that any column of M corresponds to a maximal clique of G since the column is not contained in any other column of M. So for any interval representation matrix M', there exists a permutation matrix P such that every column of M appears in PM'; otherwise, M' would not be an interval representation matrix for G. It follows that M corresponds to a minimum interval representation. ▮

It is already known that a graph is an interval graph if and only if the vertex versus maximal clique incidence matrix satisfies the consecutive 1's property [19, 21]. Gilmore and Hoffman [21] showed that a vertex versus maximal clique incidence matrix with consecutive 1's in each row corresponds to a minimum interval representation. The minimality can be easily established by Theorem 5.

THEOREM 6. *If M is a vertex versus maximal clique incidence matrix of an n-vertex interval graph G with consecutive* 1*'s in each of its rows, then M corresponds to a minimal* (*minimum*) *interval representation of G. In addition, the size of a minimal* (*minimum*) *interval representation of an interval graph G equals the number of the maximal cliques of G and is at most n.*

Proof. Since each column of M is distinct and corresponds to a maximal clique of G, it follows that no column of M contains another. By Theorem 5, M corresponds to a minimal (minimum) interval representation of G. So the size of a minimal (minimum) interval representation of an interval graph G equals the number of the maximal cliques of G. Since M has n rows, there can be at most n intersection points. It then follows from Theorem 5 that the size of a minimal (minimum) interval representation is at most n. ▮

It is interesting to note that the analogous statement of Theorem 5 is not valid for circular arc representation. Take a look at M_5 in Section 2. M_5 is a minimal circular arc representation, but it is not a minimum circular arc representation. In fact, the analogous statements of Lemma 1 and Lemma 2 for circular arc representation are not valid either. Consider matrix

$$M_7 = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix}.$$

The first column corresponds to the counterclockwise endpoint of the first circular arc (row) and the clockwise endpoint of the second circular arc (row). However, it does not correspond to any intersection point. In the matrix, no column is contained in another, but the intersection of any two circular arcs contains two points. So there are no intersection points. We will instead give the following results for the circular arc representation.

LEMMA 3. *Suppose (D, A) is a circular arc representation, and M is the corresponding matrix. A column of M is not contained in any other column if the corresponding element of D is an intersection point.*

Proof. Analogous to the "only if" part of the proof of Lemma 2. ▮

THEOREM 7. *Suppose (D, A) is a circular arc representation and M is the corresponding matrix. For the following four assertions, one implies the next:*

1. *(D, A) is a minimum circular arc representation.*
2. *(D, A) is a minimal circular arc representation.*
3. *Every element of D is an intersection point.*
4. *No column of M contains another.*

Proof. $(1 \Rightarrow 2)$ By definition.
$(2 \Rightarrow 3)$ Analogous to the corresponding part of the proof of Theorem 5.
$(3 \Rightarrow 4)$ Immediate from Lemma 3. ▮

We have shown above that Assertion 4 in Theorem 7 does not imply Assertion 3 and Assertion 2 does not imply Assertion 1, an interested reader will ask if Assertion 3 implies Assertion 2. The answer is "yes." The following theorem gives a necessary and sufficient condition for the minimal circular arc representation.

THEOREM 8. *Suppose (D, A) is a circular arc representation. The representation is minimal if and only if every element of D is an intersection point.*

Proof. $(\Rightarrow)$ By Theorem 7.
$(\Leftarrow)$ Let G be the circular arc graph corresponding to the circular arc representation (D, A). Assume every element of D is an intersection point, but the representation is not minimal. Then there exists an element, say d, in D such that the intersection graph (denoted by G') of (D', A') is isomorphic to G, where $D' = D - \{d\}$ and A' is the corresponding set of circular arcs. Recall that every element (including d) of D is an intersection point. It follows that G' contains fewer edges than G. Consequently, G and G' are not isomorphic. Thus a contradiction is derived. This completes the proof. ▮

For circular arc representation satisfying the Helly property, we have the following properties.

THEOREM 9. *Suppose (D, A) is a Θ circular arc representation and M is the corresponding matrix. The following three assertions are equivalent:*

1. *(D, A) is a minimum Θ circular arc representation.*
2. *(D, A) is a minimal Θ circular arc representation.*
3. *No column of M contains another.*

Proof. $(1 \Rightarrow 2)$ By definition.
$(2 \Rightarrow 3)$ Assume M corresponds to a minimal Θ circular arc representation, and M contains two distinct columns, say Y_i and Y_j, such that Y_i is contained in Y_j. Delete Y_i from M and denote by M' the resulting matrix. It is easy to see that M' corresponds to a Θ circular arc representation for the same graph as M does, which contradicts the assumption that M corresponds to a minimal Θ circular arc representation.
$(3 \Rightarrow 1)$ Suppose G is the Θ circular arc graph corresponding to M. Observe that each column of M corresponds to a maximal clique. For any Θ circular arc representation matrix M', there exists a permutation matrix P such that every column of M appears in PM'; otherwise, M' would not be a Θ circular arc representation matrix for G. It follows that M corresponds to a minimum Θ circular arc representation. ▮

Note that a minimal Θ circular arc representation is not necessarily a minimal circular arc representation. Let us take a look at the following example:

$$M_8 = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$

M_8 corresponds to a minimal Θ circular arc representation but not a minimal circular arc representation since the deletion of column 3 yields an equivalent circular arc representation.

Gavril [20] characterized Θ circular arc graphs as graphs whose vertex versus maximal clique incidence matrices satisfy the circular 1's property, and showed that a vertex versus maximal clique incidence matrix with circular 1's in each row corresponds to a Θ circular arc representation. It now follows easily from Theorem 9 that vertex versus maximal clique incidence matrices with circular 1's in each row actually correspond to minimum Θ circular arc representations.

4. PROCEDURES

In this section we will give concrete computational procedures for the recognition and construction of minimum interval and circular arc representations. We will also study the optimality of the procedures.

We begin with the problem of testing for minimum interval representations. We note that if an $s \times t$ matrix is a minimum interval representation matrix, then $s \geqslant t$. The reason is that if the matrix is a minimum interval representation matrix, then all columns of the matrix correspond to intersection points by Theorem 5. Thus, the number of left endpoints and therefore the number of rows will be at least t. Consequently, if $s < t$, we can conclude immediately that the representation is not minimum. Suppose we have an interval representation matrix with size $s \times t$ ($s \geqslant t$). We first check which columns correspond to the intersection points. Obtaining the intersection points is easy. By Lemma 1, a column corresponds to an intersection point if and only if it corresponds to the left endpoint of an interval and also the right endpoint of an interval. For each row (interval), we can decide its two endpoints in $O(1)$ time with t EREW PRAM processors. It then follows easily that all the intersection points can be identified in $O(1)$ time with $O(st)$ common CRCW PRAM processors. For instance, we can decide which of the t columns correspond to intersection points in the following way:

```
0  for $i := 1$ to $t$ codo $l[i] := 0$ odoc;
1  for $i := 1$ to $s$ codo $l[b[i]] := 1$ odoc
```

In the above code $b[i]$ gives the location where interval i begins. After the execution of the above code, column i is the left endpoint of an interval if and only if $l[i] = 1$.

On the EREW PRAM model, the problem cannot be solved in $O(1)$ time. In fact, we have established a lower bound stated in the following theorem.

THEOREM 10. *Deciding if an $s \times t$ interval representation matrix is minimum requires at least $\Omega(\log s)$ time on a CREW PRAM, for $s \geqslant t$.*

Proof. We prove the theorem by a reduction from computing the OR. Let $b[1], b[2], ..., b[n]$ be n bits. We construct an $n \times n$ matrix M as

$$M[i, j] = \begin{cases} 1, & \text{if} \quad b[i] = 1 \wedge 0 < j \leqslant n, \\ 0, & \text{if} \quad b[i] = 0 \wedge i \neq j \wedge 0 < j \leqslant n, \\ 1, & \text{if} \quad b[i] = 0 \wedge i = j. \end{cases}$$

Obviously, M can be constructed in constant time with $O(n^2)$ processors. According to the construction, column i is not contained in any other column and corresponds to an intersection point if and only if $b[i] = 0$, for any i. Therefore, no column of M contains another and all columns of M correspond to intersection points if and only if the OR of the n bits is 0. By Theorem 5, no column of M contains another and all columns of M correspond to intersection points if and only if M is a minimum interval representation. So, if deciding whether an interval representation with s intervals is minimum can be done in $o(\log s)$ on a CREW PRAM then the OR of n bits can also be computed in $o(\log n)$ time on a CREW PRAM, which contradicts Theorem 2. This completes the proof. ∎

In fact, the reduction in the proof of the above theorem can also be used to obtain the $\Omega(\log s)$ time lower bound for deciding if an $s \times t$ circular arc representation matrix is minimal (recall from Theorem 8 that all columns of M correspond to intersection points if and only if M is a minimal circular arc representation) and deciding if an $s \times t$ Θ circular arc representation matrix is minimum (recall from Theorem 9 that no column of M contains another if and only if M is a minimum Θ circular arc representation matrix) for $s \geqslant t$, on a CREW PRAM. We list the result as the following theorem.

THEOREM 11. *At least $\Omega(\log s)$ time is required for deciding if an $s \times t$ circular arc representation matrix is minimal and if an $s \times t$ Θ circular arc representation matrix is minimum, on a CREW PRAM, for $s \geqslant t$.*

Below we will show that deciding if an interval representation is minimum can be done in $O(\log s)$ time by an optimal EREW PRAM procedure. To implement the algorithm on EREW PRAM, we will use additional arrays to avoid concurrent access. To decide which columns correspond to left endpoints, another $s \times t$ array ml is used:

```
0  for $i := 1$ to $t$ codo $l[i] := 0$ odoc; {initialize $l$}
1  for $i := 1$ to $s$ codo {initialize $ml$}
2    for $j := 1$ to $t$ codo
3      $ml[i, j] = 0$;
4    odoc;
5  odoc;
6  for $i := 1$ to $s$ codo $ml[i, b[i]] := 1$ odoc;
   {row $i$ of $ml$ contains the first 1 of row $i$ of the
    input matrix}
7  for $i := 1$ to $t$ codo $l[i] := \bigvee_{j=1}^{s} ml[j, i]$ odoc;
```

After executing the above code, $l[i] = 1$ if and only if column i corresponds to a left endpoint. The only step that requires more than constant time is line 7, which can be done in $O(\log s)$ time on an EREW PRAM. All steps can be done optimally. In an analogous way, we can also decide which columns correspond to right endpoints. Then we know immediately which columns correspond to intersection points. Now, deciding if all columns correspond to intersection points can be done in $O(T)$ time with $O(t/T)$ EREW PRAM processors for any T, $\log t \leqslant T \leqslant t$. Recall that the intersection points are computed only when $t \leqslant s$. Therefore, we can conclude the following.

THEOREM 12. *Given an $s \times t$ interval representation matrix, deciding if the representation is minimum can be done in $O(\log s)$ time with $O(st/\log s)$ EREW PRAM processors, or in $O(1)$ time with $O(st)$ common CRCW PRAM processors. Both algorithms are time-and-work-optimal.*

If an interval representation matrix is not minimum, we can obtain an equivalent minimum one by deleting some columns. However, we cannot obtain such a matrix by simply deleting all columns that do not correspond to intersection points initially, since deleting one column may change the status of a neighboring column and make it correspond to an intersection point. Below we describe a sequential procedure for obtaining an equivalent minimum interval representation.

First compute $l[i]$ and $r[i]$ for $0 < i \leqslant t$. Then perform the following task:

```
0  for $i := 1$ to $t$ do $m[i] := l[i] \wedge r[i]$ od;
1  $i := 1$;
2  while $i <= t$ do
3    if $l[i] = 1 \wedge r[i] = 0$ then
4      while $r[i] = 0$ do $i := i + 1$ od;
5      $m[i] := 1$;
6    fi;
7    $i := i + 1$;
8  od;
```

Line 0 sets $m[i]$ to 1 if and only if column i corresponds to an intersection point. Then we scan the columns from left to right (lines 2–8). If a column corresponds to a left endpoint but not a right endpoint, we will repeatedly remove columns (keep $m[i]$ as 0) until we have reached a column that corresponds to a right endpoint (line 4). Then the column corresponds to an intersection point and will remain (set $m[i]$ to 1 at line 5), regardless of whether or not the column corresponds to an intersection point initially. When the above procedure terminates, the columns that correspond to intersection points ($m[i] = 1$) form an equivalent minimum interval representation.

The parallel procedure can work as follows. First, set $m[i]$'s as line 0. Then, for each left endpoint that does not correspond to an intersection point, find the closest right endpoint, say k, to its right, and set $m[k]$ to 1. It has been shown that finding the first 1 in a (0, 1)-array can be done in $O(1)$ time with $O(n)$ common CRCW PRAM processors [18]. So we can also set $m[k]$ in constant time on common CRCW PRAM. On the EREW PRAM, finding the closest right endpoint takes $O(\log t)$ time. Once we have identified all the columns corresponding to maximal cliques, we simply apply subarray computation on all the rows. Note that on EREW PRAM, computing an equivalent minimum interval representation by column deletion requires at least $\Omega(\log t)$ time (see Theorem 4) and at least $\Omega(\log s)$ time (see Theorem 10) and therefore at least $\Omega(\log s + \log t)$ time. It is now easy to conclude the following.

THEOREM 13. *Given an $s \times t$ interval representation matrix, an equivalent minimum interval representation can be obtained in $O(\log(st))$ time with $O(st/\log(st))$ processors by a time-and-work-optimal EREW PRAM algorithm, or in $O(\log t/\log \log t)$ time with $O(st \log \log t/\log t)$ processors by a time-and-work-optimal common CRCW PRAM algorithm, or in $O(1)$ time with $O(st)$ processors by a time-optimal BSR algorithm.*

As is mentioned above, each column of a minimum interval representation matrix corresponds to a maximal clique and the matrix gives the set of all maximal cliques.

Next we will consider the problem of deciding if a circular arc representation is minimal. It follows from Theorem 8 that the problem can be solved by checking if each column of the circular arc representation matrix corresponds to an intersection point. For the same reason as above, if the input matrix is of size $s \times t$ and $s < t$, then we can conclude immediately that the representation is not minimal. So we only need to consider the case $s \geqslant t$. For circular arc representations, we cannot use exactly the same method in recognizing intersection points since a point is not necessarily an intersection point even if it is both the clockwise endpoint and the counterclockwise endpoint of two arcs.

The procedure works as follows. We first locate all the clockwise endpoints and counterclockwise endpoints. Then for each column, say i, perform the following task. If the column corresponds to a clockwise endpoint and a counterclockwise endpoint, then find the shortest arcs, say a and b, whose clockwise and counterclockwise endpoints are i, respectively. Then i is an intersection point if and only if the size of intersection between a and b is 1. Both a and b can be found using a variation of the procedure for finding the first 1 in a (0, 1)-array, which takes $O(1)$ time and $O(t)$ common CRCW PRAM processors [18], or $O(\log t)$ time and $O(t/\log t)$ EREW PRAM processors. It is now easy to conclude the following.

THEOREM 14. *Given an $s \times t$ circular arc representation matrix, deciding if the representation is minimal can be done in $O(\log s)$ time with $O(st/\log s)$ EREW PRAM processors, or in $O(1)$ time with $O(st)$ common CRCW PRAM processors. Both algorithms are time-and-work-optimal.*

However, obtaining an equivalent circular arc representation from an arbitrary circular arc representation can not be done in an analogous way and requires additional work. Next we will consider how to construct an equivalent minimal circular arc representation from a circular arc representation. The method is sketched as follows. For the same reason as before, we only need to consider the case when $s \geqslant t$. First we obtain all the clockwise endpoints $e[i]$'s and counterclockwise endpoints $b[i]$'s of all the circular arcs. Then starting from the first column and ending at the last column, we perform the following task for each column: Check, based on the values of $b[i]$'s and $e[i]$'s, if the current column, say c, corresponds to an intersection point. If so, set $m[c]$ to 1. Otherwise, delete the column (keep $m[c]$ as 0) and update $b[i]$'s and $e[i]$'s if applicable. At the end of the iteration, each remaining column corresponds to an intersection point, and all the remaining columns form a minimal circular arc representation, by Theorem 8. Since we delete only columns that do not correspond to intersection points, the resulting circular arc representation is an equivalent minimal one.

We are now going to present an efficient implementation of the algorithm. For the convenience of the description, we assume, in the following, that the indices of the first row and the first column are both 0. We will also assume that the input matrix does not contain an all-1 row. From Theorem 8 we can easily show the following property: M corresponds to a minimal circular arc representation if and only if $[\,{}^{\mathbf{1}}_{M}\,]$ corresponds to a minimal circular arc representation, where **1** is an all-1 row. So the preceding assumption has no loss of generality.

```
0   for $i := 0$ to $s-1$ do {compute $b$ and $e$}
1     for $j := 0$ to $t-1$ do
2       if $M[i, j] = 1 \wedge M[i, (j+1) \bmod t] = 0$
        then $e[i] := j$ fi;
3       if $M[i, j] = 1 \wedge M[i, (j+t-1) \bmod t] = 0$
        then $b[i] := j$ fi;
4     od;
5   od;
6   for $i := 0$ to $t-1$ do $m[i] := 0$ od; {initialize $m$}
7   for $i := 0$ to $t-1$
8     find shortest arc, say $j$, whose clockwise endpoint
8     is $i$;
9     find shortest arc, say $k$, whose counterclockwise
      endpoint is $i$;
10    if both $j$ and $k$ exist and size of their intersection
      is 1 then $m[i] := 1$ fi;
11    if $m[i] = 0$ then {delete column $i$}
12      for $j := 0$ to $s-1$ do {update $b$ and $e$, if
        applicable}
13        if $b[j] = i$ then $b[j] := (i+1) \bmod s$ fi;
14        if $e[j] = i$ then $e[j] := (i-1+s) \bmod s$
          fi;
15      od;
16    fi;
17  od;
```

Computing the $b[i]$'s and $e[i]$'s (lines 0–5) is straightforward and takes $O(st)$ time. Lines 8–9 can be easily done in $O(t)$ time. If both j and k exist, then we have two arcs $[b[j], i]$ and $[i, e[k]]$. The size of the intersection between the two arcs is 1 if and only if $b[j] \leqslant i \leqslant e[k]$ or $i \leqslant e[k] < b[j]$ or $e[k] < b[j] \leqslant i$, which can be decided in constant time.

It is now easy to see that we can obtain an equivalent minimal circular arc representation in $O(st)$ time. We can now easily conclude the following.

THEOREM 15. *Given an $s \times t$ circular arc representation matrix, an equivalent minimal circular arc representation can be obtained in $O(st)$ time. The algorithm is optimal.*

Consider the following sample circular arc representation matrix:

$$M_9 = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}.$$

None of the columns correspond to any intersection point. When columns 0 and 1 have been deleted, column 2 corresponds to an intersection point. So column 2 is not removed, according to the above procedure. Then the rest of the columns are all deleted. So the equivalent minimal circular arc representation matrix is a column that consists of six 1's only.

The above procedure works in a sequential fashion and can not be parallelized directly. However, some observations will help us in obtaining a good parallel algorithm. It is not hard to see that deleting a column that does not correspond to an intersection point yields an equivalent circular arc representation. Moreover, being an intersection point is not affected by deleting some columns. Nevertheless, a column can become to correspond to an intersection point as a result of deleting another column, even though the two columns are not next to each other. If we delete a column, the two neighboring columns may change status and become to correspond to intersection points. Some other columns may change status, too. Suppose we have two arcs $[i, j]$ and $[j, i]$. If column i is deleted, then column j becomes to correspond to an intersection point and cannot be deleted. We say two arcs *embrace* if they intersect at both endpoints but neither is contained in the other. If we delete a column, the two neighboring columns and any embracing columns may change status and become to correspond to intersection points. It should be obvious that several columns can be deleted simultaneously if the deletion of one column does not affect the status of another column. So, if we can identify those columns efficiently, we can also obtain a more compact circular arc representation efficiently:

```
0   for $i := 0$ to $s-1$ codo {compute $b$ and $e$}
1     for $i := 0$ to $t-1$ codo
2       if $M[i, j] = 1 \wedge M[i, (j+1) \bmod t] = 0$
        then $e[i] := j$ fi;
3       if $M[i, j] = 1 \wedge M[i, (j+t-1) \bmod t] = 0$
        then $b[i] := j$ fi;
4     odoc;
5   odoc;
6   for $i := 0$ to $t-1$ codo {initialize $left$ and $right$}
7     $left[i] := (i+t-1) \bmod t$; {$left$ points to the left
      neighbor}
```

```
8     $right[i] := (i+1) \bmod t$; {$right$ points to the right neighbor}
9   odoc;
10  for $i := 0$ to $t-1$ codo $m[i] := 0$ odoc; {initialize $m$}
11  for $z := 0$ to $\lfloor \log_2(t-1) \rfloor$ do
12    for $i := 0$ to $t-1$ codo {initialize $lend$ and $rend$}
13      for $j := 0$ to $t-1$ codo
14        $lend[i, j] := 0$; $rend[i, j] := 0$;
15      odoc;
16    odoc;
17    for $i := 0$ to $s-1$ codo {compute $lend$ and $rend$}
18      $lend[b[i], e[i]] := 1$; {$lend[i, j] = 1$ means $[i, j]$ is an arc}
19      $rend[e[i], b[i]] := 1$; {$rend[i, j] = 1$ means $[j, i]$ is an arc}
20    odoc;
21    for $i := 0$ to $t-1$ codo if $i$ is an intersection point then $m[i] := 1$ fi odoc;
22    construct $G_z = (V_z, E_z)$, where $V_z = \{v_i \mid m[i] = 0 \wedge 0 < i < t \wedge i \bmod 2^z = 0 \wedge i \bmod 2^{z+1} \neq 0\}$
23      and $E_z = \{(v_i, v_j) \mid lend[i, j] = rend[i, j] = 1 \wedge v_i \in V_z \wedge v_j \in V_z\}$;
24    find a maximal independent set $V'_z$ of $G_z$;
25    $m[i] := 1$ for all $v_i \in V_z - V'_z$; {keep columns in $V_z - V'_z$}
26    for $i := 0$ to $s-1$ codo {update $b$ and $e$}
27      if $b[i] \bmod 2^z = 0 \wedge b[i] \bmod 2^{z+1} \neq 0 \wedge m[b[i]] = 0$ then $b[i] := right[b[i]]$ fi;
28      if $e[i] \bmod 2^z = 0 \wedge b[i] \bmod 2^{z+1} \neq 0 \wedge m[e[i]] = 0$ then $e[i] := left[e[i]]$ fi
29    odoc;
30    for each $v_i \in V'_z$ codo {update $left$ and $right$}
31      $right[left[i]] := right[i]$; $left[right[i]] := left[i]$;
32    odoc; {columns in $V'_z$ have been conceptually deleted}
33  od;
34  if column 0 now corresponds to an intersection point then $m[0] := 1$ fi;
35  remove column $i$ of $M$ if $m[i] = 0$, for $0 \leqslant i < t$.
```

Computing the clockwise endpoints $e[i]$'s and counterclockwise endpoints $b[i]$'s (lines 0–5) is straightforward. In order to achieve a polylogarithmic time bound, we must identify columns that can be deleted simultaneously. As observed earlier in this paper, deleting one column may make a neighboring column correspond to an intersection point. So in our algorithm, we only delete (conceptually) nonneighboring columns simultaneously. This is done by the for loop (lines 11–33). The loop iterates $\lfloor \log_2(t-1) \rfloor + 1$ times, and processes $G_0, G_1, ..., G_{\lfloor \log_2(t-1) \rfloor}$, successively. It is easy to see that these graphs have disjoint vertex sets and $\bigcup_{z=0}^{\lfloor \log_2(t-1) \rfloor} V_z \subseteq \{v_i \mid 0 < i < t\}$. Since each vertex corresponds to a column of M, we often find it convenient to use the two terms interchangeably. To show the correctness of the algorithm, we first give the following result.

LEMMA 4. *For each z, G_z does not contain neighboring columns.*

Proof. If G_z has at most one vertex, then it is trivially true that G_z does not contain neighboring columns. Suppose G_z has at least two vertices. Then it is necessary that $z < \lfloor \log_2(t-1) \rfloor$, since at most one vertex can be in $G_{\lfloor \log_2(t-1) \rfloor}$ by the construction of G_z (see line 22). Suppose columns i and j are two arbitrary columns in G_z and $i < j$. Then there exist two odd integers n_1 and n_2 such that $i = n_1 2^z$ and $j = n_2 2^z$. Let n_3 be an even number between n_1 and n_2. By definition, column $n_3 2^z$ is not in G_s, for $0 \leqslant s \leqslant z$. So when G_Z is constructed, column $n_3 2^z$ cannot have been deleted and lies to the right of column i and to the left of column j. Also note that column 0 lies on the other circular arc between columns i and j. Therefore, we can conclude that columns i and j are not neighbors. This completes the proof of the lemma. ▌

However, we may not delete several columns simultaneously even if no two of them are neighbors, since there may be some embracing arcs. To resolve this problem, we construct G_z (lines 22–23) as follows. Associate a column (and endpoint) with a vertex. If two arcs embrace and the size of their intersection is 2, then link the two vertices corresponding to the two endpoints. So, if two vertices are connected by an edge in the resulting graph, only one of the two columns can be deleted. If several vertices are mutually independent (i.e., no two of them are connected by an edge), then all of the columns can be deleted simultaneously. Therefore, we compute a maximal independent set of a graph (line 24). Columns corresponding to the maximal independent set can be deleted simultaneously. Because of the maximality, any vertex not in the independent set is connected to a vertex in the independent set. Consequently, the columns associated with the vertices outside the independent set will correspond to intersection points when the columns associated with the vertices inside the independent set have been deleted. At lines 25–32, we conceptually delete the columns associated with the vertices of G_z inside the independent set and keep the columns associated with the vertices of G_z outside the independent set (by setting m value to 1) and also update b, e, $left$, and $right$.

When the for loop (lines 11–33) terminates, all columns except column 0 have been considered. So we then check if column 0 can be deleted to obtain an equivalent circular arc representation. No columns are physically deleted until the end of the procedure. Since we make sure in the procedure that column deletion does not affect intersection relation and at the end of the procedure all columns correspond to intersection points, we can now conclude from Theorem 8 that the procedure correctly constructs an equivalent minimal circular arc representation.

If the procedure runs on M_9, lines 0–5 obtain the following result in the form of $i[b[i], e[i]]$: $0[5, 2]$, $1[1, 3]$, $2[2, 5]$, $3[1, 4]$, $4[0, 3]$, $5[0, 4]$. During the first iteration of the `for` loop (lines 11–33), lines 22–23 construct a graph G_0 with three vertices corresponding to columns 1, 3, and 5, respectively. The graph does not contain any edge. So all three columns are conceptually deleted. Lines 26–29 update b and e, and we obtain the following result: $0[0, 2]$, $1[2, 2]$, $2[2, 4]$, $3[2, 4]$, $4[0, 2]$, $5[0, 4]$.

G_1 is empty, so the second iteration of the `for` loop does nothing. G_2 constructed during the third iteration of the `for` loop has only one vertex corresponding to column 4. The column is conceptually deleted and the circular arcs after update at lines 26–29 will be: $0[0, 2]$, $1[2, 2]$, $2[2, 2]$, $3[2, 2]$, $4[0, 2]$, $5[0, 2]$. After three iterations of the `for` loop, the control goes to line 34 and checks if column 0 corresponds to an intersection point now. Since column 0 does not correspond to an intersection point, its m value remains 0. Finally, the procedure physically deletes all columns that have been conceptually deleted (i.e., columns whose m value is 0). The resulting matrix contains only column 2 with six 1's.

Before deriving some efficient resource bounds for constructing equivalent minimal circular arc representations, we will make some additional assumptions. The first assumption is that $s \leqslant t(t-1)/2 = O(t^2)$. If this is not true, the input matrix contains some identical rows. In this case, we can simply remove and make row duplicates at the beginning and at the end, respectively. We claim that this processing can be done within $O(\log(st))$ time and $O(st)$ work. Obviously, any row, say i, can be represented by a pair, $(b[i], e[i])$, where $0 \leqslant i < s$ and $0 \leqslant b[i], e[i] < t$. First we sort the rows using radix sort. Then we can identify row duplication in constant time with $O(s)$ EREW PRAM processors. We sort $e[i]$'s by actually sorting $e'[i]$'s, where $e'[i] = se[i] + i$. So each $e'[i]$ is distinct and is in the range $[0, st-1]$. Such numbers can be sorted in $O(\log(st))$ time with $O(st/\log(st))$ EREW PRAM processors [10]. Then $b[i]$'s can be sorted using the same method. Now, the validity of the claim follows immediately.

The next assumption is that $t \leqslant s = O(s)$. If this is not true, the input matrix contains at least one column that corresponds to at most one endpoint. In this case, we will remove some columns without affecting the circular arc representation so that in the resulting matrix each column corresponds to at least two endpoints (possibly for one arc). This can be done as follows (in a way similar to constructing an equivalent minimal interval representation discussed earlier in this section). Initialize the value of $m[i]$ as 0, for all i, $0 \leqslant i < t$. For each counterclockwise endpoint i, check if point i corresponds to at least two endpoints. If so, set $m[i]$ to 1. Otherwise, find the next clockwise endpoint, say j, in clockwise order, and set $m[j]$ to 1. Then, delete column i if $m[i] = 0$ for all i. All this can be done in $O(\log(st))$ time and $O(st)$ work on an EREW PRAM.

With the above assumptions, let us now consider the resource requirements of the procedure. It is easy to see that the most expensive part is the `for` loop (lines 11–33). Deciding if i is an intersection point (line 21) can be done by checking the size of the intersection between the shortest arc of type $[i, j]$ and the shortest arc of type $[k, i]$. Since we have s arcs $[b[i], e[i]]$, $0 \leqslant i < s$, and t columns, this step (line 21) takes $O(\log s)$ time and $O(st)$ work on an EREW PRAM. Constructing G_z (lines 22–23) is straightforward and can be easily done within the same resource bounds. One challenging process is to find a maximal independent set (line 24). Karp and Wigderson [26] discovered the first NC algorithm for the problem. It was later shown that the problem can be solved in $O(\log^3 v)$ time with $O((v+e)\log^2 v)$ work on an EREW PRAM [22], where v and e denote, respectively, the number of the vertices and the edges in a graph. In our case, $v \leqslant t/2 = O(t)$. Since each edge corresponds to two embracing arcs (rows), we have $e \leqslant s/2 = O(s)$. Therefore, line 24 takes $O(\log^3 t)$ time with $O((s+t)\log^2 t)$ work on an EREW PRAM. The total work of the loop body (lines 12–32) is $O(st + (s+t)\log^2 t) = O(st + t\log^2 t) = O(st)$ since $t \leqslant s$. The total time of the loop body is $O(\log s + \log^3 t)$. By Assumption 1, $\log s = O(\log t)$. So the loop body takes $O(\log^3 t)$ time. Since the loop iterates $O(\log t)$ times, the loop takes $O(\log^4 t)$ time and $O(st \log t)$ work.

Note that the preprocessing that makes the assumptions valid takes $O(\log(st))$ time and $O(st)$ work. It follows that computing an equivalent minimal circular arc representation matrix takes $O(\log s + \log^4 t)$ time and $O(st \log t)$ work on an EREW PRAM. Therefore, we have the following theorem.

Theorem 16. *Given an $s \times t$ circular arc representation matrix, an equivalent minimal circular arc representation can be obtained in $O(\log s + \log^4 t)$ time with $O(st \log t/(\log s + \log^4 t))$ processors on an EREW PRAM.*

The lower time bound we have obtained on the EREW PRAM is $\Omega(\log(st))$ and does not match the above upper bound. We conjecture that the algorithm is not time-optimal. The algorithm is work-optimal within a factor of $O(\log t)$.

If the input is a Θ circular arc representation matrix and we need to obtain an equivalent minimum Θ circular arc representation, then the problem can be solved faster as follows.

First, we eliminate all column duplication. Then, we check, for each column, if it is contained in another column. If so, delete it. According to the procedure, no column contains another in the resulting matrix. So the resulting matrix gives an equivalent minimum Θ circular arc representation by Theorem 9. Deciding whether column i contains column j

takes constant time with $O(s)$ common CRCW PRAM processors. There are $O(t^2)$ pairs of columns. So all the intersection points can be identified in constant time with $O(st^2)$ common CRCW PRAM processors. On the EREW PRAM, this can be done in $O(\log(st))$ time with $O(st^2)$ work. Once all the intersection points are identified, we can obtain the matrix corresponding to an equivalent minimum Θ circular arc representation by applying subarray computation on all the rows. Now, it is easy to conclude the following.

THEOREM 17. *Given an $s \times t$ Θ circular arc representation matrix, an equivalent minimum Θ circular arc representation can be obtained in $O(\log(st))$ time with $O(st^2/\log(st))$ EREW PRAM processors, or in $O(\log t/\log\log t)$ time with $O(st^2 \log\log t/\log t)$ common CRCW PRAM processors, or in $O(1)$ time with $O(st^2)$ BSR processors. All algorithms are time-optimal.*

5. DISCUSSION

In this paper, we have studied the properties of minimal interval and circular arc representations and have given some efficient sequential and parallel algorithms for the recognition and construction of such representations. All the recognition algorithms are optimal. The algorithms for constructing equivalent minimum interval representations are also optimal. However, for the problem of constructing equivalent minimal circular arcs, only the sequential algorithm is optimal; none of the parallel algorithms can be proved to be time-and-work-optimal. It is tempting to ask if we can obtain optimal parallel algorithms for this problem.

The models of parallel computation used in this paper include EREW PRAM, CRCW PRAM, and BSR. We have presented algorithms for each of these models and they are of independent interest. One might ask the following questions: Which model is the best? Is it sufficient to design algorithms on only one of the models? It is often debatable whether one model is better than another. There is no universal agreement on the answer. It seems premature to tell at this point. There is an interesting project on building PRAM-type computers (see, e.g., [1]). BSR is a relatively new model; the analogous project is not known currently but it is technically feasible (see, e.g., [28]). The power and the limitation of BSR are not as well understood and there is much room for further investigation. It is well known that the multiplication of two integers can be realized by a short program with repeated additions. However, multipliers are implemented in the VLSI chips of most of today's computers. We do not know for sure whether it is advantageous to include mechanism in parallel machines that supports broadcasting. As in the way of deciding Gordon Bell Prize winners, possibly a good way to compare between PRAM and BSR is to run sample programs on both types of machines, in which case efficient algorithms on both PRAM and BSR are needed. Perhaps someday, the algorithms in this paper will also be used for this purpose.

ACKNOWLEDGMENT

We thank an anonymous referee for bringing matrix M_8 to our attention.

REFERENCES

1. F. Abolhassan, R. Drefenstedt, J. Keller, W. J. Paul, and D. Scheerer, On the physical design of PRAMs, *Comput. J.* **36**(8) (1993), 756–762.
2. A. V. Aho, J. E. Hopcroft, and J. D. Ullman, "The Design and Analysis of Computer Algorithms," Addison–Wesley, Reading, MA, 1974.
3. S. G. Akl, "The Design and Analysis of Parallel Algorithms," Prentice–Hall, Englewood Cliffs, NJ, 1989.
4. S. G. Akl and L. Chen, Efficient parallel algorithms on proper circular arc graphs, *IEICE Trans. Inform. Systems* **E79-D**(8) (1996), 1015–1020.
5. S. G. Akl and G. R. Guenther, Broadcasting with selective reduction, *in* "Proceedings, 11th IFIP World Computer Congress" (G. X. Ritter, Ed.), pp. 515–520, North-Holland, Amsterdam, 1989.
6. P. W. Beame and J. Hastad, Optimal bounds for decision problems on the CRCW PRAMs, *J. Assoc. Comput. Mach.* **36**(3) (1989), 643–670.
7. S. Benzer, On the topology of the genetic fine structure, *Proc. Nat. Acad. Sci.* **45** (1959), 1607–1620.
8. R. P. Brent, The parallel evaluation of general arithmetic expressions, *J. Assoc. Comput. Mach.* **21** (1974), 201–208.
9. L. Chen, Efficient parallel algorithms for several intersection graphs, *in* "Proceedings, 22nd Int'l Symp. on Circuits and Systems," pp. 973–976, IEEE Press, New York, 1989.
10. L. Chen, Efficient deterministic parallel algorithms for integer sorting, *in* "Proc. International Conference on Computing and Information, Lecture Notes in Computer Science, Vol. 468," (S. G. Akl, F. Fiala, and W. W. Koczkodaj, Eds.), pp. 433–442, Springer-Verlag, New York/Berlin, 1990.
11. L. Chen, Optimal parallel time bounds for the maximum clique problem on intervals, *Inform. Process. Lett.* **42**(4) (1992), 197–201.
12. L. Chen, Efficient parallel recognition of some circular arc graphs, I, *Algorithmica* **9**(3) (1993), 217–238.
13. L. Chen, Revisiting circular arc graphs, *in* "Proceedings, 5th Annual International Symposium on Algorithms and Computation, Lecture Notes in Computer Science, Vol. 834," (D.-Z. Du and X.-S. Zhang, Eds.), pp. 559–566, Springer-Verlag, New York/Berlin, 1994.
14. L. Chen, Optimal circular arc representations, *in* "Proceedings, International Conference on Parallel Computing (EuroPar), Lecture Notes in Computer Science, Vol. 966" (S. Haridi, K. Ali, and P. Magnusson, Eds.), pp. 255–266, Springer-Verlag, New York/Berlin, 1995.
15. L. Chen, Efficient parallel recognition of some circular arc graphs, II, *Algorithmica* **17**(3) (1997), 266–280.
16. R. Cole and U. Vishkin, Faster optimal parallel prefix sums and list ranking, *Inform. Comput.* **81**(3) (1989), 334–352.
17. S. A. Cook, C. Dwork, and R. Reischuk, Upper and lower time bounds for parallel random access machines without simultaneous writes, *SIAM J. Comput.* **15**(1) (1986), 87–97.
18. F. E. Fich, P. L. Ragde, and A. Wigderson, Relations between concurrent-write models of parallel computation, *in* "Proc. 3rd ACM Symp. on Principles of Distributed Computing," pp. 179–189, Assoc. Comput. Mach., New York, 1984.

19. D. R. Fulkerson and O. A. Gross, Incidence matrices and interval graphs, *Pacific. J. Math.* **15** (1965), 835–855.
20. F. Gavril, Algorithms on circular-arc graphs, *Networks* **4** (1974), 357–369.
21. P. C. Gilmore and A. J. Hoffman, A characterization of comparability graphs and of interval graphs, *Canad. J. Math.* **16** (1964), 539–548.
22. M. Goldberg and T. Spencer, Constructing a maximal independent set in parallel, *SIAM J. Discrete Math.* **2**(3) (1989), 322–328.
23. M. C. Golumbic, "Algorithmic Graph Theory and Perfect Graphs," Computer Science and Applied Mathematics, Academic Press, New York, 1980.
24. U. I. Gupta, D. T. Lee, and J. Y.-T. Leung, Efficient algorithms for interval graphs and circular-arc graphs, *Networks* **12** (1982), 459–467.
25. J. L. Henessy and D. A. Patterson, "Computer Architecture: A Quantitative Approach," Morgan Kaufmann, San Mateo, CAN, 1990.
26. R. M. Karp and A. Wigderson, A fast parallel algorithm for the maximal independent set problem, *J. Assoc. Comput. Mach.* **32**(4) (1985), 762–773.
27. R. E. Ladner and M. J. Fischer, Parallel prefix computation, *J. Assoc. Comput. Mach.* **27**(4) (1980), 831–838.
28. L. F. Lindon and S. G. Akl, An optimal implementation of broadcasting with selective reduction, *IEEE Trans. Parallel Distrib. Systems* **4**(3) (1993), 256–269.
29. M. V. Marathe, H. B. Hunt, III, and S. S. Ravi, Efficient approximation algorithms for dogmatic partition and on-line coloring of circular arc graphs, *in* "Proceedings, 5th International Conference on Computing and Information" (O. Abou-Rabia, C. K. Chang, and W. W. Koczkodaj, Eds.), pp. 26–30, IEEE Press, New York, 1993.
30. A. C. Tucker, An efficient test for circular-arc graphs, *SIAM J. Comput.* **9** (1980), 1–24.