Efficient Parallel Algorithms on Circular Arcs

Lin Chen
Department of Computer and Information Science
Ohio State University
Columbus, OH 43210
USA

A circular arc graph is an intersection graph on a set of circular arcs on a circle. A Θ circular arc graph (a.k.a. Helly circular arc graph) is an intersection graph on a set of circular arcs with Helly property: for any subset of circular arcs, if any two circular arcs intersect, then the intersection of all the circular arcs in the subset is non-empty. In this paper, we give the first efficient parallel algorithm for deciding whether n circular arcs represent a Θ circular arc graph, and if so, constructing a set of circular arcs with Helly property. The algorithm runs in $O(\log^2 n)$ time with $O(n^3)$ processors on CRCW PRAM. At the same time, we also obtain an efficient parallel algorithm for computing all the maximal cliques including all the maximum cliques of a circular arc graph. We further give a new characterization of Θ circular arc graphs.

Let $\mu(V_1, \ldots, V_r)$ be the vertex versus vertex set incidence matrix, where each V_i is a vertex set. Let $C_1, \ldots, C_l$ be all the maximal cliques of G. For every vertex v_i, let G_i denote the subgraph induced by v_i and all its neighbors. Let $C_1^i, \ldots, C_{k_i}^i$ be all the maximal cliques of G_i. Denote the maximal elements of $\cup_{i=1}^n\{C_1^i, \ldots, C_{k_i}^i\}$ by $D_1, \ldots, D_k$. Gavril [6] characterized Θ circular arc graphs as follows.

Theorem 1 *G is a Θ circular arc graph if and only if $\mu(C_1, \ldots, C_l)$ has the circular 1's property.*

Based upon the characterization, Gavril gave an $O(n^3)$ sequential algorithm for recognizing Θ circular arc graphs using dynamic programming approach. That algorithm does not seem to be parallelizable. Using a different method in this paper, we obtain a parallel algorithm which runs in $O(\log^2 n)$ time with $O(n^3)$ processors. We first give the following equation.

Lemma 1 $\{D_i \mid 0 < i \leq k\} = \{C_j \mid 0 < j \leq l\}$.

Proof. Suppose C_j contains vertex p. Then C_j is C_q^p for a q. Since C_j is a maximal clique of G, C_q^p must be D_i for an i. Hence $\{D_i \mid 0 < i \leq k\} \supseteq \{C_j \mid 0 < j \leq l\}$.

Assume there exists a D_i such that $D_i \notin \{C_j \mid 0 < j \leq l\}$. Then there must be a C_j such that $D_i \subset C_j$. As we have shown above, $C_j = D_r$ for an r. This contradicts the fact that

$\{D_i \mid 0 < i \leq k\}$ is the maximal elements of $\cup_{i=1}^{n}\{C_1^i, \ldots, C_{k_i}^i\}$. Hence $\{D_i \mid 0 < i \leq k\} \subseteq \{C_j \mid 0 < j \leq l\}$. □

The following result follows immediately from Theorem 1 and Lemma 1.

Theorem 2 *G is a Θ circular arc graph if and only if $\mu(D_1, \ldots, D_k)$ has the circular 1's property.*

However, Gavril [6] characterized Δ circular arc graphs in the following way: G is a Δ circular arc graph if and only if $\mu(D_1, \ldots, D_k)$ has the circular 1's property [sic]. A Δ circular arc graph is an intersection graph on a set of circular arcs such that for any three circular arcs, if any two of them intersect, then the intersection of the three circular arcs is not empty. Gavril also gave an example showing correctly that the class of Δ circular arc graphs is not equivalent to the class of Θ circular arc graphs. So there must be some errors somewhere. We have noticed that the error was caused by Gavril's incorrect claim that each G_i is an interval graph.

Before we give an NC algorithm for recognizing Θ circular arc graphs, we first describe how to compute the maximal cliques of a circular arc graph.

Given a circular arc graph G and a circular arc representation S, we denote the endpoints of the arcs consecutively in the clockwise direction by $h_1, h_2, \ldots, h_{2n}, h_{2n+1} = h_1$. Let M_i be the set of arcs in S that contain endpoint h_i. Let M_i' be the set of arcs in S but not in M_i that intersect all arcs in M_i. Since h_i is not contained in any arcs in M_i', the arcs in M_i' do not cover the entire circle. So M_i' is an intersection representation for an interval graph. It is not difficult to see that a maximal clique of M_i' together with M_i forms a maximal clique containing h_i for the circular arc graph G.

The endpoints of the arcs can be sorted in $O(\log n)$ time with $O(n)$ processors on EREW PRAM, or in $O(\log n/\log\log 2p/n)$ time with $2n \leq p \leq n^2$ processors on CRCW PRAM using an algorithm by Cole [4]. If the range of endpoints is linear in n—in fact, a circular arc graph can always be represented by n circular arcs whose endpoints are in a range linear in n—Chen [2] has shown that sorting can be done in $O(\log n)$ time with $O(n/\log n)$ processors on EREW PRAM, or in $O(\log n/\log\log n)$ time with $O(n\log\log n/\log n)$ processors on CRCW PRAM. If the integers are not distinct but at most f_1 number of integers have multiplicity greater than f_2, then sorting can be done in $O((f_2 + g)\frac{\log n}{\log\log n})$ time with $O(\frac{f_1 n \log\log n}{g \log n})$ processors for any g satisfying $0 < g \leq f_1$. When both bounds f_1 and f_2 are constants, the algorithm is optimal.

All the maximal cliques of an interval graph can be obtained by applying prefix computation. Suppose $e_1, \ldots, e_{2n}$ are the endpoints of n intervals representing an interval graph. Define a new array $f[1:2n]$ in the following way:

$$f[i] = \begin{cases} 1 & \text{if } e_i \text{ is the left endpoint of an interval} \\ -1 & \text{otherwise} \end{cases}$$

Apply prefix sum computation on the array $f[1:2n]$ and store the result in the array $s[1:2n]$. Note that it is always true that $s[1] = 1$ and $s[2n] = 0$. Then the vertices corresponding to intervals containing endpoint i form a maximal clique if $s[i] > s[i-1]$ and $s[i] > s[i+1]$, for $0 < i < 2n$, assuming $s[0] = 0$. The largest $s[i]$ corresponds to a maximum clique of the interval graph. The maximum of $s[i]$'s can be obtained via prefix computation choosing max as the binary operator. Prefix sum can be computed in $O(\log n)$ time with $O(n/\log n)$ processors on EREW PRAM [9] [8], or in $O(\log n/\log\log n)$ time with $O(n\log\log n/\log n)$ processors on Common CRCW PRAM [5]. It is easy to see that the number of maximal cliques containing endpoint h_i is bounded by $O(n)$.

Incidentally, we can compute a maximum clique from the maximal cliques of the circular arc graphs. Total work is bounded by $O(n^3)$. The time complexity is dominated by the prefix computation and sorting. We present the result in the following theorem.

Theorem 3 *A maximum clique of a circular arc graph can be computed in $O(\log n)$ time with $O(n^3/\log n)$ processors on CREW PRAM, or in $O(\frac{\log n}{\log\log n})$ time with $O(\frac{n^3\log\log n}{\log n})$ processors on CRCW PRAM.*

Though the computation of a maximum clique is not required for the recognition of Θ circular arc graphs, the problem is interesting in its own right. Previously, one algorithm for computing a maximum clique was announced in Chen [1]. The algorithm runs in $O(\log n)$ time with $O(n^3)$ processors on CRCW PRAM. Kim [7] independently obtained an algorithm which runs in $O(\log^2 n)$ time using $O(n^3/\log n)$ CREW PRAM processors. So our new algorithm is not only faster, but also more efficient in terms of processor-time product.

We now return to the problem of recognizing Θ circular arc graphs. After we have computed the maximal cliques containing h_i, for all $0 < i \leq 2n$, we have at most $O(n^2)$ maximal cliques. Each maximal clique can be represented by an n-tuple consisting only of 0's and 1's. Now we want to delete the duplications. First sort these n-tuples according to the lexicographic order. The comparison of two n-tuples a and b can be parallelized as follows. Let c be a new tuple such that $c[i] = a[i] - b[i]$, for $0 < i \leq n$. If all the elements of c are 0, then $a = b$. Otherwise, obtain the location, say k, of the first non-zero element in c. If $c[k] < 0$, then $a < b$. The location of the first non-zero element in c can be computed by applying prefix sum computation on an n-tuple d, where $d[i]$ is the absolute value of the corresponding element of $c[i]$, for $0 < i \leq n$. Then k is the integer l satisfying $\sum_{i=1}^{l} d[i] = 1$. So the comparison of two n-tuples can be done in $O(\log n)$ time with $O(n/\log n)$ processors on EREW PRAM, or in $O(\log n/\log\log n)$ time with $O(n\log\log n/\log n)$ processors on CRCW PRAM. As is known, sorting n elements by comparison can be done in $O(\log n)$ time with $O(n)$ processors on EREW PRAM assuming each comparison can be done in unit sequential time. So sorting all those maximal cliques can be done in $O(\log^2 n)$ time with $O(n^3/\log n)$ processors on EREW PRAM, or in $O(\log^2 n/\log\log n)$ time with $O(n^3\log\log n/\log n)$ processors. Suppose the maximal cliques after sorting is $C_1, \ldots, C_s$. Compare C_i with C_{i+1} and delete the duplication. Then we get l distinct maximal cliques, say, $C_1, \ldots, C_l$. Gavril has correctly shown that the number of distinct maximal cliques in a Θ circular arc graph is at most n. Consequently, if $l > n$, the input graph is not a Θ circular arc graph. Otherwise, we decide if the graph is a Θ circular arc graph by testing the circular 1's property for $\mu(C_1, \ldots, C_l)$. Chen and Yesha [3] have shown that, deciding if a matrix has the circular 1's property for rows, and if so, transforming the matrix into one with circular 1's in each of its rows, can be done in $O(\log^2 n)$ time with $O(n^3)$ processors. If the graph is a Θ circular arc graph, then an intersection representation with the Helly property can be constructed within the same resource bounds. Assume the 1's in each row of $\mu(C_1, \ldots, C_l)$ occur in a circularly consecutive fashion. For row i, construct an arc $[m, p]$ if the 1's start from column m and end at column p. We claim that the arcs constructed have the Helly property. Suppose several arcs mutually intersect. Then the corresponding vertices must be in a maximal clique, say, C_s. According to the construction of the arcs, those arcs all contain $[s, s]$. So the intersection of those arcs is not empty. Therefore, the constructed arcs have the Helly property. It is easy to see that the complexity bounds are dominated by the recognition of the circular 1's property. We summarize the result as the following theorem.

Theorem 4 *Deciding if a graph is a Θ circular arc graph, and if so, constructing an intersection representation with the Helly property, can be done in $O(\log^2 n)$ time with $O(n^3)$ processors.*

References

[1] L. Chen. NC algorithms for circular-arc graphs. In F. Dehne, J.-R. Sack, and N. Santoro, editors, *Proc. Workshop on Algorithms and Data Structures, Lecture Notes in Computer Science, Vol. 382*, pages 291–302. Springer-Verlag, 1989.

[2] L. Chen. Efficient deterministic parallel algorithms for integer sorting. In S. G. Akl, F. Fiala, and W. W. Koczkodaj, editors, *Advances in Computing and Information, Proc. International Conference on Computing and Information*, pages 367–371, 1990.

[3] L. Chen and Y. Yesha. Parallel testing for the consecutive ones property and transformable convex bipartite graphs and finding a maximum matching. In *Proc. 27th Ann. Allerton Conf. on Communication, Control, and Computing*, pages 756–765. University of Illinois at Urbana-Champaign, 1989.

[4] R. Cole. Parallel merge sort. *SIAM Journal on Computing*, 17(4):770–785, August 1988.

[5] R. Cole and U. Vishkin. Faster optimal parallel prefix sums and list ranking. *Information and Computation*, 81(3):334–352, June 1989.

[6] F. Gavril. Algorithms on circular-arc graphs. *Networks*, 4:357–369, 1974.

[7] S. K. Kim. *A Parallel Algorithm for Finding a Maximum Clique of a Set of Circular-arcs of a Circle.* University of Washington, 1989.

[8] C. P. Kruskal, L. Rudolph, and M. Snir. The power of parallel prefix. *IEEE Transactions on Computers*, C-34:965–968, 1985.

[9] R. E. Ladner and M. J. Fischer. Parallel prefix computation. *Journal of the ACM*, 27:831–838, 1980.