OPTIMAL PARALLEL ALGORITHMS ON CIRCULAR-ARC GRAPHS

A.Srinivasa Rao and C.Pandu Rangan
Department of Computer Science & Engineering
I.I.T. Madras 600 036 India.

1. INTRODUCTION

Recently, a growing interest has been shown in designing parallel algorithms on restricted classes of graphs [10,11]. We present in this paper some optimal parallel algorithms on circular-arc graphs. Specifically, we present optimal parallel algorithms for finding (unweighted) maximum independent set, minimum clique cover and minimum dominating set in a unified way. Given the sorted array of end points of the arcs in the intersection model of the circular-arc graph, all the above mentioned problems can be solved sequentially in \(O(n)\) time where \(n\) is the number of arcs [9]. All our parallel algorithms run in \(O(\log n)\) time on an \(O(n/\log n)\)-processor EREW PRAM and hence they are optimal.

All the algorithms that we present will have the following form:

Stage 1: (Pruning) Delete a suitable subset of vertices of the given circular-arc graph \(G\) and obtain a graph \(G'\).

Stage 2: (Find Successor) Define in a suitable manner a function on the vertex set of \(G'\) called NEXT.

Stage 3: (Greedy Step) Use NEXT function greedily and solve the problem on \(G'\).

Stage 4: (Extension Step) Extend the solution of \(G'\) to \(G\) (if necessary).

2. PRELIMINARIES

All the graphs we consider in this paper are finite graphs with no multiple edges and self-loops.

Let \(G = (V,E)\) be a graph. A set of vertices \(S \subseteq V\) forms a clique in \(G\) if every pair of vertices in \(S\) are adjacent. A clique cover of \(G\) is a partition of vertex set \(V\) into \(V_1, \ldots, V_k\), such that each \(V_i, 1 \leq i \leq k\), forms a clique in \(G\); \(k\) is the size of the clique cover. A clique cover with minimum size is called a minimum clique cover (MCC).

A set of vertices \(S \subseteq V\) forms an independent set in \(G\) if no two vertices of \(S\) are adjacent. An independent set with maximum
cardinality is called a maximum independent set (MIS).

A set $S \subseteq V$ of vertices dominates a set $S' \subseteq V$ if every vertex in $S' - S$ is adjacent to some vertex in $S$. If $S$ dominates $V$, we say that $S$ is a dominating set for the graph $G$. A dominating set with minimum cardinality is called a minimum dominating set (MDS).

The MCC, MIS, and MDS problems for an arbitrary graph are known to be NP-complete [7].

A graph $G = (V,E)$ is a circular-arc graph if there exists a one-to-one correspondence of the vertex set $V$ with a family $AF$ of (closed circular) arcs on the unit circle such that two vertices are adjacent in the graph iff their corresponding arcs in the family intersect. The family $AF$ is called an intersection model of the graph $G$. An example is given in Fig.1. The family $AF$ is proper if no arc of $AF$ is contained in some other arc of $AF$.

Circular-arc graphs arise in various applications such as traffic-control and information retrieval and they are well studied in literature [2,5,6,8,13,14,15]. From now on, let $AF = \{X_1, X_2, \ldots, X_n\}$ be a family of arcs on a unit circle and $G = (V,E)$, $|V| = n$, be the circular-arc graph with $AF$ as its intersection model. The arc $X_i$ is represented by the ordered pair $(l(X_i), r(X_i))$, where $l(X_i)$ and $r(X_i)$ denote its left and right end points respectively. The arc $X_i$ exists on the circle as a traversal in the clockwise direction from $l(X_i)$ to $r(X_i)$ along the circumference of the circle.

Instead of presenting the algorithms on the graph $G$, we will be working with the intersection model $AF$ itself. For example, we will compute a maximum independent set of arcs (two arcs being independent if they do not intersect) to solve the MIS problem.

Let $X_i$ and $X_j$ be two arcs such that the arc $X_j$ is totally contained in the arc $X_i$. Then $X_i$ is called a containing arc and arc $X_j$ is called a contained arc.

Let $PL$ be the sorted array of $2n$ distinct end points of arcs in $AF$. These points appear in $PL$ in the same order as they are encountered in the clockwise traversal of the circle beginning at $l(X_1)$. We note that if the end points are not sorted, then we can use the Parallel Merge Sort [3] to sort them in $O(\log n)$ time using $O(n)$ processors.

Let $L$ be a list of elements $a_1, a_2, \ldots, a_k$, $k > 0$, not necessarily in that order. Then, the position of an element $a_i$ in the list $L$ is the number of elements that precede it in $L$ plus one.

For a nonempty set $S$ of integers, let $\text{MAX}(S)$ and $\text{MIN}(S)$ denote a maximum element and a minimum element of $S$ respectively.