NC Algorithms for Recognizing Chordal Graphs and k Trees

N. CHANDRASEKHARAN AND S. SITHARAMA IYENGAR

Abstract—**We present parallel algorithms for recognizing the chordal graphs and k trees. Under the model of computation PRAM with concurrent reading and writing allowed, these algorithms take $O(\log n)$ time and require $O(n^4)$ processors. Our algorithms have an improved processor-bound than an independent result by Edenbrandt for recognizing chordal graphs in parallel using $O(n^3m)$ processors. Furthermore, our characterizations for chordal graphs and k trees are interesting in their own right.**

Index Terms—**Chordal graphs and k trees, NC algorithm, parallel algorithm.**

I. Introduction

With the increasing use of highly parallel computers, it has become necessary to identify various computational problems that can be solved fast in parallel. In particular, there has been a considerable interest in problems which have parallel algorithms running in time bounded by a polynomial in the logarithm of the size of the input (i.e., in poly-log time) and using a number of processors polynomial in the input size. Such a class of problems is referred to as the class NC (Pippenger [18]). Some of the reasons for this interest in the class NC are the following.

1) The "speedup" of the NC algorithms is exponential compared to their sequential versions. Furthermore, they make use of a "reasonable" amount of hardware, namely, the processors.

2) The class NC is "robust" under reasonable changes in the underlying machine models of parallel computation.

The model of computation used here is the parallel random access machine or PRAM in which all the processors have access to a common memory and run synchronously (see Vishkin [28]). Both simultaneous reading as well as writing on common memory locations are allowed. Furthermore, in the case of simultaneous writing, an arbitrary processor succeeds in writing.

A class of graphs called the perfect graphs has been the focus of extensive study of late (Golumbic [13]). Perfect graphs are interesting from both the algorithmic and combinatorial points of view. Important among the perfect graphs are the chordal graphs, comparability graphs, interval graphs, k trees, permutation graphs, cographs, etc. [13]. All these graphs arise in various real-life applications and have polynomial-time algorithms for recognition (Corneil [6]). Recently, NC algorithms have been given for recognizing comparability graphs, permutation graphs, and interval graphs (Kozen *et al.* [16], Helmbold and Mayr [15]). For a broader treatment on parallel algorithms, see [31].

In this paper, we present NC algorithms for recognizing chordal graphs and k trees. These algorithms use $O(n^4)$ processors and take $O(\log n)$ time on the PRAM model of computation we have discussed earlier. We just recently came to know of an independent result by Edenbrandt [9] who gives an NC algorithm for recognizing chordal graphs. Although the spirit of our recognition algorithms for the chordal graphs is the same, there are, however, some technical differences. In particular, Edenbrandt's algorithm requires $O(n^3m)$ processors and $O(\log n)$ time. Since $O(n^3m)$ processors in the worst case would be $O(n^5)$ processors, our algorithm has an improved processor-bound. Furthermore, we feel that the characterizations we have obtained for chordal graphs and k trees are of independent interest.

II. Graph-Theoretic Terminology

A graph $G = (V, E)$, where V is the vertex set and E is the edge set, is considered here as being finite, undirected, connected, and without multiple edges or self-loops. For our purposes, the terms *subgraph* and *induced subgraph* mean the same as the terms *clique* and *complete graph*. Furthermore, we often use the graph for its vertex set. For most of our presentation, we follow the standard terminology of Golumbic [13], unless otherwise indicated.

For a pair of nonadjacent vertices u and v of a graph $G = (V, E)$ a *uv separator* is a set $S \subset V$ of vertices such that u and v are in distinct connected components of the graph induced by $V - S$. A uv separator S is *minimal* if no proper subset of S is a uv separator. If the uv separator is a clique, then we call it a *uv clique separator*. A graph G is *chordal* if every simple cycle of length greater than three has a chord. A *perfect elimination ordering* (PEO) is an ordering, say, $[x_{i_1}, \cdots, x_{i_n}]$ of the vertices of G such that $\mathrm{ADJ}(x_{i_j}) \cap \{x_{i_{j+1}}, \cdots, x_{i_n}\}$ induces a clique, for $j = 1, 2, \cdots, n$. Here, $\mathrm{ADJ}(x) = \{y : \{x, y\} \in E\}$. Furthermore, we define, for $S \subseteq V$, $\mathrm{ADJ}(S) = \cup_{x \in S} \mathrm{ADJ}(x)$. A vertex x is said to be *simplicial* if $\mathrm{ADJ}(x)$ induces a clique in G. Chordal graphs arise in various applications, the most important being the solution of sparse systems of linear equations (Rose [22], Rose *et al.* [24]). Other applications of chordal graphs occur in the study of evolutionary trees (Buneman [3]), in facility location (Chandrasekharan and Tamir [5]), and in relational database theory

Manuscript received September 24, 1986; revised May 11, 1987.

N. Chandrasekharan is with the Department of Mathematical Sciences, Clemson University, Clemson, SC 29634.

S. S. Iyengar is with the Department of Computer Science, Louisiana State University, Baton Rouge, LA 70803.

IEEE Log Number 8719408.

(Beeri *et al.* [1]). The following characterizations are well known for chordal graphs.

Theorem 2.1: The following are equivalent.

1) G is chordal.
2) (Fulkerson and Gross [11]) G has a perfect elimination ordering.
3) G can be obtained by the following recursive construction rules:
   a) Start with any clique, as the basis graph. A clique is a chordal graph.
   b) To a chordal graph H, add a new vertex and make it adjacent to some clique subgraph of H (see Proskurowski [19]).
4) (Dirac [7]) Every minimal separator of G induces a clique in G.
5) (Gavril [12]) G is the intersection graph of the subtrees of a tree.
6) Every subgraph of G is either a clique or contains two nonadjacent simplicial vertices (see Duchet [8]).
7) Every connected subgraph with $n \geq 2$ vertices contains at most $n - 1$ cliques (see [8]). □

The sequential algorithms for testing chordality make use of various procedures called the PEO *schemes* which generate a PEO if it exists. The various PEO schemes known in the literature are the following:

1) ([24]) Lexicographic Breadth First Search (LBFS),
2) (Tarjan and Yannakkakis [27]) Maximum Cardinality Search (MCS),
3) (Shier [25]) Maximal Element in Component (MEC), and
4) ([25]) Maximum Cardinality neighborhood in Component (MCC).

It is not relevant for our purposes to go into the detail of the above schemes. But the following result establishes the importance of the PEO schemes in the recognition of chordal graphs.

Theorem 2.2. [24], [25], [27]: A graph $G = (V, E)$ is chordal if and only if an ordering of the vertices of G, generated by any of the PEO schemes LBFS, MCS, MEC, and MCC, is a PEO. Furthermore, G can be so tested for chordality in $O(n + m)$ time. □

The k trees, introduced by Harary and Palmer [14], are an important subclass of chordal graphs. They have been characterized by Beineke and Pippert [2], and Rose [23] subsequently in many interesting ways. Furthermore, the k trees have been extensively investigated in [19]–[21] and Foata [10]. We give below the definition and various characterizations of k trees.

Definition 2.3: A graph is a k tree if it can be obtained by the following recursive construction rules.

a) Start with any k clique as the basis graph. A k clique is a k tree.
b) To any k tree H add a new vertex and make it adjacent to a k clique subgraph of H, to form a $(k + 1)$ clique. □

Theorem 2.4: The following are equivalent.

1) $G = (V, E)$ is a k tree.
2)
   a) G is connected,
   b) G has a k clique but no $(k + 2)$ clique, and
   c) every minimal vertex separator of G is a k clique [23].
3)
   a) G is chordal,
   b) $|E| = k|V| - k(k + 1)/2$, and
   c) G has a k clique but no $(k + 2)$ clique [23].
4)
   a) G is connected,
   b) $|E| = k|V| - k(k + 1)/2$, and
   c) every minimal vertex separator of G is a k clique [23].
5) G is chordal and the set of PEO's generated by the PEO scheme MCS is equal to that generated by the scheme MCC for G [4].
6) G is chordal and the set of PEO's generated by the scheme MEC is equal to that generated by the scheme MEI for G [4]. □

Note that none of the above characterizations for both the chordal graphs and k trees seems to lead to NC algorithms for recognition. In particular, it is not clear whether a PEO can be generated by parallelizing any of the PEO schemes into an NC algorithm. Recently, the notion of a *maximal clique separator* was explored [30]. In Sections III and IV, we introduce some simple characterizations of chordal graphs and k trees and obtain the NC algorithms for their recognition.

III. NC Algorithm for Recognizing Chordal Graphs

We note that the characterization of chordal graphs in Theorem 2.1. 4) in terms of the minimal separators is a little stronger than necessary. We weaken this result to obtain the following lemma.

Lemma 3.1: A graph $G = (V, E)$ is chordal if and only if for every nonadjacent pair of vertices u and v of V, there exists a uv clique separator in G.

Proof:

(ONLY IF): If G is chordal, then for every pair of nonadjacent vertices u and v of V there exists a minimal uv separator which is a clique [by Theorem 2.1.4)].

(IF): By contradiction. Assume that there exists a chordless simple cycle C of length $l \geq 4$ in G. Take any two nonadjacent vertices u and v in C. Let the cycle C be $u = u_1u_2 \cdots u_i\, v = v_1v_2 \cdots v_ju$, where $i + j = l$. Furthermore, $i, j \geq 2$. Any vertex separator for u and v should include some u_p and v_q, $p \in \{2, \cdots, i\}$, $q \in \{2, \cdots, j\}$. Otherwise, u and v would still be connected. But since C is a chordless cycle, u_p and v_q are nonadjacent. This contradicts the assumption that there exists a clique separator for every pair of nonadjacent vertices in G. □

Now instead of looking for minimal separators in a graph G, it is enough to look for a clique separator. We describe a sequential algorithm for testing chordality based on the above lemma.

Algorithm ChordalTest [Graph: $G = (V, E)$];

```
begin
    for every Nonadjacent unordered pair of vertices u and v of V do
        G_uv  := G − ADJ(u);
        G^v_uv := component of G_uv containing v;
        M_uv  := {x:x ∈ ADJ(u) and x is adjacent to some vertex in G^v_uv};
        if M_uv induces a clique in G then continue
        else
        begin
            print "G is not chordal";
            terminate;
        end;
    rof;
    print "G is chordal";
end.
```

□

We note here that a variant of the body of the above algorithm is used by Whitesides [29] as the initialization step of her algorithm for finding a clique separator of a graph. The correctness of the algorithm *ChordalTest* is established below.

Lemma 3.2: In the algorithm *ChordalTest*, M_{uv} is a uv seperator in G. Furthermore, if M_{uv} has two nonadjacent vertices (for some u, v in G), then there is a chordless cycle in G having a length of at least four.

Proof: It is clear that $(G - M_{uv}) - \{u\} \supseteq G^v_{uv}$. Any x in $(G - M_{uv}) - \{u\}$ and not in G^v_{uv} is not connected to the vertices in G^v_{uv}. So M_{uv} seperates u and all the vertices of G^v_{uv}. In particular, M_{uv} is a uv seperator in G. Let x and y be two nonadjacent vertices in M_{uv}, which are adjacent to vertices, say r and s, respectively, in G^v_{uv}. Let $ra_1a_2a_3 \cdots a_ps$ be a shortest chordless path in G^v_{uv}. Then it is easy to see that $uxra_1a_2a_3 \cdots a_psyu$ is a chordless path of length at least four in G. □

Lemma 3.3: $G = (V, E)$ is chordal if and only if for every nonadjacent pair of vertices u and v of V, M_{uv} (in the algorithm *ChordalTest*) induces a clique in G.

Proof:

(ONLY IF): Let G be chordal. Assume G is not a clique. Suppose M_{uv} does not induce a clique for some pair of nonadjacent vertices u and v in G. This means that there exists at least two nonadjacent vertices x and y in M_{uv}. Then by Lemma 3.2 there is a chordless cycle of length at least four in G, contradicting the assumption that G is chordal.

(IF): Follows from Lemma 3.1. □

The following result is immediate.

Theorem 3.4: The algorithm *ChordalTest* correctly tests whether G is chordal or not. □

Now we parallelize the above sequential algorithm for testing chordality.

Algorithm NCTestChordal:

```
begin
    if G is a clique then "G is chordal" and terminate;
      for every nonadjacent pair of vertices u, v do in parallel
1.        H := G − ADJ(u);
2.        F := Component of H containing the vertex v;
          M := ∅;
3.          for each x ∈ ADJ(u) and y ∈ F do in parallel
                if {x, y} ∈ E then M := M ∪ {x};
            rof;
4.        if M is a clique then return 1 else return 0;
      rof;
    if for each pair of nonadjacent vertices u, v, the answer
    returned is 1 then "G is chordal" else
    "G is not chordal";
end.
```

The following theorem establishes the correctness of the algorithm *NCTestChordal* and its complexity.

Theorem 3.5: The algorithm *NCTestChordal* correctly tests in parallel the chordality of G. Furthermore, the algorithm has a time-complexity $O(\log n)$ and makes use of $O(n^4 + n^3 + n^2m)$ processors.

Proof: The proof of correctness of the parallel algorithm follows from the proof of its sequential version (Theorem 3.4). To find out the number of processors required, we note that the number of nonadjacent pair of vertices is $O((n/2))$. For finding the connected component F, the parallel algorithm of Shiloach and Vizhkin [26] can be made use of. This algorithm takes $O(\log n)$ time and requires $O(m + n)$ processors. Furthermore, step 3 requires $O(n^2)$ processors. So in all, the algorithm requires $O(n^2(n + m) + n^4)$ processors and $O(\log n)$ time. □

IV. NC Algorithm for Recognizing k Trees

The NC algorithm for recognizing k trees is similar to the one for recognizing chordal graphs but different in the sense that it makes use of minimal separators. We need the following preliminary results.

Lemma 4.1: Let G be a noncomplete chordal graph. Let x be a simplicial vertex in G and $H = (U, F) = G - \{x\}$. Then for any nonadjacent pair of vertices u and v of H, no minimal uv separator contains x.

Proof: If H is a clique, then the lemma is vacuously true. Assume H is not a clique. Let u and v be a pair of nonadjacent vertices. Both u and v cannot be adjacent to x, otherwise u and v would themselves have to be adjacent to each other. Therefore, at most one of the vertices u and v is adjacent to x. Two cases arise.

Case 1: Without loss of generality, let $u \in \mathrm{ADJ}(x)$.

Then $v \notin \mathrm{ADJ}(x)$. Let S be a minimal uv separator contained in $\{x\} \cup \mathrm{ADJ}(x) - \{u\}$. Let C_1 and C_2 be the connected components of $G - S$, containing the vertices u and v, respectively. Then $C_1 \cap C_2 = \emptyset$. Also no vertex in C_2 is adjacent to x, because x is a simplicial vertex. Furthermore, no vertex in C_2 is adjacent to u. This is because if there is a vertex z in C_2 adjacent to u, then u and v will be connected by a path in $G - S$, unless z is adjacent to x also. But we know that no vertex in C_2 is adjacent to x. So all the vertices in C_2 are adjacent to a subset of $\mathrm{ADJ}(X) - \{u\}$. Therefore, u and v can be separated by removing $R = (\mathrm{ADJ}(x) - \{u\}) \cap \mathrm{ADJ}(C_2)$. Furthermore, it is easy to see that R is also the only minimal uv separator contained in $\mathrm{ADJ}(x) - \{u\} \cup \{x\}$. If any minimal uv separator S is to contain X, then it should be a subset of $\mathrm{ADJ}(x) \cup \{x\} - \{u\}$. But we have seen that the only minimal uv separator in $\mathrm{ADJ}(x) \cup \{x\} - \{u\}$ is R which does not contain x. Therefore, the lemma holds.

Case 2: Both $u, v \notin \mathrm{ADJ}(x)$.

Let S be a minimal uv separator contained in $\{x\} \cup \mathrm{ADJ}(x)$. Let C_1 and C_2 be the connected components of $G - S$, containing the vertices u and v, respectively. Again, no vertex in either C_1 or C_2 is adjacent to x. So the only minimal uv separator contained in $\mathrm{ADJ}(x) \cup \{x\}$ are $\mathrm{ADJ}(x) \cap \mathrm{ADJ}(C_1)$ and $\mathrm{ADJ}(x) \cap \mathrm{ADJ}(C_2)$ neither of which contains the vertex x. As in the previous case, we can see that the lemma holds. □

We give below a new characterization of k trees.

Theorem 4.2: A graph $G = (V, E)$ is a k tree if and only if

a) For every nonadjacent pair of vertices u and v of G, there exists a minimal uv separator which is a k clique, and

b) $m = kn - k(k + 1)/2$.

Proof:

(ONLY IF): If G is a clique then it is a k tree, trivially. If G is not a clique, then for every pair of nonadjacent vertices u and v, the minimal separator of G is a k clique (by Theorem 2.4 4)c). Furthermore, $m = kn - k(k + 1)/2$ (by Theorem 2.4 4)b).

(IF): The condition a) alone implies that G is chordal (by Lemma 3.1). Therefore, conditions a) and b) together fulfill the conditions of Theorem 2.4 3)a) and 2.4 3)b). It is enough to show that G has a k clique but no $(k + 2)$ clique. This we show below.

It is easy to see that if G is a clique, it cannot be a $(k + 2)$ clique. Therefore, assume that G is not a clique. Then G has a k clique because it will have a minimal separator. We show by induction on the number of vertices that G does not have a $(k + 2)$ clique. All graphs satisfying conditions a) and b) of Theorem 3 and having 1, 2, 3, and 4 vertices do not have $(k + 2)$ cliques. Assume that all graphs having fewer than or equal to $(n - 1)$ vertices and satisfying conditions a) and b) do not have $(k + 2)$ cliques. Consider a graph G on n vertices and satisfying conditions a) and b). Because G is chordal, there exists a simplicial vertex x in G. Let $H = (U, F) = G - \{x\}$ and $W = \{x\} \cup \mathrm{ADJ}(x)$. Now $|U| = n - 1$ and $|W| \leq n - 1$, otherwise G would be a clique. Now we consider two cases here.

Case 1: H is a clique with $(n - 1)$ vertices.

The conditions a) and b) are satisfied by H. By the inductive hypothesis, H does not have a $(k + 2)$ clique. Since G is not a clique, there exists at least one vertex in U not adjacent to x. Furthermore, all the minimal zx separators, where $z \notin \mathrm{ADJ}(x)$, are nothing but $\mathrm{ADJ}(x)$ itself. Since G satisfies condition a), $\mathrm{ADJ}(x)$ has to be a k clique. This implies that in G there is at least one $(k + 1)$ clique but no $(k + 2)$ clique.

Case 2: H is not a clique.

In G, for every nonadjacent pair of vertices $w, z \in U$, there exists a minimal separator which is a k clique. Since x cannot be in any such minimal wz separator, H satisfies condition a) (by Lemma 4.1). Consider the recursive construction of the graph G starting from the maximal clique W. Let y be a vertex to be added next in this recursive construction process. The vertex y is not adjacent to x but adjacent to a clique in $\mathrm{ADJ}(x)$. Furthermore, any minimal xy separator is contained in $\mathrm{ADJ}(x)$. If either $|\mathrm{ADJ}(x)|$ or $|\mathrm{ADJ}(y)| < k$, then there exists a minimal xy separator of size less than k in G which is a contradiction to our assumption. If both $|\mathrm{ADJ}(x)|$ and $|\mathrm{ADJ}(y)|$ are less than k, then again a similar contradiction arises. Therefore, $|\mathrm{ADJ}(y)| = |\mathrm{ADJ}(x)| = k$. Hence, H has $m - k = k(n - 1) - k(k + 1)/2$ edges. So H satisfies condition b) also. By the inductive hypothesis, H does not contain a $(k + 2)$ clique. By the above facts, G also does not contain a $(k + 2)$ clique. □

The following sequential algorithm tests whether a graph G is a k tree or not. The body of this algorithm is similar to that of the *ChordalTest* algorithm.

Algorithm k-treeTest [Graph: G = (V, E)];

```
begin
  if the equation m = kn − k(k + 1)/2 does not have a positive integer root ≤ n for k
  then
      begin
        print "G is not a k tree";
        terminate;
      end
      else
begin
    let k ≤ n be a positive integer root of the above equation;
  for every nonadjacent unordered pair of vertices u and v of V do
      G_uv := G − ADJ(u);
      G^v_uv := component of G_uv containing v;
      M_uv := {x : x ∈ ADJ(u) and x is adjacent to some vertex in G^v_uv};
      if M_uv induces a k clique in G then continue
      else
      begin
        print "G is not a k tree";
        terminate;
      end;
  print "G is a k tree";
  end
end.                                                                    □
```

The correctness of the above algorithm is established below.

Lemma 4.3: The graph G is a k tree if and only if

1) $m = kn - k(k+1)/2$, and
2) M_{uv} (in the algorithm *k tree*) induces a k clique for all nonadjacent pairs of vertices u and v of G.

Proof:

(IF): It is easy to see that M_{uv} is a minimal uv separator in G. Furthermore, the condition b) in Lemma 4.2 is satisfied. By Lemma 4.2, G is a k tree.

(ONLY IF): G is a k tree. Then clearly $m = kn - k(k+1)/2$ and therefore k will have a positive integer value in any solution to this equation. Furthermore, if G is not a clique, it can be easily shown that the above equation can have at most one positive integer root less than n for k. Now, assume that for some nonadjacent pair of vertices u and v, M_{uv} is not k clique. Then because M_{uv} is a minimal separator, it is not a k tree contradicting the assumption [by Theorem 2.4 4)c)]. □

The proof of the following result is straightforward.

Theorem 4.4: The algorithm *k-treeTest* correctly tests if G is a k tree or not. □

Along the same lines as the NC algorithm for recognizing chordal graphs, the sequential algorithm *k-treeTest* can be parallelized into an NC algorithm. Therefore, we have the following theorem.

Theorem 4.5: There exists a parallel algorithm for recognizing k trees which takes $O(\log n)$ time and makes use of $O(n^4 + n^2m + n^3)$ processors. □

V. Conclusions

We have shown that the chordal graphs and k trees have NC recognition algorithms. We note that the other important subclasses of graphs in the chordal hierarchy like the split graphs, indifference graphs, and the threshold graphs have straightforward NC algorithms for recognition. Of interest would be to obtain NC algorithms for recognizing the directed path, the undirected path, and the strongly chordal graphs. In a wider context, more effort is needed toward developing parallel algorithms on various classes of perfect graphs. This would include both fast parallel recognition algorithms and algorithms for solving combinatorial optimization problems on these graphs.

Acknowledgment

We thank Dr. J. Gilbert and Dr. D. Kozen, the former for bringing to our notice the work of Dr. A. Edenbrandt and the latter for his prompt and helpful replies to our questions. We note that recently Naor, Naor, and Schaffer [17] have developed NC algorithms for finding a PEO of a chordal graph and for finding clique number, etc. We thank Dr. D. Shier and an anonymous referee for bringing the above work to our notice. We are thankful to Prof. Liu for handling this paper in a timely fashion.

References

[1] C. Beeri, R. Fagin, D. Maier, A. Mendelzon, J. Ullman, and M. Yannakakis, "Properties of acyclic database schemes," in *Proc. 13th Annu. ACM Symp. Theory Comput.*, 1981, pp. 355-362.

[2] L. Beineke and R. Pippert, "Properties and characterizations of k-trees," *Mathematica*, vol. 18, pp. 141-151, 1971.

[3] P. Buneman, "The recovery of trees from measures of dissimilarity," in *Mathematics in the Archaeological and Historical Sciences*. Edinburgh, Scotland: Edinburgh Univ., 1972, pp. 387-395.

[4] N. Chandrasekharan, "New characterizations and algorithmic studies on chordal graphs and k-trees," M.Sc. (Engg.) Thesis, School of Automation, Indian Instit. Sci., Bangalore, India, 1985.

[5] R. Chandrasekharan and A. Tamir, "Polynomially bounded algorithms

for locating p-centers on a tree," *Math. Programming,* vol. 22, pp. 304–315, 1982.
[6] D. Corneil, "Algorithms for perfect graphs," in *Proc. IEEE Int. Symp. Circuits Syst.,* 1985, pp. 1187–1190.
[7] G. Dirac, "On rigid circuit graphs," in *Proc. Abh. Math. Sem.,* Univ. of Hamburg, vol. 25, 1961, pp. 71–76.
[8] P. Duchet, "Classical perfect graphs," in *Topics on Perfect Graphs,* C. Berge and V. Chvatal Eds., *Ann. Disc. Math.,* vol. 21, pp. 67–96 North-Holland, 1984.
[9] A. Edenbrandt, "Combinatorial problems in matrix computation," Ph.D. dissertation, Cornell Univ., Ithaca, NY, July 1985.
[10] D. Foata, "Enumerating k-trees," *Disc. Math.,* vol. 1, pp. 181–186, 1974.
[11] D. Fulkerson and O. Gross, "Incidence matrices and interval graphs," *Pac. J. Math.,* vol. 15, pp. 835–855, 1965.
[12] F. Gavril, "The intersection graphs of subtrees in trees are exactly the chordal graphs," *J. Combin. Theory, B,* vol. 16, pp. 47–56, 1974.
[13] M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs.* New York: Academic, 1980.
[14] F. Harary and E. Palmer, "On acyclic simplicial complexes," *Mathematica,* vol. 15, pp. 115–122, 1968.
[15] D. Helmbold and E. Mayr, "Perfect graphs and parallel algorithms," in *Proc. IEEE 1986 Int. Conf. Parallel Processing,* Aug. 1986, to be published.
[16] D. Kozen, U. Vazirani, and V. Vazirani, "NC algorithms for comparability graphs, interval graphs, and testing for unique perfect matching," in *Proc. 5th Conf. FST & TCS,* New Delhi, India, pp. 496–503, 206, *Lecture Notes in Computer Science,* Springer-Verlag, 1985.
[17] J. Naor, M. Naor, and A, Schaffer, "Fast parallel algorithm for chordal graphs," in *Proc. Symp. Theory Comput.,* New York, May 1987.
[18] N. Pippenger, "On simultaneous resource bounds," in *Proc. 20th IEEE Symp. Foundations Comput. Sci.,* 1979, pp. 307–311.
[19] A. Proskurowski, "Recursive graphs, recursive labelings and shortest paths," *SIAM J. Comput.,* vol. 10, pp. 391–397, 1981.
[20] ——, "k-trees: Representations and distances," Tech. Rep., CIS-TR-80-5, Univ. Oregon, OR, 1980.
[21] ——, "Separating subgraphs in k-trees: Cables and caterpillars," *Disc. Math.,* vol. 49, pp. 275–285, 1984.
[22] D. Rose, "A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations," in *Graph Theory and* Computing. R. Read, Ed. New York: Academic, 1972, pp. 183–217.
[23] ——, "On simple characterizations of k-trees," *Disc. Math.,* vol. 7, pp. 317–322, 1974.
[24] D. Rose, R. E. Tarjan, and G. Lueker, "Algorithmic aspects of vertex elimination on graphs," *SIAM J. Comput.,* vol. 5, pp. 266–283, 1976.
[25] D. Shier, "Some aspects of perfect elimination orderings in chordal graphs," *Disc. Appl. Math.,* vol. 7, pp. 325–331, 1984.
[26] Y. Shiloach and U. Vishkin, "An $O(\log n)$ parallel connectivity algorithm," *J. Algorithms,* vol. 3, pp. 57–67, 1982.
[27] R. Tarjan and M. Yannakakis, "Simple linear-time algorithms to test the chordality of graphs, test acyclicity of hypergraphs and selectively reduce acyclic hypergrahs," *SIAM J. Comput.,* vol. 13, pp. 566–579, 1984.
[28] U. Vishkin, "Implementation of simultaneous-memory accesses in models that forbid it," *J. Algorithms,* vol. 4, pp. 45–50, 1983.
[29] S. H. Whitesides, "An algorithm for finding clique cut-sets," *Inform. Processing Lett.,* vol. 12, pp. 31–32, 1981.
[30] N. Chandrasekharan, R. Laskar, and S. S. Iyengar, "Maximal clique-separators of chordal graphs and New separator theorems," in *Proc. South Eastern Combinatorial Conf.*, Boca Raton, FL, Feb. 1987.
[31] A. Mortra and S. S. Iyengar, "Parallel algorithms for some computation problems," *Adv. Comput.*, vol. 26, pp. 95–153, 1987.

N. Chandrashekaran is presently a Ph.D. degree student in the department of Mathematics, Clemson University, Clemson, SC. His research interests are in parallel algorithms for graph problems.

S. Sitharama Iyengar received the Ph.D. degree in engineering in 1974.

He is currently a Professor of Computer Science and Supervisor of robotic research and parallel algorithms at Louisiana State University, Baton Rouge. He has authored (coauthored) more than 75 research papers in parallel algorithms, data structures, navigation of intelligent mobile robot, etc. He is currently studying the application of neural network techniques for path planning and learning in mobile robots. His papers have appeared in the following journals: IEEE TSE, IEEE PAMI, IEEE SMC, IEEE JOURNAL OF ROBOTICS AND AUTOMATION, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, CACM, JCIS, *Computer Networks, Journal of Robotic Systems, BIT, Theoretical Computer Science,* and several other international journals and IEEE proceedings. He is an ACM National Lecturer for 1986–1988. His research has been funded by NASA, DOE, NAVY, Jet Propulsion Lab. Caltech, etc.

Dr. Iyengar has been on the program committee for several major conferences in the USA and in Europe. He is a coguest editor for a special issue in IEEE TRANSACTIONS ON SOFTWARE ENGINEERING. He is also a guest editor for a special issue on autonomous intelligent machines in IEEE COMPUTER magazine.