SIAM J. DISC. MATH.
Vol. 3, No. 4, pp. 531–536, November 1990

008

THE DOMATIC NUMBER PROBLEM IN INTERVAL GRAPHS*

TUNG-LIN LU†, PEI-HSIN HO†, AND GERARD J. CHANG†

Abstract. A set of vertices D is a dominating set of a graph $G = (V, E)$ if every vertex in $V - D$ is adjacent to a vertex in D. The domatic number $d(G)$ of a graph $G = (V, E)$ is the maximum number k such that V can be partitioned into k disjoint dominating sets $D_1, \cdots, D_k$. The main purpose of this paper is to give linear algorithms for the domatic number problem in interval graphs. This paper also proves that $d(G) = \delta(G) + 1$ for any interval graph G, where $\delta(G)$ is the minimum degree of a vertex in G.

Key words. dominating set, domatic number, interval graph, degree, linear algorithm, NP-complete

AMS(MOS) subject classifications. 05C70, 68R10

1. Introduction. A set of vertices D is a *dominating set* of a graph $G = (V, E)$ if every vertex in $V - D$ is adjacent to a vertex in D. The *domatic number* $d(G)$ of a graph $G = (V, E)$ is the maximum number k such that V can be partitioned into k disjoint dominating sets $D_1, \cdots, D_k$. The domination set problem and its variations have been extensively studied; however, the domatic number problem is much less well known.

Lower bounds and upper bounds for the domatic number were studied in [4]–[6], [9]–[11], and [13]. In particular, [6] showed that $d(G) \leqq \delta(G) + 1$ for any graph G, where $\delta(G)$ is the minimum degree of a vertex in G. G is *domatically full* if $d(G) = \delta(G) + 1$. Cockayne and Hedetniemi [6] determined $d(G)$ for some special classes of graphs; consequently, K_n, $\bar{K}_n$, C_{3n}, trees and maximal outerplanar graphs are domatically full.

The domatic number problem is NP-complete for general graphs [7] and circular-arc graphs [2]. The problem is solved in $O(n^2 \log n)$ time for proper circular-arc graphs [2], $O(n^{2.5})$ time for interval graphs [1] and $O(n \log n)$ time for proper interval graphs [1].

The main purpose of this paper is to give linear algorithms for the domatic number problem in interval graphs. As a by-product we also prove that interval graphs are domatically full.

An *interval family* is a set of intervals on the real line. An interval family is *proper* if no interval is properly contained within another interval. A graph is a (*proper*) *interval graph* if there is a one to one correspondence between the vertices of the graph and the intervals of a (proper) interval family such that two vertices are joined by an edge if and only if their corresponding intervals overlap.

Interval graphs have been extensively studied and used as models for many real world problems. In particular, they have applications in archaeology, genetics, ecology, psychology, traffic control, computer scheduling, storage information retrieval, and electronic circuit design (see [8], [12]).

Booth and Lueker [3] gave a linear algorithm for deciding whether a given graph is an interval graph and constructing, in the affirmative case, the required interval family. In this paper we begin with the assumption that G is known to be an interval graph, and a corresponding interval family is given.

* Received by the editors June 30, 1989; accepted for publication (in revised form) March 6, 1990. This research was supported by the National Science Council of the Republic of China under grant NSC-77-0208-M009-21.

† Department of Applied Mathematics, National Chiao Tung University, Hsinchu 30050, Taiwan, Republic of China.

2. Notation and assumptions. Suppose $I = \{1, \cdots, n\}$ is the interval family for an interval graph G produced by the linear algorithm in [3], where interval i is equal to $[a_i, b_i]$ for $i = 1, \cdots, n$. a_i is the *left endpoint* of interval i and b_i the *right endpoint*. Without loss of generality we may assume that $\{a_1, \cdots, a_n, b_1, \cdots, b_n\} = \{1, 2, \cdots, 2n\}$.

For the sake of simplicity we will denote an interval graph G as $G(I)$ and deal with intervals instead of vertices. In this way, the *closed neighborhood* $N[i]$ of an interval i is the set of all intervals that overlap with interval i. A dominating set for $G(I)$ corresponds to a subset S of intervals in I such that every interval in I overlaps with at least one interval in S.

For each interval i *next*(i) is the interval j such that b_j is as small as possible but satisfying $b_i < a_j$; next(i) is null if no such j exists.

3. The algorithms. In this section we will give two efficient algorithms for the domatic number problem in interval graphs $G(I)$. For technical reasons, we first augment I with two "dummy" intervals 0 and $n + 1$ such that $a_0 = -1$, $b_0 = 0$, $a_{n+1} = 2n + 1$ and $b_{n+1} = 2n + 2$. We then construct an acyclic directed graph H as follows. The nodes of H correspond to the intervals in $I' \equiv I \cup \{0, n + 1\}$. There is a directed arc (i, j) in H if and only if $j \in N[\text{next}(i)]$. Note that if (i, j) is an arc in H, then $b_i < b_j$. This guarantees that H is acyclic. Since $N[n + 1] = \{n + 1\}$, $(i, n + 1)$ is an arc in H if and only if next$(i) = n + 1$.

LEMMA 3.1. *Any directed path from node* 0 *to node* $n + 1$ *in* H *corresponds to a dominating set for* $G(I)$.

Proof. Suppose $\langle i_0, i_1, \cdots, i_r \rangle$ is a directed path from node 0 to node $n + 1$ in H. By the definition of an arc in H, $b_{i_0} < b_{i_1} < \cdots < b_{i_r}$. For any interval $j \in I$, choose an index s such that $b_{i_{s-1}} < a_j < b_{i_s}$. Suppose intervals j and i_s do not overlap. Then $b_{i_{s-1}} < a_j < b_j < a_{i_s} < b_{i_s}$. Let $k = \text{next}(i_{s-1})$. By the definition of function next, $b_k \leqq b_j$ and so intervals k and i_s do not overlap. That implies $i_s \notin N[\text{next}(i_{s-1})]$, on contradicting that (i_{s-1}, i_s) is an arc in H. Therefore $j \in N[i_s]$ and so $\{i_1, \cdots, i_{r-1}\}$ is a dominating set for $G(I)$. □

A dominating set of $G(I)$ does not necessary correspond to a directed path from node 0 to node $n + 1$ in H. So we cannot conclude immediately, as in [1], that the domatic number of $G(I)$ is equal to the maximum number of disjoint paths from node 0 to node $n + 1$ in H. In fact our definition of directed graph H is different from that in [1]. The way our algorithms work is by means of the following duality relation.

LEMMA 3.2 [6] (weak duality inequality). $d(G) \leqq \delta(G) + 1$ *for any graph* G.

The main idea of our algorithms is to find $\delta + 1$ disjoint dominating sets in $G(I)$, or equivalently $\delta + 1$ disjoint paths from node 0 to node $n + 1$ in H. We will present two algorithms for this purpose. The first algorithm finds the dominating sets one by one. The second algorithm finds all dominating sets simultaneously. Section 4 implements these algorithms and shows that their running times are linear.

ALGORITHM D1

```
initially all intervals are unlabeled; k ← 0;
loop
  i ← 0;
  while (next(i) ≠ n + 1) do
    j ← next(i);
    if N[j] has no unlabeled intervals then STOP;
    choose an unlabeled interval h ∈ N[j] with largest left endpoint;
    label h by k + 1;
```

$i \leftarrow h$;
end while;
$k \leftarrow k + 1$;
forever.

Suppose k^* is the final k when Algorithm D1 stops. Let D_i be the set of all intervals labeled by i. By Lemma 3.1, we have k^* disjoint dominating sets $D_1, \cdots, D_k^*$.

LEMMA 3.3. *There exists an interval j such that $|N[j]| = k^*$.*

Proof. When Algorithm D1 stops there exists some interval j' such that all intervals in $N[j']$ are labeled by integers between 1 and k^*. Let j be such an interval with largest left endpoint.

Suppose $N[j]$ contains two distinct intervals p and q of the same label k. Assume intervals in D_k are labeled in the order $\cdots p = p_1, p_2, \cdots, p_m = q, \cdots$. By the definition of function next,

$$(3.1) \qquad b_p = b_{p_1} < a_{\text{next}(p_1)} < b_{p_2} < a_{\text{next}(p_2)} < \cdots < b_{p_{m-1}} < a_{\text{next}(p_{m-1})} < b_{p_m} = b_q.$$

Also $a_j < b_p$ since $p \in N[j]$. Let $r = \text{next}(p_{m-1})$. Then $a_j < a_r$. By the choice of j, $N[r]$ has an interval s which is unlabeled or is labeled by $k^* + 1$. Suppose $a_s > a_q$. Since $s, q \in N[r]$, by Algorithm D1, interval s would be labeled by k before interval q being labeled by k. So $a_s < a_q$. Also $a_q < b_j$ since $q \in N[j]$. Then $a_s < b_j$. On the other hand,

$$a_j < b_p < a_r < b_s.$$

The first inequality follows from that $p \in N[j]$, the second is part of (3.1), and the third from $s \in N[r]$. Both $a_s < b_j$ and $a_j < b_s$ imply that $s \in N[j]$, a contradiction to the assumption that all intervals in $N[j]$ are labeled by integers between 1 and k^* but that s is not. Hence all intervals in $N[j]$ have distinct labels, i.e., $|N[j]| = k^*$. □

By Lemmas 3.2 and 3.3 we have

$$\delta + 1 = \min_{i \in I} |N[i]| \leqq |N[j]| = k^* \leqq d(G(I)) \leqq \delta + 1,$$

and so in fact the above inequalities are equalities.

THEOREM 3.4 (strong duality theorem). *$d(G) = \delta(G) + 1$ for any interval graph G.*

THEOREM 3.5. *Algorithm* D1 *works for solving the domatic number problem in interval graphs.*

The second algorithm follows from the fact that the outdegree of each node i in H such that $(i, n+1)$ is not an arc in H is $|N[\text{next}(i)]| \geqq \delta + 1$. We will describe our algorithm in terms of H here and implement it in terms of intervals in the next section.

ALGORITHM D2
initially all nodes in H are unlabeled;
find a topological sort $i_0 = 0, i_1, i_2, \cdots, i_n, i_{n+1} = n + 1$ for the nodes of H
(i.e., (i_p, i_q) is an arc in H implies $p < q$);
label the first r intervals (according to the topological sort)
in $N[\text{next}(0)]$ by $1, \cdots, r$ respectively;
for $r \leftarrow 1$ to n do
if $(i_r, n+1)$ is not an arc in H and i_r is labeled by k then
(*) choose an unlabeled node j with (i_r, j) is an arc in H and label j by k;
end for.

Since $i_0 = 0, i_1, i_2, \cdots, i_n, i_{n+1} = n + 1$ is a topological sort for the nodes of H, the fact that each node of H such that $(i, n+1)$ is not an arc in H has outdegree at least

$\delta + 1$ implies that each subgraph H_r of H induced by $\{i_r, i_{r+1}, \cdots, i_{n+1}\}$ has the same property. So in Step(*) of Algorithm D2, we can always find such a node j. For each k between 1 and $\delta + 1$, $\{0, n + 1\}$ together with all nodes labeled by k form a directed path from node 0 to node $n + 1$. The algorithm does produce $\delta + 1$ disjoint paths from node 0 to node $n + 1$. So we have another way to verify Theorem 3.4 and solve the domatic number problem.

THEOREM 3.6. *Algorithm* D2 *works for solving the domatic number problem in interval graphs.*

4. Implementation. This section gives two implementations for Algorithm D1 and one implementation for Algorithm D2. The implementations show that the algorithms are linear.

We assume $\{a_1, \cdots, a_n, b_1, \cdots, b_n\} = \{1, 2, \cdots, 2n\}$. We also need the information that each $p \in [1, 2n]$ is equal to a_i or b_i for some interval $i \in I$. This can be done by a simple do loop. We can use one array to indicate that p is a left or a right endpoint and another array to indicate that p is the endpoint of an interval $i \in I$.

To implement our algorithms, we first must produce function next. This can be done in $O(n)$ time as follows. The algorithm scans the endpoints from $2n$ backward to -1. In the algorithm, s is the interval with smallest right endpoint among the intervals whose left endpoints have been scanned.

```
//*Calculate function next*//
s ← n + 1;
for p ← 2n to −1 step −1 do
   case 1: p = a_i for some interval i ∈ I'
           if b_i < b_s then s ← i;
   case 2: p = b_i for some interval i ∈ I'
           next(i) ← s;
end for.
```

Having calculated function next, we now present the first implementation of Algorithm D1. The main difficulty in Algorithm D1 is how to "choose an unlabeled interval $h \in N[i]$ with largest left endpoint." We use a stack to store candidates of such intervals such that an interval with larger left endpoint is closer to the top of the stack.

```
//*First implementation of Algorithm D1*//
label (i) ← 0 for all intervals i ∈ I; k ← 0;
loop
     stack S ← φ; h ← 0;
     for p ← 1 to 2n do
          j ← next(h);
          case 1: p = a_i for some interval i ∈ I.
                  if label(i) = 0 then push i into S;
          case 2: p = b_i for some interval i ∈ I.
                  if i = j then
                     while (S ≠ φ) do
                        pop h from S;
                        if b_h > a_j then
                           [label(h) ← k + 1;
                           if next(h) = n + 1 then goto (**) else goto(*)]
                     end while;
                     STOP the algorithm;
                     end if;
```

(*) end for;
(**) $k \leftarrow k + 1$;
forever.

As in the proof after Lemma 3.3, there are $\delta + 2$ iterations in the loop. Each iteration needs $O(n)$ steps. So this is an $O(\delta n + n) = O(|E| + |V|)$ time implementation of Algorithm D1.

As a second implementation, we only have to produce "sorted" closed neighborhoods $N[i]$ for all intervals $i \in I$. Once we have sorted closed neighborhoods, Algorithm D1 is easily implemented. We now give an $O(|E| + |V|)$ algorithm for producing the closed neighborhood of all intervals in I as follows.

//*Each closed neighborhood $N[i]$ is sorted according to left endpoints*//
doubly linked list $L \leftarrow \phi$;
for $p \leftarrow 1$ to $2n$ do
    case 1: $p = a_i$ for some interval $i \in I$
        $N[i] \leftarrow L$;
        $L \leftarrow L + i$; //*remember the address of i in L*//
        for each $j \in L$ do $N[j] \leftarrow N[j] + i$;
    case 2: $p = b_i$ for some interval $i \in I$
        delete i from L;
end for.

Finally, we shall implement Algorithm D2 in terms of intervals rather than the directed graph H. Note that if (i, j) is an arc in H, then $b_i < b_j$. A simple topological sort of the nodes of H is to sort intervals in I' according to their right endpoints.

//*Implementation of algorithm D2*//
label(i) $\leftarrow 0$ for all intervals $i \in I$;
$p \leftarrow 1$;
for $k \leftarrow 1$ to $\delta + 1$ do
    if $p = a_i$ for some interval i which intersects interval next(0) then label(i) $\leftarrow k$;
    $p \leftarrow p + 1$;
end for;
for $p \leftarrow 1$ to $2n$ do
    if $p = b_i$ for some interval $i \in I$ then
        if label(i) $\neq 0$ and next(i) $\neq n + 1$
        then [get $j \in N$[next(i)] with label(j) $= 0$; label(j) $\leftarrow$ label(i)]
end for.

In the above implementation, each closed neighborhood is considered as an unsorted single linked list. To get some interval j in N[next(i)] we simply get the first element of the list and delete it from the list. If its label is nonzero, we continue to get the first element from the remaining list until an interval with label zero is found.

5. Proper interval graphs. In the case where $G(I)$ is a proper interval graph, assume $a_1 < a_2 < \cdots < a_n$. It is easy to see that each closed neighborhood $N[i]$ is a consecutive set, i.e., $N[i] = \{j, j + 1, \cdots, j + k\}$ for some j and k. Since each $|N[i]| \geqq \delta + 1$, we can construct $\delta + 1$ disjoint dominating sets $D_k = \{i \in I : i \equiv k \bmod \delta + 1\}$, $k = 1, \cdots, \delta + 1$. This is a much simpler method than that in [1].

6. Conclusion. The main results of this paper give two linear algorithms for the domatic number problem on interval graphs. As a by-product we also show that interval

graphs are domatically full. A much simpler method for the problem in proper interval graphs is also discussed. We suspect that similar results can be obtained for the problem in strongly chordal graphs or even for chordal graphs.

REFERENCES

[1] A. A. BERTOSSI, *On the domatic number of interval graphs*, Inform. Proc. Lett., 28 (1988), pp. 275–280.
[2] M. A. BONUCCELLI, *Dominating sets and domatic number of circular arc graphs*, Disc. Appl. Math., 12 (1985), pp. 203–213.
[3] K. BOOTH AND G. S. LUEKER, *Testing for consecutive ones property, interval graphs and graph planarity using* PQ-*tree algorithms*, J. Comput. System Sci., 13 (1976), pp. 335–379.
[4] E. J. COCKAYNE, *Domination of undirected graphs—a survey*, Lecture Notes in Math. 642, Springer-Verlag, Berlin, New York, (1978), pp. 141–147.
[5] E. J. COCKAYNE AND S. T. HEDETNIEMI, *Optimal domination in graphs*, IEEE Trans. Circuits Systems, CAS-22 (1975), pp. 41–44.
[6] ———, *Towards a theory of domination in graphs*, Networks, 7 (1977), pp. 247–261.
[7] M. R. GAREY AND D. S. JOHNSON, *Computer and Intractability: A Guide to the Theory of* NP-*Completeness*, W. H. Freeman, San Francisco, CA, 1979.
[8] M. C. GOLUMBIC, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.
[9] F. JAEGAR AND C. PAYAN, *Relations du type Nordhans–Gaddum pour le nombre d'absorption d'un graphe simple*, C. R. Acad. Sci. Paris, Ser. A, 274 (1972), pp. 728–730.
[10] R. LASKAR AND H. B. WALIKAR, *On domination related concepts in graph theory*, Lecture Notes in Math. 885, Springer-Verlag, Berlin, New York, (1980), pp. 308–320.
[11] O. ORE, *Theory of Graphs*, Amer. Math. Soc. Colloq. Publ. 38, Providence, RI, 1962.
[12] F. S. ROBERTS, *Graph Theory and its Applications to Problems of Society*, CBMS-NSF Regional Conference Series 29, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1978.
[13] P. J. SLATER, *R-domination in graphs*, J. ACM, 23 (1976), pp. 446–450.