Discrete Applied Mathematics 39 (1992) 87-93
North-Holland

Note

Generalized vertex covering in interval graphs

Madhav V. Marathe*, R. Ravi** and C. Pandu Rangan

Department of Computer Science, Indian Institute of Technology, Madras 600036, India

Received 11 July 1989
Revised 17 January 1991

Abstract

Marathe, M.V., R. Ravi and C. Pandu Rangan, Generalized vertex covering in interval graphs, Discrete Applied Mathematics 39 (1992) 87–93.

Given an integer i and an undirected graph G, the generalized i-vertex cover problem is to find a minimum set of vertices such that *all* cliques in G of size i contain at least one vertex from this set. This problem is known to be NP-complete for chordal graphs when i is part of the input. We present a greedy linear time algorithm for this problem in the case of interval graphs.

1. Introduction

In an undirected graph, a clique R of size j is said to *cover* a clique T of size i ($i \geq j$) if $V(R) \subseteq V(T)$. Let $S_i(G)$ denote the set of *all* cliques of size i in G. Let $X \subseteq S_j(G)$. X is an (i,j) clique cover of G if for every T in $S_i(G)$, there is at least one R in X such that R covers T. X is said to be a minimum (i,j) clique cover if $|X| \leq |Y|$ for any (i,j) clique cover Y. The cardinality of a minimum (i,j) clique cover in G is called the (i,j) clique cover number of G and is denoted by $c_{i,j}(G)$. When $j = 1$, we shall refer to its $(i,1)$ clique cover as its i-vertex cover for obvious reasons.

Correspondence to: Professor Pandu Rangan, Department of Computer Science and Engineering, Indian Institute of Technology, Madras 600036, India.

* Current address: Department of Computer Science, SUNY, Albany, NY 12222, USA.

** Current address: Box 1910, Department of Computer Science, Brown University, Providence, RI 02912, USA.

We define the $C_{i,j}$ problem as follows [2]. Given an undirected graph G and an integer k, is $c_{i,j}(G) \leq k$? This problem is shown to be NP-complete for general graphs when $i > j \geq 1$ and for chordal graphs when $i > j \geq 2$ in [2]. We shall extend their notation and define the optimization version of the decision problem $C_{i,j}$ as $C'_{i,j}$. Thus, $C'_{i,j}$ is the problem of determining the exact value of $c_{i,j}$ and finding an (i,j) clique cover of this size. As before, we shall call the $C_{i,1}$ problem as the i-vertex cover problem. A polynomial time algorithm for the i-vertex cover problem for any fixed i was presented for chordal graphs in [2]. This employs a dynamic programming approach on a rooted clique tree of a chordal graph and has running time exponential in i. Further, it was also shown there that when i is part of the input, the problem is NP-complete.

We present below a greedy linear time algorithm to solve the i-vertex cover problem when i is the part of the input for interval graphs, a subclass of perfect graphs. Interval graphs are in fact a subclass of chordal graphs and they are extensively discussed in [3]. An interval graph is a graph whose vertices can be mapped to unique intervals on the real line such that two vertices in the graph are adjacent iff their corresponding intervals intersect. It is interesting to note that the 2-vertex cover problem is that of finding the minimum set of vertices such that all the edges of G are incident to at least one vertex from this set. This is exactly the *minimum vertex cover problem*. Thus, our result above also implies a linear time algorithm for the minimum vertex cover problem on interval graphs. Further, the 3-vertex problem on the class of chordal graphs is equivalent to that of computing the *minimum feedback vertex set* of the graph. This is because of the fact that chordal graphs do not have induced chordless cycles of size greater than three. Since interval graphs are a subclass of chordal graphs, our result above also implies a linear time algorithm for the minimum feedback vertex set problem on interval graphs.

2. Definitions

Let $G=(V,E)$, $|V|=n$, $|E|=m$ be an interval graph. We assume that $V=\{1,\ldots,n\}$ and that the vertices of G are labeled according to the IG ordering [5]. This is simply the order in which we consider the intervals in the intersection model of G in the nondecreasing order of their right endpoints. Fig. 1 shows an interval graph with its interval representation and its vertices labeled in the IG ordering. An important property of this ordering is that for vertices x, y, z in V such that $x<y<z$ in the ordering, if x is adjacent to z, then y is also adjacent to z [5]. This ordering, which can be computed in linear time, has led to several efficient algorithms on interval graphs [5,1,6,4].

If vertices x and y are adjacent, then we write $x \sim y$ or $y \sim x$. The subgraph induced in G by the vertex set $\{1,2,\ldots,k\}$ is denoted by $G(k)$. The IG ordering in interval graphs is a PEO (perfect elimination ordering) [3]. Hence we can build up a representation of the clique formed by each vertex and the vertices above it in the

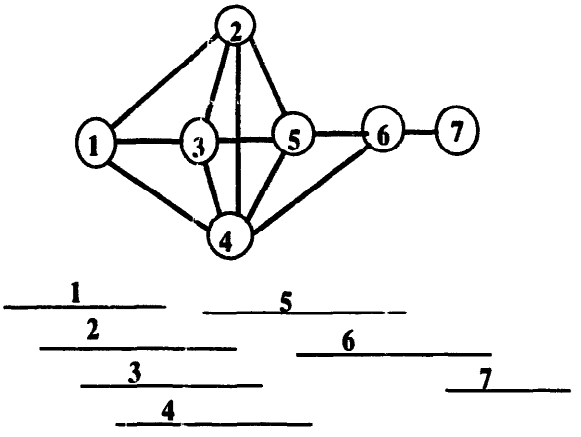

Fig. 1. An interval graph and its representation.

IG ordering that it is adjacent to. For every vertex v in V, define K_v as follows:

$$K_v = \{w\colon\ w > v \text{ and } w \sim v\}.$$

Since the IG ordering is a PEO, $\{v\} \cup K_v$ forms a clique in G. We call this the *associated clique* of vertex v and denote it by C_v. It is well known that any chordal graph, and hence any interval graph, has at most n maximal cliques. We remark that the set of all maximal cliques in an interval graph is a subset of the set of associated cliques of all the vertices of the graph. We denote by $MVC_{i,1}$, a set of all 1-cliques that together cover all the i-cliques of G, i.e., a set of vertices such that all cliques in the graph of size i contain at least one vertex from this set. In fact, we will drop the 1 from $MVC_{i,1}$ and denote it by MVC_i hereafter. We shall refer to the minimum $(i, 1)$ clique cover as the minimum i-vertex cover for brevity. We denote by $MVC_i(k)$, the minimum i-vertex cover of the subgraph $G(k)$ that covers all cliques in $G(k)$ of size i. We define $C_v(k)$ to be the part of C_v that is in $G(k)$. In other words,

$$C_v(k) = \{w\colon\ w \in C_v \text{ and } 1 \leq w \leq k\}.$$

In the subgraph $G(k)$, let $W_v(k)$ denote the number of vertices in the clique $C_v(k)$ formed so far in $G(k)$ that are not in $MVC_i(k)$. In other words,

$$W_v(k) = \{w\colon\ w \in C_v(k) \text{ and } w \notin MVC_i(k)\}$$

3. The algorithm

We follow a greedy incremental approach in choosing the minimum i-vertex cover. That is, we choose a vertex v to be included in the minimum vertex cover only when it is absolutely essential. In other words, if the addition of the vertex v to the

graph $G(v-1)$ creates new uncovered cliques of size i, then we include v in the minimum i-vertex cover. To keep track of such newly formed uncovered cliques, we incrementally build up the representation of the associated cliques of each vertex. At stage j, we compute $C_v(j)$ for all v such that $1 \leq v \leq j$. Further, for each vertex v, we maintain an uncovered weight W_v initialized to zero. The value of W_v at the jth iteration is the number of vertices in the associated clique for vertex v uncovered by the set $MVC_i(j-1)$ collected until the previous stage and is denoted by $W_v(j)$. For any vertex v, we define

$$LO(v) = \begin{cases} \min\{w: w < v \text{ in the IG order and } w \sim v\}, & \text{if such a } w \text{ exists,} \\ v, & \text{otherwise.} \end{cases}$$

This definition facilitates the updates to be effected to the counters $C_v(j)$ and $W_v(j)$. Since the definition of LO captures the regular nesting property of the maximal cliques of the interval graph, the only vertices whose counter values would be affected at iteration j are those in the set $\{LO(j), \ldots, j\}$. If this iteration causes any of these uncovered weight counters to attain value i, then we add the vertex j to the vertex cover maintained so far.

The algorithm is presented using two easy procedures "Update" and "Include" at each iteration. Procedure "Update" computes the revised weights of all the associated cliques. It then signals through a Boolean flag "Must" whether the inclusion of the present vertex has caused the formation of any clique of size i that is as yet uncovered. Thus, it indicates whether the current vertex must be included in the cover. Procedure "Include" adds the current vertex to the minimum i-vertex cover maintained so far. It also recomputes the uncovered weights of the associated cliques. This is done only if the "Must" flag was set to true in this iteration. The cover itself can be maintained as a simple bit vector.

Algorithm MIN VERTEX COVER.

Given an interval graph G with its vertices in the IG ordering and an array LO giving the value of $LO(v)$ for each vertex v and the integer i, a minimum i-vertex cover MVC is constructed as follows.

(1) Initialize a counter W_v for each vertex v to zero and MVC to null.
(2) For each vertex v from 1 to n do
  (2.1) UPDATE($v, LO(v), Must$).
  (2.2) If ($Must$) then INCLUDE($v, LO(v), MVC$).

Procedure UPDATE($v, LO(v), Must$).

(1) Set $Must$ to false.
(2) For each vertex u from $LO(v)$ to v do
  (2.1) Increment W_u by one since vertex v expands the associated cliques of exactly these vertices u.
  (2.2) If any of these counters equal i, then there is an uncovered i-clique that has been identified and so set $Must$ to true.

Procedure INCLUDE($v, LO(v), MVC$).

(1) Add v to the bit vector MVC.

(2) For each vertex u from $LO(v)$ to v do

(2.1) Decrement W_u by one since the vertex v has now been included in the set MVC.

4. Proof of correctness and complexity

Theorem 4.1. *Algorithm MIN VERTEX COVER produces the minimum i-vertex cover of an interval graph for any input i using linear time and space.*

Before we prove correctness of Algorithm MIN VERTEX COVER, we present a useful lemma.

Lemma 4.2. *A graph* $G=(V,E)$ *with* $|V|=n$ *is an interval graph iff its vertices can be numbered from* 1 *to n such that for* $i<j<k$, (i,k) *is an edge in the graph only if* (j,k) *is an edge in the graph, i.e.,*

$$(i,k)\in E \Rightarrow (j,k)\in E.$$

Such an ordering is the IG ordering of the interval graph.

For a proof of the above, see [5]. The proof of correctness of Algorithm MIN VERTEX COVER follows directly from the observation that W_u at iteration k correctly maintains $W_u(k)$ and from Lemma 4.3 below that establishes the minimality of the set $MVC_i(k)$ maintained by the algorithm. Note that inclusion of the vertex $k+1$ creates new i-cliques that are uncovered by the set $MVC_i(k)$ if and only if $W_u(k)<i$ and $W_u(k+1)=i$ for some $1\le u\le k+1$.

Lemma 4.3. *Let* $v\in V$ *be such that as the above algorithm runs,* $W_u(v)<i$ *and* $W_u(v+1)=i$ *for some u such that* $1\le u\le v+1$. *Then*

$$MVC_i(v+1)=MVC_i(v)\cup\{v+1\}.$$

Proof. We prove this in two parts. First, we show that in the above case, there exists no i-vertex cover of $G(v+1)$ of size less than $|MVC_i(v)|+1$. Then, we argue that the best vertex to include in $MVC_i(v+1)$ is $v+1$ which completes the proof.

We show that if $W_u(v)<i$ and $W_u(v+1)=i$ for some u such that $1\le u\le v+1$, then $|MVC_i(v+1)|=|MVC_i(v)|+1$. The proof is by contradiction. Assume that there exists a set S that is an i-vertex cover of $G(v+1)$ and is of cardinality less than $|MVC_i(v)|+1$. Then, we can write $S=MVC_i(v)-X\cup Y$ where X and Y are disjoint sets of vertices with $|X|\ge|Y|$. In other words, we can obtain set S from $MVC_i(v)$ by removing from it a subset X of vertices and adding instead a set of vertices Y of smaller or at most the same cardinality as X. We consider only the case when $|X|=|Y|$.

Notice that by way of formation of $MVC_i(v)$, for any vertex z included in this set at stage z, we can identify at least one unique i-clique that it covers in the subgraph $G(z)$. Thus, for each vertex $x_r \in X$, we can identify a corresponding vertex $y_r \in Y$ such that y_r covers the i-clique that x_r covered in the graph $G(x_r)$. Thus, $y_r < x_r$. Furthermore, there exists a $y_j \in Y$ such that it covers not only the i-clique covered by the corresponding x_j but also the new i-clique formed by the addition of the vertex $v+1$ to $G(v)$ which made $W_u(v+1)=i$. But now since x_j occurs later than y_j in the IG ordering, it will form a new uncovered i-clique with the same vertices that y_i did in the graph $G(v+1)$ due to the property of the IG ordering in Lemma 4.2. Thus, the set S is not an i-vertex cover since we identified an uncovered i-clique, a contradiction. This proves that $|MVC_i(v+1)| = |MVC_i(v)| + 1$.

To prove that $v+1$ is the best choice for inclusion in the set $MVC_i(v+1)$, observe that there is no other vertex y such that $y \le v+1$ that can be added to $MVC_i(v+1)$ and help in covering the new uncovered i-clique identified at this iteration and also cover more i-cliques at a later iteration. We can rule out $1 \le y < LO(v+1)$ directly since these vertices cannot even cover the new i-clique involving $v+1$ by the definition of LO. For any other y, if y forms an i-clique with vertices like z where $y < v+1 < z$ and $z \in \{v+2, \ldots, n\}$, then Lemma 4.2 implies that $v+1$ would also be part of such a clique and would cover it. Thus, the best choice for inclusion in $MVC_i(v+1)$ is $v+1$ itself and the minimality of this set is maintained from $G(v)$ to $G(v+1)$ proving that Algorithm MIN VERTEX COVER outputs a correct solution. □

As for complexity, the IG ordering and LO array calculations take linear time and space [5]. Since at the jth iteration of the algorithm, we do exactly d_j updates, where d_j is the degree of the vertex numbered j, the overall time complexity of the algorithm is $O(\sum_{j=1}^{n} d_j) = O(m)$. Both MVC and the array W_u take linear space to maintain. Thus the whole algorithm is of linear time and space complexity.

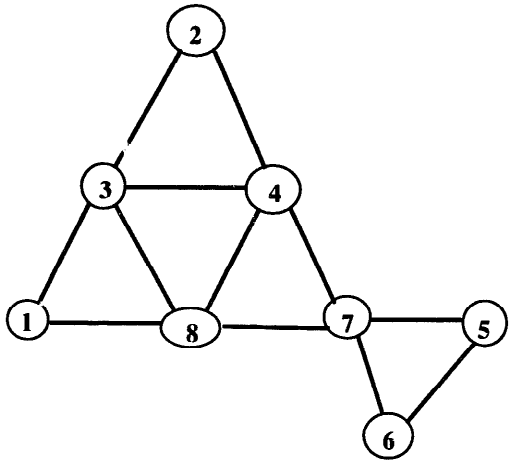

Fig. 2. An example of a chordal graph where the greedy method doesn't work.

5. Remarks

We draw attention to a remark in [2] that the greedy approach used above cannot be applied to the $C_{i,1}$ problem on chordal graphs. This can be seen in the example shown in Fig. 2 for the case $i=3$. As we proceed incrementally and greedily, we are forced to choose vertices 4 and 7 when they are encountered since they are required to cover the 3-cliques $\{2,3,4\}$ and $\{5,6,7\}$ respectively in the graph $G(7)$. But in the graph $G(8)$, we are again forced to include vertex 8 to cover $\{1,3,8\}$ while a smaller set $\{3,7\}$ would suffice to cover all the 3-cliques in G. The failure of the greedy approach here is because the nesting of the cliques is not as linear and regular in chordal graphs as in interval graphs.

It would be interesting to investigate if a polynomial algorithm exists for the $C(i,j)$ problem for any fixed j in the case of interval graphs.

References

[1] S. Aravind, R. Mahesh and C. Pandu Rangan, Bandwidth optimization for interval graphs revisted, Inform. and Comput., to appear.

[2] D.G. Corneil and J. Fonlupt, The complexity of generalized clique covering, Discrete Appl. Math. 22 (1988/89) 109-118.

[3] M.C. Golumbic, Algorithmic Graph Theory and Perfect Graphs (Academic Press, New York, 1980).

[4] J.M. Keil, Finding Hamiltonian circuits in interval graphs, Inform. Process. Lett. 20 (1985) 201-206.

[5] G. Ramalingam and C. Pandu Rangan, A unified approach to domination problems in interval graphs, Inform. Process. Lett. 27 (1988) 271-274.

[6] A. Srinivas Rao and C. Pandu Rangan, A linear algorithm for domatic number of interval graphs, Inform. Process. Lett. 33 (1989/90) 29-33.