TOTAL DOMINATION IN INTERVAL GRAPHS REVISITED

G. RAMALINGAM and C. PANDU RANGAN
Department of Computer Science and Engineering, Indian Institute of Technology, Madras 600036, India

Communicated by David Gries
Received 23 January 1987
Revised 20 July 1987

Keywords: Total domination, interval graph, algorithm

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 \( D \). A set of vertices \( D \) is a total-dominating set of a graph \( G = (V, E) \) if every vertex in \( V \) is adjacent to a vertex in \( D \). The problems of finding a minimum cardinality dominating set or a minimum cardinality total-dominating set are both NP-complete for many classes of graphs [6,2], though polynomial-time algorithms exist for some classes [1,2].

A graph is an intersection graph of a family \( F \) of sets if there exists a one-to-one correspondence between its vertices and the sets of \( F \) such that two vertices are adjacent in the graph iff the corresponding two sets intersect. If \( F \) is a family of intervals on a linearly ordered set (like the real line), then the graph is called an interval graph [5].

Bertossi [1] gives an \( O(n^2) \) algorithm for computing a minimum cardinality total-dominating set of an interval graph. Keil [7] proposed a linear algorithm for the same problem. Keil’s algorithm (from now on referred to as algorithm A) has a flaw; we present an example where it does not work and point out the error in the algorithm. A closer analysis of the problem is presented, followed by a revised version of the algorithm.

1.1. Definitions

The notations used are mostly as in [7]. Gilmore and Hoffman [4] have shown that the maximal cliques of an interval graph \( G \) can be linearly ordered such that, for every vertex \( x \) of \( G \), the cliques containing \( x \) occur consecutively. Let \( C_1, C_2, \ldots, C_m \) represent such an ordering of the maximal cliques of the given graph.

For every vertex \( v \) in \( G \), define low\([v]\) as the minimum \( i \) such that \( v \) is in \( C_i \) and level\([v]\) as the maximum \( i \) such that \( v \) is in \( C_i \). Thus, \( v \) occurs only in the cliques \( C_{\text{low}([v])} \) to \( C_{\text{level}([v])} \). Let \( L_i \) be the set of vertices \( v \) such that level\([v]\) is less than or equal to \( i \). Let \( R_i \) be the set of vertices \( v \) such that low\([v]\) is greater than or equal to \( i \).

If \( X \) is a set of vertices, let TD\((X)\) denote a minimum cardinality total-dominating set of \( X \). (A set \( Y \) of vertices is a total-dominating set of a set \( X \) if every vertex in \( X \) is adjacent to some vertex in \( Y \).) Note that TD\((X)\) is not unique since there may be more than one minimum cardinality total-dominating set.

If a set of vertices \( D \) total-dominates \( L_i \), for some \( i \), then the span of \( D \) (span\([D]\)) is defined as the maximum \( j \) such that \( D \) total-dominates \( L_{j-1} \).

Thus, given an interval graph \( G = (V, E) \), the problem is to compute a set TD\((V)\), that is, a TD\((L_m)\) or a TD\((R_1)\) since \( L_m = R_1 = V \). The
The graph is assumed to be connected. Otherwise, the connected components can be handled separately. Further, it is assumed that the graph has more than one vertex. (Otherwise, the graph has no total-dominating set.)

2. Counterexample

First, an example is given where algorithm A fails. Consider the graph, shown in Fig. 1, with eight vertices, whose maximal cliques, in order, are

$C_1 = \{1, 2\}$, $C_2 = \{2, 3\}$, $C_3 = \{3, 4\}$,
$C_4 = \{4, 5, 6\}$, $C_5 = \{5, 7\}$, $C_6 = \{7, 8\}$.

The set computed by algorithm A is $(2, 3, 4, 7, 8)$, whereas $(2, 3, 5, 7)$ is a smaller total-dominating set. A similar pattern repeated again and again gives the graph shown in Fig. 2.

The difference in the cardinality of the set yielded by algorithm A and the cardinality of a minimum total-dominating set can be arbitrarily large in graphs as in Fig. 2.

3. Results

The problem is viewed as one of computing a TD($R_i$). While executing the algorithm, a set that total-dominates vertices not in $R_i$ would have been computed and a set to total-dominate $R_i$ would be required. Hence, vertices not in $R_i$ are referred to as dominated and vertices in $R_i$ are referred to as undominated.

3.1. Lemma. Let

$j = \min_{x \in R_i} \text{level}[x].$

Then, there exists a TD($R_i$) that includes a vertex in $C_j$ (and all of whose vertices have level $\geq j$).

Proof. Assume that no vertex of a given TD($R_i$) is in $C_j$. Let $x$ be a vertex of $R_i$ such that level[$x$] = $j$. Let $y$ be a vertex of TD($R_i$) that dominates (i.e., is adjacent to) $x$. Then, $y$ occurs in at least one of the cliques in which $x$ occurs. From the assumption, $y$ does not occur in $C_j$. This means that level[$y$] $< j$. Since the cliques are maximal, there is a vertex $z$ in $C_j$ that does not occur in $C_{j-1}$. Replace $y$ by $z$ in TD($R_i$). The resulting set also total-dominates $R_i$ since any vertex of $R_i$ dominated by $y$ occurs in $C_j$ (by definition of $j$) and hence will be dominated by $z$ also. This proves that there exists a TD($R_i$) that includes a vertex occurring in $C_j$. Similarly, any vertex in TD($R_i$) that has level less than $j$ can be removed or replaced by vertices that have level greater than or equal to $j$. Hence, the result follows. $\square$

3.2. Lemma. If there exists a TD($R_i$) that includes a dominated vertex $x$ of $C_j$ (where $j$ is as defined in Lemma 3.1), then there exists a TD($R_i$) that includes a dominated vertex $y$ of $C_j$ of maximum level.

Proof. Assume the converse. Then, level[$x$] $< \text{level}[y]$. Replace $x$ by $y$ in TD($R_i$). The set continues to total-dominate $R_i$ because any vertex of $R_i$ dominated by $x$ occurs in at least one of $C_j$ to $C_{\text{level}[x]}$. Since $y$ also occurs in all these cliques, $y$ also dominates all the vertices in $R_i$ that are dominated by $x$. This proves the lemma. We now show how to compute a TD($R_i$) that includes $y$.

Let $X = \text{TD}(R_i) - \{y\}$. Since $y$ does not dominate any vertex in $R_{\text{level}[y]+1}$, $X$ should total-dominate $R_{\text{level}[y]+1}$. Further, $y$ dominates all vertices of $R_i$ which are not in $R_{\text{level}[y]+1}$. Thus, it is also sufficient if $X$ total-dominates $R_{\text{level}[y]+1}$.

Hence,

$$\text{TD}(R_i) = \begin{cases} \{y\} \cup \text{TD}(R_{\text{level}[y]+1}) & \text{if level}[y] < m, \\ \{y\} & \text{if level}[y] = m. \end{cases} \square$$
3.3. Lemma. If there exists a TD($R_i$) that includes an undominated vertex $x$ of $C_j$ ($j$ as defined in Lemma 3.1), then there exists a TD($R_i$) that includes the undominated vertex $y$, in $C_j$, of maximum level.

Proof. Assume the converse. As above, $y$ total-dominates any vertex of $R_i$ that $x$ total-dominates, other than $y$ itself. The vertex that total-dominates $x$ will total-dominate $y$ also. (This vertex is not $y$ since, by assumption, $y$ is not in TD($R_i$). Further, this vertex has level $\geq j$, from Lemma 3.1. Hence, this vertex will dominate $y$.) Hence, we can replace $x$ by $y$ in TD($R_i$). This proves the lemma.

We now show how to compute a TD($R_i$) that includes $y$.

Let $X = TD(R_i) - \{y\}$. Obviously, $X$ must be a TD($R_{level[y]+1} \cup \{y\}$). Thus, $X$ must include some vertex adjacent to $y$, say $u$. Let $z$ be the vertex in $C_{level[y]}$ of maximum level (other than $y$ itself). Then, $z$ is adjacent to any vertex of $R_{level[y]+1}$ that $u$ is adjacent to. Thus, if we replace $u$ by $z$ in $X$, the resulting set will also be a TD($R_{level[y]+1} \cup \{y\}$). Hence, we get

$$X = \begin{cases} 
\{z\} \cup TD(R_{level[z]+1}) & \text{if level}[z] < m, \\
\{z\} & \text{if level}[z] = m.
\end{cases} \quad \square$$

4. Revised algorithm

The above results yield a linear-time algorithm for computing TD($R_i$). Assume that we have computed a set $X$ such that $X \cup TD(R_i)$, for some $i$, yields TD($R_i$).

Let

$$j = \min_{x \in R_i} \text{level}[x].$$

Let $d_1$ be the dominated vertex in $C_j$ of maximum level and let $u_1$ be the undominated vertex in $C_j$ of maximum level. Let $k = \text{level}[d_1] + 1$ and $s = \text{level}[u_1] + 1$. At least one of $\{d_1\} \cup TD(R_{k})$ or $\{u_1\} \cup TD(\{u_1\} \cup R_{s})$ is TD($R_i$) (from the above lemmas).

In the next step, we consider the computation of TD($R_k$), proceeding just as above.

Let

$$t = \min_{x \in R_k} \text{level}[x].$$

Let $d_2$ and $u_2$ be the dominated and undominated vertices of maximum level in $C_i$. Let $p = \text{level}[d_2] + 1$ and $q = \text{level}[u_2] + 1$. Then, at least one of $\{d_2\} \cup TD(R_{p})$ or $\{u_2\} \cup TD(\{u_2\} \cup R_{q})$ is TD($R_i$).

We next consider the computation of TD($\{u_1\} \cup R_{s}$). As shown in the proof of Lemma 3.3, if $d_3$ is the vertex in $C_{s-1}$ of maximum level, then $\{d_3\} \cup TD(R_i)$ is a TD($\{u_1\} \cup R_{s}$), where $r = \text{level}[d_3] + 1$.

Combining the above two, at least one of $\{d_1, d_2\} \cup TD(R_{p})$, $\{d_1, u_2\} \cup TD(\{u_2\} \cup R_{q})$ or $\{u_1, d_3\} \cup TD(R_{r})$ is a TD($R_i$). Now, if $p < r$, then $\{d_1, d_2\} \cup TD(R_{p})$ can be ignored since TD($R_i$) will total-dominate $R_i$, also and hence will have no fewer vertices than TD($R_i$). Thus, in this case, at least one of $\{d_1, u_2\} \cup TD(\{u_2\} \cup R_{q})$ or $\{u_1, d_3\} \cup TD(R_{r})$ is a TD($R_i$). Conversely, if $p > r$, $\{u_1, d_3\} \cup TD(R_{r})$ can be ignored.

Similarly, we can continue, maintaining at any time at most two sets, both of the same cardinality, such that at least one of them will eventually yield TD($R_i$). The two sets, $D$ and $U$ say, would be such that the last vertex added to them would be a dominated one, $d$ say, and an undominated one, $u$ say, respectively. And the smaller of $D \cup TD(R_{level[d]+1})$ and $U \cup TD(\{u\} \cup R_{level[u]+1})$ will be TD($R_i$).

Though the algorithm can be implemented as above, the following analysis will be used to get a slightly modified algorithm. The analysis also serves to point out the error in algorithm A. In what follows, $D$, $U$, $d$, and $u$ are as above.

Let

$$\text{span}[D] = \begin{cases} 
\min_{x \in R_{level[d]+1}} (\text{level}[x]) & \text{if level}[d] < m, \\
m + 1 & \text{if level}[d] = m.
\end{cases}$$

Obviously, level$[d] < \text{span}[D]$.

Case 1: level$[u] \leq \text{level}[d]$. In this case, $D \cup TD(R_{level[d]+1})$ will be a TD($R_i$) since the other
set is no smaller than this one. This follows since
\[ \text{TD}(R_{\text{level}[u]+1} \cup \{u\}) \] will total-dominate
\[ R_{\text{level}[d]+1} \] also.

**Case 2:** span\([D]\) \leq \text{level}[u]. In this case, \( U \cup \text{TD}(R_{\text{level}[u]+1} \cup \{u\}) \) will be a TD\((R)\) since \( \text{TD}(R_{\text{level}[d]+1}) \) will total-dominate \( R_{\text{level}[u]+1} \cup \{u\} \) also.

**Case 3:** \( \text{level}[d] < \text{level}[u] < \text{span}[D] \). In this case, a choice cannot be made straightaway. Algorithm A chooses vertices correctly in Cases 1 and 2. However, in this case it chooses \( d \) which may not be a correct choice. However, both sets can be maintained, as in the above algorithm, postponing the choice to a later stage. The next step can be simplified as explained below.

We have \( \text{level}[d] < \text{level}[u] < \text{span}[D] \).

To compute \( \text{TD}(R_{\text{level}[u]+1}) \), first choose either \( d_1 \), a dominated vertex of maximum level in \( C_{\text{span}[D]} \), or \( u_1 \), an undominated vertex of maximum level in \( C_{\text{span}[D]} \). Since \( d_1 \) is dominated, \( d_1 \) occurs in \( C_{\text{level}[d]} \) and hence in \( C_{\text{level}[u]} \). The next vertex to be added to \( U \), from the proof of Lemma 3.3, is vertex \( d_2 \) in \( C_{\text{level}[u]} \) of maximum level. Since \( d_1 \) also is in \( C_{\text{level}[u]} \), \( \text{level}[d_2] \geq \text{level}[d_1] \). Hence, \( U \cup \{d_1\} \) can be ignored and \( U \cup \{d_2\} \) can be chosen. Thus, the remaining two possible sets are \( U \cup \{d_2\} \) and \( D \cup \{u_1\} \).

The algorithm is described below. Note that the terms 'dominated vertex' and 'undominated vertex' are used with respect to the set \( D \). At any stage, a vertex \( x \) is a dominated vertex iff \( \text{low}[x] < \text{level}[d] \) (except initially when all the vertices are undominated and \( d \) is undefined).

---

**Revised algorithm**

**Input:** An interval graph \( G = (V, E) \), represented by \( m \) ordered maximal cliques \( C_1 \) to \( C_m \). Given the graph, this ordering can be computed in linear time, as in [3].

**Output:** A minimum cardinality total-dominating set of \( G \).

```plaintext
for each vertex \( u \) compute \( \text{low}[u] \) and \( \text{level}[u] \);
\( D := \{\} \); \( \text{span}[D] := 1 \); \( U := \text{undefined} \);
while \( D \) undefined or \( \text{span}[D] \neq m + 1 \) do
    if \( D \) and \( U \) are defined
        \( \text{next}_d := \) the vertex in \( C_{\text{level}[u]} \) of maximum level;
        \( \text{next}_u := \) the undominated vertex in \( C_{\text{span}[D]} \) of maximum level;
        \( d := \text{next}_d \); \( D := U \cup \{d\} \);
        \( u := \text{next}_u \); \( U := D \cup \{u\} \);
    else
        if \( D \) defined
            \( u := \) the undominated vertex in \( C_{\text{span}[D]} \) of maximum level;
            \( U := D \cup \{u\} \);
            \( d := \) the dominated vertex in \( C_{\text{span}[D]} \) of maximum level (= undefined if none exists);
            if \( d \) defined
                then \( D := U \cup \{d\} \)
                else \( D := \text{undefined} \)
            fi
        else (only \( U \) is defined)
            \( d := \) the vertex in \( C_{\text{level}[u]} \) of maximum level;
            \( D := U \cup \{d\} \);
            \( U := \text{undefined} \)
        fi
    fi
```

20
if;
if $D$ is defined
then
  if $\text{level}[d] < m$
  then $\text{span}[D] := \min_{x \in K \text{level}[d] + 1} \text{level}[x]$
  else $\text{span}[D] := m + 1$
fi
fi;
if $D$ and $U$ are defined
if $\text{level}[d] \geq \text{level}[u]$
then $U := \text{undefined}$
else
  if $\text{level}[u] \geq \text{span}[D]$
  then $D := \text{undefined}$
fi
fi
{ $D$ is a minimum cardinality total-dominating set }

The correctness of the algorithm follows from the earlier results.

5. Time complexity

The algorithm can run in linear time. For an interval graph $G$ with $n$ vertices and $e$ edges, the number of maximal cliques $m$ is less than or equal to $n$. The summation of the cardinalities of the maximal cliques is less than or equal to $e + n$.

Each step of the algorithm can be implemented so that the whole algorithm runs in linear time. Computation of low and level is simple enough and requires just one pass through all the maximal cliques. Finding $\text{span}[D]$ can be done as below:

$$
\text{for } i := \text{level}[d] + 1 \text{ step 1 until } \text{span}[D] \text{ found do}
\quad \text{for all vertices } v \text{ in } C, \text{ do}
\quad \quad \text{if } (\text{low}[v] \geq \text{level}[d] + 1) \text{ and } \text{level}[v] = i)
\quad \quad \text{then } \text{span}[D] = i; \{ \text{span}[D] \text{ found.} \}
$$

Thus, the running time is

$$O\left(\sum_{i=1}^{m} |C_i|\right) = O(n + e).$$

References