

# Bottleneck Domination and Bottleneck Independent Domination on Graphs<sup>\*</sup>

WILLIAM CHUNG-KUNG YEN

*Department of Graphic Communications and Technology*

*Shih Hsin University*

*Taipei, 116 Taiwan*

*E-mail: ckyen001@ms7.hinet.net*

In this paper, the bottleneck dominating set problem and one of its variants, the bottleneck independent dominating set problem, are considered. Let  $G(V, E, W)$  denote a graph with  $n$ -vertex-set  $V$  and  $m$ -edge-set  $E$ , where each vertex  $v$  is associated with a real cost  $W(v)$ . Given any subset  $V'$  of  $V$ , the bottleneck cost of  $V'$  is defined as  $\max\{W(x) \mid x \in V'\}$ . The major task involves identifying a dominating set / independent dominating set of  $G$  such that their bottleneck costs are minimized.

This paper first proposes an  $O(n \log n + m)$  time algorithm for solving the Bottleneck Dominating Set problem on weighted general graphs using the binary search technique. Second, an  $O(n)$  time algorithm is designed for the problem on weighted trees. Then, we show that the situation is greatly different when the Bottleneck Independent Dominating Set problem (the BIDS problem) is considered. This paper proves that the BIDS problem is NP-hard on planar graphs and presents a linear-time optimal algorithm for the BIDS problem on weighted interval graphs.

**Keywords:** bottleneck cost, dominating set, independent dominating set, tree, planar graph, interval graph

## 1. INTRODUCTION

Suppose that  $G(V, E, W)$  represents a connected and undirected graph, where  $V$  is the set of  $n$  vertices and  $E$  is the set of  $m$  edges, respectively. Meanwhile, each vertex  $v$  is associated with a real cost  $W(v)$ . During the last twenty-five years, the fastest-growing area of study within graph theory has been domination and some of its related problems [31, 32]. The authors in [31, 32] pointed out some real applications of domination in graphs, e.g., radio station allocation, the maximal structurally equivalent set in social network theory, kernels in 2-person games, etc. From theoretical point of view, the major goal of all these applications is to find a set  $H$  of vertices in  $G$  such that every vertex in  $V - H$  must be adjacent to at least one vertex in  $H$ . Since each vertex is associated with a real cost, the most natural objective is to determine a set such that the *sum cost (total cost)* is minimized. Many previous works have dealt with this research issue [1-3, 5-7, 11-23, 30, 34, 37, 42, 44, 49].

In service facility location problems, besides minimizing the sum of the costs related to building service facilities, another significant goal is to minimize the "*largest cost*" of allocating these service facilities. This is very important if these facilities are invested

---

Received May 25, 2000; revised November 10, 2000; accepted January 9, 2001.

Communicated by Jang-Ping Sheu.

<sup>\*</sup>This research was supported by National Science Council, ROC, under contract number NSC-89-2213-E-159-001.

by different investors. It is reasonable and natural to keep the largest amount invested by the investors *as small as possible*. This cost is termed the *bottleneck cost*. Some papers have considered bottleneck cost minimization on graph problems, e.g., the Bottleneck Traveling Salesman Problem [10, 46], the Bottleneck  $k$ -Supplier Problem [35], the Bottleneck Spanning Tree Problem and the Bottleneck Maximum Cardinality Matching Problem [25], the Bottleneck Graph Partition Problem [36], etc. This paper examines the problem of finding dominating sets as well as variants with the minimum bottleneck costs on weighted graphs.

In the following, we will describe our problems in detail. Traditionally, a *dominating set*  $Q$  of a graph  $G(V, E)$  is a subset of  $V$  in which each vertex  $v \in (V - Q)$  is adjacent to at least one vertex  $u \in Q$ , i.e.,  $(v, u) \in E$ . Many types of dominating sets on graphs have been proposed and studied over the years, such as connected dominating sets (where the subgraph induced by  $Q$  is connected) [8, 33, 37, 52], independent dominating sets (where  $Q$  is independent, i.e.,  $(u, v) \notin E$ , for all  $u, v \in Q$ ) [22, 23, 38, 41], total dominating sets (where there is no isolated vertex in  $Q$ ) [6, 12, 39, 47, 48], and perfect dominating sets (where every vertex  $u \in V - Q$  is adjacent to exact one vertex in  $Q$ ) [42, 53, 54]. Most of these previous studies focused on finding a certain type of dominating set  $D$  such that the sum cost of  $D$  is minimized. For any  $H \subseteq V$ , the *bottleneck cost* is defined as  $\max\{W(x) \mid x \in H\}$ . The two problems considered in this paper are defined below:

**The Bottleneck Dominating Set problem (The BDS problem):** Identify a dominating set  $S \subseteq V$  with the minimum bottleneck cost among all possible dominating sets of a weighted graph  $G(V, E, W)$ .  $S$  denotes an optimal solution of  $G$ .

**The Bottleneck Independent Dominating Set problem (The BIDS problem):** Identify an independent dominating set (ID set)  $H \subseteq V$  with the minimum bottleneck cost among all possible ID sets of a weighted graph  $G(V, E, W)$ .  $H$  denotes an optimal solution of  $G$ .

Fig. 1 shows a graph with real costs on vertices. Examples of optimal solutions for the above two problems are described in the following.

- (1) For the BDS problem, the sets  $\{a, e, f\}$  and  $\{c, d, f\}$  are two dominating sets. The set  $\{a, e, f\}$  is a dominating set with the minimum bottleneck cost, 3, since  $\max\{W(a), W(e), W(f)\} = \max\{1, 1, 3\} = 3$ .
- (2) For the BIDS problem, the sets  $\{c, f\}$  and  $\{a, f\}$  are two ID sets. The set  $\{a, f\}$  is an ID set with the minimum bottleneck cost which is equal to 3.

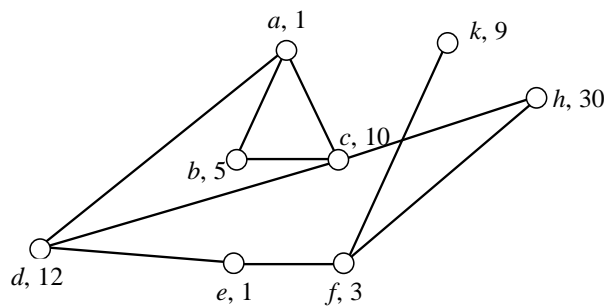


Fig. 1. A graph with real costs on vertices.

In this paper, we will first prove that the BDS problem can be solved in  $O(n \log n + m)$  time on weighted general graphs using the binary search technique. Next, we will show that the time-complexity can be reduced to  $O(n)$  if the input graphs are restricted on trees. Then, we will give a very interesting result: the BIDS problem is NP-hard even though the class of input graphs is restricted on planar graphs. After this, an  $O(n)$  time algorithm for the BIDS problem on weighted interval graphs will be designed using the dynamic programming strategy [4, 45]. This strategy has been applied to solve some other dominating set problems [44, 53]. The lower bounds of the BDS problem and the BIDS problem are both  $O(n)$  because each vertex must be examined at least once in order to obtain an optimal solution. Thus, both the algorithm for the BDS problem on weighted trees and the algorithm for the BIDS problem on weighted interval graphs are time-optimal.

## 2. THE BDS PROBLEM ON WEIGHTED GENERAL GRAPHS

To solve the BDS problem, a corresponding decision problem is defined.

**The Bottleneck Dominating Set decision problem (The BDS decision problem):** Given an undirected and connected graph  $G(V, E, W)$  as well as a real constant  $\eta$ , determine whether a dominating set  $S \subseteq V$  exists such that the bottleneck cost of  $S$  is less than or equal to  $\eta$ .

For each  $x \in V$ , define  $N(x) = \{y \mid y \in V \text{ and } (x, y) \in E\}$ . The following key lemma is established.

**Lemma 1.** Suppose that  $\psi(G(V, E, W) \text{ and } \eta)$  is an instance for the BDS decision problem. Let  $V' = \{x \mid x \in V \text{ and } W(x) \leq \eta\}$ . Then,  $V - V' = \{x \mid x \in V \text{ and } W(x) > \eta\}$ . The answer for  $\psi$  is 'YES' iff  $N(v) \cap V' \neq \emptyset$ , for all  $v \in (V - V')$ .

**Proof.** If the answer for  $\psi$  is 'YES', then this implies that a dominating set  $S$  exists such that its bottleneck cost is less than or equal to  $\eta$ , i.e.,  $S \subseteq V'$ . From this, we can directly derive that  $N(v) \cap V' \neq \emptyset$ , for all  $v \in (V - V')$ .

On the other hand, consider the case in which  $N(v) \cap V' \neq \emptyset$ , for all  $v \in (V - V')$ . The set  $V'$  is a dominating set with a bottleneck cost less than or equal to  $\eta$ . Therefore, the answer for  $\psi$  is certainly 'YES'. ■

Based on Lemma 1, we give the following algorithm for correctly solving the BDS problem on weighted general graphs.

### Algorithm BDS

#### Input:

A weighted graph  $G(V, E, W)$ , in which  $V = \{x_1, \dots, x_n\}$ .

#### Output:

A dominating set  $S$  of  $G$  such that the bottleneck cost of  $S$  is minimized.

**Method:**

Step 1: Sort the vertices of  $G$  into non-decreasing order using their costs as keys;  
 /\*  $W(x_1) \leq \dots \leq W(x_n)$  after sorting. \*/

Step 2:  $lb = 1$ ;  $ub = n$ ; /\* lower bound and upper bound of the indices \*/

Step 3:  $S =$  empty set;

Step 4: **while** ( $lb \leq ub$ )

Step 5:      $mid = (lb + ub) / 2$ ;

Step 6:      $V' = \{x_1, \dots, x_{mid}\}$ ;

Step 7:      $flag = \text{Test-Neighbor}(V')$ ;

Step 8:     **if**  $flag == \text{TRUE}$

Step 9:     {

Step 10:          $S = V'$ ;  $ub = mid - 1$ ;

Step 11:     }

Step 12:     **else**

Step 13:          $lb = mid + 1$ ;

Step 14:     **endif**

Step 15: **endwhile**

Step 16: **if** ( $S$  is not empty)  
           output  $S$ ;

Step 17: **endif**

**End BDS**

The output of the procedure  $\text{Test-Neighbor}(V')$  is TRUE if  $V' = V$  or  $N(v) \cap V' \neq \emptyset$ , for all  $v \in (V - V')$ . Otherwise, its return value is FALSE. First, we analyze its time-complexity. For each  $x \in V$ , suppose that  $N(x) = \{x_{j_1}, \dots, x_{j_p}\}$ . Define  $\ln(x) = j_p$ , i.e.,  $\ln(x)$  is the smallest index value of the vertices in  $N(x)$ . Then, it is easy to prove that  $N(v) \cap V' = \emptyset$  iff  $\ln(v) > mid$ , for each  $v \in (V - V')$ . Therefore, the procedure  $\text{Test-Neighbor}(V')$  can be conducted in  $O(|V - V'|)$  time if  $\ln(x)$  have been pre-computed, for all  $x \in V$ .

The following theorem can be established.

**Theorem 1.** The BDS problem can be solved in  $O(n \log n + m)$  time on weighted general graphs.

**Proof.** Let  $T(n)$  denote the time-complexity of Algorithm BDS. Step 1 involves sorting of  $n$  numbers, and its time-complexity is  $O(n \log n)$ . Based on the above discussion, the while loop from Step 4 to Step 15 can be performed in  $O(n \log n)$  time if  $\ln(x)$  has been pre-computed, for all  $x \in V$ . Computing all  $\ln(x)$  requires examining each edge once, and the time-complexity of this task is  $O(m)$ .

Combining the above results, it is easy to verify that  $T(n) = O(n \log n + m)$ . ■

### 3. AN $O(n)$ TIME ALGORITHM FOR THE BDS PROBLEM ON WEIGHTED TREES

Given a tree  $T$  and any vertex  $r$  of  $T$ , this tree is denoted by  $T(r)$ , i.e.,  $r$  is the root.

For any dominating set  $D$  of  $T(r)$ ,  $\beta(D)$  represents the bottleneck cost of  $D$ ; that is,  $\beta(D)$  can be expressed as  $\max_{v \in D} \{W(v)\}$ . For the sake of clear presentation, let  $\delta(r)$  denote the value of the minimum bottleneck cost of all optimal solutions of the BDS problem on  $T(r)$ ; i.e.,  $\delta(r) = \min\{\beta(D) \mid D \text{ is a dominating set of } T(r)\}$ .

It is clear that an optimal solution of the BDS problem either includes the root  $r$  or does not. This leads us to introduce the following two new related problems,  $(P_{r-})$  and  $(P_r)$ , which are in fact the original problem with additional constraints.

$(P_{r-})$  Find a dominating set  $H$  with the minimum bottleneck cost under the additional constraint that  $r \notin H$ .

$(P_r)$  Find a dominating set  $H$  with the minimum bottleneck cost under the additional constraint that  $r \in H$ .

Let  $\delta_0(r)$  and  $\delta_1(r)$  denote the minimum bottleneck costs of all optimal solutions of both problems  $(P_{r-})$  and  $(P_r)$ , respectively; i.e.,  $\delta_0(r) = \min\{\beta(D) \mid r \notin D \text{ and } D \text{ is a dominating set of } T(r)\}$  and  $\delta_1(r) = \min\{\beta(D) \mid r \in D \text{ and } D \text{ is a dominating set of } T(r)\}$ . Based on the definitions of the BDS problem and problems  $(P_{r-})$  and  $(P_r)$ , the following formula can be easily obtained:

$$\delta(r) = \min\{\delta_0(r), \delta_1(r)\}. \tag{3.1}$$

Formula (3.1) implies that  $\delta(r)$  can be computed in constant time when  $\delta_0(r)$  and  $\delta_1(r)$  have been obtained.

First, consider the boundary case in which  $T(r)$  only consists of vertex  $r$ . In this situation,  $\{r\}$  is the only dominating set. Therefore,  $\delta_0(r) = \infty$ ,  $\delta(r) = \delta_1(r) = W(r)$  under this boundary condition.

Another boundary case is the situation in which  $T(r)$  only consists of vertex  $r$  and its children, say  $x_1, \dots, x_k$ ,  $k \geq 1$ ; i.e.,  $x_1, \dots, x_k$  are all leaf vertices. Let  $H$  be any optimal solution. If  $r \in H$ , then  $x_1, \dots, x_k$  are dominated by  $r$  and  $\delta(r) \geq W(r)$ . It is easy to check that including any vertex in  $\{x_1, \dots, x_k\}$  will lead to a solution with a bottleneck cost greater than or equal to  $W(r)$ . Thus,  $\{r\}$  is already an optimal solution of problem  $(P_{r-})$ . On the other hand, if  $r \notin H$ , then  $\{x_1, \dots, x_k\}$  is the only dominating set left in  $T(r)$ .

Based upon the above discussion, the correctness of the following formulas for computing  $\delta_0(r)$ ,  $\delta_1(r)$ , and  $\delta(r)$  under this boundary condition can be easily verified:

$$\delta_0(r) = \max_{1 \leq i \leq k} \{W(x_i)\}, \tag{3.2}$$

$$\delta_1(r) = W(r), \tag{3.3}$$

$$\delta(r) = \min\{\delta_0(r), \delta_1(r)\}. \tag{3.4}$$

Second, consider a general tree  $T(r)$ . The vertex-set of  $T(r)$  is  $\{r\} \cup \Omega \cup \Gamma \cup V^+$ , where  $\Omega$ ,  $\Gamma$ , and  $V^+$  can be described as follows:

$$\Omega = \{y_1, \dots, y_q\} \text{ (\Omega may be empty),}$$

$$\Gamma = \{x_1, \dots, x_k\} \text{ (\Gamma \neq \emptyset), and}$$

$$V^+ \neq \emptyset.$$

A general tree  $T(r)$  is shown in Fig. 2.

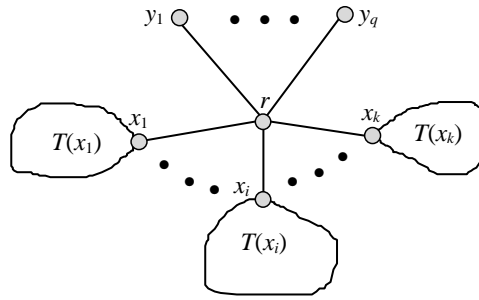


Fig. 2. A general tree  $T(r)$ , where  $y_1, \dots, y_q$  are leaf children of  $r$  and  $T(x_i)$  contains at least two vertices,  $i = 1, \dots, k, k \geq 1$ .

Now, consider problem  $(P_r)$ , in which no feasible solution includes  $r$ . Examine the set  $\Omega$ . Two situations can occur.

**Case I.**  $\Omega$  is not empty.

All the vertices in  $\Omega$  must be included in any optimal solution according to the definition of dominating sets. After that,  $r$  is dominated by all the vertices in  $\Omega$ . Therefore,  $x_1, \dots, x_k$  can be either included or excluded. Let  $\delta_0^{(I)}(r)$  denote the minimum bottleneck cost of  $T(r)$  in this case. Then, the following formula can be easily derived:

$$\delta_0^{(I)}(r) = \max\{\max_{v \in \Omega}\{W(v)\}, \delta(x_1), \dots, \delta(x_i), \dots, \delta(x_k)\}. \quad (3.5)$$

**Case II.**  $\Omega$  is empty.

In this case, an optimal solution must contain at least one vertex belonging to  $\Gamma$  in order to dominate  $r$ . Let  $\delta_0^{(II)}(r)$  denote the minimum bottleneck cost of  $T(r)$  in this case. The following formula can be easily derived:

$$\delta_0^{(II)}(r) = \min\{\Pi_1, \dots, \Pi_k\}, \text{ where } \Pi_i = \max\{\delta(x_1), \dots, \delta(x_{i-1}), \delta_1(x_i), \delta(x_{i+1}), \dots, \delta(x_k)\}. \quad (3.6)$$

Based on the discussion of the above two cases,  $\delta_0(r)$  can be easily computed using the following formula:

$$\delta_0(r) = \begin{cases} \delta_0^{(I)}(r), & \Omega \neq \emptyset \\ \delta_0^{(II)}(r), & \Omega = \emptyset. \end{cases} \quad (3.7)$$

Next, we will deal with problem  $(P_r)$ . If  $H$  is an optimal solution of problem  $(P_r)$ , then  $r \in H$  and  $\delta_1(r) \geq W(r)$ . If  $\Omega$  is not empty, then it is easy to verify that including any vertex in  $\Omega$  will lead to a solution with a bottleneck cost greater than or equal to  $W(r)$ . Meanwhile, the vertices in  $\Gamma$  can be either included or not included. Verifying the correctness of the following formula is a simple task when the optimal solutions of the BDS problem on the subtrees  $T(x_i)$ ,  $1 \leq i \leq k$ , have been computed:

$$\delta_1(r) = \max\{W(r), \delta(x_1), \delta(x_2), \dots, \delta(x_k)\}. \quad (3.8)$$

In the following, we will prove that the time-complexity of implementing the above formulas is  $O(n)$ . First, the following lemmas need to be established.

**Lemma 2.** If the  $2k$  values  $\delta_1(x_i)$  and  $\delta(x_i)$ ,  $i = 1, \dots, k$ , have been computed, then  $\delta_0(r)$  can be obtained in  $O(q + k)$  time for any subtree  $T(r)$  as shown in Fig. 2.

**Proof.** Formula (3.7) states that  $\delta_0(r)$  is equal to either  $\delta_0^{(1)}(r)$  or  $\delta_0^{(II)}(r)$ .  $\delta_0^{(1)}(r)$  can be computed in  $O(q + k)$  time based on Formula (3.5). The remaining task is to prove that  $\delta_0^{(II)}(r)$  can be computed in  $O(k)$  time.

First, we can compute the largest value, *maxval*, and the second largest value, *secval*, of the  $k$  values  $\delta(x_i)$ ,  $i = 1, \dots, k$ , in  $O(k)$  time.

Next, the value  $\delta_0^{(II)}(r)$  can be derived using a loop in  $O(k)$  time as follows:

```

 $\delta_0^{(II)}(r) = \text{maxval};$  /* initialization */
loop  $i = 1$  to  $k$ 
     $\text{maxval}_i = \text{maxval};$ 
    if  $\delta(x_i) = \text{maxval}$  then
        if  $\delta_1(x_i) \geq \text{secval}$  then
             $\text{mxaval}_i = \delta_1(x_i);$ 
            /*  $\delta_1(x_i)$  is the new maximum in  $\{\delta(x_1), \dots, \delta(x_{i-1}),$ 
                 $\delta_1(x_i), \delta(x_{i+1}), \dots, \delta(x_k)\}.$  */
        else
             $\text{mxaval}_i = \text{secval};$ 
            /*  $\delta_1(x_i)$  is the second largest in  $\{\delta(x_1), \dots, \delta(x_{i-1}),$ 
                 $\delta_1(x_i), \delta(x_{i+1}), \dots, \delta(x_k)\}$ , so  $\Pi_i =$  the second largest in
                 $\{\delta(x_1), \dots, \delta(x_{i-1}), \delta(x_i), \delta(x_{i+1}), \dots, \delta(x_k)\}.$  */
        endif
    else /*  $\delta(x_i) \leq \text{maxval};$ 
        if  $\delta_1(x_i) \geq \text{maxval}$  then
             $\text{maxval}_i = \delta_1(x_i);$ 
        endif
    endif

    /* Indeed,  $\text{maxval}_i = \Pi_i$  in Formula (3.6) */
 $\delta_0^{(II)}(r) = \min\{\delta_0^{(II)}(r), \text{maxval}_i\};$ 
endloop.

```

**Lemma 3.** For any internal vertex  $r$ ,  $\delta_1(r)$  can be computed in  $O(k + 1)$  time when the values  $\delta(x_i)$ ,  $1 \leq i \leq k$ , are available.

**Proof.** This lemma can be directly derived from Formula (3.8). ■

From Lemma 2, Lemma 3, and the fact that  $\delta(r)$  can be computed in constant time by simply comparing  $\delta_0(r)$  and  $\delta_1(r)$  for any internal vertex  $r$ , we can easily conclude that

the number of operations executed on each vertex is bounded by a constant.

Finally, an optimal solution can be identified by examining each vertex once from the root  $r$  after  $\delta(r)$  has been computed, and its time-complexity is also  $O(n)$ . Therefore, the following main result can be established.

**Theorem 2.** The BDS problem can be solved in  $O(n)$  time on weighted trees.

#### 4. THE BIDS PROBLEM IS NP-HARD ON PLANAR GRAPHS

This section will examine the complexity of the BIDS problem on planar graphs. A graph  $G(V, E)$  is said to be *planar* if it is possible to draw  $G$  in the plane so that the edges of  $G$  intersect only at end vertices [28]. Fig. 3 depicts an example of a planar graph.

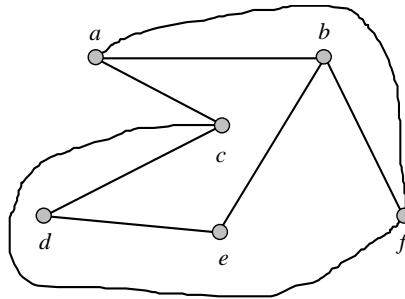


Fig. 3. An example of a planar graph.

A graph  $G(V, E)$  is called a *bipartite graph* if  $V$  can be partitioned into two disjoint sets  $X$  and  $Y$  such that both  $X$  and  $Y$  are independent sets [29]. A bipartite graph is called a *complete-bipartite graph* if  $(u, v) \in E$ , for all  $u \in X$  and  $v \in Y$ . In the rest of this paper,  $K_{m,n}$  will denote a complete-bipartite graph in which  $|X| = m$  and  $|Y| = n$ .

The following theorem states the necessary and sufficient conditions for testing the planarity of graphs [29].

**Theorem 3 (Kuratowski's Theorem).** A graph  $G$  is planar if and only if  $G$  contains no subgraph homeomorphic with  $K_5$  (a clique with 5 vertices) or  $K_{3,3}$ .

To analyze the complexity of the BIDS problem, a decision problem corresponding to the BIDS problem is defined.

**The Bottleneck Independent Dominating Set decision problem (the BIDS decision problem):** Given an undirected and connected graph  $G(V, E, W)$  as well as a real constant  $\eta$ , determine whether an independent dominating set  $S \subseteq V$  exists such that the bottleneck cost of  $S$  is less than or equal to  $\eta$ .

To simplify our proof and make it clearer, a variant of the BIDS decision problem is introduced below.

**The Constrained Independent Dominating Set decision problem (the CIDS decision problem).** Given an undirected and connected graph  $G(V, E)$  and a set of vertices  $V' \subseteq V$ , determine whether or not there exists an independent set  $S \subseteq (V - V')$  which dominates  $V'$ .

Now, the following lemma is established.

**Lemma 4.** The BIDS decision problem is polynomially equivalent to the CIDS decision problem.

**Proof.** First, we will prove that the BIDS decision problem can be reduced to the CIDS decision problem in polynomial time.

Let  $G_{\text{BIDS-decision}}(V, E)$  and the constant  $\eta$  comprise an instance of the BIDS decision problem. The corresponding instance of the CIDS decision problem can be constructed by setting  $V'$  as the set of the vertices with weights greater than  $\eta$  in  $V$ .

Suppose that there exists an independent set  $S \subseteq (V - V')$  which also dominates  $V'$  in the CIDS decision problem. There are two cases to be considered.

**Case I.**  $S = V - V'$

In this case, it is trivial that  $S$  is also an independent dominating set of the graph, and that the bottleneck cost of  $S$  must be less than or equal to  $\eta$ .

**Case II.**  $S \subset V - V'$

In this situation, the set  $Q = (V - V' - S)$  is not empty. Now, let  $Q' = \{v \mid v \in Q \text{ and } (u, v) \notin E, \text{ for all } u \in S\}$ ; i.e.,  $Q'$  is the subset of  $Q$  in which  $(u, v) \notin E$ , for all pairs  $u \in S$  and  $v \in Q$ . If  $Q'$  is not empty, then let  $H$  be any independent dominating set of  $Q'$ . This means that  $H \subseteq Q'$  and any vertex  $x \in Q' - H$  must be adjacent to at least one vertex in  $H$ . It is easy to verify that  $H$  can be found using a simple greedy algorithm, and that the time-complexity is polynomial. If  $Q'$  is empty, then set  $H$  to be empty. It is easy to check that  $S \cup H$  is an independent dominating set of the BIDS decision problem where the bottleneck cost is less than or equal to  $\eta$ .

From the above two cases, we can conclude that if the CIDS decision problem can be solved in polynomial time, then the BIDS decision problem can also be solved in polynomial time.

The next required task is to prove that the CIDS decision problem can be reduced to the BIDS decision problem in polynomial time.

The corresponding input of the BIDS decision problem can be obtained from an instance of the CIDS decision problem in the following way.

1. Set a function  $W$  which maps  $V$  to real values as follows:
  - $W(v) = 2$ , for all  $v \in V'$ ;
  - $W(v) = 1$ , for all  $v \in (V - V')$ .

2. Let  $\eta = 1.5$ .

Since  $\eta = 1.5$ , if  $S = \{x_1, \dots, x_q\}$  is a solution of the BIDS decision problem, then  $W(x_j) = 1$ , for all  $j = 1, \dots, q$ . It is clear that  $S \subseteq (V - V')$  and  $S$  also comprises a solution of the CIDS decision problem. Therefore, if the BIDS decision problem can be solved in polynomial time, then the CIDS decision problem can also be solved in polynomial time.

Based on the above two polynomial time reductions, a solution for the BIDS decision problem can be easily obtained from a solution of the corresponding CIDS decision problem in polynomial time, and vice versa. Therefore, these two problems are polynomially equivalent. ■

Now, the task of proving that the BIDS decision problem is NP-complete [26] is equivalent to the task of showing that the CIDS decision problem is NP-complete. First, a known NP-complete problem is introduced [26].

**The Planar Three Satisfiability problem (the P3SAT problem).** Given a set  $C$  of Boolean clauses in the conjunctive normal form over a set  $U$  of variables, determine whether the given Boolean formula is satisfiable or not under the following constraints:

- (1) Each clause contains at most three literals.
- (2) The bipartite graph  $G^B(V^B, E^B)$ , where  $V^B = U \cup C$  and  $E^B$  contains exactly those pairs  $\{u, c\}$  such that either  $u$  or  $\bar{u}$  belongs to the clause  $c$ , is planar.

Now, the following lemma is established.

**Lemma 5.** The CIDS decision problem is NP-complete on planar graphs.

**Proof.** We will first show that the CIDS decision problem belongs to the NP class of problems by designing an NP algorithm for it as follows:

**Algorithm NP\_CIDS\_decision**

**Input:**

An undirected graph  $G(V, E)$  and a subset  $H$  of  $V$ .

**Output:**

‘YES’ if an independent set  $S \subseteq (V - H)$  which dominates  $H$  exists, and ‘NO’, otherwise.

**Guessing Phase:**

Let  $(V - H) = \{y_1, \dots, y_z\}$ ;

**for**  $j \leftarrow 1$  to  $z$

$b_j = \text{choice}(\{0,1\})$ ; /\*  $b_j = 1$  iff  $y_j$  is selected. \*/

**endfor**

$S = \{y_j \mid b_j = 1\}$ ;

**Checking Phase:**

**for** each pair  $u, v \in S$

if  $(u, v) \in E$

```

    return 'NO';
  endif
endfor
for each  $u \in H$ 
  if  $N(u) \cap (V - H)$  is an empty set
    return 'NO';
  endif
endfor
return 'YES';

```

### End NP\_CIDS\_decision

It is a simple matter to show that the Checking Phase of the above algorithm can be completed in polynomial time with respect to  $n$  and  $m$ . This means that the CIDS decision problem belongs to the NP class of problems.

We will now show that the P3SAT problem can be reduced to the CIDS decision problem on planar graphs in polynomial time. Suppose that there is an instance of the P3SAT problem with the  $h$  variable-set  $U = \{u_1, \dots, u_h\}$  and  $r$  clause-set  $C = \{c_1, \dots, c_r\}$ . The bipartite graph  $G^B(V^B, E^B)$ , where  $V^B = U \cup C$  and  $E^B$  contains exactly the pairs  $\{u_i, c_j\}$  such that either  $u_i$  or  $\bar{u}_i$  belongs to the clause  $c_j$ , is planar. Consider the graph  $G_{\text{CIDS-decision}}(V, E)$  constructed in the following way:

$$\begin{aligned}
 V &= \{c_1, \dots, c_r\} \cup \{u_1, \dots, u_h\} \cup \{\bar{u}_1, \dots, \bar{u}_h\}, \\
 E &= \{(c_s, u_i) \mid c_s \text{ contains } u_i\} \cup \{(c_t, \bar{u}_i) \mid c_t \text{ contains } \bar{u}_i\} \\
 &\quad \cup \{(u_i, \bar{u}_i) \mid 1 \leq i \leq h\}.
 \end{aligned}$$

Indeed, the graph  $G_{\text{CIDS-decision}}(V, E)$  can be constructed from  $G^B(V^B, E^B)$  as follows:

- (1) Replace vertex  $u_i$  with the edge  $(u_i, \bar{u}_i)$ ,  $1 \leq i \leq h$ .
- (2) For each pair  $c_t$  and  $u_i$ , the following three cases must be handled.

**Case I.**  $c_t$  contains both  $u_i$  and  $\bar{u}_i$ .  
Add the edge  $(c_t, \bar{u}_i)$ .

**Case II.**  $c_t$  contains only  $u_i$ .  
Reserve the edge  $(c_t, u_i)$ .

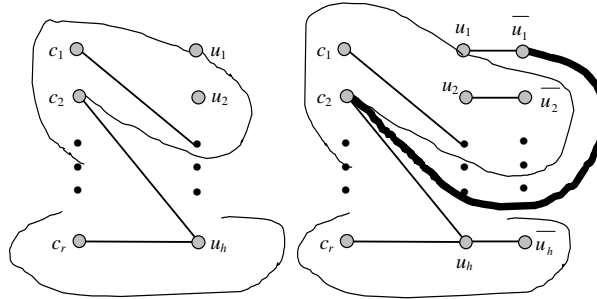
**Case III.**  $c_t$  contains only  $\bar{u}_i$ .  
Remove the edge  $(c_t, u_i)$  and add the edge  $(c_t, \bar{u}_i)$ .

It is easy to check that the graph  $G_{\text{CIDS-decision}}(V, E)$  contains no subgraph homeomorphic with  $K_5$  or  $K_{3,3}$ . Therefore, it is also a planar graph according to the Kuratowski's Theorem.

The time-complexity of constructing  $G_{\text{CIDS-decision}}$  from  $G^B$  can be easily proved to be polynomial. Now, let  $V' = \{c_1, \dots, c_r\}$ . Then,  $(V - V') = \{u_1, \dots, u_h, \bar{u}_1, \dots, \bar{u}_h\}$ .

The goal of the CIDS decision problem is to determine whether there exists an independent set  $S \subseteq (V - V')$  which dominates  $V'$ .

The remaining task is to show that there exists an independent set of  $(V - V')$  which dominates  $V'$  in  $G$  if and only if the given Boolean formula  $c_1 \bullet \dots \bullet c_r$  is satisfiable.



Remarks:  
 This figure assumes that  $c_2$  contains both  $u_1$  and  $\bar{u}_1$ .  
 Therefore, the edge  $(c_2, \bar{u}_1)$  is added.

Fig. 4. Construction of an input instance of the CIDS decision problem from an instance of the P3SAT problem.

- (1) Assume that there is an assignment satisfying the Boolean formula. Let  $u_{z_1} = \dots = u_{z_\alpha} = \text{TRUE}$  and  $u_{w_1} = \dots = u_{w_\varepsilon} = \text{FALSE}$ , where  $\alpha + \varepsilon = h$ . Then,  $z_i \neq w_j$ , for any  $i$  and  $j$ . Let  $S = \{u_{z_1}, \dots, u_{z_\alpha}, u_{w_1}, \dots, u_{w_\varepsilon}\} \subseteq (V - V')$ . Verifying that  $S$  is an independent set of  $(V - V')$ , and that  $S$  dominates all of the vertices in  $V'$  is a simple matter.
- (2) Assume that  $S \subseteq (V - V')$  is an independent set, and that it also dominates  $V'$ . Let  $Q = \{u_{z_1}, \dots, u_{z_\alpha}, u_{w_1}, \dots, u_{w_\varepsilon}\}$  be the set of literals corresponding to  $S$ ,  $\alpha + \varepsilon \leq h$ . Assign the literals in  $Q$  to be TRUE, i.e., assign  $u_{z_1} = \dots = u_{z_\alpha} = \text{True}$  and  $u_{w_1} = \dots = u_{w_\varepsilon} = \text{FALSE}$ . All other literals (if they exist) can be assigned to be either TRUE or FALSE. It is easy to ascertain that this assignment satisfies the given Boolean formula is easy.

Based on the discussion so far, we can draw the following conclusion. There exists an independent set in  $(V - V')$  dominating  $V'$  in the planar graph  $G_{\text{CIDS-decision}}$  if and only if the given Boolean formula is satisfiable. Therefore, the CIDS decision problem is NP-complete on planar graphs. ■

Consequently, the following theorem can be obtained.

**Theorem 3.** The BIDS problem is NP-hard on planar graphs.

### 5. AN $O(n)$ TIME ALGORITHM FOR THE BIDS PROBLEM ON WEIGHTED INTERVAL GRAPHS

This section deals with the BIDS problem on weighted interval graphs. A graph

$G(V, E)$  is called an *interval graph* if its vertices can be put into a one-to-one correspondence with a set  $F$  of intervals on the real line  $R$  so that two vertices are adjacent in  $G$  iff their corresponding intervals have a nonempty intersection.  $F$  is called an *intersection model* for  $G$  [28]. Fig. 5 and Fig. 6 show an interval graph and its intersection model, respectively.

In the rest of this paper, a highly effective tree structure, called a *clique path*, will serve to represent any interval graph [9, 15, 27]. In [27], Gavril proposed the use of *clique trees* to represent chordal graphs. Interval graphs form a subclass of the class of chordal graphs. Thus, we will first introduce chordal graphs. An edge is a *chord* of a cycle if it connects any two nonconsecutive vertices of the cycle. A graph  $G$  is *chordal* iff every cycle with length greater than three has a chord. Chordal graphs are intersection graphs if the sets are restricted to subtrees of a tree [27]. In [27], Gavril also demonstrated that the intersection models can be selected so that the nodes of the tree are always the *maximal cliques* of the original graphs. Each vertex then corresponds to the subtree consisting of exactly those cliques to which it belongs. Such an intersection model is called a *clique tree* of the graph. If the intersection model is further restricted such that the clique tree itself is a path, called a *clique path*, then it also defines the class of interval graphs [24, 40]. A clique tree of a chordal graph can be constructed in linear-time [51]. Fig. 7 shows the clique path of the interval graph shown in Fig. 5, where  $C_1, \dots, C_7$  are the maximal cliques of  $G$ .

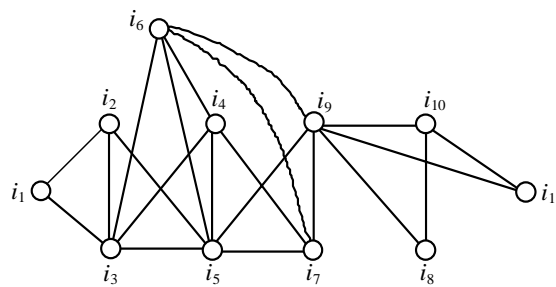


Fig. 5. An interval graph  $G$ .

Suppose that the clique path of the input interval graph  $G$  has  $z$  cliques,  $C_1, \dots, C_z$ , where  $C_j$  denotes the  $j$ th clique from left to right in the clique path. Define that  $C_i < C_j$  iff  $C_i$  stands to the left of  $C_j$ .  $C_{j+1}$  is said to be the *immediate successor* of  $C_j$  in this ordering scheme, for any  $1 \leq j < z$ , and we will hereafter use the notation  $IS(C_j)$  to denote the immediate successor of  $C_j$ . In addition, the *index number* of  $C_j$  is just  $j$  and is denoted by  $INDEX(C_j)$ .  $C_{i,j}$  here represents the subpath of the clique path consisting of the cliques  $C_i, C_{i+1}, \dots, C_j$ , for all  $i$  and  $j$ ,  $1 \leq i \leq j \leq z$ .

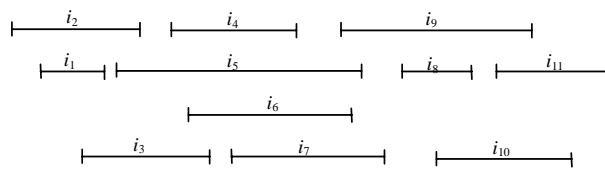


Fig. 6. An intersection model  $F$  of the interval graph shown in Fig. 5.

Each interval  $v$  can belong to more than one clique, say  $C_{v_1}, \dots, C_{v_p}, p > 1$ . The following properties can be directly derived.

**Property 1.** For any interval  $v$ , if  $v \in C_{v_1}, \dots, C_{v_p}, p > 1$ , then these  $p$  cliques form a subpath in the clique path.

**Property 2.** For any two maximal cliques  $C_i$  and  $C_j$  in the clique path,  $i \neq j, C_i - C_j \neq \emptyset$  and  $C_j - C_i \neq \emptyset$ ; i.e.,  $C_i$  can not contain  $C_j$  and vice versa.

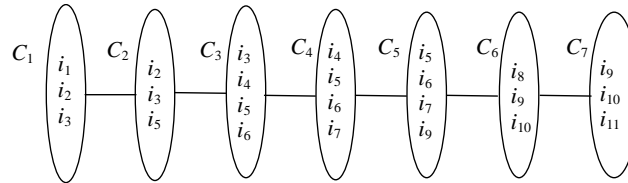


Fig. 7. A clique tree of the interval graph  $G$  shown in Fig. 5.

Assume that  $\langle LC(v), RC(v) \rangle$  denotes the leftmost clique and the rightmost clique in the subpath, respectively. If  $v$  only belongs to some clique  $C_v$ , then  $\langle LC(v), RC(v) \rangle = \langle C_v, C_v \rangle$ . In Fig. 7,  $i_5$  belongs to four cliques,  $C_2, C_3, C_4$ , and  $C_5$ , and they form a subpath of the clique path. Therefore,  $\langle LC(i_5), RC(i_5) \rangle = \langle C_2, C_5 \rangle$ . On the other hand,  $\langle LC(i_{11}), RC(i_{11}) \rangle = \langle C_7, C_7 \rangle$  because  $i_{11}$  only belongs to  $C_7$ .

Table 1 illustrates  $\langle LC(v), RC(v) \rangle$  for each vertex  $v$  of the interval graph shown in Fig. 5. For any two vertices  $u$  and  $v$ , we define  $u < v$  iff  $RC(u) < RC(v)$  or  $(RC(u) = RC(v) \text{ and } LC(u) \leq LC(v))$ . In Table 1,  $i_1 < \dots < i_{11}$ . Without a loss of generality, the vertices are assumed to be sorted according to this order in the rest of the paper.

Efficient algorithms exist for finding dominating sets [15, 49], path covering [50], and Hamiltonian circuits [43] on interval graphs. In this section, we will design an  $O(n)$  time algorithm for solving the BIDS problem on weighted interval graphs.

**Table 1. The  $\langle LC(v), RC(v) \rangle$  of all vertices of the interval graph shown in Fig. 5.**

Vertex $v$	$\langle LC(v), RC(v) \rangle$	Vertex $v$	$\langle LC(v), RC(v) \rangle$
$i_1$	$\langle C_1, C_1 \rangle$	$i_7$	$\langle C_4, C_5 \rangle$
$i_2$	$\langle C_1, C_2 \rangle$	$i_8$	$\langle C_6, C_6 \rangle$
$i_3$	$\langle C_1, C_3 \rangle$	$i_9$	$\langle C_5, C_7 \rangle$
$i_4$	$\langle C_3, C_4 \rangle$	$i_{10}$	$\langle C_6, C_7 \rangle$
$i_5$	$\langle C_2, C_5 \rangle$	$i_{11}$	$\langle C_7, C_7 \rangle$
$i_6$	$\langle C_3, C_5 \rangle$		

Any vertex  $v$  is called a *black vertex* if  $v$  must be included in any optimal solution, and  $v$  is called a *white vertex* if  $v$  must be excluded from any optimal solution. For any pair  $s, h$  in which  $1 \leq s < h \leq z$ , let  $\delta(C_{s,h})$  be the bottleneck cost of all optimal solutions of the BIDS problem on  $C_{s,h}$ ; i.e.,  $\delta(C_{s,h}) = \min\{\beta(D) \mid D \text{ is an ID set of } C_{s,h}, \text{ and } \beta(D) \text{ is the bottleneck cost of } D\}$ , where for any two pairs of real numbers  $(a, b)$  and  $(c, d)$ ,  $(a, b) < (c, d)$

$(c, d)$  iff  $(a < c)$  or  $(a = c \text{ and } b < d)$ , and  $(a, b) = (c, d)$  iff  $(a = c \text{ and } b = d)$ . The main objective of the BIDS problem is to compute  $\delta(C_{1,z})$ .

Since each node of the clique path corresponds to a maximal clique of the original interval graph, at most one vertex in  $C_s$  can be black,  $1 \leq s \leq z$ . The following property can be easily established.

**Property 3.** For each  $C_s$ , if  $v$  is a black vertex, then all vertices  $x \in ((C_s \cup \dots \cup RC(v)) - \{v\})$  are dominated by  $v$  and they must be white.

Let us first consider node (clique)  $C_1$ . Assume that  $Q = C_1 \cap C_2$ . By the definition of clique paths and Property 3, we can be sure that  $C_1 - Q$  must be nonempty. Therefore, one of the vertices in  $C_1$  must be black in order to dominate the vertices in  $C_1 - Q$ . This implies that the case in which all vertices in  $C_1$  are white can not occur.

Let  $v^*$  be the only black vertex in  $C_1$ . Assume that  $C_t$  is the immediate successor of  $RC(v^*)$ . Let  $H = RC(v^*) \cap C_t$ . Based on Property 3, in  $C_{t,z}$ , all vertices in  $H$  are dominated by  $v^*$ . This implies that all vertices in  $H$  can be “ignored” when we consider  $C_{t,z}$  in the remaining steps. Meanwhile, it is easy to verify that  $LC(y) = C_t$ , for all  $y \in C_t - H$ . Therefore, we refine the original BIDS problem to obtain a more general problem on  $C_{t,z}$  as follows:

( $P_R$ ) For any  $C_{t,z}$  and a given  $R \subset C_t$ , in which  $LC(y) = C_t$ , and for all  $y \in C_t - R$ , find an ID set with the minimum bottleneck cost on  $C_{t,z}$  such that all vertices in  $R$  can be “ignored.”

It is easy to verify that the original BIDS problem on  $C_{t,z}$  is just a special case of problem ( $P_R$ ), in which  $R$  is an empty set.

Let  $\delta_R(C_{t,z})$  be the minimum bottleneck cost of problem ( $P_R$ ) on  $C_{t,z}$ . After solving problem ( $P_R$ ) on  $C_{t,z}$  in which  $t = \text{INDEX}(\text{IS}(RC(v^*)))$  and  $R$  is given as  $RC(v^*) \cap \text{IS}(RC(v^*))$ , for each  $v^* \in C_1$ , we can easily derive the following formula.

$$\delta(C_{1,z}) = \min_{v^* \in C_1} \{ \max\{W(v^*), \delta_R(C_{t,z})\} \}, \text{ where for each } v^* \in C_1, t = \text{INDEX}(\text{IS}(RC(v^*))) \text{ and } R \text{ is the set } RC(v^*) \cap \text{IS}(RC(v^*)). \tag{5.1}$$

The remaining task is to solve problem ( $P_R$ ) on  $C_{t,z}$ ,  $t > 1$ , for some given  $R \subset C_t$ . The following lemma is established.

**Lemma 6.** Consider problem ( $P_R$ ) on  $C_{t,z}$ ,  $t > 1$ . If  $(C_t - R) \cap C_{t+1} \neq \emptyset$ , then there must exist a black vertex in  $(C_t - R)$  in order to dominate the vertices in  $(C_t - C_{t+1})$ . Otherwise, the problem becomes equivalent to problem ( $P_Q$ ) on  $C_{t+1,z}$ , where  $Q = R \cap C_{t+1}$ .

**Proof.** If  $(C_t - R) \cap C_{t+1} \neq \emptyset$ , then there exists  $u \in C_t - R$ , but  $u \notin C_{t+1}$ . Therefore, there must be one black vertex in  $(C_t - R)$  in this case.

The other case implies that  $(C_t - R) \subset C_{t+1}$ . Therefore, the problem becomes

equivalent to problem  $(P_Q)$  on  $C_{t+1,z}$ , where  $Q = R \cap C_{t+1}$ . ■

Based on Lemma 6 and using the similar reasoning in Formula (5.1), the following formulas can be obtained:

$$\delta_R(C_{t,z}) = \min_{u^* \in (C_t - R)} \{\max\{W(u^*), \delta_Q(C_{y,z})\}\}, \text{ if } (C_t - R) \cap C_{t+1} \neq \emptyset, \text{ where for each } u^* \in (C_t - R), y = \text{INDEX}(\text{IS}(\text{RC}(u^*))) \text{ and } Q \text{ is the set } \text{RC}(u^*) \cap \text{IS}(\text{RC}(u^*)); \quad (5.2)$$

$$\delta_R(C_{t,z}) = \delta_Q(C_{t+1,z}), \text{ if } (C_t - R) \subset C_{t+1}. \quad (5.3)$$

The boundary conditions of the problem occur when the interval graph only contains the maximal clique  $C_z$ . Verifying the correctness of the following two formulas is a simple matter:

$$\delta(C_z) = \min_{v^* \in C_z} \{W(v^*)\}, \quad (5.4)$$

$$\delta_R(C_z) = \min_{u^* \in (C_z - R)} \{W(u^*)\}, \text{ for any nonempty set } R \subset C_z. \quad (5.5)$$

Based on the computational results obtained so far, the following algorithm can be designed for solving the BIDS problem on weighted interval graphs using the dynamic programming strategy.

#### Algorithm BIDS\_Interval\_Graphs

/\* The BIDS problem on a weighted interval graph. \*/

##### Input:

The clique path  $C_{1,z}$  of the interval graph  $G$ .

##### Output:

The value  $\delta(C_{1,z})$ .

##### Method:

```

if  $z = 1$ 
{
  delta =  $\min_{u^* \in C_z} \{W(u^*)\}$ ;
  return(delta);
}
endif
delta =  $\infty$ ;
for each  $v^* \in C_1$  do
   $t = \text{INDEX}(\text{IS}(\text{RC}(v^*)))$ ;
   $R = \text{RC}(v^*) \cap \text{IS}(\text{RC}(v^*))$ ;
  delta_P_R_prime = BIDS_P_R_prime( $t, z$ );

```

```

    if max( $W(v^*)$ , delta_P_R_prime) < delta
        delta = max( $W(v^*)$ , delta_P_R_prime);
    endif
endfor
return(delta);
End BIDS_Interval_Graph

Algorithm BIDS_P_R_prime
/* The problem ( $P_R$ ) on  $C_{t,z}$ . */
Input:
    The clique path  $C_{t,z}$  of the interval graph  $G$ .
Output:
    The value  $\delta_R(C_{t,z})$ .
Method:
    if  $t = z$  /*  $|R| = |C_z| - 1$ . */
    {
        delta_P_R_prime =  $W(u^*)$ ; /*  $u^*$  is the only vertex in  $C_z - R$ . */
        return(delta_P_R_prime);
    }
    endif
    if  $(C_t - R) \cap C_{t+1}$  is not empty
        for each  $u^* \in (C_t - R)$ 
             $y = \text{INDEX}(\text{IS}(\text{RC}(u^*)))$ ;
             $Q = \text{RC}(u^*) \cap \text{IS}(\text{RC}(u^*))$ ;
            delta_P_Q_prime = BIDS_P_R_prime( $y, z$ );
            if max( $W(u^*)$ , delta_P_Q_prime) < delta_P_R_prime
                delta_P_R_prime = max( $W(v^*)$ , delta_P_Q_prime);
            endif
        endfor
    else
    {
         $Q = R \cap C_{t+1}$ ;
        delta_P_R_prime = BIDS_P_Q_prime( $t + 1, z$ );
    }
    endif
return(delta_P_R_prime);
End BIDS_P_R_prime

```

The time-complexity of the above algorithm is analyzed in the following. Let  $T(P_R, C_{t,z})$  represent the time-complexity for solving problem ( $P_R$ ) on interval graph  $C_{t,z}$  and let  $T(\text{BIDS}, C_{1,z})$  denote the time-complexity of the proposed algorithm. Summarizing the results obtained so far, the following formulas can be easily derived:

$$T(\text{BIDS}, C_{1,z}) = \sum_{v^* \in C_1} (T(P_R, C_{t,z}) + 1), \text{ where for each } v^* \in C_1, t = \text{INDEX}(\text{IS}(\text{RC}(v^*))) \text{ and } R \text{ is the set } \text{RC}(v^*) \cap \text{IS}(\text{RC}(v^*)); \quad (5.6)$$

$$T(P_R; C_{t,z}) = \sum_{u^* \in (C_t - R)} (T(P_{Q'}, C_{y,z}) + 1) + T(P_{F'}, C_{t+1,z}), \text{ where for each } u^* \in (C_t - R), y = \text{INDEX}(\text{IS}(\text{RC}(u^*))), \text{ and } Q \text{ is the set } \text{RC}(u^*) \cap \text{IS}(\text{RC}(u^*)), \text{ and } F = R \cap C_{t+1}. \quad (5.7)$$

Finally, Formula (5.4) and (5.5) for the boundary condition yield the following formula:

$$T(\text{BIDS}, C_z) = \sum_{R \subset C_z} T(P_{R'}, C_z) = O(|C_z|) \quad (5.8)$$

It is easy to ascertain that for each vertex  $v$ , we only need to examine it and  $\text{LC}(v)$  and  $\text{RC}(v)$  a set number of times. This implies that  $T(\text{BIDS}, C_{1,z}) = O(n)$ , where  $n$  is the number of vertices of the input interval graph.

**Theorem 4.** The BIDS problem can be solved in  $O(n)$  time on weighted interval graphs.

## 6. CONCLUSIONS

The problem of finding a dominating set with the minimum sum cost on weighted graphs is a well-known NP-hard problem. In this paper, we have derived a very interesting and meaningful result stating that the problem is solvable in  $O(n \log n + m)$  time if the cost considered is the bottleneck cost. The time-complexity can be reduced to  $O(n)$  if the input graph is always a tree. However, the NP-hardness property remains when we restrict the outputs to independent dominating sets. In addition, an  $O(n)$  time-optimal algorithm for the BIDS problem on weighted interval graphs has been proposed which employs the dynamic programming strategy.

In the future, the approach taken in this study can be easily applied to solve the BDS problem and the BIDS problem on other classes of graphs, such as block graphs and two-terminal series-parallel graphs. Study of the complexities of finding other types of dominating sets, e.g., perfect dominating sets and connected dominating set, with minimum bottleneck costs on weighted graphs is another important research direction, as is determining the relationships between bottleneck problems and summation problems on weighted graphs.

## REFERENCES

1. R. B. Allan and R. C. Laskar, "On domination and independent domination number of graphs," *Discrete Mathematics*, Vol. 23, 1978, pp. 73-76.
2. R. B. Allan and R. C. Laskar, "On domination and some related concepts in graph theory," in *Proceedings of 9th S-E Conference on Combinatorics, Graph Theory and Computing*, 1978, pp. 43-56.
3. M. J. Atallah, G. K. Manacher, and J. Urrutia, "Finding a maximum independent dominating set in a permutation graph," *Discrete Applied Mathematics*, Vol. 21, 1988, pp. 177-183.

4. R. E. Bellman and S. E. Dreyfus, *Applied Dynamic Programming*, Princeton University Press, Princeton, N.J., 1962.
5. A. A. Bertossi, "Dominating sets for split and bipartite graphs," *Information Processing Letters*, Vol. 23, 1984, pp. 37-40.
6. A. A. Bertossi, "Total domination in interval graphs," *Information Processing Letters*, Vol. 23, 1986, pp. 131-134.
7. A. A. Bertossi, "On the domatic number of interval graphs," *Information Processing Letters*, Vol. 28, 1988, pp. 257-280.
8. C. Bo and B. Liu, "Some inequalities about connected domination number," *Discrete Mathematics*, Vol. 159, 1996, pp. 241-245.
9. P. Buneman, "A characterization on rigid circuit graphs," *Discrete Mathematics*, Vol. 9, 1974, pp. 205-212.
10. G. Carpaneto, S. Martello, and P. Toth, "An algorithm for the bottleneck traveling salesman problem," *Operations Research*, Vol. 32, 1984, pp. 380-389.
11. G. J. Chang, "Labeling algorithms for domination problems in sun-free chordal graphs," *Discrete Applied Mathematics*, Vol. 22, 1988/89, pp. 21-34.
12. G. J. Chang, "Total domination in block graphs," *Operations Research Letters*, 1989, Vol. 8, pp. 53-57.
13. G. J. Chang and G. L. Nemhauser, "R-domination on block graphs," *Operations Research Letters*, Vol. 1, 1982, pp. 214-218.
14. G. J. Chang and G. L. Nemhauser, "The  $k$ -domination and  $k$ -stability problems on sun-free chordal graphs," *SIAM Journal on Algebraic Discrete Methods*, Vol. 5, 1984, pp. 332-345.
15. M. S. Chang, "Efficient algorithms for the domination problems on interval graphs and circular-arc graphs," *SIAM Journal on Computing*, Vol. 27, 1998, pp. 1671-1694.
16. E. J. Cockayne, "Domination of undirected graphs - A survey," in Y. Alavi and D. R. Lick, ed., *Theory and Applications of Graphs in American's Bicentennial Year*, Springer, Berlin, 1978, pp. 141-147.
17. E. J. Cockayne and S. T. Hedetniemi, "Optimal domination in graphs," *IEEE Transactions on Circuit and Systems*, CAS-22, 1975, pp. 855-957.
18. E. J. Cockayne and S. T. Hedetniemi, "Towards a theory of domination in graphs," *Networks*, Vol. 7, 1977, pp. 247-261.
19. E. J. Cockayne and S. T. Hedetniemi, "On the diagonal queens domination problem," *Journal of Combinatorial Theory Series A*, Vol. 42, 1986, pp. 137-139.
20. P. Damaschke, H. Muller, and D. Kratsch, "Domination in convex and chordal bipartite graphs," *Information Processing Letters*, Vol. 36, 1990, pp. 231-236.
21. Y. L. Daniel, "Dominations in trapezoid graphs," *Information Processing Letters*, Vol. 52, 1994, pp. 309-315.
22. M. Farber, "Independent domination in chordal graphs," *Operation Research Letters*, Vol. 1, 1982, pp. 134-138.
23. M. Farber, "Domination, independent domination and duality in strongly chordal graphs," *Discrete Applied Mathematics*, Vol. 7, 1984, pp. 115-130.
24. D. R. Fulkerson and O. A. Gross, "Incidence matrices and interval graphs," *Pacific Journal of Mathematics*, Vol. 15, 1965, pp. 835-855.
25. H. N. Gabow and R. E. Tarjan, "Algorithms for two bottleneck optimization problems," *Journal of Algorithms*, Vol. 9, 1988, pp. 411-417.

26. M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Bell Laboratories, Murray Hill, NJ, 1978.
27. F. Gavril, "The intersection graphs of subtrees in tree are exactly the chordal graphs," *Journal of Combinatorial Theory Series B*, Vol. 16, 1974, pp. 47-56.
28. M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.
29. R. Gould, *Graph Theory*, The Benjamin/Cummings Publishing Company, Inc., Menlo Park, California, 1988.
30. D. L. Grinstead, P. J. Slater, N. A. Sherwani, and N. D. Holmes, "Efficient edge domination problems in graphs," *Information Processing Letters*, Vol. 48, 1993, pp. 221-228.
31. T. W. Haynes, S. T. Hedetniemi, and P. J. Salter, *Fundamentals of Domination in Graphs*, Marcel Dekker, Inc., New York, Basel, Hong Kong, 1998.
32. T. W. Haynes, S. T. Hedetniemi, and P. J. Salter, *Domination in Graphs: Advanced Topics*, Marcel Dekker, Inc., New York, Basel, Hong Kong, 1998.
33. S. T. Hedetniemi and R. C. Laskar, "Connected domination in graphs," in B. Bollobas, editor, *Graph Theory and Combinatorics*, Academic Press, London, 1984, pp. 209-218.
34. S. T. Hedetniemi, R. C. Laskar, and J. Pfaff, "A linear algorithm for finding a minimum dominating set in a cactus," *Discrete Applied Mathematics*, Vol. 13, 1986, pp. 287-292.
35. D. S. Hochbaun and D. B. Shmoys, "A unified approach to approximation algorithms for bottleneck problems," *Journal of the ACM*, Vol. 33, 1986, pp. 533-550.
36. D. S. Hochbaun and A. Pathria, "The bottleneck graph partition problem," *Networks*, Vol. 28, 1996, pp. 221-225.
37. O. H. Ibarra and Q. Zheng, "Some efficient algorithms for permutation graphs," *Journal of Algorithms*, Vol. 16, 1994, pp. 453-469.
38. R. W. Irving, "On approximating the minimum independent dominating set," *Information Processing Letters*, Vol. 37, 1991, pp. 197-200.
39. J. K. Keil, "Total domination in interval graphs," *Information Processing Letters*, Vol. 22, 1986, pp. 171-174.
40. C. G. Kekkerkerker and J. CH. Boland, "Representation of a finite graph by a set of intervals on the real line," *Fundament Mathematicae*, Vol. 51, 1962, pp. 45-64.
41. H. Kim, "Finding a maximum independent set in a permutation graph," *Information Processing Letters*, Vol. 36, 1990, pp. 19-23.
42. Y. C. Liu and M. S. Chang, "Polynomial algorithms for various weighted perfect domination problems on some classes of graphs," Master Theses, Dept. of Computer Science and Information Engineering, National Chung Cheng University, 1993.
43. G. K. Manacher, T. A. Mankus, and C. J. Smith, "An optimum  $\Theta(n \log n)$  algorithm for finding a canonical hamiltonian path and a canonical hamiltonian circuit in a set of intervals," *Information Processing Letters*, Vol. 35, 1990, pp. 205-211.
44. K. S. Natarajan and L. J. White, "Optimum domination in weighted trees," *Information Processing Letters*, Vol. 7, 1987, pp. 261-265.
45. G. L. Nemhauser, *Introduction to Dynamic Programming*, Wiley, New York, 1966.
46. R. G. Parker and R. L. Rardin, "Guaranteed performance heuristics for the bottleneck traveling salesperson problem," *Operations Research Letters*, Vol. 2, 1984, pp.

- 269-272.
47. J. Pfaff, R. Laskar, and S. T. Hedetniemi, "Linear algorithm for independent domination and total domination in series-parallel graphs," Technical Report 441, Dept. of Mathematical Sciences, Clemson University, Clemson, SC, 1984.
  48. G. Ramalingam and C. P. Rangan, "Total domination in interval graphs revisited," *Information Processing Letters*, Vol. 27, 1988, pp. 17-21.
  49. G. Ramalingam and C. P. Rangan, "A unified approach to domination problems on interval graphs," *Information Processing Letters*, Vol. 27, 1988, pp. 217-274.
  50. A. S. Rao and C. P. Rangan, "Linear algorithm for optimal path cover problem on interval graphs," *Information Processing Letters*, Vol. 35, 1990, pp. 149-153.
  51. D. J. Rose, R. E. Tarjan, and G. S. Lueker, "Algorithmic aspects of vertex elimination on graphs," *SIAM Journal on Computing*, Vol. 5, 1976, pp. 266-283.
  52. K. White, M. Farber, and W. R. Pulleyblank, "Steiner trees, connected domination, and strongly chordal graphs," *Networks*, 1985, Vol. 15, pp. 109-124.
  53. C. C. Yen and R. C. T. Lee, "The weighted perfect domination problem," *Information Processing Letters*, 1990, Vol. 35, pp. 295-299.
  54. C. C. Yen and R. C. T. Lee, "Linear time algorithms to solve the weighted perfect domination problem in series-parallel graphs," *European Journal of Operations Research*, Vol. 73, 1994, pp. 19-26.



**William Chung-Kung Yen (顏重功)** received his B.S. degree from Department of Computer Science at Soochow University in 1988. After one-year TA (Teacher Assistant) experience in the same department, he started his research career. He received his Master and Ph.D. degree from Department of Computer Science at National Tsing Hua University in Jun. 1991 and Nov. 1996, respectively. Meanwhile, he was a technical staff of Technical Office at Data Communication Institute from Dec. 1993 to Mar. 1996. After then, from Mar. 1996 to Jul. 2000, he has ever been an associate professor of Department of Electronic Engineering at Jin-Wen College of Technology and Department of Business Administration at Minghsin Institute of Technology, respectively. He is now an associate professor of Department of Graphic Communications and Technology at Shih Hsin University since Aug. 2000. The current research interests of Dr. Yen include graph theory, network algorithms, computer networks, and data structures.