Domination and Perfect Domination under Bottleneck Cost Minimization with Extensions[1] (瓶頸成本最佳化支配與完美支配問題及其擴充)

William Chung-Kung Yen (顏重功副教授)
Department of Graphic Communications and Technology,
Shih Hsin University, Taipei, Taiwan, R.O.C.
e-mail: ckyen001@ms7.hinet.net
(世新大學平面傳播科技學系)

摘要

設 $G(V, E, W)$為一個 n 個頂點與 m 個邊之頂點具權重的圖形。任給一個 V 之子集合 H，其瓶頸成本定義為$\beta(H) = \max_{x\in H}\{W(x)\}$。本論文主要目標為找出具最小瓶頸成本之支配集與完美支配集。

最小瓶頸支配集已經被證明在一般圖形上可以於 $O(n\log n + m)$的時間求解，且如果是樹形圖還可以降低到 $O(n)$。本論文進一步證明在區塊圖形上仍能於 $O(n)$時間求解。然而，完美支配集則情況大不相同。我們將證明在二裂圖形是 NP-hard 但是樹形圖上可以有 $O(n)$時間的演算法。最後我們延伸到同時考慮瓶頸配置成本與總和通訊成本，並且證明在一般化圖形上求支配集之時間複雜度為 $O(m\log n)$。

關鍵詞:（完美）支配集、瓶頸配置成本與總和通訊成本、二裂圖形、區塊圖形、樹

Abstract

Let $G(V, E, W)$ be a graph with n-vertex-set V and m-edge-set E, where W is a cost function which maps V to real costs. For any subset H of V, the bottleneck cost of H is defined as $\beta(H) = \max_{x\in H}\{W(x)\}$. The main goal of this research is to identify a certain type of dominating set of G such that its bottleneck cost is minimized.

The problem of identifying a dominating set with the minimum bottleneck cost has been known to be $O(n\log n + m)$ time solvable on weighted general graphs; and the time-complexity can be reduced to $O(n)$ on trees. This paper first shows that the time-complexity remains $O(n)$ on weighted block graphs. Second, we show that the situation is greatly different when the Bottleneck Perfect Dominating Set Problem (the BPDS problem) is considered. This paper claims a very meaningful algorithmic result: the BPDS problem is NP-hard on bipartite graphs but $O(n)$ time solvable on weighted trees. Finally, we extend the result of the BDS problem to consider the sum communication costs simultaneously and the time-complexity is $O(m\log n)$ on weighted general graphs

Keywords: (perfect) dominating set, bottleneck placement cost and sum communication cost, bipartite graph, block graph, tree

1. Introduction

In this paper, $G(V, E, W)$ denotes a graph with n-vertex-set V and m-edge-set E in which W represents a cost function mapping V to real costs. Due to the rapid growth of the application areas of domination in the past thirty years, many researches have been focused on studying domination and its related problems on graphs. These application areas include computer science, electrical and computer engineering, telecommunication and network design, and operations research, etc. [13, 14] Let (u, v) be an edge of G, we say that *u dominates v*, and vice versa. For any subset H of V, H is called a *dominating set* of G iff each vertex in $V – H$ is dominated by at least one vertex in H. Some practical variants of dominating sets have been proposed and studied. H is called a *total dominating set* iff H contains no isolated vertex [3, 5, 18, 22, 23]. H is called a *connected dominating set* iff the subgraph induced by H is connected [4, 15, 24]. H is called an *independent*

[1] This research was supported by National Science Council, Taiwan, ROC, under contract number NSC-89-2213-E-159-001

dominating set iff H forms an independent set, that is, $(u, v) \notin E$, for all $u, v \in H$ [7-9, 17, 19, 28-30]. Finally, H is called a *perfect dominating set* iff each vertex $u \in V - H$ is dominated by only one vertex in H [11, 20, 25, 26].

Traditionally, researchers emphasize on finding a certain type of dominating set H such that its sum cost, $\sum_{x \in H} W(x)$, is minimized. This paper addresses another important cost measurement, the bottleneck cost. For any subset H of V, *bottleneck cost* of H is defined as $\beta(H) = \max_{x \in H}\{W(x)\}$. The goal of this research is to find a certain type of dominating set such that its bottleneck cost is minimized.

The problems studied in this paper are defined below.

The Bottleneck Dominating Set problem (***The BDS problem***): Given a weighted graph $G(V, E, W)$, find a dominating set H of G such that $\beta(H)$ is minimized. H denotes an optimal solution of G.

The Bottleneck Perfect Dominating Set problem (***The BPDS problem***): Given a weighted graph $G(V, E, W)$, find a perfect dominating set R of G such that $\beta(R)$ is minimized. R denotes an optimal solution of G.

Fig. 1 illustrates a graph with real costs on vertices. Examples of optimal solutions for the problems studied in this paper 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 BPDS problem, the sets $\{b, e, f\}$ and $\{a, f\}$ are two perfect dominating sets. The set $\{a, f\}$ is a perfect dominating set with the minimum bottleneck cost which is equal to 3.

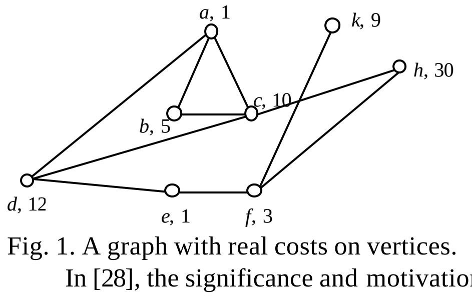

Fig. 1. A graph with real costs on vertices.

In [28], the significance and motivations of finding various dominating sets with minimum bottleneck costs have been proposed. We have proven that the BDS problem is $O(n\log n + m)$ time solvable on weighted general graphs and $O(n)$ time solvable on weighted trees. This paper will first show that the time-complexity of the BDS problem remains $O(n)$ on weighted block graphs. Second, the situation will be shown greatly different when the BPDS problem is considered. We will claim a very worthy algorithmic results on graphs: the BPDS problem is NP-hard on bipartite graphs but $O(n)$ time solvable on weighted trees. Finally, we extend the result of the BDS problem to consider the sum communication costs simultaneously and the time-complexity is $O(m\log n)$ time on weighted general graphs

2. An $O(n)$ Time Algorithm for the BDS Problem on Weighted Block Graphs

This section will design an $O(n)$ time algorithm for the BDS problem on weighted block graphs. Given a graph $G(V, E)$, a *cut vertex* is a vertex v such that deleting v and all its incident edges can increase the number of connected components. A block of G is a maximal connected induced subgraph containing no cut vertex. In [12], Harary have defined a block graph BG(G) of a graph G as the intersection graph of the blocks of G. Let $\text{BK}_1, ..., \text{BK}_h$ be the blocks of G. The block graph of G, denoted by $\text{BG}(G) = (V_{\text{BG}}, E_{\text{BG}})$, is defined as follows: $V_{\text{BG}} = \{\text{BK}_1, ..., \text{Bk}_h\}$, and $E_{\text{BG}} = \{(\text{BK}_i, \text{Bk}_j) \mid i \neq j \text{ and } \text{BK}_i \cap \text{BK}_j \neq \emptyset\}$.

Harary has also proven that a graph is the block graph of some graph iff all of whose blocks are complete graphs. Thus, researchers focused on the type of block graphs whose blocks are all complete graphs. Fig. 2 shows a block graph of this type. The graph has four blocks and $\{d, g\}$ are the cut vertices.

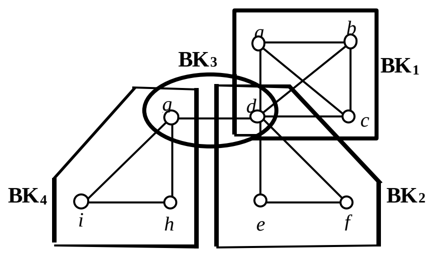

Fig. 2. A block graph with four blocks.

A very useful tree-like decomposition structure, called a block-tree, will be used to represent a block graph hereafter. The block-tree is refined from the cut-tree structure proposed by Aho [1]. The *cut-tree*, denoted by $T^*(V^*, E^*)$, of a block graph $\text{BG}(V, E)$ with h blocks $\text{BK}_1, ..., \text{BK}_h$ and p cut vertices $v_1, ..., v_p$ is defined as follows:

$V^* = \{\text{BK}_1, ..., \text{BK}_h, v_1, ..., v_p\}$, and $E^* = \{(\text{BK}_i, v_j) \mid v_j \in \text{BK}_i, 1 \leq i \leq h, 1 \leq j \leq p\}$.

The cut-tree of the block graph in Fig. 2 is illustrated in Fig. 3.

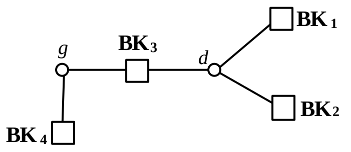

Fig. 3. The cut-tree of the block graph in Fig. 2.

In [1], Aho has shown that the cut-tree of a block graph can be constructed in linear-time by the depth-first-search. For any block BK_i, define $B_i = \{v \in \mathrm{BK}_i \mid v \text{ is not a cut vertex}\}$, $1 \le i \le h$. For example, in Fig. 2, B_1 represents the set of the vertices $\{a, b, c\}$. Note that B_i can be empty under this definition. Let us take Fig. 2 as the example again. B_3 is empty because the original block BK_3 corresponds to the set of vertices, $\{d, g\}$, which are all cut vertices. The *block-tree* $T^{\mathrm{BG}}(V^{\mathrm{BG}}, E^{\mathrm{BG}})$ of a block graph $\mathrm{BG}(V, E)$ can now be defined as follows, where B_i, $1 \le i \le h$, will be called *block vertices* hereafter. $V^{\mathrm{BG}} = \{B_1, ..., B_h, v_1, ..., v_p\}$, and $E^{\mathrm{BG}} = \{(B_i, v_j) \mid v_j \in \mathrm{BK}_i, 1 \le i \le h, 1 \le j \le p\}$.

The block-tree of the block graph shown in Fig. 2 is depicted in Fig. 4. Some papers have given efficient algorithms for solving problems on block graphs [5, 6, 16, 27]. This section will solve the BDS problem on weighted block graphs by the dynamic programming strategy [2, 21].

Definition 1: Any vertex v is called a *black vertex* if v must be included by any optimal solution, and v is called a *white vertex* if v must be excluded from any optimal solution.

Definition 2: For any nonempty block vertex B, $B^{\min}$ denotes any vertex in B such that $W(B^{\min}) = \min_{v \in B}\{W(v)\}$, i.e., $B^{\min}$ is a vertex in B with the minimum cost.

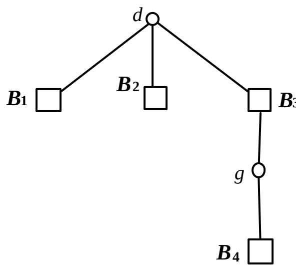

Fig. 4. The block-tree of the block graph in Fig. 2.

For any block graph BG, if BG only consists of a non-empty block vertex B, then BG has no cut vertex according to the definition of block graphs. In this situation, it is clear that $\{B^{\min}\}$ is a dominating set with the minimum bottleneck cost since all vertices in B form a clique.

In the rest of this section, the input of the BDS problem will be the block-tree of the original block graph. It is reasonable to further assume that at least one cut vertex exist based upon the above discussion.

Given a block graph BG and its corresponding block-tree T, suppose that r is any cut vertex as shown in Fig. 5. The block-tree will be denoted by $T(r)$. In Fig. 5, each C_i represents the clique $\{x_{i_1}, ..., x_{i_{j_i}}\} \cup B_i$, $1 \le i \le k$. It is clear that any optimal solution either includes r or does not. So, the following two new related problems, $(\mathrm{P}_{\bar{r}})$ and (P_r) are introduced.

$(\mathrm{P}_{\bar{r}})$ Find a dominating set H of BG with the minimum bottleneck cost under the additional constraint that $r \notin H$.

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

Let $\beta_{\bar{r}}(\mathrm{BG})$ and $\beta_r(\mathrm{BG})$ denote the bottleneck cost of all optimal solutions of Problem $(\mathrm{P}_{\bar{r}})$ and Problem (P_r) on BG, respectively. The bottleneck cost of all optimal solutions of the BDS problem, denoted by $\beta(\mathrm{BG})$, can be easily derived using the following formula.

$$\beta(\mathrm{BG}) = \min\{\beta_{\bar{r}}(\mathrm{BG}), \beta_r(\mathrm{BG})\} \quad \text{—(2.1)}$$

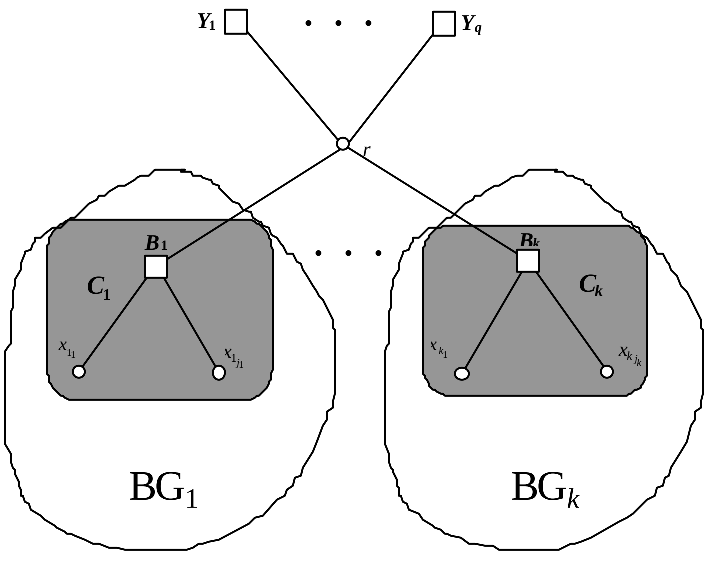

Fig. 5. A block-tree $T(r)$, where the square vertices represent block vertices and $Y_1, \ldots, Y_q$ are nonempty block vertices if exist.

Formula (2.1) indicates that $\beta(\mathrm{BG})$ can be computed in $O(1)$ time if $\beta_{\bar{r}}(\mathrm{BG})$ and $\beta_r(\mathrm{BG})$ have been available.

First, consider the boundary case in which $T(r)$ only consists of cut vertex r and its children, say $B_1, ..., B_k$, $k \geq 2$. Let H be any optimal solution. If $r \in H$, then all vertices in $B_1 \cup ... \cup B_k$ are dominated by r and $\beta(BG) \geq W(r)$. It is easy to verify that including any vertex in $B_1 \cup ... \cup B_k$ will obtain 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 we can easily verify that $\{B_1^{\min}, ..., B_k^{\min}\}$ is a dominating set with the minimum bottleneck cost of Problem ($P_{\bar{r}}$).

The correctness of the following formulas for computing $\beta_{\bar{r}}(BG)$, $\beta_r(BG)$, and $\beta(BG)$ under this boundary condition can be easily proved.

$$\beta_{\bar{r}}(BG) = \max_{1 \leq i \leq k}\{W(B_k^{\min})\} \text{ —(2.2)}$$

$$\beta_r(BG) = W(r) \text{ —(2.3)}$$

$$\beta(BG) = \min\{\beta_{\bar{r}}(BG), \beta_r(BG)\} \text{ —(2.4)}$$

Consider a general block-tree $T(r)$ as shown in Fig. 5. Let Ω denote the set $\{Y_1, \ldots Y_q\}$. To solve Problem ($P_{\bar{r}}$), the following two cases should be handled.

Case 1. Ω is not empty

Since Problem ($P_{\bar{r}}$) deals with the case in which all feasible solution can not include r, at least one black vertex must be included from each block vertex Y_i, $1 \leq i \leq q$, respectively, according to the definition of dominating sets. Then, r is dominated by the black vertices in $Y_1 \cup ... \cup Y_q$. This implies that we can solve the BDS problem on $BG_1, \ldots, BG_k$ independently and recursively without considering r. Let $\beta^{(I)}(BG)$ denote the minimum bottleneck cost of BG in this case. The following formula can be easily established.

$$\beta^{(I)}(BG) = \max\{\max_{B \in \Omega}\{W(B^{\min})\}, \beta(BG_1), \ldots, \beta(BG_k)\} \text{ —(2.5)}$$

Case 2. Ω is empty

There must exist a black vertex $v_i \in C_i$, for some i, in order to dominate the vertex r. Let $\beta^{(II)}(BG)$ denote the minimum bottleneck cost of BG in this case and $\beta_{v_i}(BG_i)$ represent the minimum bottleneck cost of BG_i under the condition that some vertex v_i must be black. Verifying the correctness of the following formula is simple.

$$\beta^{(II)}(BG) = \max\{\Pi_1, \ldots, \Pi_k\},$$

where $\Pi_i = \max\{\beta(BG_1), \ldots, \beta(BG_{i-1}), \beta_{v_i}(BG_i), \beta(BG_{i+1}), \ldots, \beta(BG_k)\}$

—(2.6)

Based upon the above two cases, $\beta_{\bar{r}}(BG)$ can be easily computed as follows:

$$\beta_{\bar{r}}(BG) = \begin{cases} \beta_{\bar{r}}^{(I)}(BG), \Omega \neq \varnothing \\ \beta_r^{(II)}(BG), \Omega = \varnothing \end{cases} \text{ —(2.7)}$$

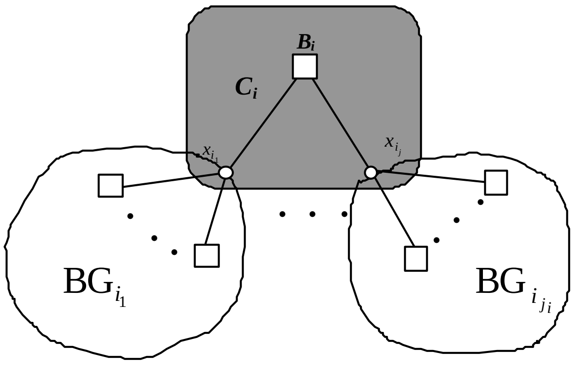

Fig. 6. The block-tree corresponding to block graph BG_i shown in Fig. 5.

The remaining problem is to compute $\beta_{v_i}(BG_i)$, $v \in C_i$, for all i. Consider the block-tree corresponding to BG_i shown in Fig. 6. Two situations can occur.

Case 1. $v_i \in B_i$

It is clear that B_i is not empty in this case and v_i can be assigned as $B_i^{\min}$. Let $\beta_{v_i}^{(1)}(BG_i)$ denote the minimum bottleneck cost of BG_i here. The following formula can be easily derived.

$$\beta_{v_i}^{(1)}(BG_i) = \max\{W(B_i^{\min}), \beta(BG_{i_1}), \ldots, \beta(BG_{i_{j_i}})\} \text{ —(2.8)}$$

Case 2. $v_i \in \{x_{i_1}, \ldots, x_{i_{j_i}}\}$

All vertices in C_i are dominated by v_i. Let $\beta_{v_i}^{(2)}(BG_i)$ denote the minimum bottleneck cost of BG_i in this case. The following formula can be easily derived.

$\beta_{v_i}{}^{(2)}(\mathrm{BG}_i) = \max\{\Psi_1, \ldots, \Psi_{j_i}\}$,

where $\Psi_s = \max\{\beta(\mathrm{BG}_{i_1}), \ldots, \beta(\mathrm{BG}_{i_{s-1}}), \beta_{x_s}(\mathrm{BG}_{i_s}), \beta(\mathrm{BG}_{i_{s+1}}), \ldots, \beta(\mathrm{BG}_{i_{j_i}})\}$ —(2.9)

Using the reasoning of the above two cases, $\beta_{v_i}(\mathrm{BG}_i)$ can be easily computed by the following formula.

$$\beta_{v_i}(\mathrm{BG}_i) = \begin{cases} \beta_{v_i}{}^{(1)}(\mathrm{BG}_i), v_i \in B_i \\ \beta_{v_i}{}^{(2)}(\mathrm{BG}_i), v_i \in \{x_{i_1}, \ldots, x_{i_{j_i}}\} \end{cases} \text{—(2.10)}$$

Next, we will deal with Problem (P_r). If H is an optimal solution of Problem (P_r), then $r \in H$ and $\beta_r(\mathrm{BG}) \geq W(r)$. If Ω is not empty, then it is easy to show that including any vertex in $Y_1 \cup \ldots \cup Y_q$ will obtain to a solution with a bottleneck cost greater than or equal to $W(r)$. Verifying the correctness of the following formula is an easy task when the optimal solutions of the BDS problem on the subgraphs BG_i, $1 \leq i \leq k$, have been computed.

$$\beta_r(\mathrm{BG}) = \max\{W(r), \beta(\mathrm{BG}_1), \ldots, \beta(\mathrm{BG}_k)\} \text{—(2.11)}$$

In the following, we will prove that the time-complexity for implementing the above formulas is $O(n)$. It is trivial to see that formulas (2.1) to (2.11) can be performed in either $O(1)$ time or linear-time, except Formula (2.6) and Formula (2.9). The following lemmas will prove that Formula (2.6) and Formula (2.9) can also be implemented in linear-time.

Lemma 1: If the values $\beta_{x_s}(\mathrm{BG}_{i_s})$ and $\beta(\mathrm{BG}_{i_s})$, $s = 1, \ldots, j_i$, have been computed, then $\beta_{v_i}{}^{(2)}(\mathrm{BG}_i)$ can be obtained in $O(j_i)$ time, i.e., Formula (2.9) can be done in $O(j_i)$ time.

***Proof*:** First, we can compute the largest value, *maxval*, and the second largest value, *secval*, of the j_i values $\beta(\mathrm{BG}_{i_s})$, $s = 1, \ldots, j_i$, in $O(j_i)$ time.

Next, the value $\beta_{v_i}{}^{(2)}(\mathrm{BG}_i)$ can be derived using a loop in $O(j_i)$ time as follows:

```
$\beta_{v_i}{}^{(2)}(\mathrm{BG}_i)$ = maxval; /* initialization */
**loop** $s = 1$ to $j_i$
    maxval$_s$ = maxval;
    if $\beta(\mathrm{BG}_{i_s})$ = maxval then
        if $\beta_{x_s}(\mathrm{BG}_{i_s}) \geq$ secval then
            maxval$_s$ = $\beta_{x_s}(\mathrm{BG}_{i_s})$;
            /* $\beta_{x_s}(\mathrm{BG}_{i_s})$ is the new maximum in $\{\beta(\mathrm{BG}_{i_1}), \ldots, \beta(\mathrm{BG}_{i_{s-1}}), \beta_{x_s}(\mathrm{BG}_{i_s}), \beta(\mathrm{BG}_{i_{s+1}}), \ldots, \beta(\mathrm{BG}_{i_{j_i}})\}$. */
        else
            maxval$_s$ = secval;
            /* $\beta_{x_s}(\mathrm{BG}_{i_s})$ is the second largest in $\{\beta(\mathrm{BG}_{i_1}), \ldots, \beta(\mathrm{BG}_{i_{s-1}}), \beta_{x_s}(\mathrm{BG}_{i_s}), \beta(\mathrm{BG}_{i_{s+1}}), \ldots, \beta(\mathrm{BG}_{i_{j_i}})\}$, so $\Psi_s$ = the second largest in $\{\beta(\mathrm{BG}_{i_1}), \ldots, \beta(\mathrm{BG}_{i_{s-1}}), \beta_{x_s}(\mathrm{BG}_{i_s}), \beta(\mathrm{BG}_{i_{s+1}}), \ldots, \beta(\mathrm{BG}_{i_{j_i}})\}$. */
        endif
    else /* $\beta(\mathrm{BG}_{i_s}) \leq$ maxval$_s$;
        if $\beta_{x_s}(\mathrm{BG}_{i_s}) \geq$ maxval then
            maxval$_s$ = $\beta_{x_s}(\mathrm{BG}_{i_s})$;
    endif

    /* Indeed, maxval$_s$ = $\Psi_s$ in Formula (2.9) */
```

```
        $\beta_{v_i}^{(2)}(BG_i) = \min\{\beta_{v_i}^{(2)}(BG_i), maxval_s\}$;
    **endloop**.
```

Lemma 2: Formula (2.6) can be implemented in $O(\sum_{i=1}^{k}|C_i| + k)$ time.

***Proof*:** Using the similar technique as Lemma 1, $\beta^{(II)}(BG)$ can be computed in $O(k)$ time when the values $\beta(BG_i)$ and $\beta_{v_i}(BG_i)$, $i = 1, ..., k$, have been computed. Based upon the result of Lemma 1 and the facts that Formula (2.8) and Formula (2.10) can be done in $O(j_i + 1)$ time and $O(1)$ time, respectively. We can easily conclude that $\beta^{(II)}(BG)$ can be computed in $O(\sum_{i=1}^{k}|C_i| + k)$ time.

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

Theorem 1: The BDS problem can be solved in $O(n)$ time on weighted block graphs.

3. NP-hardness of the BPDS Problem on Bipartite Graphs

To examine the complexity of the BPDS problem, its corresponding decision problem is considered.

***The Bottleneck Perfect Dominating Set decision problem* (*The BPDS decision problem*):** Given an undirected and connected graph $G(V, E, W)$ and a real constant η, determine whether a perfect dominating set $R \subseteq V$ exists such that its bottleneck cost is less than or equal to η.

Next, a variant of the BPDS decision problem is introduced.

***The Constrained Perfect Dominating Set decision problem* (*The CPDS decision problem*):** Given an undirected and connected graph $G(V, E)$ and a set of vertices $V' \subseteq V$, determine whether there exists a perfect dominating set R of G such that $R \subseteq (V - V')$.

The following lemma can then be directly derived.

Lemma 3: The BPDS decision problem is polynomially equivalent to the CPDS decision problem.

This section will consider the complexity about the BPDS problem on *bipartite graphs* [29]. A graph $G(V, E)$ is called a *bipartite graph* if V can be partitioned into two disjoint sets I and J such that I and J are both independent sets, that is, there is no edge (u, v) such that $u, v \in I$ or $u, v \in J$. We denote a bipartite graph with $V = I \cup J$ by $G(I \cup J, E)$ hereafter. Fig. 7 depicts an instance of bipartite graphs.

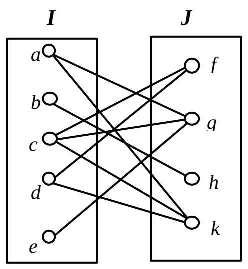

Fig. 7. A bipartite graph $G(I \cup J, E)$.

Showing that the BPDS decision problem is NP-complete is equivalent to proving that the CPDS decision problem is NP-complete. A known NP-complete problem is introduced and Lemma 4 can be established.

<u>The Exact Cover problem</u> [10] : Given a family of $F = \{S_1, ..., S_p\}$ of sets where each S_i is a subset of a set $X = \{x_1, ..., x_q\}$, does there exist a subfamily of pairwise disjoint sets of F whose union is equal to X?

Lemma 4: The CPDS decision problem is NP-complete on bipartite graphs.

***Proof*:** It is easy to check that the CPDS decision problem belongs to the NP class of problems. We now show that the Exact Cover problem can be reduced to the CPDS decision problem on bipartite graphs in polynomial time. Given an instance of the Exact Cover problem in which $F = \{S_1, ..., S_p\}$ and $X = \{x_1, ..., x_q\}$, a bipartite graph $G(I \cup J, E)$ is constructed by the following transformation rules. $I = F \cup \Pi$, $J = X \cup \Phi$ and $E = \{(x_z, S_t) \mid x_z \text{ belongs to } S_t\} \cup E^*$, where Π, Ω, and E^* are derived using the following loop.

$$\Pi = \Omega = \varnothing;\ E^* = \varnothing;\ i = j = 0;$$

```
for each pair $S_z$ and $S_t$ in $F$ do
{
  if $S_z \cap S_t \neq \emptyset$ then
    {
      $i = i + 1; \Pi = \Pi \cup \{u_i\}$;
      $j = j + 2; \Phi = \Phi \cup \{y_{j-1}, y_j\}$;
      $E^* = E^* \cup \{(S_z, y_{j-1}), (y_j, S_t), (y_{j-1}, u_i), (y_j, u_i)\}$
  }
}
endfor
```

Assume that $\Pi = \{u_1, ..., u_\beta\}$ and $\Phi = \{y_1, ..., y_\lambda\}$ after performing the above procedure. Some useful properties of G can be easily derived.

Property 1: $0 \leq \beta \leq \binom{p}{2}$, $\lambda = 2\beta$, *and* $|E^*| = 4\beta$.

Property 2: *Each* u_i *is adjacent to exactly two vertices* $y_{(2i-1)}$ *and* $y_{(2i)}$ *in* Φ.

It is easy to ascertain that the time-complexity of the transformation procedure is polynomial based on the above properties.

Let $V¢ = X$. Then, $(V - V¢)$ corresponds to the set $F \cup \Phi \cup \Pi$. The goal of the CPDS decision problem is to determine whether there exists a perfect dominating set D of G such that $D \subseteq (V - V¢) = F \cup \Phi \cup \Pi$.

For any vertex v of G, w is said to be a *neighbor* of v iff $(v, w) \in E$ and define $N(v) = \{w \mid (v, w) \in E\}$. Suppose that K is a solution of the Exact Cover problem. Without a loss of generality, we can assume that $K = \{S_1, ..., S_\alpha\}$, $1 \leq \alpha \leq p$. Then, $S_z \cap S_t = \emptyset$, for all S_z and S_t in K. It implies that each x_h in X will be dominated by exactly one vertex in K. The situation can be illustrated in Fig. 8. The task left is to determine which vertices in $(F - K) \cup \Phi \cup \Pi$. could be selected to obtain a perfect dominating set $D \subseteq F \cup \Phi \cup \Pi$.

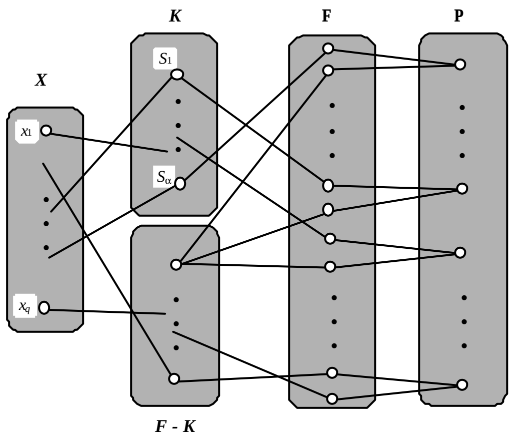

Fig. 8. A bipartite graph constructed from an instance of the Exact Cover Problem and K is a solution of it.

For each $S \in F - K$, there must exist $S^+ \in F - K$ such that $S \cap S^+ \neq \emptyset$. This implies that there exists $x_h \in X$ such that $(S, x_h) \in E$ and $(S^+, x_h) \in E$. Therefore, all vertices in $F - K$ can not be included into any perfect dominating set $D \subseteq F \cup \Phi \cup \Pi$.

The following loop will determine which vertices in $\Phi \cup \Pi$ could be added to obtain a solution of the CPDS decision problem. Its correctness can be easily verified based on the reasoning so far as well as the comments within it.

```
$D = K$;
$Q = F - K$; $H = \Phi$; $R = \Pi$; /* initialization */
for each $S \in Q$ do
{
    if only one $S^+ \in K$ exists such that $(S, y_j)$ — $(y_j, u_i)$ — $(u_i, y_{j-1})$ —
    $(y_{j-1}, S^+)$ is a path of $G$, for some $y_j$, $y_{j-1}$, and $u_i$, then
    {
        $D = D \cup \{y_j\}$;
        /* $y_{j-1}$ is only dominated by $S^+$; $u_i$ and $S$ is only dominated by $y_j$. */
        $H = H - \{y_{j-1}, y_j\}$; $R = R - \{u_i\}$;
    }
    else
    {
        Let $\{S^{(1)}, ..., S^{(\varepsilon)}\}$ be the vertices in $K$ such that $(S, y^{(t)}_j)$ — $(y^{(t)}_j, u^{(t)}_i)$ — $(u^{(t)}_i,$
        $y^{(t)}_{j-1})$ — $(y^{(t)}_{j-1}, S^{(t)})$ be a path of $G$, $\varepsilon > 1$ and $1 \leq t \leq \varepsilon$;
        $D = D \cup \{y^{(1)}_j\}$;
        /* $y^{(1)}_{j-1}$ is only dominated by $S^{(1)}$ and $u^{(1)}_i$ is only dominated by $y^{(1)}_j$. */
        $H = H - \{y^{(1)}_{j-1}, y^{(1)}_j\}$;          $R = R - \{u^{(1)}_i\}$;
        for $z$ = 2 to $\varepsilon$ do
```

```
                {
                        $D = D \cup \{y^{(z)}_{j-1}, u^{(z)}_i\}$; $H = H - \{y^{(z)}_{j-1}, y^{(z)}_j\}$; $R = R - \{u^{(z)}_i\}$;
                }
                endfor
        }
}
endfor
$D = D \cup R$; /* The remaining vertices in $\Pi$ are all included. */
```

Next, we prove another direction. If D is a perfect dominating set $D \subseteq (V - V')= F \cup \Phi \cup \Pi$, then each x_h in X must be dominated by exactly one vertex in D, denoted by $x^{(D)}_h$. It is easily checkable that $x^{(D)}_h \in F$, for all h, and either $x^{(D)}_z \cap x^{(D)}_s = \emptyset$ or $x^{(D)}_z = x^{(D)}_s$, for all $z \neq s$. Therefore, $K = \cup_{1 \le h \le q}\{x^{(D)}_h\}$ is a solution of the Exact Cover Problem.

From the discussions so far, a perfect dominating set $D \subseteq (V - V')$ exists in the bipartite graph G iff there exists a subfamily of pairwise disjoint sets of F such that its union is equal to X. This implies that the CPDS decision problem is NP-complete on bipartite graphs.

Theorem 2: The BPDS problem is NP-hard on bipartite graphs.

4. An $O(n)$ Time Algorithm for the BPDS Problem on Weighted Trees

Given a tree T and any vertex r, the tree is denoted by $T(r)$. For any perfect dominating set R of $T(r)$, $\beta(R)$ is the value of 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, denote $\delta(r)$ to be the value of the minimum bottleneck cost of all optimal solutions of the BPDS problem on $T(r)$, i.e., $\delta(r) = \min\{\beta(R) \mid R$ is a perfect dominating set of $T(r)\}$.

Given a tree $T(r)$, any optimal solution either includes the root r or not. This leads us to introduce the following two new related problems, $(P_{r'})$ and (P_r), which are in fact the original BPDS problem with additional constraints.

$(P_{r'})$ Compute $\delta_0(r) = \min\{\beta(R) \mid r \notin R$ and R is a perfect dominating set of $T(r)\}$.
(P_r) Compute $\delta_1(r) = \min\{\beta(R) \mid r \in R$ and R is a perfect dominating set of $T(r)\}$.

From the definitions of the BPDS problem and the problems $(P_{r'})$ and (P_r), the following formula can be easily obtained.

$$\delta(r) = \min\{\delta_0(r), \delta_1(r)\} \text{ —(4.1)}$$

Formula (4.1) implies that the bottleneck cost of all optimal solutions can be obtained in $O(1)$ time when the problems $(P_{r'})$ and (P_r) have been solved. Therefore, the followings will concentrate on solving these two problems.

First, consider the boundary case when $T(r)$ only consists of vertex r. In this situation, $\{r\}$ is the only one perfect dominating set. Therefore, the bottleneck costs of the problems $(P_{r'})$, (P_r), and the original problem in this boundary condition are as follows: $\delta^{\text{bndy-I}}_0(r) = \infty$, $\delta^{\text{bndy-I}}(r) = \delta^{\text{bndy-I}}_1(r) = W(r)$.

Another boundary case is the situation that $T(r)$ only consists of vertex r and its children, say $x_1, \ldots, x_k$. If $k = 1$, $\{r\}$, $\{x_1\}$, and $\{r, x_1\}$ are the all perfect dominating sets of $T(r)$. Otherwise, the only perfect dominating set is $\{r\}$. It is easy to verify the correctness of the following formulas.

$$\delta^{\text{bndy-II}}_0(r) = \begin{cases} W(x_1), k = 1 \\ \infty, k \ge 2 \end{cases} \text{ —(4.2)}$$

$$\delta^{\text{bndy-II}}_1(r) = W(r) \text{ —(4.3)}$$

$$\delta^{\text{bndy-II}}(r) = \min\{\delta^{\text{bndy-II}}_0(r), \delta^{\text{bndy-II}}_1(r)\} \text{ —(4.4)}$$

Now, consider a general tree $T(r)$ as shown in Fig. 9. The problem $(P_{r'})$ considers the cases in which r must be excluded. From the definition of perfect dominating sets, any feasible solution of the problem $(P_{r'})$ must contain exactly one vertex belonging to $\{x_1, \ldots, x_k\}$. This can be expressed as the following formula.

$$\delta_0(r) = \min\{\max\{\mathbf{d_1(x_1)}, \delta_0(x_2), \ldots, \delta_0(x_k)\}, \max\{\delta_0(x_1), \mathbf{d_1(x_2)}, \ldots, \delta_0(x_k)\}, \ldots, \max\{\delta_0(x_1), \delta_0(x_2), \ldots, \mathbf{d_1(x_k)}\}\} \text{ —(4.5)}$$

Formula (4.5) indicates that the cases where x_i, $1 \le i \le k$, is the only one vertex that must be included are dealt with, respectively.

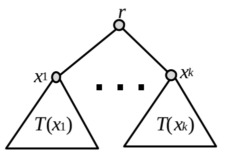

Fig. 9. A tree $T(r)$ and the subtrees of r.

Next, turn to deal with the problem (P_r). If R is a solution of the problem (P_r), then $r \in R$. The bottleneck cost of D will be equal to or greater than $W(r)$. Now, for each x_i in $\{x_1, ..., x_k\}$, it can be either included or excluded. Suppose that $T(x_{i_1})$, ..., $T(x_{i_{j_i}})$ be the subtrees of vertex x_i as illustrated in Fig. 10. If x_i is excluded, i.e., x_i has been dominated by r, then all the vertices in $\{x_{i_1}, ..., x_{i_{j_i}}\}$ must also be excluded since each vertex not in R can be dominated by only one vertex in R. In another, if x_i is included, then every vertex in $\{x_{i_1}, ..., x_{i_{j_i}}\}$ can be either included or excluded. Now, the following formula can be easily obtained when the problem ($P_{r'}$) on the subtrees $T(x_{i_p})$, $1 \le i \le k$ and $1 \le p \le j_i$, and the problem (P_r) on $T(x_i)$, $1 \le i \le k$, have been solved.

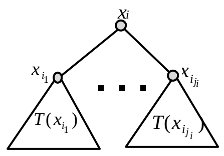

Fig. 10. The subtrees of vertex x_i, $1 \le i \le k$.

$\delta_1(r) = \max\{W(r), \min_{1 \le i \le k}\{\max\{\Phi_i, \vartheta_i\}\}\}$ —(4.6)

, where $\Phi_i = \max\{W(x_i), \max\{\delta(x_{i_1}), \delta(x_{i_2}), ..., \delta(x_{i_{j_i}})\}\}$, and $\vartheta_i = \max\{\delta_0(x_{i_1}), \delta_0(x_{i_2}), ..., \delta_0(x_{i_{j_i}})\}$.

The following will prove that the time-complexity for implementing the above formulas is $O(n)$. First, the following lemmas are established.

Lemma 5: If the $2k$ values $\delta_1(x_i)$ and $\delta_0(x_i)$, $i = 1, ..., k$, have been computed, then $\delta_0(r)$ can be obtained in $O(k)$ time for any internal vertex r with k children $x_1, ..., x_k$.

***Proof*:** First, we compute the largest value, *maxval*, and the second largest value, *secval*, of the k values $\delta_0(x_i)$, $i = 1, ..., k$, in $O(k)$ time.

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

```
$\delta_0(r)$ = maxval; /* initialization */
**loop** $i$ = 1 to $k$
        maxval$_i$ = maxval;
        if $\delta_0(x_i)$ = maxval then
                if $\delta_1(x_i) \ge$ secval then
                mxaval$_i$ = secval;
                else
                if $\delta_1(x_i) \ge$ maxval then
                        maxval$_i = \delta_1(x_i)$;
                $\delta_0(r) = \min\{\delta_0(r)$, maxval$_i\}$;
**endloop**.
```

Lemma 6: For any internal vertex r, $\delta_1(r)$ can be computed in $O(\sum_{i=1}^{k}(j_i+1) + 1)$ time when the values $\delta_0(x_{i_p})$, $1 \le i \le k$, $1 \le p \le j_i$, and $\delta_1(x_i)$, $1 \le i \le k$, are available.

***Proof*:** This lemma can be directly derived from Formula (4.6).

From Lemma 5, Lemma 6, 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 conclude that the number of operations executed on each vertex is bounded by a constant.

Finally, an optimal solution can be derived from the root of the input tree in $O(n)$ time since for the set $\{r, x_1, ..., x_k\}$, one of the following cases could occur.

Case 1. r is selected. For each subtree $T(x_i)$, if $\max\{W(x_i), \max\{\delta(x_{i_1}), \delta(x_{i_2}), ..., \delta(x_{i_{j_i}})\}\} < \max\{\delta_0(x_{i_1}), \delta_0(x_{i_2}), ..., \delta_0(x_{i_{j_i}})\}$, then x_i is included, else x_i can be excluded.

Case 2. r is not selected. Only one vertex in $\{x_1, ..., x_k\}$ must be included.

Based on the above discussion, an optimal solution can be identified by examining each vertex constant times from the root r after $\delta(r)$ has been computed, and its time- complexity is also $O(n)$.

Theorem 3: The BPDS problem can be solved in $O(n)$ time on weighted trees.

5. Extension to Consider Sum Communication Costs Simultaneously

In addition to the costs of allocating service facilities, the communication costs between the vertices with no facility and the vertices with facilities are also a key factor related to the costs and performance of real systems. This section will extend the results of the BDS problem for dealing with the two cost measurements simultaneously.

For any dominating set D, the *bottleneck-placement-cost* of D is defined to be $BP(D) = \max_{x \in D}\{C(x)\}$ and its *sum-*

communication-cost is defined to be SC(*D*) = $\sum_{x \in V-D} W(x, ì\,(x))$, where μ(*x*) is a vertex *y* in which *W*(*x*, *y*) is minimized among all vertices dominating *x*. The pair (BP(*D*), SC(*D*)) is called the *BS-dominating-cost* of *D*. The goal is to identify a dominating set *D* such that BP(*D*) is minimized. Furthermore, if more than one dominating set whose bottleneck-placement-costs are minimized, then any one with the minimum sum-communication-cost will be selected.

Now, the following definitions are first made.

Definition 3: Given any two pairs of real numbers (a_1, b_1) and (a_2, b_2), the following relations are defined.
1. (a_1, b_1) < (a_2, b_2) if and only if (iff) ($a_1 < a_2$) or ($a_1 = a_2$ and $b_1 < b_2$).
2. (a_1, b_1) = (a_2, b_2) iff ($a_1 = a_2$ and $b_1 = b_2$).

Definition 4: Let (a_1, b_1), ..., (a_n, b_n) be any *n* pairs of real numbers.
1. The pair (a_i, b_i) is a *minimum pair* among these *n* pairs iff (a_i, b_i) ≤ (a_j, b_j), for all $j \neq i$.
2. The pair (a_t, b_t) is a *maximum pair* among these *n* pairs iff (a_t, b_t) ≥ (a_s, b_s), for all $s \neq t$.

The extended problem in this section is now described formally as follows:

The Minimum BS-dominating-cost Dominating Set problem (***The MBS_DS problem***): Find a dominating set *S* ⊆ *V* such that its BS-dominating-cost is minimized for a weighted graph *G*(*V*, *E*, *C*, *W*). *S* is called an optimal solution of *G*.

Fig. 11 illustrates another graph with positive costs both on vertices and edges. Example of optimal solution for the MBS_DS problem is described as follows: The sets D_1 = {*a*, *e*, *f*} and D_2 = {*c*, *d*, *f*} are two dominating sets. BP(D_1) = max{*W*(*a*), *W*(*e*), *W*(*f*)} = max{1, 1, 3} = 3 and SC(D_1) = (*b*, μ(*b*)) + (*c*, μ(*c*)) + (*d*, μ(*d*)) + (*h*, μ(*h*)) + (*k*, μ(*k*)) = *W*(*b*, *f*) + *W*(*c*, *a*) + *W*(*d*, *a*) + *W*(*h*, *f*) + *W*(*k*, *f*) = 3 + 2 + 5 + 4 + 3 = 17. It is easy to check that D_1 is an optimal solution.

This section will prove that the MBS_DS problem can be solved in *O*(*m*log*n*) time on weighted general graphs using the binary search technique. To investigate the complexity of the MBS_DS problem, a corresponding constrained optimization problem is introduced.

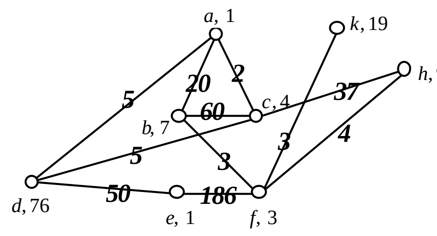

Fig. 11. A graph with real costs both on vertices and edges.

The Constrained Minimum BS-dominating-cost Dominating Set problem (***The CMBS_DS problem***): Given a weighted graph *G*(*V*, *E*, *C*, *W*) and a positive constant λ, find a dominating set *S* such that BP(*S*) ≤ λ. Any dominating set *H* satisfying the condition BP(*H*) ≤ λ is called a feasible solution. If S_1, …, S_p, $p \geq 2$, are the feasible solutions, then identify a dominating set S_i such that SC(S_i) ≤ SC(S_t), for all $t \neq i$, furthermore. We call that S_i is an optimal solution of the CMBS_DS problem.

For each *x* ∈ *V*, define *N*(*x*) = {*y* | *y* ∈ *V* and (*x*, *y*) ∈ *E*}. The following key lemma is established.

Lemma 7: Suppose that ψ (*G*(*V*, *E*, *C*, *W*) and λ) is an input instance for the CMBS_DS problem. Let *V*¢ = {*x* | *x* ∈ *V* and *C*(*x*) ≤ λ}. Then, *V* - *V*¢ = {*x* | *x* ∈ *V* and *C*(*x*) > λ}. The following statements hold.
(1) A feasible solution of ψ exists iff *N*(*v*) ∩ *V*¢ ≠ ∅, for all *v* ∈ (*V* - *V*¢).
(2) Suppose that two or more feasible solutions of ψ exist. For each *x* ∈ *V* - *V*¢, denote that $N_{V'}(x)$ = {*z* ∈ *N*(*x*) ∩ *V*¢ | *W*(*z*, *x*) ≤ *W*(*y*, *x*), for all *y* ∈ *N*(*x*) ∩ *V*¢}. Then, *H* = $\cup_{x \in (V-V')}$ {*y* | *y* is any one vertex in $N_{V'}(x)$} is an optimal solution of ψ.

***Proof*:** (1) If a feasible solution of ψ exists, then it implies that a dominating set *S* exists such that its bottleneck-placement-cost is less than or equal to λ, i.e., *S* ⊆ *V*¢. This can directly derive that *N*(*v*) ∩ *V*¢ ≠ ∅, for all *v* ∈ (*V* - *V*¢).

On the other hand, consider the case *N*(*v*) ∩ *V*¢ ≠ ∅, for all *v* ∈ (*V* - *V*¢). We simply put *S* as *V*¢ and it is easy to verify that *S* a feasible solution of ψ.
(2) From (1), we know that *H* ⊆ *V*¢. Assume that *Q* is a feasible solution other than *H* such that SC(*Q*) < SC(*H*). *Q* must also be a subset of *V*¢. Let *u* ∈ *Q* – *H*. It is easy to verify that *N*(*u*) ∩ (*V* - *V*¢) must contain at least one vertex, otherwise we can merely discard *u* to obtain a better solution.

For any *v* ∈ *N*(*u*) ∩ (*V* - *V*¢), if *v* is only dominated by *u*, then $N_{V'}(v)$ = {*u*}. This will imply that $u \in H$. Therefore, v must also be dominated by some other vertex $y \in H$ such that $W(y, v) < W(u, v)$, for all $v \in N(u) \cap (V$ - $V¢)$. This can easily derive that SC(Q) > SC(H). A contradiction occurs.

Based on Lemma 7, the following algorithm can be designed to solve the MBS_DS problem on weighted general graphs correctly.

Algorithm MBS_DS
Input: A weighted graph $G(V, E, C, W)$ in which $V = \{x_1, ..., x_n\}$.
Output: A dominating set S of G such that the BS-dominating-cost of S is minimized.
Method:

```
Step 1: Sort the vertices of $G$ into non-decreasing order using their costs as keys;
        /* $C(x_1) \le \ldots \le C(x_n)$ 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 ≤ ub)
Step 5:         mid = (lb + ub) / 2;
Step 6:         $V¢ = \{x_1, ..., x_{\text{mid}}\}$;
Step 7:         flag = Test-Neighbor($V¢$);
Step 8:         if flag == TRUE
Step 9:         {
Step 10:                $S = \cup_{u \in (V - V')} \{y \mid y$ is any one vertex in $N_{V'}(x)\}$;
                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 $S$;
End MBS_DS
```

The output of the procedure Test-Neighbor($V¢$) is TRUE if $V¢ = V$ or $N(v) \cap V¢ \neq \varnothing$, for all $v \in (V$ - $V¢)$. Otherwise, its return value is FALSE.

The time-complexity of procedure Test-Neighbor($V¢$) can be analyzed as follows. For each $x \in V$, suppose that $N(x) = \{x_{j_1}, \ldots, x_{j_\rho}\}$, $j_1 < \ldots < j_\rho$. Define ln(x) = j_1, i.e., ln(x) is the smallest index value of the vertices in $N(x)$. Then, it is easy to prove that $N(v) \cap V¢ = \varnothing$ iff ln(v) > mid, for each $v \in (V$ - $V¢)$. Therefore, the procedure Test-Neighbor($V¢$) can be finished in $O(|V$ - $V¢|)$ time if ln(x) have been pre-computed, for all $x \in V$.

The following theorem can be established consequently.

Theorem 4: The MBS_DS problem can be solved in $O(m\log n)$ time on weighted general graphs.

Proof: Let $T(n)$ denote the time-complexity of Algorithm MBS_DS. Step 1 involves a sorting of n numbers and its time-complexity is $O(n\log n)$. Step 10 can be performed in $O(m)$ time. Based on the above discussions, the while loop from Step 4 to Step 15 can be performed in $O((n + m)\log n)$ time if ln(x) have been pre-computed, for all $x \in V$. Computing all ln(x) requires to examine each edge once and its time-complexity is $O(m)$.

Combining the above results, it is easily verifiable that $T(n) = O(n\log n) + O((n + m)\log n) = O(m\log n)$.

6. The Conclusions

This paper has discussed the BDS problem and the BPDS problem on graphs with real costs on vertices. This paper has shown that the time-complexity of the BDS problem remains $O(n)$ on weighted block graphs. Second, this paper has claimed a very meaningful algorithmic result on graphs: the BPDS problem is NP-hard on bipartite graphs but $O(n)$ time solvable on weighted trees. Finally, we extend the result of the BDS problem to consider the sum communication costs simultaneously and the time-complexity is $O(m\log n)$ time on weighted general graphs

Some directions are worthy to continue in the future.

1. The approach used in this study can be easily applied to solve this problem on other classes of graphs, such as perfect graphs.
2. Consider the combinations of placement costs and communication costs simultaneously.
3. Find out the properties and the relationships between bottleneck minimization problems and summation minimization problems on weighted graphs. This is a very important and practical research direction.

Acknowledgment

This research was supported by National Science Council, Taiwan, ROC, under contract number NSC-89-2213-E-159-001.

References

[1] A. V. Aho, J. E. Hopcropt and J. D. Ullman, The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, MA, 1974.

[2] R. E. Bellman and S. E. Dreyfus, Applied Dynamic Programming, Princeton University Press,

Princeton, N.J., 1962.

[3] A. A. Bertossi, "Total Domination in Interval Graphs," Information Processing Letters, vol. 23, pp. 131-134, 1986.

[4] C. Bo and B. Liu, "Some Inequalities about Connected Domination Number," Discrete Mathematics, vol. 159, pp. 241-245, 1996.

[5] G. J. Chang, "Total Domination in Block Graphs," Operations Research Letters, vol. 8, pp. 53-57, 1989.

[6] G. J. Chang and G. L. Nemhauser, "R-domination on Block Graphs," Operations Research Letters, vol. 1, pp. 214-218, 1982.

[7] M. S. Chang, "Efficient Algorithms for the Domination Problems on Interval Graphs and Circular-Arc Graphs," SIAM Journal on Computing, vol. 27, No. 6, pp. 1671-1694, 1998.

[8] M. Farber, "Independent Domination in Chordal Graphs," Operation Research Letters, vol. 1, pp. 134-138, 1982.

[9] M. Farber, "Domination, Independent Domination and Duality in Strongly Chordal Graphs," Discrete Applied Mathematics, vol. 7, pp. 115-130, 1984.

[10] M. R. Garey and David S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Bell Laboratories, Murray Hill, Freeman & Co., N. J., 1978.

[11] D. L. Grinstead, P. J. Slater, N. A. Sherwani, and N. D. Holmes, "Efficient Edge Domination Problems in Graphs," Information Processing Letters, vol. 48, pp. 221-228, 1993.

[12] F. Harary, "A Characterization of Block Graphs," Canadian Mathematics Bull, vol. 6, pp. 1-6, 1963.

[13] T. W. Haynes, S. T. Hedetniemi, and P. J. Salter, Fundamentals of Domination in Graphs, Marcel Dekker, Inc., New York, Basel, Hong Kong, 1998.

[14] T. W. Haynes, S. T. Hedetniemi, and P. J. Salter, Domination in Graphs: Advanced Topics, Marcel Dekker, Inc., New York, Basel, Hong Kong, 1998.

[15] S. T. Hedetniemi and R. Laskar, "Connected Domination in Graphs," In B. Bollobas, editor, Graph Theory and Combinatorics, Academic Press, London, pp. 209-218, 1984.

[16] J. Y. Hsiao and C. Y. Tang, "Single Step Searching in Weighted Block Graphs," Information Sciences, vol. 81, pp. 1-29, 1994.

[17] R. W. Irving, "On Approximating the Minimum Independent Dominating Set," Information Processing Letters, vol. 37, pp. 197-200, 1991.

[18] J. K. Keil, "Total Domination in Interval Graphs," Information Processing Letters, vol. 22, pp. 171-174, 1986.

[19] H. Kim, "Finding a Maximum Independent Set in a Permutation Graph," Information Processing Letters, vol. 36, pp. 19-23, 1990.

[20] Y. C. Liu and M. S. Chang, Polynomial Algorithms for Various Weighted Perfect Domination Problems on Some Classes of Graphs, Master Theses, National Chung Cheng University, 1993.

[21] G. L. Nemhauser, Introduction to Dynamic Programming, Wiley, New York, 1966.

[22] J. Pfaff, R. Laskar, and S. T. Hedetniemi, Linear Algorithm for Independent Domination and Total Domination in Series-Parallel Graphs, Technical Report 441, Clemson University, Clemson, SC, 1984.

[23] G. Ramalingam and C. P. Rangan, "Total Domination in Interval Graphs Revisited," Information Processing Letters, vol. 27, pp. 17-21, 1988.

[24] E. Sampathkumar and H. B. Walikar, "The Connected Domination Number of A Graph," Journal of Mathematical Physical Sciences, vol. 13, pp. 607-613, 1979.

[25] C. C. Yen and R. C. T. Lee, "The Weighted Perfect Domination Problem," Information Processing Letters, vol. 35, pp. 295-299, 1990.

[26] 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, pp. 19-26, 1994.

[27] C. C. Yen and R. C. T. Lee, "The Weighted Perfect Domination and Its Variants," Discrete Applied Mathematics, vol. 66, pp. 147-160, 1996.

[28] W. C. K. Yen, "Bottleneck Domination and Bottleneck Independent Domination on Graphs," to appear in Journal of Information Science and Engineering, 2001.

[29] W. C. K. Yen, "Bottleneck Independent Dominating on Some Classes of Graphs," Proc. International Computer Symposium'2000: Workshop on Algorithms and Computation, 2000, pp. 1-8.

[30] W. C. K. Yen and C. Y. Tang, "The Bottleneck Independent Dominating on Permutation Graphs," Proc. International Computer Symposium'92, Vol. I, 1992, pp. 455-462.