Solving the weighted efficient edge domination problem on bipartite permutation graphs

Chin Lung Lu, Chuan Yi Tang*

Department of Computer Science, National Tsing Hua University, Hsin-Chu 30043, Taiwan

Received 7 June 1996; received in revised form 5 March 1998; accepted 16 March 1998

Abstract

Given a simple graph $G = (V, E)$, an edge $(u, v) \in E$ is said to dominate itself and any edge $(u, x)$ or $(v, x)$, where $x \in V$. A subset $D \subseteq E$ is called an efficient edge dominating set of $G$ if all edges in $E$ are dominated by exactly one edge of $D$. The efficient edge domination problem is to find an efficient edge dominating set of minimum size in $G$. Suppose that each edge $e \in E$ is associated with a real number $w(e)$, called the weight of $e$. The weighted efficient edge domination problem is to calculate an efficient edge dominating set $D$ of $G$ such that the weight $w(D)$ of $D$ is minimum, where $w(D) = \sum\{w(e) | e \in D\}$. In this paper, we show that the problem of determining whether $G$ has an efficient edge dominating set is NP-complete when $G$ is restricted to a bipartite graph. Consequently, the decision problem of efficient (vertex) domination remains NP-complete for the line graphs of bipartite graphs. Moreover, we present a linear time algorithm to solve the weighted efficient edge domination problem on bipartite permutation graphs, which form a subclass of bipartite graphs, using the technique of dynamic programming. © 1998 Elsevier Science B.V. All rights reserved.

Keywords: Algorithms; Efficient edge domination; NP-complete; Bipartite graphs; Bipartite permutation graphs

1. Introduction

Let $G = (V, E)$ be a simple graph, i.e., finite, undirected, and loopless graph without multiple edge. A vertex $u \in V$ is said to dominate itself and all vertices $v \in V$ such that $(u, v) \in E$. A subset $D_V$ of $V$ is called a (vertex) dominating set if every vertex of $V$ is dominated by at least one vertex in $D_V$. Similarly, an edge $(u, v) \in E$ dominates itself and any edge $(u, x)$ or $(v, x)$ in $E$. A subset $D_E$ of $E$ is called an edge dominating set if every edge of $E$ is dominated by at least one edge in $D_E$. An edge (resp. vertex) dominating set $D_E$ (resp. $D_V$) is efficient if every edge in $E$ (resp. vertex in $V$) is dominated by exactly one edge of $D_E$ (resp. vertex of $D_V$). Note that not all graphs have efficient edge dominating (EED) sets and efficient (vertex) dominating (EVD)
sets (see Fig. 1). Those graphs that have EED sets include path \( P_n \) for all \( n \), cycle \( C_n \) iff \( n \equiv 0 \pmod{3} \) [6], complete bipartite graph \( K_{m,n} \) iff \( m = 1 \) or \( n = 1 \), and complete graph \( K_n \) for \( n \leq 3 \). Those graphs that have EVD sets include \( P_n \) for all \( n \), \( C_n \) iff \( n \equiv 0 \pmod{3} \), \( K_{m,n} \) iff \( m = 1 \) or \( n = 1 \), and \( K_n \) for all \( n \) [1].

Given a simple graph \( G \), an edge-packing is an edge subset \( B \subseteq E \) such that no edge in \( E \) is dominated by more than one edge of \( B \). If \( B \) is an edge-packing, then \( B \) is said to efficiently dominate the collection \( C(B) \) of edges, where \( C(B) = \bigcup_{(v,u) \in B} \{(u,x) \mid x \in V \} \cup \{(v,x) \mid x \in V \} \). It is clear that \( G \) has an EED set if there is an edge-packing \( B \) in \( G \) with \( C(B) = E \). The efficient edge (resp. vertex) domination problem is to find an EED (resp. EVD) set of minimum size in \( G \). Suppose that each edge \( e \in E \) (resp. vertex \( u \in V \)) is associated with a real number \( w(e) \) (resp. \( w(u) \)), called the weight of \( e \) (resp. \( u \)). The weighted efficient edge (resp. vertex) domination problem is to calculate an EED set \( D_E \) (resp. EVD set \( D_V \)) of \( G \) such that the weight \( w(D_E) \) (resp. \( w(D_V) \)) is minimum, where \( w(D_E) = \sum \{w(e) \mid e \in D_E\} \) and \( w(D_V) = \sum \{w(u) \mid u \in D_V\} \).

There are many applications for the efficient edge domination problem in the resource allocation of parallel processing system [9], encoding theory and network routing problems [6]. Grinstead et al. [6] proved that the problem of determining if a given general graph has an EED set is NP-complete and presented linear time algorithms for computing the maximum number of edges that can be efficiently dominated on trees and series-parallel graphs. Pal [11] proposed a linear time algorithm for calculating an edge-packing with the maximum weight on interval graphs.

As to the efficient (vertex) domination problem, Bange et al. [1] showed that this problem is NP-complete on general graphs. Note that the concept of the EVD set is equivalent to that of independent perfect dominating (IPD) set [4]. The problem of finding an IPD set of \( G \) has been shown to be NP-complete on bipartite graphs and chordal graphs [13, 16]. Therefore, the efficient domination problem is also NP-complete on bipartite graphs and chordal graphs. Recently, Lu et al. [10] proved that the efficient domination problem on chordal bipartite graphs, which are a subclass of bipartite graphs, but a super-class of bipartite permutation graphs, is still NP-complete. The weighted efficient domination problem can be solved on cocomparability graphs and trapezoid graphs in \( O(|V||E|) \) [4] and \( O(|V| \log \log |V| + |E|) \) [8] time respectively, where \( |E| \) is the number of edges in the complement \( \overline{G} = (V, \overline{E}) \) of \( G \). There are linear time algorithms for the weighted efficient domination problem on trees [1, 13],
In this paper, we first show that the problem of determining whether $G$ has an $EED$ set is NP-complete when $G$ is restricted to a bipartite graph. Consequently, the decision problem of efficient (vertex) domination, i.e., the problem of determining whether a graph has an $EVD$ set, remains NP-complete for the line graphs of bipartite graphs. Finally, we present a linear time algorithm to solve the weighted efficient edge domination problem on bipartite permutation graphs using the technique of dynamic programming.

2. The NP-completeness of the efficient edge domination problem on bipartite graphs

**Problem EC** (Exact Cover)

*Instance:* A family of sets $F = \{S_1, S_2, \ldots, S_n\}$.

*Question:* Does $F$ contain an exact cover, i.e., a subfamily of pairwise disjoint sets whose union is equal to $X$, where $X = \bigcup_{1 \leq j \leq n} S_j$?

**Problem EED** (Efficient Edge Domination)

*Instance:* A graph $G = (V, E)$.

*Question:* Does $G$ have an $EED$ set?

From [5], we know that Problem EC is NP-complete. Problem EED is the decision problem of efficient edge domination on a graph. In this section, we will show that Problem EED is NP-complete even when $G$ is restricted to a bipartite graph by reducing Problem EC to it.

**Theorem 2.1.** Problem EED on bipartite graphs is NP-complete.

**Proof.** Obviously, there is an NP algorithm for deciding whether a bipartite graph has an $EED$ set. In the following, we will show that Problem EC is reducible to Problem EED on a bipartite graph in polynomial time.

Given an instance $F$ of Problem EC, we construct a bipartite graph $G_F = (V_F, E_F)$ as follows. Let $F = \{S_1, S_2, \ldots, S_n\}$ and $X = \{x_1, x_2, \ldots, x_m\}$.

$V_F = \{S_j, a_j, b_j \mid 1 \leq j \leq n\} \cup \{x_i \mid 1 \leq i \leq m\} \cup \{y_{ij} \mid 1 \leq i \leq m, 1 \leq j \leq n \text{ and } x_i \in S_j\}$,

$E_F = \{(S_j, a_j), (a_j, b_j) \mid 1 \leq j \leq n\} \cup \{(x_i, y_{ij}), (y_{ij}, S_j) \mid 1 \leq i \leq m, 1 \leq j \leq n \text{ and } x_i \in S_j\}$.

See Fig. 2 for an example with $F = \{S_1, S_2, S_3\} = \{(x_1, x_2, x_3), (x_4), (x_2, x_3)\}$. Next, we claim that $F$ has an exact cover $F'$ if and only if $G_F$ has an $EED$ set $D$. First, assume that $F$ has an exact cover $F'$. Define $D = \{(x_i, y_{ij}), (a_j, b_j) \mid S_j \in F'\} \cup \{(S_j, a_j) \mid S_j \in F'\}$. It is easy to verify that $D$ is an $EED$ set of $G_F$. Conversely, assume that $G_F$
has an EED set \( D \). Note that \( D \) contains no edges of the form \((y_{ij}, S_j)\); otherwise, \((y_{ij}, S_j)\) in \( D \) would imply that \((S_j, a_j)\) not in \( D \) and \((a_j, b_j)\) not in \( D \) and so \( D \) does not dominate \((a_j, b_j)\). Consequently, \( D \) contains exactly one edge adjacent to \( x_i \) for \( i = 1, 2, \ldots , m \) and exactly one edge adjacent to \( a_j \) for \( j = 1, 2, \ldots , n \). Moreover, if \( D \) contains \((a_j, b_j)\), then \( D \) contains edges \((x_i, y_{ij})\) for all \( x_i \) in \( S_j \); if \( D \) contains \((S_j, a_j)\), then \( D \) contains none of the edges \((x_i, y_{ij})\) for \( x_i \) in \( S_j \). Let \( F' \) be defined by \( S_j \in F' \) if \((a_j, b_j) \in D\). Clearly, \( F' \) is a subfamily of pairwise disjoint sets whose union is equal to \( X \). That is, \( F' \) is an exact cover of \( F \).

3. The weighted efficient edge domination problem on bipartite permutation graphs

In this section, we use \( G=(A,B,E) \) to denote a bipartite graph with a bipartition \((A,B)\) of its vertex set. For any vertex \( u \in A \cup B \), the open neighborhood of \( u \) is \( N(u) = \{ v \in A \cup B \mid (u, v) \in E \} \). A strong ordering of the vertices of \( G \) consists of an ordering of \( A \) and an ordering of \( B \) such that for all \((a, b'), (a', b) \in E \), where \( a, a' \in A \) and \( b, b' \in B \), \( a < a' \) and \( b < b' \) imply \((a, b), (a', b') \in E \). An ordering of the vertices of \( A \) has the adjacency property if for each vertex \( b \in B \), \( N(b) \) consists of vertices which are consecutive in the ordering of \( A \). An ordering of the vertices of \( A \) has the enclosure property if for every two vertices \( b, b' \in B \) with \( N(b) \subset N(b') \), vertices in \( N(b') \setminus N(b) \) occur consecutively in the ordering of \( A \). A bipartite permutation graph is a both bipartite and permutation graph, or, equivalently, a bipartite graph \( G \) with a strong ordering of \( A \cup B \) [12]. Note that given a strong ordering of \( A \cup B \), both \( A \) and \( B \) have the adjacency and the enclosure properties if all isolated vertices of \( G \) appear at the beginning of the orderings of \( A \) and \( B \) [2]. Spinrad et al. [12] gave a linear time algorithm for recognizing whether a given graph is a bipartite permutation graph and producing a strong ordering of the vertices if so. Let \( A = \{a_1, a_2, \ldots , a_m\} \) and \( B = \{b_1, b_2, \ldots , b_n\} \) be the vertices of \( A \) and \( B \) in the strong ordering such that \( a_i < a_{i'} \) iff \( 1 \leq i < i' \leq m \) and \( b_j < b_{j'} \) iff \( 1 \leq j < j' \leq n \), respectively.
For simplicity of illustrating algorithms, we assume that the given bipartite permutation graph \( G = (A, B, E) \) is connected.

**Lemma 3.1.** \( (a_1, b_1) \in E \) and \( (a_m, b_n) \in E \).

**Proof.** Suppose that \( (a_1, b_1) \notin E \). Since \( G \) is connected, there are vertices \( a_i \in A \) with \( i > 1 \) and \( b_j \in B \) with \( j > 1 \) such that \( (a_1, b_j) \) and \( (a_i, b_1) \) are in \( E \). By the strong ordering of \( A \cup B \), we have \( (a_1, b_1) \in E \), a contradiction. Similarly, \( (a_m, b_n) \in E \). 

For each vertex \( u \in A \cup B \) and each edge \( (a_i, b_j) \in E \), we define the following notation:

- \( s(u) = \min N(u) \), i.e., the smallest vertex adjacent to \( u \).
- \( l(u) = \max N(u) \), i.e., the largest vertex adjacent to \( u \).
- \( V(a_i, b_j) = \{ a_k \in A \mid k \leq i \} \cup \{ b_k \in B \mid k \leq j \} \).
- \( G(a_i, b_j) \) is the subgraph of \( G \) induced by \( V(a_i, b_j) \).
- \( M(a_i, b_j) \) is a minimum weighted \( EED \) set of \( G(a_i, b_j) \).
- \( M_a(a_i, b_j) \) is a minimum weighted \( EED \) set of \( G(a_i, b_j) \) with the condition that \( (a_i, b_j) \) is dominated by some edge \( (a_i, b_k) \) in \( D \).
- \( M_b(a_i, b_j) \) is a minimum weighted \( EED \) set of \( G(a_i, b_j) \) with the condition that \( (a_i, b_j) \) is dominated by some edge \( (a_k, b_j) \) in \( D \).
- \( M'(a_i, b_j) \) is a minimum weighted \( EED \) set of \( G(a_i, b_j) \) with the condition that \( (a_i, b_j) \) is in \( D \).

Clearly, if \( G \) has an \( EED \) set, then \( M(a_m, b_n) \) is a minimum weighted \( EED \) set of \( G \). For a set \( S \) of sets of edges, \( \text{Min}S \) denotes the set with minimum weight in \( S \) if \( S \) is not empty; otherwise, \( \text{Min}S \) denotes a set of infinite positive weight. The following four properties are clear and useful for the design of our algorithms.

- **(P1)** If \( a_i < a_{i'} \), then \( s(a_i) \leq s(a_{i'}) \) and \( l(a_i) \leq l(a_{i'}) \).
- **(P2)** If \( b_j < b_{j'} \), then \( s(b_j) \leq s(b_{j'}) \) and \( l(b_j) \leq l(b_{j'}) \).
- **(P3)** \( (a_i, b_j) \) is an edge in \( G \) for \( s(a_i) \leq b_j \leq l(b_j) \).
- **(P4)** \( (a_i, b_j) \) is an edge in \( G \) for \( s(b_j) \leq a_i \leq l(b_j) \).

**Lemma 3.2.** Any \( EED \) set of a general graph \( H \) contains no edge in a 4-cycle.

**Proof.** Let \( D \) be any \( EED \) set of \( H \) and \( C = (v_1, v_2, v_3, v_4, v_1) \) be a 4-cycle in \( H \). Assume that \( D \) contains an edge of \( C \), say \( (v_1, v_2) \). Then, all other edges of \( C \) are not in \( D \). To efficiently dominate \((v_3, v_4)\), \( D \) contains an edge \( e \) adjacent to \((v_3, v_4)\). Then, \( e \) and \((v_1, v_2)\) both dominate \((v_2, v_3)\) (or \((v_1, v_4)\)), a contradiction.

**Lemma 3.3.** For each edge \( (a_i, b_j) \in E \), if \( s(a_i) < b_j \) and \( s(b_j) < a_i \), then \( G(a_i, b_j) \) has no \( EED \) set.

**Proof.** Suppose that \( G(a_i, b_j) \) has an \( EED \) set \( D \). Then, \((a_i, b_j)\) is efficiently dominated by some edge \( (a_h, b_k) \) in \( D \) with \( h = i \) or \( k = j \), where \( s(b_j) \leq a_h \leq a_i \) and \( s(a_i) \leq b_k \leq b_j \). Choose \( h' \neq h \) with \( s(b_j) \leq a_{h'} \leq a_i \) and \( k' \neq k \) with \( s(a_i) \leq b_{k'} \leq b_j \). By (P1) to (P4) and the strong ordering of \( A \cup B \), \((a_p, b_q)\) is an edge in \( G \) for \( s(b_j) \leq a_p \leq a_i \) and
s(a_i) \leq b_i \leq b_j. Therefore, \((a_k, b_k, a_{k'}, b_{k'}, a_h)\) is a 4-cycle with \((a_h, b_k) \in D\), which contradicts Lemma 3.2.  

For each edge \((a_i, b_j) \in E\), if \(s(a_i) < b_j\) and \(s(b_j) < a_i\), then \(G(a_i, b_j)\) has no EED set according to Lemma 3.3. For the case \(s(a_i) = b_j\), we note that \(s(b_j) \leq l(b_{j-1}) < a_i\) by the connectivity of \(G\). Similarly, we have \(s(a_i) \leq l(a_{i-1}) < b_j\) for the case \(s(b_j) = a_i\). For convenience of illustrating algorithms, we introduce a pseudo-edge \((a_0, b_0)\) and let \(l(a_0) = b_1, l(b_0) = a_1\), and \(M_a(a_0, b_0)\) and \(M_b(a_0, b_0)\) be empty sets.

**Theorem 3.1.** Suppose that \(s(a_i) = b_j\) and \(s(b_j) < a_i\). Let \(s(b_j) = a_p\) and \(e_{ij} = (a_q, b_j)\) be an edge with a minimum weight in the set \(\{(a_k, b_j) | l(b_{j-1}) < a_k \leq a_i\}\). Then,

1. \(M'(a_i, b_j) = M_a(a_i, b_j) = \{\emptyset\}, \text{ if } j > 1 \text{ and } s(a_p) < b_{j-1},\)
   \(= \{\{(a_i, b_j)\} \cup M_b(a_{p-1}, b_{j-1})\}, \text{ if } j = 1 \text{ or } s(a_p) = b_{j-1}.\)

2. \(M(a_i, b_j) = M_b(a_i, b_j) = M'(a_q, b_j), \text{ if } j > 1 \text{ and } l(b_{j-1}) > a_p,\)
   \(= \{|\text{Min}\{M'(a_p, b_j), M'(a_i, b_j)\}\}, \text{ if } j = 1 \text{ or } l(b_{j-1}) = a_p.\)

**Proof.** (1) It is clear that \(M'(a_i, b_j) = M_b(a_i, b_j)\). For the case in which \(j > 1 \text{ and } s(a_p) < b_{j-1}\), we claim that \(M'(a_i, b_j) \neq \emptyset\). Suppose that \(M'(a_i, b_j) = \emptyset\). Then, \(M'(a_i, b_j)\) contains \((a_i, b_{j-1})\) with \(t \neq p\) to efficiently dominate \((a_p, b_{j-1})\). If \(t > p\), then \((a_i, b_{j-1}, a_p, b_{j-1}, a_1)\) is a 4-cycle with \((a_i, b_{j-1}) \in M'(a_i, b_j)\), which contradicts Lemma 3.2. If \(t < p\), then \((a_i, b_{j-1}, a_p, b_{j-1}, a_1)\) is a 4-cycle with \((a_i, b_{j-1}) \in M'(a_i, b_j)\), a contradiction again.

For the case in which \(j = 1 \text{ or } s(a_p) = b_{j-1}\), it is easy to verify that \(M'(a_i, b_j) = \{\{(a_i, b_j)\} \cup M_b(a_{p-1}, b_{j-1})\}.

(2) It is clear that \(M(a_i, b_j) = M_b(a_i, b_j)\). Let \((a_i, b_j) \in M(a_i, b_j)\), where \(p \leq t \leq i\). Note that we have \(a_p < l(b_{j-1}) < a_i\). Consider the case in which \(j > 1 \text{ and } l(b_{j-1}) > a_p\). If \(a_t < l(b_{j-1})\), then \((a_t, b_j, l(b_{j-1}), b_{j-1}, a_t)\) is a 4-cycle with \((a_t, b_j) \in M(a_i, b_j)\), which contradicts Lemma 3.2. If \(a_t = l(b_{j-1})\), then \((a_t, b_j, a_p, b_{j-1}, a_1)\) is a 4-cycle with \((a_t, b_j) \in M(a_i, b_j)\), a contradiction again. Therefore, we have \(l(b_{j-1}) < a_t \leq a_i\). For any two edges \(e_1\) and \(e_2\) in the set \(\{(a_k, b_j) | l(b_{j-1}) < a_k \leq a_i\}\), the set of the edges dominated by \(e_1\) is equal to the set of the edges dominated by \(e_2\). In other words, \(a_t = a_q\). Therefore, \(M(a_i, b_j) = M'(a_q, b_j).\)

For the case in which \(j = 1 \text{ or } l(b_{j-1}) = a_p\), it is easy to verify that \(M(a_i, b_j) = \text{Min}\{M'(a_p, b_j), M'(a_i, b_j)\}. \)

**Theorem 3.2.** Suppose that \(s(b_j) = a_i\) and \(s(a_i) < b_j\). Let \(s(a_i) = b_p\) and \(f_{ij} = (a_i, b_q)\) be an edge with a minimum weight in the set \(\{(a_i, b_k) | l(a_{i-1}) < b_k \leq b_j\}\). Then,

1. \(M'(a_i, b_j) = M_b(a_i, b_j) = \{\emptyset\}, \text{ if } i > 1 \text{ and } s(b_p) < a_{i-1},\)
   \(= \{\{(a_i, b_j)\} \cup M_a(a_{i-1}, b_{p-1})\}, \text{ if } i = 1 \text{ or } s(b_p) = a_{i-1}.\)

2. \(M(a_i, b_j) = M_b(a_i, b_j) = \{M'(a_i, b_q), \text{ if } i > 1 \text{ and } l(a_{i-1}) > b_p,\)
   \(= \{|\text{Min}\{M'(a_i, b_p), M'(a_i, b_q)\}\}, \text{ if } i = 1 \text{ or } l(a_{i-1}) = b_p.\)
Proof. Similar to Theorem 3.1. □

Lemma 3.4. For all edges \((a_i, b_j) \in E\) with \(s(a_i) = b_j\) and \(s(b_j) < a_i\), all edges \(e_{ij}\) can be computed in \(O(|E|)\) time.

Proof. We give the following algorithm for computing all \(e_{ij}\). Since the number of the total iterations is \(O(|E|)\), the time complexity of Algorithm 3.1 is \(O(|E|)\). □

Algorithm 3.1 /* Compute all \(e_{ij}\).*

\[
\text{for } j = 1 \text{ to } n \text{ do}
\]
\[
\text{for } a_i = s(b_j) \text{ to } l(b_j) \text{ do}
\]
\[
\text{if } a_i > l(b_{j-1}) \text{ then}
\]
\[
\text{if } w((a_i, b_j)) < w((a_t, b_j)) \text{ then } t = i;
\]
\[
e_{ij} = (a_i, b_j);
\]
\[
\text{endif}
\]
\[
\text{endfor}
\]
\[
\text{endfor}.
\]

Lemma 3.5. For all edges \((a_i, b_j) \in E\) with \(s(b_j) = a_i\) and \(s(a_i) < b_j\), all edges \(f_{ij}\) can be computed in \(O(|E|)\) time.

Proof. Similar to Lemma 3.4. □

By Theorems 3.1 and 3.2, we can design the following Algorithm \textit{MWEEDS} to solve the weighted efficient edge domination problem on the connected bipartite permutation graphs using the technique of dynamic programming.

Algorithm \textit{MWEEDS}.

\textit{Input:} A weighted and connected bipartite permutation graph \(G = (A, B, E)\).

\textit{Output:} A minimum weighted EED set \(D\) of \(G\).

Step 1: \(l(a_0) = b_1, l(b_0) = a_1\) and \(M_b(a_0, b_0) = M_b(a_1, b_1) = \emptyset\); \(M(a_1, b_1) = M_b(a_1, b_1) = M_b(a_1, b_1) = M'(a_1, b_1) = \{(a_1, b_1)\}\).

\textbf{for} each vertex \(u \in A \cup B\) \textbf{do} calculate \(s(u)\) and \(l(u)\); 

Step 2: compute all \(e_{ij}\) and \(f_{ij}\);

Step 3: \textbf{for} \(j = 1\) to \(n\) \textbf{do}

\textbf{for} \(a_i = s(b_j)\) \textbf{to} \(l(b_j)\) \textbf{do}

\textbf{case} 1: \(s(a_i) < b_j\) and \(s(b_j) < a_i\) \textbf{then}

let \(M(a_i, b_j), M_b(a_i, b_j), M_b(a_i, b_j)\) and \(M'(a_i, b_j)\) be empty sets with weights \(\infty\); /* by Lemma 3.3 */

\textbf{endcase};

\textbf{case} 2: \(s(a_i) = b_j\) and \(s(b_j) < a_i\) \textbf{then}

compute \(M'(a_i, b_j)\) and \(M_b(a_i, b_j)\) by (1) of Theorem 3.1, and compute \(M(a_i, b_j)\) and \(M_b(a_i, b_j)\) by (2) of Theorem 3.1;
endcase;

\textbf{case 3:} \( s(a_i) < b_j \) and \( s(b_j) = a_i \) then

\begin{align*}
&\text{compute } M'(a_i, b_j) \text{ and } M_0(a_i, b_j) \text{ by (1) of Theorem } 3.2, \\
&\text{and compute } M(a_i, b_j) \text{ and } M_0(a_i, b_j) \text{ by (2) of Theorem } 3.2;
\end{align*}

endcase

endfor

donefor;

\textbf{Step 4:} \textbf{if} \( w(M(a_m, b_n)) \neq \infty \) \textbf{then} \( D = M(a_m, b_n) \) \textbf{else} there is no \textit{EED} set in \( G \).

\textbf{Theorem 3.3.} Algorithm \textit{MWEEDS} finds a minimum weighted efficient edge dominating set of a connected bipartite permutation graph in \( O(|A| + |B| + |E|) \) time.

\textbf{Proof.} The correctness of Algorithm \textit{MWEEDS} immediately follows from the theorems in this section. In the following, we analyze the time complexity of Algorithm \textit{MWEEDS}. It is clear that Step 1 can be implemented in \( O(|A| + |B| + |E|) \). Step 2 takes \( O(|E|) \) time according to Lemmas 3.4 and 3.5. The time complexity of Step 3 is \( O(|E|) \) since the computation of each case costs constant time and the number of the total iterations is \( O(|E|) \). Therefore, the time complexity of Algorithm \textit{MWEEDS} is \( O(|A| + |B| + |E|) \). \( \square \)

\textbf{Theorem 3.4.} The weighted efficient edge domination problem on bipartite permutation graphs can be solved in linear time.

\textbf{Proof.} It immediately follows from Theorem 3.3. \( \square \)

\section{Conclusion}

In this paper, we first showed that the problem of determining whether a given graph has an efficient edge dominating set is NP-complete when restricted to bipartite graphs. Consequently, the decision problem of efficient (vertex) domination remains NP-complete for the line graphs of bipartite graphs. Finally, we presented a linear time algorithm to solve the weighted efficient edge domination problem on bipartite permutation graphs using the technique of dynamic programming. For further research, we are interested in the (weighted) efficient edge domination problem for other classes of graphs, such as chordal graphs and permutation graphs.

\textbf{Acknowledgements}

The authors would like to thank the anonymous referees for many constructive suggestions for the presentation of this paper.
References