Red Domination in Perfect Elimination Bipartite Graphs

Nesrine Abbas

MacEwan University, 5-173, 10700 104 Avenue, Edmonton, AB, T5J 4S2, Canada

Abstract
The \( k \) red domination problem for a bipartite graph \( G = (X, Y, E) \) is to find a subset \( D \subseteq X \) of cardinality at most \( k \) that dominates vertices of \( Y \). The decision version of this problem is NP-complete for general bipartite graphs but solvable in polynomial time for chordal bipartite graphs. We strengthen that result by showing that it is NP-complete for perfect elimination bipartite graphs. We present a tight upper bound on the number of such sets in bipartite graphs, and show that we can calculate that number in linear time for convex bipartite graphs. We present a linear space linear delay enumeration algorithm that needs only linear preprocessing time.

Keywords: Perfect elimination bipartite graph, Convex bipartite graph, Counting, Enumeration, Generation, Red dominating set, Blue dominating set

1. Introduction

A red dominating set in a bipartite graph \( G = (X, Y, E) \) is equivalent to a set cover where each set is represented by a vertex in \( X \), vertices of \( Y \) represent elements of the universe, and an edge between \( x \in X \) and \( y \in Y \) is added to the graph if the corresponding set contains the corresponding element. The set cover problem is to identify the smallest collection of subsets whose union equals the universe. The decision version of the set cover problem is NP-complete \(^1\). There are well-known greedy approximation algorithms to compute an approximate set cover, see \(^2\) for example. The aim of this paper is to study the problem for special bipartite graphs. The term red dominating set is better suited, and is what has been used in the literature, for graph contexts. The name originates from the view of a graph as partitioned into Red and Blue sets. The problem has been studied for general graphs in \(^3\) where the authors develop an exact exponential-time algorithm for connected red dominating set. The problem has practical application in the assignment of trains to stations.

\(^1\) Garey and Johnson, \(\text{The Design and Analysis of Algorithms} \)
\(^2\) Christofides, \(\text{Graph Theory with Applications to Engineering and Computer Science} \)
\(^3\) Yuan, \(\text{Algorithmic Graph Theory} \)
In [4], the author develops an algorithm based on real data to find a subset of the stations that covers all trains. In this paper we prove that the decision problem is NP-complete for perfect elimination bipartite graphs. We develop algorithms for counting and enumerating minimum cardinality red dominating sets in convex bipartite graphs. We present a tight upper bound on the number of such sets in bipartite graphs.

Convex bipartite graphs arise in scheduling problems. In [5], for example, a convex bipartite graph represents jobs in partition $X$ and time slots in partition $Y$. A red dominating set on such a graph is a subset of jobs that covers the time range. The red dominating set problem has been studied for chordal bipartite graphs, which properly include convex bipartite graphs, in [6]. In that paper the authors present an $O(n \cdot \min(|E| \cdot \log n, n^2))$ delay enumeration algorithm.

Other domination, counting, and enumeration problems have been studied for convex bipartite graphs and classes of bipartite graphs that include them. Enumerating spanning trees in bipartite graphs has been studied in [7]. The authors in [8] present a linear time algorithm to count independent sets in tree-convex graphs. Counting independent sets in other bipartite graph subclasses that include convex bipartite graphs has been studied in [9]. In [10], the authors show that various domination problems are solvable in polynomial time for convex bipartite graphs.

The next section contains definitions that will be used throughout the paper. We will then prove that the red dominating set decision problem is NP-complete for perfect elimination bipartite graphs. In the section that follows we present a tight upper bound on the number of vertices in a red dominating set in a bipartite graph, show how to calculate the number of such sets in linear time for convex bipartite graphs and develop a linear space linear delay enumeration algorithm. We then conclude the paper.

2. Definitions

A graph $G = (V, E)$ in this paper is finite, connected, undirected, with no loops and no parallel edges. We will use $n$ to denote $|V|$. The neighbourhood set of vertex $v$ is $N(v) = \{u : uv \in E\}$. The degree of $v$ is $\deg(v) = |N(v)|$. The graph induced by a subset $D \subseteq V$ is denoted by $G[D]$. $G[D]$ (or $D$) is an independent set if it induces a graph that has no edges. A set of vertices $D \subseteq V$ dominates another set $U \subseteq V$ if $\forall u \in U$, there is $v \in D$ such that $u = v$ or $uv \in E$. When $U$ is a single vertex $u$, we say that $D$ dominates $u$. A set $D \subseteq V$ is dominating if it dominates $V$. A path of length $k$, or a $k$-path, in $G$ is a sequence of distinct vertices $(u_1, \ldots, u_{k+1})$ such that $u_ju_{j+1} \in E, \forall j = 1, \ldots, k$. A connected graph is one that has a path between each pair in its vertex set. A cycle of length $k$ in $G$ is a sequence of distinct vertices $(u_1, \ldots, u_k)$ such that $u_ju_{j+1} \in E, \forall j = 1, \ldots, k-1, u_1u_k \in E$. A chord in a cycle is an edge joining non-consecutive vertices.

A bipartite graph $G = (X, Y, E)$ is one whose vertex set $X \cup Y$ can be partitioned into two independent sets $X$ and $Y$. We will denote $|X|$ by $n_X$ and $|Y|$ by $n_Y$. A complete bipartite graph $K_{n_X,n_Y}$ contains all possible edges.
between its two partitions. An edge \(xy\) is bisimplicial if \(N(x) \cup N(y)\) induces a complete bipartite graph. Let \(\sigma = (e_1, \ldots, e_p)\), where \(e_i = x_iy_i\), be an ordering of pairwise disjoint edges of \(G\), \(S_j = \{x_1, x_2, \ldots, x_j\} \cup \{y_1, y_2, \ldots, y_j\}\), and \(S_0 = \emptyset\). \(\sigma\) is said to be a perfect edge elimination scheme if \(e_{j+1}\) is bisimplicial in \(G[(X \cup Y) \setminus S_j]\), for \(j = 0, \ldots, p - 1\), and \(G[(X \cup Y) \setminus S_p]\) has no edges.

A bipartite graph that admits a perfect edge elimination scheme is called a perfect elimination bipartite graph. A bipartite graph \(G = (X,Y,E)\) is said to be chordal bipartite if each cycle of length greater than 4 has a chord. A bipartite graph \(G = (X,Y,E)\) is said to be \(Y\)-convex if vertices of \(Y\) can be ordered so that for each \(x \in X\) neighbours of \(x\) appear consecutively in \(Y\). We will simply refer to that graph as convex bipartite. Such an ordering is called a convex ordering and can be calculated in linear time [11]. Figure 1 shows a convex bipartite graph. Convex bipartite graphs are a proper subset of chordal bipartite graphs [12] which in turn are properly contained in perfect elimination bipartite graphs [13].

Let the neighbourhood set of some vertex \(x, x \in X\), in the convex bipartite graph \(G = (X,Y,E)\) whose vertices are in convex ordering be \(N(x) = \{y_a, y_{a+1}, \ldots, y_{a+b}\}\), \(y_a < \ldots < y_{a+b}\). Then \(\text{left}(x) = y_a\) is the leftmost neighbour of \(x\), and \(\text{right}(x) = y_{a+b}\) is the rightmost neighbour of \(x\). The neighbourhood array \(\mathcal{N}\) stores the two values \(\text{left}(x)\) and \(\text{right}(x)\) for each vertex \(x \in X\), i.e., \(\mathcal{N}[i] = (\text{left}(x_i), \text{right}(x_i))\). Given a convex ordering, values of \(\text{left}(x)\) may be calculated in \(O(n + |E|)\) time using a simple procedure that starts at \(y = y_1\), moves sequentially in \(Y\), and assigns the smallest value to \(\text{left}(x)\) for all \(x \in N(y)\). Values of \(\text{right}(x)\) may be calculated similarly. We will use \(\mathcal{N}\) as the data structure to represent a convex bipartite graph in our algorithms.

A lexicographic convex ordering (lex-convex) is a convex ordering and for \(x_i, x_j\), \(i < j\) if \(\text{left}(x_i) < \text{left}(x_j)\) or \(\text{left}(x_i) = \text{left}(x_j)\) and \(\text{right}(x_i) < \text{right}(x_j)\). Cases where \(\text{left}(x_i) = \text{left}(x_j)\) and \(\text{right}(x_i) = \text{right}(x_j)\), i.e., \(N(x_i) = N(x_j)\), can be numbered arbitrarily without disturbing other vertices’ order. Given the array \(\mathcal{N}\), a simple LSD (Least Significant Digit) radix sort may be used to obtain a lex-convex ordering of the vertices of \(X\) in \(O(n)\) time. The keys will be \(\text{left}(x)\text{right}(x)\) for each \(x \in X\). The ordering of the vertices of the graph in Figure 4 is lex-convex. We assume a lex-convex ordering for all convex bipartite graphs in this paper.

The reader is referred to [14] and [13] for any missing graph theory definitions or notations.

We formally define the \(k\) RED DOMINATION decision problem and its minimum cardinality version.

**Definition 1.** \(k\) RED DOMINATION: Given a bipartite graph \(G = (X,Y,E)\) and integer \(k\), \(1 \leq k \leq |X|\), is there a subset \(D \subseteq X\) of cardinality at most \(k\) that dominates \(Y\), i.e., \(\forall y \in Y\), does there exist some \(x \in D\), s.t. \(xy \in E\) ?

**Definition 2.** MCRD (Minimum Cardinality Red Domination): Given a bipartite graph \(G = (X,Y,E)\), find a minimum cardinality red dominating set in \(G\).
3. NP-completeness for Perfect Elimination Bipartite Graphs

Theorem 3.1. \( k \) RED DOMINATION is NP-complete for perfect elimination bipartite graphs.

Proof. Clearly \( k \) RED DOMINATION is in NP because given a set \( D \), we can verify in polynomial time if it is a solution for \( k \) RED DOMINATION for the bipartite graph \( G = (X,Y,E) \).

We complete the proof by reduction from \( k \) RED DOMINATION for bipartite graphs, which is NP-complete [1]. Given a connected bipartite graph \( G = (X,Y,E) \) and a positive integer \( k, 1 \leq k \leq n_X \), we construct a perfect elimination bipartite graph \( G_P = (X_P,Y_P,E_P) \) such that \( G \) has a red dominating set of cardinality \( k \) if and only if \( G_P \) has a red dominating set of cardinality \( k \). Let \( Y_P = Y \), \( X_P = X \cup T \) where \( T = \{ t_i : y_i \in Y \} \), and \( E_P = E \cup \{ t_i y_i, i = 1, \ldots, n_Y \} \), i.e., we include all edges in the original graph and attach a vertex \( t \) to each vertex \( y \). Clearly \( G_P \) is bipartite and can be constructed in polynomial time. \( G_P \) has the perfect edge elimination scheme \( \sigma = (t_1 y_1, t_2 y_2, \ldots, t_{n_Y} y_{n_Y}) \).

Let \( D \) be a red dominating set in \( G \). Since \( Y_P = Y \), then \( D \) induces a red dominating set in \( G_P \).

Now let \( D \) be a red dominating set in \( G_P \). If \( D \) contains no vertices from \( T \), then \( D \) induces a red dominating set in \( G \). If \( D \cap T \neq \emptyset \), then there is a vertex \( x \) in \( X_P \setminus T \) such that \( N(x) \cap N(t) \neq \emptyset \) that may replace each such vertex \( t \in D \cap T \) because we started with a connected graph \( G \). Such a set \( D \), with no vertices from \( T \), will correspond to a red dominating set with the same cardinality in \( G \). □
4. Counting and Enumerating All MCRD Sets for a Convex Bipartite Graph

In this section, we show a tight upper bound on the number of MCRD sets of cardinality $k$ in an arbitrary bipartite graph and show that there are convex bipartite graphs that possess that number of MCRD sets. We present an algorithm that enumerates all MCRD sets in an arbitrary convex bipartite graph $G = (X, Y, E)$ in linear space and linear delay after a linear time preprocessing step. The algorithm operates in two stages. The first is a labelling preprocessing stage. The second is a branch and bound search on the labelled vertices of $X$ to output the MCRD sets. The labelling algorithm has been extended to calculate the number of MCRD sets.

**Theorem 4.1.** Let $d$ be the maximum degree of vertices in $Y$ in the bipartite graph $G = (X, Y, E)$, $d = \max_{y \in Y} \deg(y)$, $d \leq n_X$. The number of MCRD sets in an arbitrary bipartite graph $G = (X, Y, E)$ is $O(d^k)$, where $k$ is the cardinality of an MCRD set.

**Proof.** Let $d$ be the maximum degree of vertices in $Y$ in the bipartite graph $G = (X, Y, E)$, $d = \max_{y \in Y} \deg(y)$, $d \leq n_X$. Choose an arbitrary vertex $y_1 \in Y$. There are $\deg(y_1) \leq d$ choices for the first vertex in an MCRD set for $G$. Given vertices $u_1, u_2, \ldots, u_a, a \geq 1$, in an MCRD set $D$, we only need to consider a vertex from $N(y)$ for an arbitrary vertex $y \in Y$ that is not dominated by $u_1, u_2, \ldots, u_a$ for the next vertex in $D$. At each step in the process to build $D$, at most $d$ vertices need to be considered. Using this process, some of the sets obtained may not be MCRD sets of cardinality $k$, however all MCRD sets of cardinality $k$ will be generated. Hence the number of MCRD sets is $O(d^k)$. □

Figure 2 shows a convex bipartite graph that possesses that upper bound of MCRD sets. Not all vertices and edges are shown in the figure. The graph consists of $k, k > 1$, complete bipartite graphs $K_{d-1,d-1}$, with an edge joining pairs of consecutive complete bipartite graphs — the thick edges shown in the figure. The maximum degree of vertices in $Y$ is $d$ and it is that of vertices incident with edges joining the complete bipartite graphs, shown filled with black in the figure. Any MCRD set will contain exactly one vertex from each complete bipartite graph and will be of cardinality $k$. Thus the number of MCRD sets is $(d-1)^k = O(d^k)$. It is worth noting that we can construct a graph with $d^k$ MCRD sets by removing the thick edges joining the complete bipartite graphs but this will render the graph disconnected.

In the next lemma we prove some algorithmic properties for MCRD sets in convex bipartite graphs.

**Lemma 4.2.** Let $D = \{u_1, u_2, \ldots, u_k\}$, $u_1 < u_2 < \ldots u_k$, be an MCRD set of cardinality $k$ for the convex bipartite graph $G = (X, Y, E)$. The following is true.

- For distinct vertices $u, v \in D$, it cannot be the case that $left(u) \leq left(v) \leq right(v) \leq right(u)$, i.e., $N(v) \not\subseteq N(u), \forall u, v \in D, u \neq v.$
For all consecutive pairs $u, v, u < v$ in $D$, $\text{left}(u) < \text{left}(v) \leq \text{right}(u) + 1 \leq \text{right}(v)$, i.e., $N(u) \cup N(v)$ consists of consecutive vertices in $Y$.

For each vertex $u \in D$, $D$ cannot contain more than one vertex $v > u$ such that $\text{left}(u) < \text{left}(v) \leq \text{right}(u) + 1 \leq \text{right}(v)$.

Proof. Let $D$ be an MCRD set of cardinality $k$ for the convex bipartite graph $G = (X, Y, E)$. $N(v) \subseteq N(u)$ for some pair $u, v \in D$, i.e., $\text{left}(u) \leq \text{left}(v) \leq \text{right}(u) \leq \text{right}(v)$, implies $D \setminus \{v\}$ is a red dominating set of cardinality $k-1$, which contradicts the minimality of $D$.

Note that this implies that for all vertices $u, v \in D$, $u \neq v$, $\text{left}(u) \neq \text{left}(v)$ and $\text{right}(u) \neq \text{right}(v)$.

The union of the neighbourhood sets, $N(u) \cup N(v)$, of two consecutive vertices $u, v, u < v$ in a red dominating set in a convex bipartite graph whose vertices are in lex-convex ordering must consist of consecutive vertices in $Y$, i.e., $\text{right}(u) + 1 \in N(v)$, for otherwise some vertices in $Y$ would not be dominated. Therefore, $\text{left}(u) < \text{left}(v) \leq \text{right}(u) + 1 \leq \text{right}(v)$ by the above paragraph.

Suppose there are vertices $u < v < w \in D$ such that $\text{left}(u) < \text{left}(v) \leq \text{right}(u) + 1 \leq \text{right}(v)$ and $\text{left}(u) < \text{left}(w) \leq \text{right}(u) + 1 \leq \text{right}(w)$. Then $N(u) \cup N(v) \subseteq N(u) \cup N(w)$ if $\text{right}(v) < \text{right}(w)$, and $N(u) \cup N(w) \subseteq N(u) \cup N(v)$ otherwise. In either case one of $v$ or $w$ may be removed from $D$ to produce a $k-1$ red dominating set, contradicting the minimality of $D$. □

The enumeration algorithm operates in two stages. The first is a preprocessing labelling procedure that assigns labels to vertices of $X$. The second stage is a fast branch and bound search on vertices of $X$ that starts at a vertex adjacent to $y_n$ with a particular label and proceeds backwards to a vertex adjacent to $y_1$. The preprocessing stage is presented in Algorithm 1. Vertices adjacent to
$y_1$ are assigned label 0 and other vertices start with an initial label of \( \infty \). Subsequently, a vertex \( v \) is assigned label \( a \) if the union of its neighbourhood set with that of a smaller vertex \( u \) with label \( a - 1 \) consists of consecutive vertices in \( Y \), \( N(v) \nsubseteq N(u) \), and \( N(u) \nsubseteq N(v) \). The label \( a \) will be the smallest such label for \( v \). We use a FIFO queue to process the vertices. The algorithm can also be used to calculate the number of MCRD sets in linear time as we will prove. The array \( \text{Count} \) is used for that purpose.

**Algorithm 1** Assigning labels and counts to vertices of \( X \) in the convex bipartite graph \( G = (X,Y,E) \)

**Input:** The neighbourhood array \( N \) of the convex bipartite graph \( G = (X,Y,E) \)

**Output:** Arrays \( \text{Label}, \text{Count}, |\text{Label}| = |\text{Count}| = n_X \). \( \text{Label}[x] \) stores the label of \( x \). \( \text{Count}[x] \), for \( x \in N(y_{n_Y}) \), stores the number of MCRD sets containing \( x \).

1: \( \text{Label}[x] \leftarrow \infty, \forall x \in X \);
2: \( \text{Count}[x] \leftarrow 0, \forall x \in X \);
3: \text{enqueue}(\text{Queue}, N(y_1)); // enqueue neighbours of \( y_1 \) into queue in order
4: \( \text{Label}[x] \leftarrow 0, \forall x \in N(y_1) \);
5: \( \text{Count}[x] \leftarrow 1, \forall x \in N(y_1) \);
6: \textbf{while} Queue is not empty \textbf{do}
7: \( x \leftarrow \text{dequeue}(\text{Queue}) \);
8: \textbf{for all} \( x_j \in \{ x' : x' > x, x' \leq x_{n_X}, \text{left}(x) < \text{left}(x') \leq \text{right}(x') \leq \text{right}(x) + 1 \leq \text{right}(x'), \text{Label}[x'] \geq \text{Label}[x] + 1 \} \) \textbf{do}
9: \textbf{if} \( \text{Label}[x_j] > \text{Label}[x] + 1 \) \textbf{then}
10: \text{enqueue}(\text{Queue}, x_j); // enqueue vertex only if it is not in queue
11: \textbf{end if}
12: \( \text{Label}[x_j] \leftarrow \text{Label}[x] + 1 \);
13: \( \text{Count}[x_j] \leftarrow \text{Count}[x_j] + \text{Count}[x] \);
14: \textbf{end for}
15: \textbf{end while}

Running Algorithm 1 on the graph in Figure 1 results in the array \( \text{Label} \) with values \([0, 0, 1, 1, \infty, 2, 2, 2, \infty] \) and the array \( \text{Count} \) with values \([1, 1, 2, 1, 0, 2, 3, 1, 0] \).

**Theorem 4.3.** Algorithm 1 runs in \( O(n_X + |E|) \) time.

**Proof.** The set built in line 8 will be of size at most \( \text{deg}(\text{right}(x) + 1) \). Adding up those sets over all iterations and since each vertex in \( X \) is visited at most once, we can see that this step is executed at most \( |E| \) times. Since all other operations in the while loop take constant time and operations outside the loop take at most \( O(n_X) \) time, therefore, Algorithm 1 runs in \( O(n_X + |E|) \) time. \( \square \)

Let \( k \) be the cardinality of an MCRD for a given convex bipartite graph \( G = (X,Y,E) \). We prove the following properties for the output of Algorithm 1.
Lemma 4.4. The following is true for the output of Algorithm 7.

(a) Each vertex adjacent to \( y_1 \) retains a label of 0, and each other vertex has a label larger than 0.

(b) If \( u < v \) and \( \text{Label}[u] \neq \infty \) then \( \text{Label}[u] \leq \text{Label}[v] \).

(c) Once a vertex is assigned a label \( < \infty \), its label is not modified.

(d) If \( \text{left}(u) = \text{left}(v) \) and both \( u, v \) have labels \( \neq \infty \), then \( \text{Label}[u] = \text{Label}[v] \).

(e) A vertex \( v \) is assigned label \( a > 0 \) if and only if there is a vertex \( u \) whose label is \( a - 1 \), \( u < v \), \( N(v) \cup N(u) \) consists of consecutive vertices, \( N(v) \not\subseteq N(u) \), and \( N(u) \not\subseteq N(v) \).

Proof. The first statement is easy to see. It is also easy to note that at the start of each iteration of the while loop the queue will contain only vertices with labels smaller than \( \infty \).

We prove statement \((b)\) by induction on the index of vertices in \( X \). It is true for \( x \in N(y_1) \). Assume it is true for all \( x < x_p, 1 < p \leq n_X \). The first time the label of \( x_p, x_p \not\in N(y_1) \), is modified is when it is first added to the set built in line 8. Let \( x_1 \) be assigned label \( a \) for the first time during an iteration when \( x = x_1 < x_p, \text{Label}[x_1] = a - 1 \). For \( \text{Label}[x_p] \) to change, it must be added to a set built in line 8 at an iteration when \( x = x', x_i < x' < x_p \). Then \( \text{Label}[x_p] = a \geq \text{Label}[x'] + 1 \) and \( \text{Label}[x_1] \leq \text{Label}[x'] \) by the induction hypothesis, imply \( \text{Label}[x'] = a - 1 \). Hence \( \text{Label}[x_p] \) will not change, and \( \text{Label}[x_p] \geq \text{Label}[x], \forall x < x_p \). Therefore, by mathematical induction, if \( u < v \) and \( \text{Label}[u] \neq \infty \) then \( \text{Label}[u] \leq \text{Label}[v] \).

The construct above also proves statement \((c)\).

Let \( u < v \) and \( \text{left}(u) = \text{left}(v) \), then \( N(u) \subseteq N(v) \). Suppose they don’t have the same labels, then \( N(u) \subset N(v) \), i.e., \( \text{right}(u) < \text{right}(v) \), and \( \text{Label}[u] = a < \text{Label}[v] \), by statement \((b)\). This means that \( u \) is added to the set built in line 8 when \( x = x_i \) and \( \text{Label}[x_i] = a - 1 \), \( \text{left}(x_i) < \text{left}(u) = \text{left}(v) \leq \text{right}(x) + 1 \leq \text{right}(u) < \text{right}(v) \) and \( \text{Label}[x_i] + 1 \leq \text{Label}[u] < \text{Label}[v] \) imply that \( v \) too is added to that set during the same iteration and will receive label \( a \). By statement \((c)\) the label of \( v \) will not change. This contradicts our supposition. Therefore, \( \text{left}(u) = \text{left}(v) \) and both \( u, v \) have labels \( \neq \infty \) implies \( \text{Label}[u] = \text{Label}[v] \). This proves statement \((d)\).

A vertex \( v \) is assigned label \( a \) when it is added to the set in line 8 when \( x = u < v \) has label \( a - 1 \) and \( \text{left}(u) < \text{left}(v) \leq \text{right}(u) + 1 \leq \text{right}(v) \). \( N(v) \cup N(u) \) consists of consecutive vertices and \( N(v) \not\subseteq N(u) \) because \( \text{right}(u) + 1 \in N(v) \). \( N(u) \not\subseteq N(v) \) because \( \text{left}(u) < \text{left}(v) \). Since the label of \( v \) will not change by statement \((c)\) this proves the if-then part \((\Longrightarrow)\) direction in statement \((e)\).

If there is a vertex \( u \) with label \( a - 1 \), and a vertex \( v > u \), whose label has not been modified from \( \infty \), such that \( N(v) \cup N(u) \) consists of consecutive vertices, \( N(v) \not\subseteq N(u) \), and \( N(u) \not\subseteq N(v) \), then \( \text{left}(u) < \text{left}(v) \leq \text{right}(u) + 1 \leq \text{right}(v) \).
right(v) and v will be added to the set in line 8 when x = u and will receive a label of a. This concludes the proof for statement (e) and the lemma. □

Lemma 4.5. Let $D = \{u_1, u_2, \ldots, u_k\}, u_1 < u_2, \ldots, < u_k$, be a subset of $X$ for the convex bipartite graph $G = (X, Y, E)$, and Label be the output of Algorithm 7. $D$ is an MCRD set if and only if all of the following is true.

- $u_k$ is adjacent to $y_{n_Y}$
- $\text{Label}[u_i] = i - 1$ for all $u_i \in D$
- $k - 1 = \min_{x \in N(y_{n_Y})} \text{Label}[x]$,
  i.e., it is the smallest label of any vertex adjacent to $y_{n_Y}$
- for any consecutive vertices $u_i, u_{i+1} \in D$, $i = 1, \ldots, k - 1$, $\text{left}(u_i) \leq \text{left}(u_{i+1}) - 1 \leq \text{right}(u_i) < \text{right}(u_{i+1})$

Proof. $\implies$: By the definition of red domination and the lex convex ordering, $u_1$ is adjacent to $y_1$ and $u_k$ is adjacent to $y_{n_Y}$. By statement (a) and statement (e) in Lemma 4.4, $\text{Label}[u_1] = 0$ and $\text{Label}[u_k] = i - 1$. By the minimality of $D$, $k - 1 = \min_{x \in N(y_{n_Y})} \text{Label}[x]$. By Lemma 4.2, for consecutive vertices $u_i$ and $u_{i+1}$, $\text{left}(u_i) < \text{left}(u_{i+1}) \leq \text{right}(u_i) + 1 \leq \text{right}(u_{i+1})$. Therefore, $\text{left}(u_i) \leq \text{left}(u_{i+1}) - 1 \leq \text{right}(u_i) < \text{right}(u_{i+1})$.

$\impliedby$: If a set $D = \{u_1, \ldots, u_k\} \subseteq X$ satisfies the conditions, then it is easy to see that it is a red dominating set. If it is not an MCRD set then $k > p$, where $p$ is the cardinality of an MCRD set of $G$. Then there is a vertex $x \in N(y_{n_Y})$ with a label of $p - 1$ by the first part of this proof. This contradicts the definition of $k$ in the hypothesis. □

Theorem 4.6. The number of MCRD sets for a graph $G = (X, Y, E)$ is equal to $\sum_{x \in N(y_{n_Y})} \text{Count}[x]$, where Count is the output of Algorithm 7.

Proof. By Lemma 4.5, it suffices to prove that the number of MCRD sets containing $u$ for each $u \in \{u : u \in N(y_{n_Y}), \text{Label}[u] = k - 1\}$ is equal to $\text{Count}[u]$. We will prove the more general statement that the number of MCRD sets containing $u$ for each $u \in N(y_{n_Y})$ is equal to $\text{Count}[u]$. We will first prove the following claim.

Claim 1. If $D = \{u_1, u_2, \ldots, u_k\}, u_1 < u_2, \ldots, < u_k$, is an MCRD set in $G$, then each $D_i \subseteq D$, where $D_i = \{u_1, \ldots, u_i\}$, is an MCRD set in the connected subgraph $H_i$ induced by $\{x : x \in X, x \leq u_i\} \cup \{y : y \in N(x), x \leq u_i\}$.

Proof. If not then there is a set $D' \subseteq X, D' = \{v_1, \ldots, v_j\}, v_1 < \ldots, < v_j \leq u_i$, such that $|D'| = j < i$ and $D'$ is an MCRD set in $H_i$. Then $v_j$ is adjacent to right($u_i$) and, therefore, $N(u_i) \subseteq N(v_j)$, and $\{v_1, \ldots, v_j, u_{i+1}, \ldots, u_k\}$ is a red dominating set for $G$ of a smaller cardinality than $D$ which contradicts the minimality of $D$. □
We use induction on the index of a vertex \( x_i \in X \) to prove that for all \( x_i \in X \), \( \text{Count}[x_i] \) equals the number of MCRD sets in \( H_i \), where \( H_i \) is defined as in the above claim. The hypothesis is true for all \( x \in N(y_1) \), because \( \text{Count}[x] = 1 \) for all \( x \in N(y_1) \). Assume the hypothesis is true for all \( x < x_p, 1 < p \leq n_X \).

Let \( D \) be any MCRD set that contains \( x_p, D = \{ u_1, \ldots, u_p-1, u_p = x_p, \ldots, u_k \} \).

By the above claim, each \( D_i = \{ u_1, \ldots, u_i \}, 1 \leq i \leq k \), is an MCRD set in the connected subgraph \( H_i \). The number of MCRD sets that contain \( x_p \) in the connected subgraph \( H_p \) is equal to the sum, over all \( i \), of the number of MCRD sets \( D_i = \{ u_1, u_2, \ldots, u_i \} \) where \( u_i < x_p, \left( \text{left}(u_i) \leq \text{left}(x_p) - 1 \leq \text{right}(u_i) < \text{right}(x_p) \right) \), and \( \text{Label}[x_p] = \text{Label}[u_i] + 1 \) by Lemma 4.5. By the induction hypothesis, the number of such sets, for each \( i \), is equal to \( \text{Count}[u_i] \).

For all such vertices \( u_i \), where \( \text{left}(u_i) < \text{left}(x_p) \leq \text{right}(u_i) + 1 \leq \text{right}(x_p) \) and \( \text{Label}[x_p] = \text{Label}[u_i] + 1 \), \( x_p \) will be added to the set built in line 8 in Algorithm 1 when \( x = u_i \). Hence \( \text{Count}[x_p] \) is equal to the number of MCRD sets in the connected subgraph \( H_p \).

Therefore, by mathematical induction, \( \text{Count}[x] \) is equal to the number of MCRD sets that contain \( x \), for all \( x \in N(y_{n_Y}) \).

**Corollary 4.7.** The number of MCRD sets for an arbitrary convex bipartite graph can be computed in \( O(n_X + |E|) \) time.

To enumerate all MCRD sets, we perform the second stage of the enumeration process. Algorithm 2, presented below, performs a branch and bound search on vertices of \( X \) starting from a vertex with label \( k - 1 \) that is adjacent to \( y_{n_Y} \) going backwards up to a vertex with label 0. In the process we only add to the current output \( D \) vertices whose neighbourhood sets together with that of the previous vertex in \( D \) make a set of consecutive vertices in \( Y \) and are not contained in each other. Once we reach label 0, we output the set. As we will show, unlike general recursive backtracking techniques, the branch and bound algorithm does not generate failed partial solutions. Applying the algorithm to the graph in Figure 1 outputs the sets \( \{ x_8, x_4, x_2 \}, \{ x_7, x_4, x_2 \}, \{ x_7, x_3, x_2 \}, \) and \( \{ x_7, x_3, x_1 \} \).

**Theorem 4.8.** Algorithm 2 correctly enumerates all MCRD sets in linear space and linear delay.

**Proof.** We first show that each call to the function \( \text{GENERATE} \) in line 3 will result in at least one output. To see this we note that by statement (e) in Lemma 4.4, the set built in line 10 is always non-empty, which implies that with each recursive call to \( \text{GENERATE} \) the cardinality of \( D \) increases.

The output of the algorithm is an MCRD set by Lemma 4.5. The algorithm will output at least one MCRD set by the above paragraph.

Suppose there is an MCRD set \( T = \{ t_1, \ldots, t_k \} \) of cardinality \( k \) that is not output by the algorithm. Among all such sets, assume \( T \) has the longest sequence of vertices that agrees with an output of Algorithm 2, i.e., there is a red dominating set \( D = \{ d_1, \ldots, d_k \} \), that is output by the algorithm, \( t_i = d_i, a \leq i \leq k \), \( T \) and \( D \) have the smallest such index \( a \). Since the loop in line 1
Algorithm 2: Enumerating all MCRD sets for the convex bipartite graph $G = (X, Y, E)$

**Input:** The neighbourhood array $N$ of the convex bipartite graph $G = (X, Y, E)$, array $Label$, $k - 1$: as defined in Lemma 4.5

**Output:** All MCRD sets of $G$

1. for all $x$ such that $x \in \{x' : x' \in N(y_n), Label[x'] = k - 1\}$ do
2.   $D \leftarrow \{x\}$;
3.   GENERATE($D, x$);
4. end for
5. GENERATE($D, x$) // recursive function
6. if $Label[x] = 0$ then
7.   output $D$;
8. return
9. end if

// the next loop finds all candidate vertices to extend the MCRD set
10. for all $x_j \in \{x' : x' < x, left(x') \leq left(x) - 1 \leq right(x') < right(x), Label[x'] = Label[x] - 1\}$ do
11.   $D \leftarrow D \cup \{x_j\}$;
12.   GENERATE($D, x_j$);
13. end for
14. return

The algorithm needs space to represent $G$ via the array $N$, array $Label$, and each output $D$. Therefore, the space requirement is $O(n_X)$.

The time between each output is the time to go from a vertex with label $k - 1$ adjacent to $y_n$, down to a vertex with label 0 adjacent to $y_1$, choosing those vertices in line 10. Since with each recursive call to GENERATE in line 12 the cardinality of $D$ increases, therefore, the delay is $O(k)$. □

5. Conclusion

We have shown that the red dominating set problem is NP-complete for perfect elimination bipartite graphs. We have shown that $O(d^k)$ is a tight upper bound on the number of minimum red dominating sets in bipartite graphs, where $d$ is the maximum degree in $Y$ and $k$ is the minimum cardinality of a red dominating set. We have presented an algorithm to enumerate all minimum cardinality red dominating sets in convex bipartite graphs in linear space and linear
delay. One advantage of that algorithm is that its preprocessing step takes only linear time. The preprocessing step has been used to calculate the number of minimum cardinality red dominating sets in a convex bipartite graph. An interesting extension to the results presented in this paper is to explore whether the two stage labelling branch and bound enumeration technique is general enough to be applied to other enumeration problems while maintaining linearity. Because convex bipartite graphs are not symmetric, it may be worthwhile studying the enumeration of blue domination for convex bipartite graphs.

References

[1] Karp, R.M.: Reducibility Among Combinatorial Problems. In: Miller R.E., Thatcher J.W., Bohlinger J.D. (eds) Complexity of Computer Computations. The IBM Research Symposia Series. Springer, Boston, MA 85-103(1972)

[2] Chvatal, V.: A greedy heuristic for the set-covering problem. Mathematics of Operations Research 4(3), 233-235 (1979)

[3] Abu-Khzam, F. N., Mouawad A. E., Liedloff M.: An exact algorithm for connected red-blue dominating set. J. Discrete Algorithms 9, 252-262 (2011)

[4] Weihe, K.: Covering trains by stations or the power of data reduction. Proc. of the 1st Conference on Algorithms and Experiments, ALEX, 1-8 (1998)

[5] Dekel, E., Sahni, S.: A Parallel Matching Algorithm for Convex Bipartite Graphs and Applications to Scheduling. J. Parallel and Distributed Computing 1. 185-205 (1984)

[6] Golovach, P. A., Heggernes, P., Kanté, M. M., Kratsch, D., Villanger, Y.: Enumerating minimal dominating sets in chordal bipartite graphs. Discrete Applied Mathematics 199, 30-36 (2016)

[7] Zhou, J., Bu, C.: The enumeration of spanning tree of weighted graphs. J Algebr. Comb. August, 231-236 (2020)

[8] Lin, M., Chen, C.: Counting independent sets in tree convex bipartite graphs. Discrete Applied Mathematics 218, 113-122 (2017)

[9] Lin, M.: Counting independent sets and maximal independent sets in some subclasses of bipartite graphs. Discrete Applied Mathematics 251, 236–244 (2018)

[10] Damaschke, P., Müller, H., Kratsch D.: Domination in convex and chordal bipartite graphs. Information Processing Letters 36, 231-236 (1990)

[11] Booth, K.S., Lueker, G.S.: Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. Journal of Computer and System Sciences 13(3), 335-379 (1976)
[12] Brandstädt, A, Le, V. B., Spinrad, J. P.: Graph Classes: A Survey. SIAM, (1999)

[13] Golumbic, M. C.: Graphs. North-Holland, Amsterdam, Academic Press, New York (1980)

[14] Berge, C.: Graphs. North-Holland, Amsterdam, (1985)