AMO - Advanced Modeling and Optimization, Volume 13, Number 3, 2011

The 2-Neighbourhood Covering Problem on Permutation Graphs

Akul Rana*, Anita Pal‡ and Madhumangal Pal†

* Department of Mathematics, Narajole Raj College
Narajole, Paschim Medinipur- 721 211, INDIA.
e-mail: rana_akul@rediffmail.com

‡ Department of Mathematics,
National Institute of Technology Durgapur,
Durgapur-713209, INDIA.
e-mail: anita.buie@gmail.com

† Department of Applied Mathematics with Oceanology and Computer Programming,
Vidyasagar University,
Midnapore-721 102, INDIA.
e-mail: mmpalvu@gmail.com

Abstract

Given a simple graph $G = (V, E)$ and a fixed positive integer k. A vertex w is said to be k-neighbourhood covers an edge (u, v) if $d(u, w) \leq k$ and $d(v, w) \leq k$, where $d(u, v)$ denotes the shortest distance between the vertices u and v. A set $D \subseteq V$ is called a k-neighbourhood covering set if every edge in E is k-neighbourhood covered by some vertices of D. The k-neighbourhood covering problem is to find a k-neighbourhood covering set of minimum cardinality in G. This problem is NP-complete for general graphs, even it remains NP-complete for chordal graphs. Here, $O(n + \overline{m})$ time algorithm is presented to solve the 2-neighbourhood covering problem on permutation graphs, where n is the number of vertices in G and $\overline{m}$ is the number of edges in the complement of G. A dynamic programming approach is used to solve the problem.

***AMO - Advanced Modeling and Optimization. ISSN: 1841-4311**

Keywords: *Design of algorithms, analysis of algorithms, permutation graph, k-neighbourhood covering.*

1 Introduction

A dominating set of a graph $G = (V, E)$ is a subset D of V such that every vertex not in D is adjacent to some vertex in D. The concept of domination in graph theory arises naturally from the facility location problem in operations research. Depending on the different requirements of various location problems, domination has many variants, e.g., independent domination, connected domination, total domination, edge domination, k-domination, etc. The k-neighbourhood covering problem is a variation of well studied domination problem. A vertex x, *k-dominates* another vertex y if $d(x, y) \leq k$ where $d(x, y)$ is the shortest distance between the vertices x and y. A vertex z, *k-neighbourhood covers* (k-NCs, for short) an edge (x, y) if $d(x, z) \leq k$ and $d(y, z) \leq k$, i.e., the vertex z, k-dominates both x and y. Conversely, if $d(x, z) \leq k$ and $d(y, z) \leq k$, the edge (x, y) is said to be k-neighbourhood covered by the vertex z. A *k-neighbourhood covering set* of the graph $G = (V, E)$ is a subset D of the vertex set V, so that every edge in E is k-neighbourhood covered (k-NC) by at least one vertex of D. The *k-neighbourhood covering problem* is the problem of selecting a k-neighbourhood covering set of a graph G with the minimum cardinality. The minimum cardinality of a k-neighbourhood covering set of G is called the *k-neighbourhood covering number* and is denoted by $\rho(G, k)$. Finding a k-neighbourhood covering number of an arbitrary graph is known to be NP-complete [10]. Polynomial time algorithms are, however, available for some restricted classes such as interval graphs [3, 18], block graphs [13].

A class of perfect graphs which is practically important and mathematically interesting is the class of permutation graphs. Permutation graphs are perfect since they are cocomparability graphs. In this paper, a special case of k-neighbourhood covering problem on permutation graphs is considered. For the case, we take $k = 2$ i.e., 2-neighbourhood covering problem is solved on permutation graphs. A formal description of the problem is as follows. Let $G = (V, E)$ be a permutation graph with $V = \{1, 2, \ldots, n\}$ and $|E| = m$. A vertex z, 2-NCs an edge $(x, y) \in E$ if $d(x, z) \leq 2$ and $d(y, z) \leq 2$. The problem is to find a set of vertices D with minimum cardinality such that every edge in E is 2-neighbourhood covered by at least one member in D.

The permutation graph is defined below.

Definition 1 *Let $G = (V, E)$ be an undirected graph with vertices $V = \{1, 2, \ldots, n\}$ and their exists a permutation $\pi = \{\pi(1), \pi(2), \ldots, \pi(n)\}$ on the set $\{1, 2, \ldots, n\}$. The graph G is said to be a permutation graph if for all $i, j \in V$, $(i, j) \in E$ if and only if*

$$(i - j)(\pi^{-1}(i) - \pi^{-1}(j)) < 0,$$

where $\pi^{-1}(i)$ denotes the position of the number i in $\pi = \{\pi(1), \pi(2), \ldots, \pi(n)\}$, for each $i \in V$.

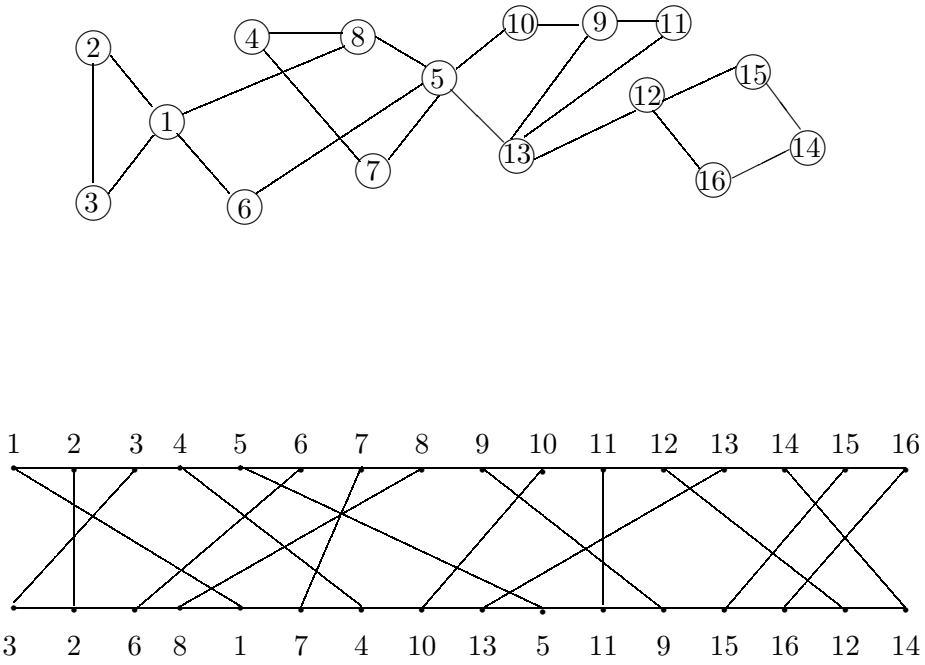


Figure 1: A permutation graph G and its corresponding matching diagram.

A permutation graph can also be visualized by its corresponding matching diagram. The matching diagram consists of two horizontal parallel lines, called the top channel and the bottom channel. The numbers $1, 2, \ldots, n$ are assigned on the top channel, in their order, from left to right, and for each $i = 1, 2, \ldots, n$ the number $\pi(i)$ on the bottom channel are assigned just below the number i on the top channel. Then, for each $i \in V$, a straight line is drawn joining two i's, one on the top channel and other on the bottom channel. The same number i is used to label the resulting line segment. Note that, the line segment i intersect the line

segment j if and only if i and j appear in reversed order in π. That is, the line segments i and j intersect if and only if the vertices i and j of the corresponding permutation graph are adjacent. This is the same as the criterion for the vertices i and j of the permutation graph to be adjacent. Therefore, an intersection graph of the lines of a matching diagram is exactly the corresponding permutation graph. Figure 1 shows a permutation graph and its corresponding matching diagram. Throughout the paper we refer to this example whenever necessary for illustration.

For a given permutation graph in the form of an adjacency matrix or an adjacency list, one can construct the matching diagram in $O(n^2)$ time [21]. Henceforth, it is assumed that a matching diagram is given for the input graph, the permutation is stored in the array $\pi(i)$, $i = 1, 2, \ldots, n$ and the inverse permutation of π is stored in array $\pi^{-1}(i)$, $i = 1, 2, \ldots, n$. The array $\pi^{-1}(i)$ can be computed in $O(n)$ time from the array $\pi(i)$. In the matching diagram $\pi^{-1}(i)$ gives the position of i on the bottom channel. In Figure 1, $\pi^{-1}(7)$ gives the position of 7 on the bottom channel which is 6.

For better illustration of the proposed approach, the matching diagram of the permutation graph is used. The words 'line' and 'vertex' are used interchangeably throughout this paper to refer a member of V. Two dummy vertices 0 and $n + 1$ are added to V such that $\pi(0) = 0$ and $\pi(n + 1) = n + 1$, for convenience. Throughout this paper it is assumed that all graphs are connected, finite, simple and undirected.

1.1 Review of previous work

Permutation graphs have been a favorite graph class for researchers designing polynomial time domination algorithms since early eighties. Many researchers [1-2, 4-8, 15-17, 19-21] have been devoted to the study of permutation graphs. Pnueli *et al.* [19] described an $O(n^3)$ time algorithm for testing if a given undirected graph G is a permutation graph. Spinrad [21] improved the above result by deriving an $O(n^2)$ algorithm for orienting comparability graphs.

There has been a growing interest in the development of efficient sequential algorithm for domination problems in graphs. An entire issue was dedicated to domination problems on graphs containing a wealth of results and references; see [12]. It is known that dominating set problem on arbitrary graphs is NP-complete [10]. The k-neighbourhood domination problem was first introduced in [9]. Hwang and Chang [13] gave a linear time sequential algorithm for k-domination problem on block graphs. For $k = 1$, Lehel *et al.* [15] have presented a linear time

algorithm for computing $\rho(G, 1)$ for an interval graph G. Hwang and Chang [14] have presented a linear time algorithm to compute $\rho(G, 1)$ for a strongly chordal graph G provided that strong elimination ordering is given. Hwang *et al.* [14] also proved that k-neighbourhood covering problem is NP-complete for chordal graphs. For $k = 2$, Mondal *et al.* [18] have presented a linear time algorithm for computing $\rho(G, 1)$ for an interval graph G. Recently, Barman *et al.* [3] solved k-neighbourhood covering problem on interval graphs. Many algorithms are available for solving several dominating set problems on permutation graphs [1-2, 4-8, 15-17, 19-20].

1.2 Application

Permutation graph have many applications in scheduling problem. See for example [6] where permutation graphs are used to describe the memory requirements of a number of programs at a certain time. See [11] for other practical applications of permutation graphs. The k-neighbourhood covering problem is an important problem in graph theory and it has many applications in real life problems. The domination is a natural model for location problems in operations research, networking, etc. The k-neighbourhood covering problem is a variant of the domination problem.

1.3 Main result

To the best of our knowledge, k-neighbourhood covering problem is not solved for $k \geq 2$ on permutation graphs. Using structural properties of the matching diagram of permutation graph, an $O(n+\overline{m})$ time algorithm is designed to solve 2-neighbourhood covering problem on permutation graphs.

1.4 Organization of the paper

The rest of the paper is organized as follows. In Section 2, some useful notations and observations are introduced. In Section 3, some important properties related to 2-neighbourhood covering set are provided for understanding the algorithm. In Section 4, an $O(n + \overline{m})$ time algorithm is designed for solving 2-neighbourhood covering problem on a permutation graph. The time complexity is also calculated in this section. Finally, in Section 5, a concluding remarks is given.

2 Notations and preliminaries

In this section, some notations are introduced which are used in the rest of the paper.

For each $i \in V$, we define *left span* of i, denoted by $ls(i)$, to be the line segment with lowest number on the top channel, say k_1, such that all edges $(x, y) \in E$, $k_1 \leq x, y \leq i$ are 2-NC by i, i.e., $ls(i) = k_1$, if k_1 is the lowest vertex covered by i and there is no vertex j, $k_1 \leq j \leq i$ such that $d(i, j) > 2$. For example, $ls(8) = 1$, $ls(4) = 4$, etc.

Likewise, the *right span* of the vertex i is the line segment with highest number on the top channel, say k_2, such that all edges $(x, y) \in E$, $i \leq x, y \leq k_2$ are 2-NC by i and is denoted by $rs(i)$, i.e., $rs(i) = k_2$, if k_2 is the highest vertex covered by i and there is no vertex j, $i \leq j \leq k_2$ such that $d(i, j) > 2$. For example, $rs(8) = 8$, $rs(4) = 8$, etc.

A line i is *left* to the line j if $i < j$ and $\pi^{-1}(i) < \pi^{-1}(j)$. Similarly, a line i is *right* to the line j if $i > j$ and $\pi^{-1}(i) > \pi^{-1}(j)$. $Right(i)$ is the set of lines right to the line segment i and $Left(i)$ is the set of lines left to the line segment i.

Define two arrays $T(i)$ and $B(i)$ as follows.

$T(i)$ is the highest vertex on the top channel intersecting the line segment i such that $T(i) > i$. If there is no such vertex then $T(i) = i$. For example, $T(1) = 8$, $T(2) = 3$, $T(5) = 13$ and $T(10) = 10$.

$B(i)$ is the vertex with highest position on the bottom channel intersecting i such that $\pi^{-1}(B(i)) > \pi^{-1}(i)$. If there is no such vertex then $B(i) = i$. For example, $B(1) = 1$, $B(3) = 1$ and $B(13) = 12$.

$T(i)$ and $B(i)$ plays an important role regarding the solution procedure of our problem. From above definitions, it is observed that $T(i) \geq i$ and $\pi^{-1}(B(i)) \geq i$.

$L(i)$ is defined to be the set of lines between the vertices i and $T(i)$ on top channel not intersecting the line segment $T(i)$. For example, $L(1) = \{2, 3, 6\}$. $R(i)$ denotes the set of lines between the vertices $T(i)$ and $T(B(T(i)))$ on top channel not intersecting the line segment $B(T(i))$. For example, $R(1) = \{9\}$ and $R(2) = \{4\}$.

From above definitions, it is clear that $L(i)$ and $R(i)$ are subsets of $Left(i)$ and $Right(i)$ respectively.

Using $Right(i)$ and $Left(i)$, one can compute $L(i)$ and $R(i)$.

Define, r_i as the smallest line of $R(i)$ and l_i is the smallest line of $L(i)$.

For each i, let m_i be the line segment with smallest position on bottom channel to the right of $T(i)$ i.e., $m_i = \{k : k \in Right(T(i))\ and\ \pi^{-1}(k)\ is\ minimum\}$. For example, $m_1 = 10$, $m_3 = 6$,

etc.

The symbol $x \sim y$ is used to denote the adjacency between the vertices x and y, i.e., there is an edge between x and y. It should be noted that $x \sim y$ implies $y \sim x$.

To compute all $T(i)$, the lines on top and bottom channels of the matching diagram are scanned, using x and y for index of the top and bottom channels respectively. The following algorithm computes all $T(i)$.

Algorithm TOP

Input: The permutation graph $G = (V, E)$ with its permutation representation.

Output: All $T(i)$, $i = 1, 2, \ldots, n$.

Initially $x = y = n$.

Step 1: While $\pi^{-1}(x) \leq y$ and $y \geq 1$ then

$T(\pi(y)) = x$, $y = y - 1$

endwhile

Step 2: While $\pi^{-1}(x) > y$ do

$x = x - 1$

endwhile

goto step 1.

end TOP

Clearly, algorithm TOP takes $O(n)$ time to compute all $T(i)$, $i = 1, 2, \ldots, n$.

A similar algorithm can be designed to compute all $B(i)$, $i = 1, 2, \ldots, n$.

Lemma 1 *All Right*(i)*,* $i = 1, 2, \ldots, n$ *can be computed in* $O(n + \overline{m})$ *time.*

Proof: To compute $Right(i)$ for all $i \in V$, line segments on the top channel are scanned from 0 to n and maintain a list L of scanned lines in increasing order of their positions on bottom channel. When the line $j \in V$ is scanned, find the first line in L with position greater than that of j on the bottom channel. Let this line be k (if exists). For each line i in L before k, add j to $Right(i)$ and insert j before k in L. This procedure takes $O(\sum_{i=0}^{n} |Right(i)|) = O(n + \overline{m})$ time to compute all $Right(i)$. □

For example, let $\pi = \{0, 3, 1, 4, 2\}$. Initially, $L = \emptyset$. First add 0 to L. Next, add 1 to L, after 0, since there exists no line in L with position > 3 (position of 1 on bottom channel). For $j = 3$, add 3 to L before 1 in L as $\pi^{-1}(1) > \pi^{-1}(3)$. Similarly, for $j = 4$, add 4 to L before 2

in L as $\pi^{-1}(2) > \pi^{-1}(4)$. Finally, $L = \{0, 3, 1, 4, 2\}$, $Right(0) = \{1, 2, 3, 4\}$, $Right(1) = \{2, 4\}$, $Right(2) = \emptyset$, $Right(3) = \{4\}$ and $Right(4) = \emptyset$.

Lemma 2 *All $Left(i)$, $i = 1, 2, \ldots, n$ can be computed in $O(n + \overline{m})$ time.*

Proof: Scan all the lines in the permutation diagram from $n+1$ down to 1 on the top channel and maintain a list K of scanned lines in descending order of their position on the bottom channel. When scan the line $j \in V$, find the first line in K with position less than that of j on bottom channel. Let this line be m (if exists). For each line i in K before m, add j to $Left(i)$ and insert j before m in K. This procedure takes $O(\sum_{i=0}^{n} |Left(i)|) = O(n + \overline{m})$ time to compute all $Left(i)$. □

For example, let $\pi = \{3, 1, 4, 2, 5\}$. Initially $K = \emptyset$. In this case first add 5 to K. Since there is no line in K with position less than that of 4, K becomes $\{5, 4\}$ and 4 is a member of $Left(5)$. Continuing this process, finally we get $K = \{5, 2, 4, 1, 3\}$, $Left(5) = \{4, 3, 2, 1\}$, $Left(4) = \{3, 1\}$, $Left(3) = \emptyset$, $Left(2) = \{1\}$, $Left(1) = \emptyset$.

From the above lemmas, it is clear that the lines in $Right(i)$ obtained from Lemma 1 are in increasing order, i.e., if $Right(i) = \{i_1, i_2, \cdots, i_r\}$ then $i_1 < i_2 < \cdots < i_r$.

Also, the lines in $Left(i)$ obtained from Lemma 2 are in descending order, i.e., if $Left(i) = \{j_1, j_2, \cdots, j_r\}$ then $j_1 > j_2 > \cdots > j_r$.

3 Some results

Before presenting the proposed algorithm for finding 2-neighbourhood covering set of a permutation graph, some important results relating to 2-neighbourhood covering set of the permutation graph are proved.

Observe that, a vertex $z \in V$, 2-NCs an edge $(x, y) \in E$, if z intersects at least one of the line segments x and y or if both the line segments x and y intersects a line segment which intersects the line z. So, the aim is to find a set of lines D with minimum cardinality such that for every $(x, y) \in E$, there exists at least one member of D that intersects at least one of the lines x and y or both x and y intersects a line which intersects the line z.

The following result due to Folklore is true for any permutation graph.

Lemma 3 *Let G be a permutation graph and u, v and w be three vertices of G such that $u < v < w$. If u is adjacent to w, then v is adjacent to at least one of u or w.*

Based on the above result, the following lemmas are proved that are the foundation of our algorithm.

Lemma 4 *The vertex $B(i)$, 2-NCs all the edges $(x, y) \in E$, where $B(i) \leq x, y \leq T(B(i))$.*

Proof: To prove this lemma, it is suffices to prove that $d(B(i), x) \leq 2$ for all $B(i) \leq x \leq T(B(i))$. Let $B(i) \leq x \leq T(B(i))$. Then, x is adjacent to at least one of $B(i)$ and $T(B(i))$. Recall that, $B(i)$ is the line with highest position on bottom channel intersecting i. Also, $T(B(i))$ is the highest line on the top channel intersecting $B(i)$. Now, if x is adjacent to $B(i)$ then $d(B(i), x) = 1 < 2$. Again, if x is adjacent to $T(B(i))$ then, also $d(B(i), x) \leq 2$ as $B(i) \sim T(B(i)) \sim x$. Hence the lemma holds. □

From the above lemma, it is observed that the vertex $B(i)$, 2-NCs all the edges $(x, y) \in E$ such that $B(i) \leq x, y \leq T(B(i))$. But, since the graph is connected, if $T(B(i)) < n$ then there exists at least one edge $(x, y) \in E$ such that $B(i) \leq x \leq T(B(i))$ and $y > T(B(i))$. These edges are not covered by $B(i)$. To cover these edges, we select the vertex $B(m_{B(i)})$ as another member of D.

Lemma 5 *The vertex $B(m_{B(i)})$, 2-NCs all the edges $(x, y) \in E$ such that $B(i) \leq x \leq T(B(i))$ and $y > T(B(i))$.*

Proof: From definition, it follows that $m_{B(i)}$ is the line segment with smallest position on the bottom channel that lies to the right of $T(B(i))$, i.e., $T(B(i)) < m_{B(i)}$ and $\pi^{-1}(T(B(i))) < \pi^{-1}(m_{B(i)})$. Since, $(x, y) \in E$ such that $B(i) \leq x \leq T(B(i))$, $y > T(B(i))$ and $x \leq B(m_{B(i)}) < y$, y must intersects $B(m_{B(i)})$. Therefore, the vertex $B(m_{B(i)})$, 2-NCs all edges $(x, y) \in E$ such that $B(i) \leq x \leq T(B(i))$ and $y > T(B(i))$ as $B(m_{B(i)}) \sim y \sim x$. □

Lemma 6 *If $R(i) = \emptyset$, then the vertex $T(i)$, 2-NCs all the edges $(x, y) \in E$, where $i \leq x, y \leq T(B(T(i)))$.*

Proof: Clearly, $(i, T(i)) \in E$. Let $(x, y) \in E$ such that $i \leq x, y \leq T(B(T(i)))$. There are three cases may arise.

Case 1: $i \leq x, y \leq T(i)$. Then x and y are adjacent to at least one of i and $T(i)$. Therefore, $T(i)$, 2-NCs the edge (x, y), since, $T(i) \sim i \sim x$ or $T(i) \sim x$.

Case 2: $T(i) \leq x, y \leq T(B(T(i))$. Since $(T(i), B(T(i))) \in E$ and $R(i) = \emptyset$, $T(i)$, 2-NCs the edge (x, y), since, $T(i) \sim B(T(i)) \sim x$ and $T(i) \sim B(T(i)) \sim y$.

Case 3: $i \leq x \leq T(i)$ and $T(i) < y \leq T(B(T(i))$. In this case the line x must intersects the line $T(i)$. Therefore, $T(i)$, 2-NCs the edge (x, y), as $T(i) \sim x \sim y$.

Hence the lemma holds. □

It is easy to verify that, if $R(i) \neq \emptyset$ and $T(B(T(i))) = n$ then also the vertex $T(i)$, 2-NCs all the edges $(x, y) \in E$, where $i \leq x, y \leq T(B(T(i)))$.

Lemma 7 *If $R(i) \neq \emptyset$ and $T(B(T(i))) < n$, then the vertex $T(i)$, 2-NCs all edges in between the lines i and $r_i - 1$.*

Proof: Recall that, r_i is the smallest line of $R(i)$, that is, r_i is the smallest line in between the vertices $T(i)$ and $T(B(T(i))$ on the top channel that does not intersect the line $B(T(i))$. Since $(r_i, B(T(i))) \notin E$, $(r_i, T(i)) \notin E$. Therefore $T(i)$ does not 2-NC the vertex r_i.

Now, $T(i)$ covers all the edges $(x, y) \in E$, $i \leq x, y \leq T(i)$ as $i \sim x$ or $i \sim T(i) \sim x$. Also, $T(i)$ covers all the edges $(x, y) \in E$, $T(i) \leq x, y \leq r_i - 1$ as $T(i) \sim B(T(i)) \sim x$ and $T(i) \sim B(T(i)) \sim y$. Lastly, if $(x, y) \in E$ such that $i \leq x \leq T(i)$ and $T(i) \leq y \leq r_i - 1$ then $T(i)$ intersects at least one of x and y since $(x, y) \in E$. Hence the lemma holds. □

From the above lemmas, it is observed that the right span of $B(i)$ is $T(B(i))$. If $R(i) \neq \emptyset$ and $T(B(T(i))) < n$, the right span of $T(i)$ is $r_i - 1$. If $R(i) = \emptyset$ or $R(i) \neq \emptyset$ and $T(B(T(i))) = n$ then the right span of $T(i)$ is $T(B(T(i)))$.

Lemma 8 *If $L(i) = \emptyset$, then the line $B(T(i))$ has the maximum span among all the lines those have left span i.*

Proof: The span of the vertex $B(T(i))$ is $T(B(T(i))) - i$. Observe that no line right to $T(i)$ on the top channel can have left span i. Let j be a line other than $B(T(i))$ such that $j \leq T(i)$ and has left span i. To prove the lemma, it is suffices to prove that $T(B(T(i))) \geq rs(j)$. There are two cases may arise:

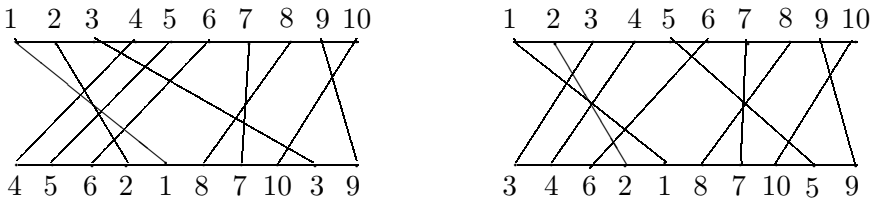
Case 1: j intersects $B(T(i))$ and Case 2: j does not intersect $B(T(i))$.

If j intersects $B(T(i))$ then $B(T(i)) < j \leq T(i)$. Therefore, $\pi^{-1}(B(T(i))) \geq \pi^{-1}(j)$ and hence $T(B(T(i))) \geq T(j)$. But in this case $rs(j) = T(j)$. Hence the result follows.

If j does not intersect $B(T(i))$ then $j < B(T(i))$, since, otherwise $j = B(T(i))$ which contradicts our assumption. Therefore $T(j) < T(B(T(i)))$. There are two subcases may occur:

Subcase 1: j intersects $T(i)$. In this case, $rs(j) = T(j)$ and since $T(j) < T(B(T(i))$, $T(B(T(i))) > rs(j)$.

Subcase 2: j does not intersect $T(i)$. In this case, right span of j is $\pi^{-1}(B(j))$, if $R(j) = \emptyset$ (by Lemma 6). Since j does not intersect $B(T(i))$ and $T(i)$, clearly, $T(B(T(i))) > rs(j)$. Hence the lemma holds. □



$B(l_1)$ has maximum span $T(1)$ has maximum span

Figure 2: Illustration of Lemma 9

Lemma 9 *If $L(i) \neq \emptyset$, then either $T(i)$ or $B(l_i)$ has the maximum span among all the lines those have left span i.*

Proof: Similar to the previous lemma. □

4 The algorithm

The main basic idea of Algorithm 2-NC is described below. The proposed algorithm proceeds by covering edges from left to right as on permutation diagram.

Let D be the 2-neighbourhood covering set of the given permutation graph G. If $L(1) = \emptyset$ then select $B(T(1))$ as the first member of D, otherwise, $T(1)$ or $B(l_1)$ will be the first member of D. Let the first member of D be t. If $rs(t) = n$ then stop. Otherwise, replace 1 by $rs(t) + 1$ or m_t. This selection is continued till right span of newly selected vertex of D, becomes greater than or equal to n.

A formal description of the algorithm is given in Algorithm 2-NC.

Algorithm 2-NC

Input: A permutation graph $G = (V, E)$ and its permutation representation.

Output: A minimum cardinality 2-neighbourhood covering set D in G.

Initially $D = \emptyset$ (empty set) and $i = 1$.

Step 1: Compute the arrays $T(i), B(i)$ for each vertex $i \in V$.

Step 2: Compute the sets $L(i), R(i)$ for each vertex $i \in V$.

Step 3: If $L(i) = \emptyset$ then

$t = B(T(i))$, $D = D \cup \{t\}$ and goto Step 5. (Lemma 8)

elseif $T(B(l_i)) > rs(T(i))$ then

$t = B(l_i)$, $D = D \cup \{t\}$ and goto Step 5. (Lemma 9)

else

$t = T(i)$, $D = D \cup \{t\}$, goto Step 4. (Lemma 6)

endif

Step 4: If $rs(t) = n$ then

stop

else

$i = rs(t) + 1$, goto Step 3.

endif.

Step 5: If $rs(t) = n$ then

stop

else

$i = m_t$, goto Step 3.

endif.

end 2-NC

The proof of the correctness of the algorithm directly follows from the Lemmas 8 and 9.

Theorem 1 *Algorithm 2-NC finds a minimum cardinality 2-neighbourhood covering set on permutation graphs in $O(n + \overline{m})$ time.*

Proof: Each of $T(i)$ and $B(i)$ can be computed in $O(n)$ time. Computation of $Right(i)$ and $Left(i)$ requires $O(n + \overline{m})$ steps (Lemma 1). Using the sets $Right(i)$ and $Left(i)$, the sets $L(i)$ and $R(i)$ can be computed in constant time. From the set $Right(i)$, m_i can be computed in constant time. Therefore, overall time complexity is $O(n + \overline{m})$. □

5 Concluding remarks

Although many domination algorithms have been proposed for permutation graphs, to date, no algorithm is available for solving k-domination problem on permutation graphs for $k \geq 2$. In this paper, an $O(n + \overline{m})$ time sequential algorithm is presented to solve 2-neighbourhood covering problem on permutation graphs. This approach can be extended to solve 2-neighbourhood covering problem on trapezoid graphs which properly contain both permutation graphs and interval graphs.

References

[1] Arvind, K. and Pandu Regan, C., Connected domination and steiner set on weighted permutation graphs, *Information Processing Letters*, **41** (1992), 215-220.

[2] Atallah, M. J., Manacher, G. K. and Urrutia, J., Finding a minimum independent dominating set in a permutation graph, *Discrete Appl. Math.*, **21** (1988), 177-183.

[3] Barman, S., Pal, M. and Mondal, S., The k-neighbourhood-covering problem on interval graphs, *Intern. J. of Computer Math.*, (2010), DOI: 10.1080/00207160802676570.

[4] Corneil, D. G. and Stewart, L. K., Dominating sets in perfect gaphs, *Discrete Math.*, **86** (1990), 145-164.

[5] Brandstadt, A. and Kratsch, D., On domination problems on permutation and other graphs, *Theoret. Comput. Sci.*, **54** (1987), 181-198.

[6] Even, S., Pnueli, A. and Lampel, A., Permutation graphs and transitive graphs, *J. Assoc. Comput. Mach.*, **19** (1972), 400-410.

[7] Corneil, D. G. and Stewart, L. K., Dominating sets in perfect gaphs, *Discrete Math.*, **86** (1990), 145-164.

[8] Farber, M. and Keil, J., M., Domination in permutation garphs, *J. Algorithms*, **6** (1985), 309-321.

[9] Fink, J. F. and Jacobson, M. S., On n-domination and n dependence and forbidden subgraphs, in *Procedings of the 5th international conference*, Wiley, New York (1984), 301-311.

[10] Garey, M. R. and Jhonson, D. S., *Computers and Interactibility: A guide to the theory of NP completeness*, W. H. Freeman and Company, San Fransisco, 1979.

[11] Golumbic, M. C., *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.

[12] Hedetniemi, S. T. and Laskar, R. C., Special volume: Topics on domination, *Discrete Math.*, **86(1-3)**(1990).

[13] Hwang, S., F. and Chang, G., J., The k-neighbor domination problem, *Europian J. Oper. Res.*, **52**(1991), 373-377.

[14] Hwang, S., F. and Chang, G., J., k-neighbourhood covering and independence problem for chordal graphs, *SIAM J. Discrete Math.*, **11(4)**(1998), 633-643.

[15] Lahel, J. and Tuza, Z., Neighborhood perfect graphs, *Discrete Math.*, **61** (1986), 93-101.

[16] Liang, Y., Rhee, C., Dhall, S., K. and Lakshmivarahan, S., A new approach for domination problem on permutation graphs, *Information Processing Letters*, **37** (1991), 219-224.

[17] Liang, Y. D., Lu, C. L. and Tang, C. Y., Efficient domination on permutation graphs and Trapezoid graphs, *LNCS* , **1276** (1997), 232-241.

[18] Mondal, S., Pal M. and Pal T. K., An optimal algorithm to solve 2-neighbourhood covering problem on interval graphs, *Intern. J. Computer Math.*, **79** (2002), 189-204.

[19] Pnueli, A., Lampel, A. and Even, S., Transitive orientation of graphs and identification of permutation graphs, *Canadian J. Math.*, **23** (1971), 160-175.

[20] Rhee, C., Liang, Y. D., Dhall, S. K. and Lakshmivarahan, S., An $O(n+m)$-time algorithm for finding a minimum-weight dominating set in a permutation graph, *SIAM J. Comput.*, **25(2)** (1996), 404-419.

[21] Spinrad, J. R., On comparability and permutation graphs, *SIAM J. Comput.*, **14** (1985), 658-670.