Research Article

An Efficient Algorithm to Solve the Conditional Covering Problem on Trapezoid Graphs

Akul Rana,[1] Anita Pal,[2] and Madhumangal Pal[3]

[1] *Department of Mathematics, Narajole Raj College, Paschim Medinipur Narajole 721 211, India*

[2] *Department of Mathematics, National Institute of Technology, Durgapur 713209, India*

[3] *Department of Applied Mathematics with Oceanology and Computer Programming,*
  Vidyasagar University, Midnapore 721 102, India

Correspondence should be addressed to Akul Rana, rana_akul@rediffmail.com

Let $G = (V, E)$ be a simple connected undirected graph. Each vertex $v \in V$ has a cost $c(v)$ and provides a positive coverage radius $R(v)$. A distance d_{uv} is associated with each edge $\{u, v\} \in E$, and $d(u, v)$ is the shortest distance between every pair of vertices $u, v \in V$. A vertex v can cover all vertices that lie within the distance $R(v)$, except the vertex itself. The conditional covering problem is to minimize the sum of the costs required to cover all the vertices in G. This problem is NP-complete for general graphs, even it remains NP-complete for chordal graphs. In this paper, an $O(n^2)$ time algorithm to solve a special case of the problem in a trapezoid graph is proposed, where n is the number of vertices of the graph. In this special case, $d_{uv} = 1$ for every edge $\{u, v\} \in E$, $c(v) = c$ for every $v \in V(G)$, and $R(v) = R$, an integer >1, for every $v \in V(G)$. A new data structure on trapezoid graphs is used to solve the problem.

1. Introduction

Let $G = (V, E)$ be a finite, connected, undirected, and simple graph where $V = \{1, 2, \ldots, n\}$ is the set of vertices and E is the set of edges with $|E| = m$. A distance d_{uv} is associated with each edge $\{u, v\} \in E$, and $d(u, v)$ is the shortest distance between every pair of vertices $u, v \in V$. Each vertex $v \in V$ has a cost $c(v)$ and provides a positive coverage radius $R(v)$. The vertex $u \in V$ is covered by a vertex $v \in V$ if and only if $v \neq u$ and $d(v, u) \leq R(v)$. The *conditional covering problem* (CCP, for short) is to minimize the sum of the costs of the vertices required to cover all the vertices in V. One closely related problem to the CCP is the total dominating set problem, which is a special case of the CCP in which all distances and coverage radii are equal to 1. The total dominating set problem is NP-complete even on bipartite graphs [1]. Since the total dominating set is a special case of CCP, the CCP is also NP-complete for general graphs.

In this paper a special case of CCP on trapezoid graphs is considered. For the case, we take $d_{uv} = 1$ for all $\{u, v\} \in E$, $c(v) = c$ for all $v \in V$, and the coverage radius $R(v) = R$, a fixed positive integer > 1 for all $v \in V$. That is, all edge weights are unity, coverage radii are uniform, and vertex costs are constant.

Trapezoid graphs were first investigated in [2, 3]. Trapezoid graphs are the intersection graphs of finite collections of trapezoids between two parallel lines [3]. Both the interval graphs and the permutation graphs form subclasses of the trapezoid graphs [3] and trapezoid graphs are subclasses of cocomparability graphs [4].

A trapezoid diagram consists of two horizontal lines (top channel and bottom channel) and a set of trapezoids. A trapezoid i between these lines has four corner points $\text{tl}(i)$, $\text{tr}(i)$, $\text{bl}(i)$, and $\text{br}(i)$ which represent the top left, top right, bottom left, and bottom right corner points of the trapezoid i, respectively. Let $T = 1, 2, \ldots, n$ denote the set of trapezoids in the trapezoid diagram for a trapezoid graph $G = (V, E)$ with $|V| = n$. For trapezoid i, $\text{tl}(i) < \text{tr}(i)$ and $\text{bl}(i) < \text{br}(i)$ holds. A graph, $G = (V, E)$, is a trapezoid graph when a trapezoid diagram exists with trapezoid set T, such that each vertex $i \in V$ corresponds to a trapezoid $i \in T$ and an edge $\{i, j\} \in E$ if and only if trapezoids i and j intersect within the trapezoid diagram. There is an $O(n^2)$ time recognition algorithm for trapezoid graphs [5]. Moreover, this algorithm also computes a trapezoid diagram, if the given graph G is a trapezoid graph. Any trapezoid diagram can be transformed into another trapezoid diagram with all corner points distinct, when the two representations still correspond to the same trapezoid graph. Without loss of generality, it is assumed that no two trapezoids share a common endpoint. The points on each horizontal line of the trapezoid diagram are labeled with distinct integers between 1 and $2n$ from left to right. The terms vertex and trapezoid are used interchangeably whenever the context is unambiguous. In this paper, it is assumed that a trapezoid diagram is given and the trapezoids are labeled in increasing order of their top right corner point, that is, $i < j$ if and only if $\text{tr}(i) < \text{tr}(j)$.

As mentioned earlier, the CCP on arbitrary graphs is NP-complete. Several important graph problems that are NP-hard in general case have polynomial time algorithms for trapezoid graphs. Solving these problems on special graphs is of great importance because solutions to a problem on special classes of graphs might shed new light on solutions for the problem on general graphs. Also, solutions to problems on special graphs can be used to approximate solutions to the problems on general graphs. Many real world problems can be modeled as special graphs, and hence simpler solutions are needed compared to the ones for general graphs.

1.1. Review of Previous Work

Moon and Chaudhry [6] were the first to address CCP as constrained facility location model. They present an integer programming model for this problem. Efficient heuristic algorithms for the CCP have been extensively studied in [7–10]. Moon and Papayanopoulos [11] consider one variation of CCP on trees. Complexity issues regarding the placement of facilities in CCP were considered in [1, 12, 13]. For the CCP on a path graph with uniform coverage radii, Lunday et al. [13] presented a linear time algorithm to solve the unweighted cost CCP and an $O(n^2)$ dynamic programming algorithm to solve the weighted cost CCP. Horne and Smith [1] studied the weighted cost CCP on path and extended star graphs with nonuniform coverage radius and developed an $O(n^2)$ dynamic programming algorithm. In an another article, Horne and Smith [12] considered weighted cost CCP on tree graphs and presented an $O(n^4)$ dynamic programming algorithm. Recently, Sivan et al. [14] have shown

that algorithm for CCP on paths, presented by Horne and Smith, is erroneous and further present a correct $O(n^3)$ algorithm for the same. They also propose an $O(n^2)$ algorithm for the CCP on paths when all vertices are assigned unit costs and further extend this algorithm to interval graphs without an increase in time complexity. In our previous work [15], we solved the CCP on unweighted interval graphs with uniform coverage radius which runs in $O(n)$ time.

1.2. Application

Trapezoid graphs are applied in various fields, including modeling channel routing problems in VLSI design [3] and identifying the optimal chain of nonoverlapping fragments in bioinformatics [16]. See also [17] for other practical applications of trapezoid graphs. The application area of the CCP includes locating facilities in distribution systems, emergency systems, communication systems, energy supply systems, and so forth.

1.3. Main Result

To the best of our knowledge, no algorithm is available to solve the CCP on trapezoid graphs. In this paper, a special case of the CCP on trapezoid graphs is considered with unit edge weights, uniform coverage radius $R > 1$ and uniform vertex costs. An $O(n^2)$ time algorithm is designed to solve the problem.

1.4. Organization of the Paper

The rest of this paper is organized as follows. Section 2 establishes basic notations and some properties of trapezoid graphs. In Section 3, a new data structure, called caterpillar, is constructed and important properties of caterpillar are proved. In Section 4, an $O(n^2)$ time algorithm is designed for solving conditional covering problem on a trapezoid graph. The time complexity is also calculated in this section. Finally, Section 5 contains some conclusions.

2. Preliminaries

This section presents the preliminaries on which the desired algorithm depends.

A trapezoid i is *left* to the trapezoid j if $i < j$ and i does not intersect j in the trapezoid diagram. Similarly, a trapezoid i is *right* to the trapezoid j if $i > j$ and i does not intersect j in the trapezoid diagram.

For a vertex $v \in V$ of a graph $G = (V, E)$, the *open neighborhood* of v are defined as $N(v) = \{u : \{u, v\} \in E\}$. For each $v \in V$, $N(v)$ can be computed in $O(n^2)$ time.

$T(i)$ is the highest trapezoid intersecting the trapezoid i such that $\text{tr}(T(i)) > \text{tr}(i)$ or $T(i) = i$ if such trapezoid does not exist.

$B(i)$ is the trapezoid with highest bottom right point intersecting the trapezoid i such that $\text{br}(B(i)) > \text{br}(i)$ or $B(i) = i$ if such trapezoid does not exist.

For example, as occurred in Figure 1, $T(1) = 2$, $T(3) = 6$, $B(1) = 1$, $B(2) = 3$, and so forth.

The symbol $x \sim y$ is used to indicate the adjacency between the vertices x and y, that is, there is an edge between x and y. It should be noted that $x \sim y$ implies $y \sim x$.
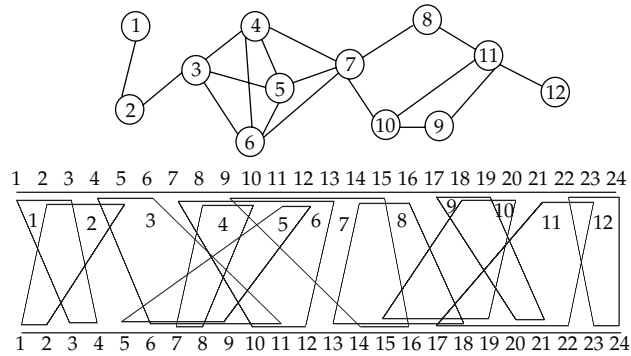
Figure 1: A trapezoid graph and one of its trapezoid representation.

The following properties of trapezoid graphs are useful for our proofs.

Lemma 2.1. *For $i < j$ two vertices i and j are adjacent in G if and only if either $\mathrm{tl}(j) < \mathrm{tr}(i)$ or $\mathrm{bl}(j) < \mathrm{br}(i)$.*

Proof. Let us consider the case $i < j$ and $\mathrm{tl}(j) < \mathrm{tr}(i)$. By the ordering of trapezoids, $\mathrm{tr}(i) < \mathrm{tr}(j)$ as $i < j$. Then $\mathrm{tl}(j) < \mathrm{tr}(i) < \mathrm{tr}(j)$. Therefore, in the corresponding trapezoid representation, the trapezoid i intersects the trapezoid j, so i is adjacent to j in G. Now consider the case when $i < j$ and $\mathrm{bl}(j) < \mathrm{br}(i)$. In this case, $\mathrm{bl}(j) < \mathrm{br}(i)$ and $tr(i) < \mathrm{tr}(j)$. Therefore in trapezoid representation, the chord joining the corner points $\mathrm{bl}(j)$ and $\mathrm{tr}(j)$ of j always intersects i, that is, the trapezoids i and j have a nonempty intersection in the trapezoid representation and hence i is adjacent to j in G. Conversely, let i is adjacent to j in G and $i < j$. Therefore in the trapezoid representation i and j have a nonempty intersection and $\mathrm{tr}(i) < \mathrm{tr}(j)$. This is possible only when either $\mathrm{tl}(j) < \mathrm{tr}(i)$ or $\mathrm{bl}(j) < \mathrm{br}(i)$ or both. Hence the lemma holds.   □

Lemma 2.2. *For any three vertices $u, v, w \in V$, if $u < v < w$ and $\{u, w\} \in E$ then either, $\{u, v\} \in E$ or $\{v, w\} \in E$.*

Proof. As $u < v < w$, so by the ordering of trapezoids we have $\mathrm{tr}(u) < \mathrm{tr}(v) < \mathrm{tr}(w)$. The condition $\{u, w\} \in E$ implies that either $\mathrm{tl}(w) < \mathrm{tr}(u)$ or $\mathrm{bl}(w) < \mathrm{br}(u)$. Therefore any trapezoid v, satisfying the inequality $\mathrm{tr}(u) < \mathrm{tr}(v) < \mathrm{tr}(w)$, either intersects u or w or both, that is, either $\{u, v\} \in E$ or $\{v, w\} \in E$.   □

The following result is used to test whether two trapezoids are adjacent or not.

Lemma 2.3. *For $i < j$ two vertices i and j of the trapezoid graph G are not adjacent if and only if $\mathrm{tr}(i) < \mathrm{tl}(j)$ and $\mathrm{br}(i) < \mathrm{bl}(j)$.*

Proof. Let $\mathrm{tr}(i) < \mathrm{tl}(j)$ and $\mathrm{br}(i) < \mathrm{bl}(j)$. Since always $\mathrm{tl}(i) < \mathrm{tr}(i), \mathrm{bl}(i) < \mathrm{br}(i), \mathrm{tl}(j) < \mathrm{tr}(j)$, and $\mathrm{bl}(j) < \mathrm{br}(j)$ holds. It follows that $\mathrm{tl}(i) < \mathrm{tr}(i) < \mathrm{tl}(j) < \mathrm{tr}(j)$ and $\mathrm{bl}(i) < \mathrm{br}(i) < \mathrm{bl}(j) < \mathrm{br}(j)$. This means that in the trapezoid representation, the trapezoid i lie on the left of the trapezoid j, implying that in the graph the vertices i and j are not adjacent. Conversely, in the trapezoid graph the vertices i and j are nonadjacent only when in the trapezoid representation the trapezoids i and j have no common region, that is, only when i is completely on the left of j, that is, $\mathrm{tr}(i) < \mathrm{tl}(j)$ and $\mathrm{br}(i) < \mathrm{bl}(j)$. Hence the lemma follows.   □

To compute all $T(i)$, the trapezoids on the top and bottom channels of the matching diagram are scanned. The following lemma computes all $T(i)$.

Lemma 2.4. *For all $u \in V$, $T(u)$ can be computed in $O(n)$ time.*

Proof. Recall that $T(u)$ is the trapezoid having highest top right point among all trapezoids intersecting the trapezoid u. Let t be the last top right point in the trapezoid representation. Then $t = 2n$. Let $t_1 = b_1 = 2n$, t_2 be the tl(k) and let b_2 be the bl(k) where k is the trapezoid whose top right point is t. Initially, we scan the points on the top channel between t_2 and t_1 and on the bottom channel between b_2 and b_1. For each such point p, let i be its corresponding trapezoid. If $T(i)$ is not found yet, then $T(i)$ is q, where q corresponds to the trapezoid whose top right point is t. Continue the same process with the following adjustment on t, t_1, b_1, t_2, and b_2 until all points are scanned. Let t_1 be $t_2 - 1$, $b_1 = b_2 - 1$, t be the previous top right point before t, $t_2 = \min\{t_2, \text{tl}(k)\}$ and $b_2 = \min\{b_2, \text{bl}(k)\}$ where k corresponds to the trapezoid whose top right point is t.

Clearly, algorithm TOP takes $O(n)$ time to compute all $T(i)$, $i = 1, 2, \ldots, n$.
Similarly, all $B(i)$, $i = 1, 2, \ldots, n$ can be computed in $O(n)$ time. □

The difficulty of the problem comes from the fact that the neighbor of a vertex of trapezoid graphs can be spread everywhere in the two channels of the trapezoid diagram. To overcome this difficulty a caterpillar is constructed from the given trapezoid diagram.

3. Scheme to Construct a Caterpillar

A *caterpillar* is a tree in which a single path is incident to every edge. The single path is called the *spine* of the caterpillar. The vertices not on the spine of a caterpillar are leaves of the caterpillar.

The caterpillar $T^*(G)$ from a given trapezoid diagram is constructed as follows.

First, compute two arrays $T(i)$ and $B(i)$ for all $i \in V$. Then construct a path such that in the path the distance between any two vertices in the trapezoid graph is shortest. Initialize $i = 1$. If $T(i) = n$ or $B(T(i))) \neq B(i)$ or $BTBT(i) \neq BTB(i)$ ($BTBT(i)$ is the composition of B and T, that is, $BTBT(i) = B(T(B(T(i)))))$ then i is adjacent to $T(i)$, otherwise i is adjacent to $B(i)$. Let i be adjacent to the vertex t (either $T(i)$ or $B(i)$). Replace i by t and repeat the process until t is adjacent to n. This path is the spine of the caterpillar. Let $P(G)$ be the spine of the caterpillar. The *length* of a vertex u in the spine is the number of edges in the path from the vertex 1 to u and is denoted by $\text{dist}_1(u)$. We define the length of the vertex 1 as 0. The vertex on the spine at length l is denoted by u_l^*. The set of leaves of the caterpillar at length l is denoted by X_l. Next, compute the leaves of the caterpillar as follows:

$$X_l = N(u_l^*) - \{u_{l-1}^*\},$$

$$X_{l-1} = N(u_{l-1}^*) - \{u_{l-2}^*, u_l^*\} \cup X_l,$$

$$X_{l-i} = N(u_{l-i}^*) - \{u_{l-(i+1)}^*, u_{l-(i-1)}^*\} \cup X_{l-(i-1)} \cup X_{l-(i-2)}, \quad \text{for } i = 2, 3, \ldots, l-1, \tag{3.1}$$

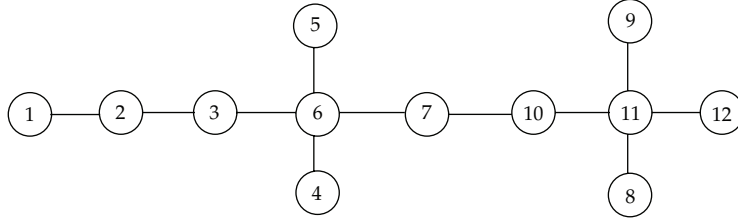$$X_0 = N(1) - \{u_1^*\} \cup X_1 \cup X_2.$$

Figure 2: The caterpillar of the graph of Figure 1.

Finally, connect u and u_i^* by an edge, for all $u \in X_i$. The resulting graph is the caterpillar $T^*(G)$. The caterpillar $T^*(G)$ for the graph of Figure 1 is shown in Figure 2.

If $u \in X_i$ then we define $\text{dist}_1(u) = i$, $i = 0, 1, \ldots, l$. The elements of the set X_i are the leaves of u_i^*, $i = 0, 1, \ldots, l$. Let v_i be any leave at length i and and Y_i be the collection of elements of the caterpillar at length i, That is, $v_i \in X_i$ and $Y_i = X_i \cup \{u_i^*\}$.

Lemma 3.1. *The caterpillar $T^*(G)$ of a connected trapezoid graph G exists and is unique for a given trapezoid diagram.*

Proof. The existence of the caterpillar $T^*(G)$ follows from the construction of caterpillar. Since for a given trapezoid diagram the order of a vertex $i \in V$ is unique, $B(i)$ and $T(i)$ are also unique. Thus the caterpillar is unique for any trapezoid graph G.   □

3.1. Properties of the Caterpillar

The vertices of $P(G)$ satisfy the following results.

By the construction of the caterpillar it is easy to see that, if $u, v \in P(G)$ such that $|\text{dist}_1(u) - \text{dist}_1(v)| > 1$ then $\{u, v\} \notin E$.

Lemma 3.2. *If $u, v \in P(G)$ such that $\text{dist}_1(u) - \text{dist}_1(v) > 1$ then $u > v$.*

Proof. Let u and v be two arbitrary vertices of $P(G)$ such that $\text{dist}_1(u) - \text{dist}_1(v) > 1$. Observe that there exists a vertex $w \in P(G)$ in between u and v such that $v \sim w$. Then there are two cases that may arise.

Case 1. $\{w, u\} \in E$. In this case $\text{tr}(u) > \text{tr}(v)$, since else $\text{tr}(u) < \text{tr}(v)$ implies that $\{v, u\} \in E$ which contradicts Lemma 3.1. Hence $u > v$.

Case 2. $\{w, u\} \notin E$. It is clear that $\text{dist}_1(v) < \text{dist}_1(w) < \text{dist}_1(u)$. Then, since $\{u, v\} \notin E$ (Lemma 3.1) and $\{v, w\} \in E$, in the trapezoid diagram u lie on the right of v. Therefore $u > v$.
                                                                                                                 □

Lemma 3.3. *If $u \in P(G)$ and $v \in X_i$ such that $\text{dist}_1(v) - \text{dist}_1(u) > 2$ then $\{u, v\} \notin E$.*

Proof. Without loss of generality, let $u = u_l^*$ and $v \in X_{l+i}$ where $i > 2$. Then $(v, u_{l+i}^*) \in E$. There exists a path in $T^*(G)$ such that $u_l^* \sim u_{l+1}^* \sim \cdots \sim u_{l+i}^* \sim v$.

Since v is adjacent to u_{l+i}^*, if v is adjacent to u_l^* then v has highest top right and bottom right point intersecting u_l^* in the trapezoid diagram. But u_{l+1}^* has highest top right or bottom right point intersecting u_l^*, a contradiction.

Hence v is not adjacent to u_l^* and the lemma follows. □

The following lemma can be proved by arguments similar to those used in the above lemma.

Lemma 3.4. *If $u \in X_l$ and $v \in X_{l+3}$ such that $\{u,v\} \in E$ then $\{v, u_{l+1}^*\} \in E$.*

Lemma 3.5. *If $u, v \in T^*(G)$ such that $dist_1(u) - dist_1(v) > 3$ then $\{u, v\} \notin E$.*

Proof. There are four cases that may arise.

Case 1. $u, v \in P(G)$. Since $dist_1(u) - dist_1(v) > 3$ and $u, v \in P(G)$, by the construction of the caterpillar, it follows that $\{u, v\} \notin E$.

Case 2. $u, v \notin P(G)$. If $\{u, v\} \notin E$, then u or v must be a member of $P(G)$ which is a contradiction.

Case 3. $u \in P(G)$ and $v \notin P(G)$. Since $dist_1(u) > dist_1(v)$, $u \in P(G)$ and $v \notin P(G)$, therefore $\{u, v\} \notin E$.

Case 4. $u \in P(G)$ and $v \notin P(G)$. If $\{u, v\} \notin E$, then v must be a member of $P(G)$ which is a contradiction. Hence the lemma follows. □

The following lemma follows from the properties of the caterpillar.

Lemma 3.6. *If u, v, and w be three vertices of the caterpillar of length l_1, l_2, and l_3, respectively such that $l_1 < l_2 < l_3$. If $d(u, w) \leq R$ then, $d(v, w) \leq R$.*

Observe that, if $\{u, v\} \in E$ such that $u \in X_i$ and $v \in X_{i+3}$ then there does not exist $w \in X_{i-j}$, $j > 2$ with $\{w, u\} \in E$. Also, there does not exist $t \in X_{i+j}$, $j > 2$ with $\{v, t\} \in E$.

Lemma 3.7. *For any two vertices $u, v \in P(G)$, the distance between u and v is minimum along $P(G)$.*

Proof. Let u and v be two arbitrary vertex of $P(G)$ such that $d(u, v) = m$. We will show that there does not exist any path from u to v with $d(u, v) < m$. Without loss of generality let $u = u_i^*$ and $v = u_{i+m}^*$. For $m = 1$, the possible paths from u to v are $u_i^* \sim u_{i+1}^*$ and $u_i^* \sim v_i \sim v_{i+1} \sim u_{i+1}^*$. Therefore, the distance between u and v is minimum along $P(G)$, that is, the lemma is true for $m = 1$. For $m = 2$, the possible paths from u to v are $u_i^* \sim u_{i+1}^* \sim u_{i+2}^*, u_i^* \sim v_i \sim v_{i+1} \sim u_{i+1}^* \sim u_{i+2}^*$, $u_i^* \sim v_i \sim v_{i+1} \sim v_{i+2} \sim u_{i+2}^*$, and $u_i^* \sim v_i \sim v_{i+2} \sim u_{i+2}^*$. That is, the lemma is true for $m = 2$ also.

Let the lemma be true for $m = k$. Then a shortest path from u to v is $u_i^* \sim u_{i+1}^* \sim \cdots \sim u_{i+k}^*$. Since v_{i+k+1} may adjacent to the leaves v_{i+k}, v_{i+k-1} and v_{i+k-2}, the possible shortest paths from u to v are $u_i^* \sim u_{i+1}^* \sim \cdots \sim u_{i+k}^* \sim u_{i+k+1}^*$ and $u_i^* \sim u_{i+1}^* \sim \cdots \sim u_{i+k-2}^* \sim v_{i+k-2} \sim v_{i+k+1} \sim u_{i+k+1}^*$. Hence $u_i^* \sim u_{i+1}^* \sim \cdots \sim u_{i+k}^* \sim u_{i+k+1}^*$ is a shortest path. □

The following definitions are crucial in designing our algorithm.

Definition 3.8. Two vertices u, v ($u < v$) in a set form a dominating pair if and only if $d(u, v) \leq R$ and there is no other vertex w in the set with $d(v, w) \leq R$.

Definition 3.9. Three vertices u, v, and w ($u < v < w$) in a set form a dominating triple if and only if $d(u, v) \leq R$, $d(v, w) \leq R$ and and there is no other vertex w in the set with $d(v, w) \leq R$.

Input: A trapezoid graph with a trapezoid diagram and coverage radius R.
Output: A conditional covering set D in V.
   Initially $D = \phi$ and $i = 0$.
Step 1: Compute the sets $T(u)$, $B(u)$ and $N(u)$ for each vertex $u \in V$.
Step 2: Construct the caterpillar. Compute the vertices of the spine and the sets X_i for
         $i = 0, 1, 2, \ldots, l$; where l is the highest length of the caterpillar.
Step 3: Compute $p = i + R$.
         If $d(u, v_j) \leq R$, $j > p + 2$ and for all $u \in Y_i$ then
               $D = D \cup \{v_j\}$ such that j is maximum, replace i by $j - 1$, $p = i + R$, goto Step 4;
         elseif $X_p = \emptyset$ or $u \in N(u_{p-1}^*) \cup N(u_{p-2}^*)$ for all $u \in X_p$ then
               $D = D \cup \{u_p^*\}$, replace i by p, $p = i + R$, goto Step 5;
         else $D = D \cup \{u_{p-1}^*\}$, replace i by $p - 1$, $p = i + R$, goto Step 5;
         endif
Step 4: If $p > l$ then
               $D = D \cup \{v\}$ such that $d(v, v_j) \leq R$ and finish;
         elseif $p = l$ then
               $D = D \cup \{u_p^*\}$ and finish;
         else go to step 6;
         endif;
Step 5: If $p > l$ then
               $D = D \cup \{v\}$ such that $d(v, u) \leq R$ and finish; // u is the latest selected member
                                                                            of D //
         elseif $p = l$ then $D = D \cup \{u_p^*\}$ and finish;
         else go to step 6;
         endif;
Step 6: If $d(u, v_j) \leq R$, $j > p + 2$ and for all $u \in Y_i$ then
               $D = D \cup \{v_j\}$ such that j is maximum, replace i by $j - 1$, $p = i + R$, goto Step 7;
         elseif $X_p = \emptyset$ or $u \in N(u_{p-1}^*) \cup N(u_{p-2}^*)$ for all $u \in X_p$ then
               $D = D \cup \{u_p^*\}$, replace i by p, $p = i + R$, goto Step 7;
         else $D = D \cup \{u_{p-1}^*\}$, replace i by $p - 1$, $p = i + R$, goto Step 7;
         endif;
Step 7: If $p > l$ then
               D is the required solution;
         elseif $p = l$ then
               If $X_p = \emptyset$ or $u \in N(u_{p-1}^*) \cup N(u_{p-2}^*)$ for all $u \in X_p$ then
                     D is the required solution;
               else $D = D \cup \{u_p^*\}$ and finish;
               endif;
         else go to Step 8;
         endif;
Step 8: If $d(u, v_j) \leq R$, $j > p + 2$ and for all $u \in Y_i$ and $j - 1 + R > l$ then
               $D = D \cup \{v_j\}$ and finish;
         elseif $p + R > l$ then
               $D = D \cup \{u_p^*\}$ and stop;
         elseif $p + R = l$ and $X_{p+R} = \emptyset$ or $u \in N(u_{p+R-1}^*) \cup N(u_{p+R-2}^*)$ for all $u \in X_{p+R}$ then
               $D = D \cup \{u_p^*\}$ and stop;
         else goto next Step;
         endif;
Step 9: If $u \in N(u_{p-1}^*) \cup N(u_{p-2}^*)$, for all $u \in X_p$ then
               replace i by $p + 1$ and goto Step 3;
         else replace i by p and goto Step 3;
         endif;

Algorithm 1: Algorithm CCP.

4. Minimum Cardinality Conditional Covering

The algorithm proceeds by covering the vertices of the caterpillar from left to right. The algorithm selects the members of conditional covering set in such a way that each member belongs to a dominating pair or a dominating triple.

A formal algorithm is presented in Algorithm 1.

4.1. Proof of Correctness

By the process of computation of D, it is easy to see that every vertex of the caterpillar is covered by a vertex in D. Also, any member of D is included in a dominating pair or triple. Therefore each vertex of the induced subgraph of D in G is covered by at least one other vertex in D.

Observe that the set D consists entirely of dominating pairs and triples of vertices. If not, then there exists a minimum cardinality conditional covering set D that does not contain entirely is covering pairs and triples. So, there must exists a set of four vertices x, y, z, w such that $d(x, y) \leq R$, $d(y, z) \leq R$ and $d(z, w) \leq R$. Now, if a vertex t be chosen as a member of D instead of z such that $\mathrm{dist}_1(t) > \mathrm{dist}_1(w)$ and $d(t, w) \leq R$ then also each vertex of the induced subgraph of D in G is covered by at least one other vertex in D. Again, the set of vertices covered by x, y, z, w is a proper subset of the set of vertices covered by x, y, w, t. Since the aim is to minimize $|D|$, later selection is better. Again, at each stage, the algorithm selects the member of D of maximum length of the caterpillar, such that the newly selected member covers the vertices of maximum length on the right and covers all uncovered vertices on the left. Therefore the solution set D is of minimum cardinality.

Theorem 4.1. *Algorithm 1 finds a minimum cardinality conditional covering set on trapezoidgraphs in $O(n^2)$ time.*

Proof. Step 1 can be computed using $O(n)$ time. Step 2 requires $O(n^2)$ time. Step 3 can be done in constant time. So the caterpillar is constructed in $O(n^2)$ time. In worst case Step 4 is to be repeated for $O(n)$ time. Therefore, the total running time of the algorithm CCP is $O(n^2)$. □

5. Concluding Remarks

To date no algorithms have been proposed for CCP on trapezoid graphs. In this paper, we proposed an $O(n^2)$ time algorithm for solving CCP on trapezoid graphs. The proposed algorithm was designed based on the strategy: first, construct the caterpillar and then solve the problem on the constructed caterpillar. Since the class of permutation graphs is a subclass of trapezoid graphs, this problem on permutation graphs can also be solved in $O(n^2)$ time by our algorithm. It would be interesting to develop more efficient algorithms of low complexity for these problems. A future study could also continue to examine one of several practical variations of the CCP.

References

[1] J. A. Horne and J. C. Smith, "Dynamic programming algorithms for the conditional covering problem on path and extended star graphs," *Networks*, vol. 46, no. 4, pp. 177–185, 2005.

 [2] D. G. Corneil and P. A. Kamula, "Extensions of permutation and interval graphs," in *Proceedings of the 18th Southeastern International Conference on Combinatorics, Graph Theory, and Computing (Boca Raton, Fla., 1987)*, Congr. Numer. 58, pp. 267–275, 1987.

 [3] I. Dagan, M. C. Golumbic, and R. Y. Pinter, "Trapezoid graphs and their coloring," *Discrete Applied Mathematics*, vol. 21, no. 1, pp. 35–46, 1988.

 [4] F. Cheah and D. G. Corneil, "On the structure of trapezoid graphs," *Discrete Applied Mathematics*, vol. 66, no. 2, pp. 109–133, 1996.

 [5] T. H. Ma and J. P. Spinrad, "On the 2-chain subgraph cover and related problems," *Journal of Algorithms*, vol. 17, no. 2, pp. 251–268, 1994.

 [6] I. D. Moon and S. S. Chaudhry, "An analysis of network location problems with distance constraints," *Management Science*, vol. 30, no. 3, pp. 290–307, 1984.

 [7] S. S. Chaudhry, "New heuristics for the conditional covering problem," *Opsearch*, vol. 30, pp. 42–47, 1993.

 [8] S. S. Chaudhry, I. D. Moon, and S. T. McCormick, "Conditional covering: greedy heuristics and computational results," *Computers & Operations Research*, vol. 14, no. 1, pp. 11–18, 1987.

 [9] V. Lotfi and I. D. Moon, "Hybrid heuristics for conditional covering problems," *International Journal of Modelling and Simulation*, vol. 17, no. 3, pp. 185–190, 1997.

[10] I. D. Moon, "Conditional covering: worst case analysis of greedy heuristics," *Journal of the Korean Operations Research and Management Science Society*, vol. 15, no. 2, pp. 97–104, 1990.

[11] I. D. Moon and L. Papayanopoulos, "Facility location on a tree with maximum distance constraints," *Computers & Operations Research*, vol. 22, no. 9, pp. 905–914, 1995.

[12] J. A. Horne and J. C. Smith, "A dynamic programming algorithm for the conditional covering problem on tree graphs," *Networks*, vol. 46, no. 4, pp. 186–197, 2005.

[13] B. J. Lunday, J. C. Smith, and J. B. Goldberg, "Algorithms for solving the conditional covering problem on paths," *Naval Research Logistics*, vol. 52, no. 4, pp. 293–301, 2005.

[14] B. Sivan, S. Harini, and C. Pandu Rangan, "On conditional covering problem," *Mathematics in Computer Science*, vol. 3, no. 1, pp. 97–107, 2010.

[15] A. Rana, A. Pal, and M. Pal, "The conditional covering problem on unweighted interval graphs," *Journal of Applied Mathematics and Informatics*, vol. 28, pp. 1–11, 2010.

[16] M. I. Abouelhoda and E. Ohlebusch, "Chaining algorithms for multiple genome comparison," *Journal of Discrete Algorithms*, vol. 3, no. 2–4, pp. 321–341, 2005.

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