On Conditional Covering Problem

Balasubramanian Sivan, S. Harini and C. Pandu Rangan

Abstract. The Conditional Covering Problem (CCP) aims to locate facilities on a graph, where the vertex set represents both the demand points and the potential facility locations. The problem has a constraint that each vertex can cover only those vertices that lie within its *covering radius* and no vertex can cover itself. The objective of the problem is to find a set that minimizes the sum of the facility costs required to cover all the demand points. An algorithm for CCP on paths was presented by Horne and Smith (Networks, 46(4):177185, 2005). We show that their algorithm is wrong and further present a correct $O(n^3)$ algorithm for the same. We 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.

Mathematics Subject Classification (2000). Primary 05C85; Secondary 90C39.

Keywords. Conditional Covering Problem, Facility Location, Total Domination Problem, Paths, Interval Graphs.

1. Introduction

Consider a simple, undirected, connected graph $G = (V, E)$. The vertex set V denotes the set of sites / demand points that must be covered by some facility and facilities must be located only on the vertices of the graph. Each vertex $i \in V$ is associated with a positive cost of locating a facility at i, denoted by $c(i)$. Each edge (i, j) is associated with a positive length e_{ij}. Let D_{ij} be the shortest distance between i and j. Each vertex $i \in V$, is associated with a positive *covering radius* R_i such that, when we locate a facility at i, all the vertices within a distance of R_i from i (except i itself) are covered by i, i.e. all the vertices in the set $\{j | D_{ij} \leq R_i$ and $j \neq i\}$ are covered(No facility can cover itself). A set $S \subset V$ covers all the vertices in the set $\{j | \exists\ i \in S$ and $D_{ij} \leq R_i\}$.

The objective of the Conditional Covering Problem, CCP, is to minimize the sum of the costs of the facilities required to cover all the vertices in V. A *Conditional*

Covering set or CCS, is a set of vertices which covers all the vertices of the graph. The cost of a set S of vertices is given by $c(S) = \sum\limits_{i \in S} c(i)$. The CCP aims to find a CCS S, that minimizes $c(S)$.

In this paper, we give a counter example for the algorithm given in [5] for solving CCP on paths and provide a corrected $O(n^3)$ algorithm for the same in Section 2. Additionally, when all the vertices are assigned the same cost (taken to be unity without loss of generality), we give an $O(n^2)$ algorithm to solve the CCP on paths in Section 3 and show how to extend this algorithm to interval graphs without any increase in time-complexity in Section 4. Since, the algorithm proposed in [5] to solve CCP on paths is incorrect, it can be seen that the extension of the algorithm to solve the CCP in star-graphs and trees given in [4] is also incorrect.

The CCP arises as an underlying graph theoretic problem in many real life facility location problems. Consider the problem of locating rescue centers in a district. Each potential site is associated with a *covering radius* and a cost of constructing a facility there. Also no rescue center can help the site at which it is located in case of a calamity at that site. Hence every facility should be covered by another facility. This problem can be modeled as CCP on a graph. CCP can similarly be used to model distribution centers and demand points where transhipment between distribution centers is required [2]. In [1], an application involving the strategic location of facilities having minimum and maximum coverage radii such as missile defense systems is discussed.

The CCP is a generalized version of the Total Dominating Set problem. Let $A \subset V$ and $B \subset V$ be two sets of vertices. A dominates B if every vertex in B -A is adjacent to at least one vertex in A; A totally dominates B if every vertex in B is adjacent to at least one vertex in A. A is called a dominating set of the graph if A dominates V and is called a totally dominating set if it totally dominates V. The Total Dominating Set problem is a special case of the CCP when $R_i = 1$, $\forall i$ and all the edge lengths e_{ij} and facility location costs $c(i)$ are unity. Since the problem of finding the minimum Total Dominating Set in general graphs has been proved to be NP-Complete in the strong sense [9], it follows that CCP is NP-Complete on general graphs.

The CCP was first introduced by Moon and Chaudhry in [7] where an integer programming model for this problem was proposed and linear programming relaxation methods were applied to them. In [2], the authors consider seven greedy heuristics for solving CCP and provide computational results for the same. In [8], Moon and Papayanopoulos discuss a slight variation of CCP on tree graphs. In this problem, each demand point has a specific radius such that a facility has to be located in that radii. In [1] Smith et al. present an $O(n^2)$ algorithm for CCP on paths when the *covering radius* is uniform for all the vertices and arbitrary

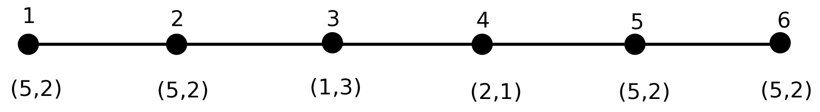

FIGURE 1. Diagram to depict the counter example. $(c(i), R_i)$ for each vertex i is given below the vertex.

positive costs are assigned to vertices. They also present an $O(n)$ time algorithm when the *covering radius* is uniform and cost is unity for all vertices of the path. In [5] Horne and Smith extend the $O(n^2)$ algorithm in [1] to the case when vertices are assigned arbitrary *covering radius.*

Here, we show that the algorithm in [5] is wrong. Let P=$123\ldots n$ be a path graph. The authors use the following equation to find the cost of the optimal CCS.

$$z^* = \underset{i \in V, g(i)=n}{Min} p(i)$$

where z^* is the cost of the optimal CCS, $g(i) = \text{Max}\{k|D_{ik} \leq R_i\}$ is the highest indexed vertex that i can cover i.e. the upper-reach of i and $p(i)$ is the cost required to optimally cover vertices 1 through i, by locating a facility at i and placing no facilities at vertices $i+1$, $i+2$, $\ldots n$. Consider the path in *Figure* 1. Their algorithm will consider the $p(i)$ values for all the vertices that have an upper-reach equal to n and choose the minimum among them. In this case, vertices 3, 5 and 6 have an upper-reach equal to n i.e. $g(3) = g(5) = g(6) = 6$. Hence the algorithm will compute the values $p(3)$, $p(5)$, $p(6)$ and declare the minimum among them as the cost of optimal solution to CCP. The algorithm used by them to compute $p(i)$ values, will compute these values as $p(3) = p(5) = 6$, $p(6) = 8$ and hence the cost of the optimal solution will be declared as 6. However, we can observe that the set $\{3, 4\}$ is the optimal CCS with a cost of three.

Their algorithm assumes that in any optimal CCS S, the highest indexed vertex in S will always cover the vertex n. However, in the above example shown by *Figure* 1, where the optimal CCS is $\{3, 4\}$, the highest indexed vertex among $\{3, 4\}$ is 4 and still 4 does not cover 6. Instead 6 is covered by 3. Similar techniques are used by them for solving CCP in extended star and trees in [5, 4] and consequently these algorithms also fail due to similar reasons. We also note that their algorithms are incorrect even when all the vertices have unit costs as shown in *Figure* 2.

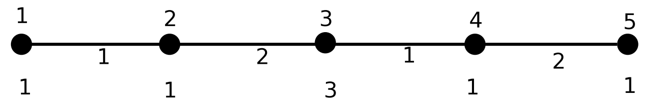

FIGURE 2. Diagram to depict the counter example when all the vertex costs are constant. R_i for each vertex i is given below the vertex. The cost given below each edge denotes the edge weight. The optimal solution is {3,4} but the algorithm in [5] will not even output a CCS.

2. Conditional Covering Problem on Paths

Let P be any path graph where $V = \{1, 2, \ldots, n\}$ and $E = \{(i, i+1)|1 \leq i \leq n-1\}$. The vertices are assigned arbitrary positive costs and the edges are assigned arbitrary positive lengths. Let P_i denote the path $123\ldots i$.
In this section, we give a polynomial time algorithm for solving the CCP on paths.

2.1. Preliminaries

From now on, we refer any two vertices i and j satisfying $i < j$ as, i being to the $left$ of j and j being to the right of i. Let (a, b) denote all the vertices in the path between the vertices indexed a and b not including a and b. Let $[a, b]$ denote all the vertices in the path between the vertices indexed a and b including a and b.

We borrow the following terminologies from [5]. For every vertex of the path, we define the *upper-reach* $g(i)$ of a vertex i as the largest indexed vertex lying within the *covering radius* of i i.e. $g(i) = \text{Max}\{k \in V|D_{ik} \leq R_i\}$ and *lower-reach* of a vertex i, given by $h(i)$, as defined as the smallest indexed vertex lying within the *covering radius* of i i.e. $h(i) = \text{Min}\{k|D_{ki} \leq R_i\}$.

The *protected cost* for vertex $i \in V$, denoted by $p(i)$, is the minimum cost to place a facility at vertex i and cover vertices 1 through i, with no facilities placed at vertices $i+1, \ldots, n$. (The protected cost for vertex 1 is taken to be infinity as it is not possible to cover 1 as per the requirements of protected cost. Also whenever it is not possible to meet the requirements of the protected cost for a particular vertex, we take it to be infinity). The *unprotected cost* $u(i)$ for vertex $i \in V$ is the minimum cost to locate a facility at vertex i and cover vertices 1 through $i-1$, with no facilities placed at vertices $i+1, \ldots, n$. For the unprotected cost, vertex i is not necessarily required to be covered by smaller indexed vertices.

Further, we use the following terminologies from [5] to compute $p(i)$ and $u(i)$ $\forall i \in V$.

In order to calculate the protected cost for a vertex i, all vertices between 1 and $i-1$ that can cover vertex i are identified in two separate sets as follows.

$GA_1(i) = \{k \in V \ : \ k+1 \leq i \leq g(k) \text{ and } h(i) > k\} \, \forall i \in V$
$GB_1(i) = \{k \in V \ : \ k+1 \leq i \leq g(k) \text{ and } h(i) \leq k\} \, \forall i \in V$

The set $GA_1(i)$ includes all vertices between 1 and $i-1$ that cover vertex i, but are not covered by vertex i.
The set $GB_1(i)$ includes all vertices between 1 and $i-1$ that cover vertex i, and are covered by vertex i.

In order to calculate the unprotected cost for vertex i, all vertices $k \in V$ that can, along with i, cover vertices $k+1$ through $i-1$, but such that k does not cover i, are identified in two separate sets as follows.

$GA_2(i) = \{k \in V \ : \ k+1 \leq h(i) \leq g(k)+1 \leq i\} \, \forall i \in V$
$GB_2(i) = \{k \in V \ : \ g(k)+1 \leq i \text{ and } h(i) \leq k\} \, \forall i \in V$

The set $GA_2(i)$ includes all vertices k between 1 and $i-1$ such that i and k do not cover each other, but do cover all vertices $k+1$ through $i-1$.
The set $GB_2(i)$ includes all vertices between 1 and $i-1$ that are covered by vertex i, but do not themselves cover vertex i.

2.2. Computing the optimal CCS

We give a property of the optimal solutions to CCP using the following lemma

Lemma 2.1. *Consider the CCP for any path P_i. Let r be any vertex such that $g(r) = i$ and let S^* be any optimal CCS of P_i. If $r \in S^*$, then atmost one vertex $v \in (r, i]$ will be in S^*.*

Proof. Assume on the contrary that there exists an optimal CCS S' such that $r \in S'$ and, two vertices j and j' between r and i are in S'. Assume without loss of generality that $h(j') > h(j)$. All the vertices to the right of j' till i are covered by r ($g(r) = i$) and all vertices to the left of j' till $h(j')$ are covered by j. Now, it can be clearly seen that the facility at j' does not cover any vertex that is not covered by r and j put together. Hence even with the removal of the facility at j' all the vertices of P_i continue to remain covered and consequently we have a solution of lesser cost which is a contradiction. □

For computing the optimal CCS, we define the following terms:

Let $\alpha(k,i) = \text{Min}\{h(k), h(i)\}$-1. Let $Z^*_j(i)$ denote the cost of optimally covering all the vertices in $[1, i]$ by using vertices from $[1, j]$. We define $Z^*_j(i)$ only when $j \geq i$. Note that $Z^*_1(1)$ is taken to be infinity. We can use the following recursion to compute $Z_j^*(i)$:

$$Z_j^*(i) = Min\left\{Z_i^*(i), \underset{k\in[i+1,j]:\ h(k)\leq i}{Min}\{c(k)+Z_i^*(h(k)-1)\}\right\} \tag{1}$$

Similar to the argument given in Lemma 2.1, we can see that, atmost one vertex can be chosen from $[i+1,j]$ for optimally covering $[1,i]$ with $[1,j]$. Hence, we have just two cases, namely

1. No vertex is chosen from $[i+1,j]$: In this case, we have to cover all the vertices in $[1,i]$ by locating facilities only in $[1,i]$. The optimal cost for doing this is $Z_i^*(i)$
2. Exactly one vertex k is chosen from $[i+1,j]$: All the vertices to the right of $h(k)-1$ are covered by the facility at k. So, we have to cover the vertices in $[1,h(k)-1]$ by locating facilities only in $[1,i]$. The optimal cost for doing this is $Z_i^*(h(k)-1)$

Both the above cases have been accounted for in (1)
Clearly $Z_n^*(n)$(cost of optimally covering vertices $[1,n]$ using vertices $[1,n]$.) is the cost of the optimal solution to CCP. For computing $Z_n^*(n)$, we can use the following equation:

$$Z_n^*(n) = \underset{i\in V:\ g(i)=n}{Min}\left\{p(i), \underset{k\in(i,n+1):\ h(k)\leq i \text{ and } g(k)<n}{Min}\{c(k)+u(i)\}\right\} \tag{2}$$

Consider all the vertices with an upper-reach equal to n (i.e. $g(i)=n$). Certainly, a facility must be located in one of these vertices for n to get covered. Because of Lemma 2.1, it follows that if we locate a facility at any vertex i for which $g(i)=n$, we can locate atmost one facility to the right of i. Hence, we have the following two cases,

1. We choose to locate no facility to the right of i : In this case, we must cover all the vertices in $[1,i]$ by locating facilities only in $[1,i-1]$ along with the facility already located at i. The optimal cost for doing this task is precisely $p(i)$.
2. We choose to locate exactly one facility k to the right of i (with $h(k)\leq i$ because otherwise there is no purpose in choosing k) : We restrict the facility k to be such that $g(k)<n$ because, the solution with $g(k)=n$ would have been accounted by the previous case i.e. the case where no facility is placed to the right of k and $g(k)=n$ and this would have been one of the solutions considered in the previous case. So, in the present case, we have to cover all vertices in $[1,\alpha(k,i)]$ by locating facilities in $[1,i-1]$ along with the facility already located in i. The optimal cost of doing this task is $Z_{i-1}^*\alpha(k,i)+c(i)$. But in this case, since $g(i)>g(k)$ and $i<k$, we have $h(i)<h(k)$ and hence $Z_{i-1}^*\alpha(k,i)+c(i)$ is the same as $u(i)$. Finally, in such a case we would incur a cost of $u(i)+c(k)$ as given in (2).

The following two *lemmas* give the equations to compute $p(i)$ and $u(i)$ recursively.

Lemma 2.2. *The following equation can be used to compute $p(i)$ when $p(k)$ and $u(k)$ is known $\forall k < i$ and $Z_b^*(a)$ is known $\forall$ a, b such that $a \leq b$, $b \leq i$-1.*

$$p(i) = c(i) + Min\left\{ \underset{k \in GA_1(i)}{Min} \{p(k)\}, \underset{k \in GA_1(i)}{Min} \left\{ u(k) + \underset{j \in (k,i):\ h(j) \leq k \ and \ g(j) < i}{Min} \{c(j)\} \right\}, \underset{k \in GB_1(i)}{Min} \{c(k) + Z_{k-1}^* \alpha(k,i)\} \right\}$$

Proof. Consider all the vertices k that have an upper-reach equal to i(i.e. $g(k) = i$). From the definition of protected cost, it follows that a facility must be located at one of these vertices for i to get covered. Because of Lemma 2.1, it follows that, we can place atmost one facility to the right of k. Hence, we have the following two cases.

1. We consider the first case when no facility is placed to the right of k. Vertex k covers i. Clearly, $k \in GA_1(i) \cup GB_1(i)$.
   If $k \in GA_1(i)$, the vertex k will be covered by a vertex to its left and k covers all vertices to its right till i. Hence, we should now cover vertices from 1 through k using vertices 1 through k(Note that a facility has already been placed at k). We account for all such $k \in GA_1(i)$ in the equation using the term, $c(i) + \underset{k \in GA_1(i)}{Min} p(k)$.
   If $k \in GB_1(i)$ then k and i cover all vertices between $min(h(i), h(k))$ and i. The vertices 1 through $\alpha(k,i)$ should be covered using vertices from 1 to $k-1$, with optimum cost, which is given by $Z_{k-1}^*(\alpha(k,i))$. We account for such $k \in GB_1(i)$ in the equation using the term $\underset{k \in GB_1(i)}{Min} \{c(k) + Z_{k-1}^*(k,i)\} + c(i)$.
2. We now consider the second case when exactly one facility namely j is placed to the right of k. Again k covers i. Clearly, $k \in GA_1(i) \cup GB_1(i)$.
   Let $k \in GA_1(i)$. Clearly, $h(j) \leq k$ (If $h(j) > k$ the purpose of j is lost) and $g(j) < i$ (If $g(j) \geq i$, such $j \in GA_1(i)$ and they would get accounted in the previous case). The only purpose of j is to cover k. Hence 1 through $k - 1$ remains to be covered optimally using 1 through k. This can be done in $u(k)$ cost. Such $k \in GA_1(i)$ are accounted in the equation using the term $\underset{k \in GA_1(i)}{Min} \{u(k) + \underset{j \in (k,i):\ h(j) \leq k \text{ and } g(j) < i}{Min} \{c(j)\}\} + c(i)$.
   If $k \in GB_1(i)$, then any j placed to the right of k will be useful only when j covers at least one vertex that k does not cover. This means that $h(j) < h(k)$. But since $j > k$, it follows that $g(j) \geq g(k)$ and hence $g(j) \geq i$. So $j \in GB_1(i)$. Such a solution would have been accounted for in the previous case and hence we need not redundantly consider such solutions again.

□

Lemma 2.3. *The following equation can be used to compute $u(i)$ when $p(k)$ and $u(k)$ is known $\forall k < i$ and $Z_b^*(a)$ is known $\forall$ a, b such that $a \leq b$, $b \leq i$-1.*

$$u(i) = Min\left\{ p(i), \underset{k \in GA_2(i)}{Min} \{p(k)\} + c(i), \underset{k \in GA_2(i)}{Min} \left\{ u(k) + \underset{j \in (k,i):\ h(j) \leq k \ and \ g(j) < h(i)-1}{Min} \{c(j)\} \right\} + \right.$$

$$c(i),$$
$$\left.\underset{k \in GB_2(i)}{Min} \left\{c(k) + Z^*_{k-1}\alpha(k,i)\right\} + c(i)\right\}$$

Proof. Sometimes, it might be cheaper to cover i also when covering $[1, i-1]$ than to leave it uncovered and so we have included $p(i)$ as the first term in the equation for $u(i)$.

Now we consider the cases where i is left uncovered. Consider all the vertices k that have $g(k) < i$ and $g(k) \geq h(i) - 1$ From the definition of unprotected cost, it follows that a facility must be located at one of these vertices for every vertex between k and i to get covered without covering i. Because of Lemma 2.1, it follows that, we can place atmost one facility to the right of k. Hence, we have the following two cases.

1. We consider the first case when no facility is placed to the right of k. Clearly, $k \in GA_2(i) \cup GB_2(i)$.
   If $k \in GA_2(i)$, the vertex k will be covered by a vertex to its left and k along with i cover all vertices between them. We account for all such $k \in GA_2(i)$ in the equation using the term, $c(i) + \underset{k \in GA_2(i)}{Min}\, p(k)$.
   If $k \in GB_2(i)$ then k and i cover all vertices between $min\{h(i), h(k)\}$ and i. The vertices 1 through $\alpha(k,i)$ should be covered using vertices from 1 to $k-1$. The optimum cost for doing this is given by $Z^*_{k-1}(\alpha(k,i))$. We account for such $k \in GB_2(i)$ in the equation using the term $\underset{k \in GB_2(i)}{Min}\{c(k) + Z^*_{k-1}(k,i)\} + c(i)$.
2. We consider the second case when exactly one facility namely j is placed to the right of k. Again k and i together cover all the vertices between them. So $k \in GA_2(i) \cup GB_2(i)$.
   Let $k \in GA_2(i)$. Clearly, $h(j) \leq k$ (If $h(j) > k$ the purpose of j is lost)and $g(j) < h(i) - 1$ (If $g(j) \geq h(i) - 1$, such j would get included in $GA_2(i)$ and they would get accounted in the previous case). The only purpose of j is to cover k. Hence 1 through $k-1$ remains to be covered optimally using 1 through $k-1$ along with the facility already placed at k. This can be done in $u(k)$ cost. Such $k \in GA_2(i)$ are accounted in the equation using the term $\underset{k \in GA_2(i)}{Min}\{u(k) + \underset{j \in (k,i):\ h(j) \leq k \text{ and } g(j) < h(i)-1}{Min}\{c(j)\}\} + c(i)$.
   If $k \in GB_2(i)$, then any j placed to the right of k will be useful only when j covers at least one vertex that k does not cover. This means that $h(j) < h(k)$. But since $j > k$, it follows that $g(j) \geq g(k)$ and hence $g(j) \geq h(i) - 1$. So $j \in GB_2(i)$. Such a solution would have been accounted for in the previous case and hence we need not redundantly consider such solutions again.

□

For each i, we compute $p(i)$, $u(i)$ and $Z^*_j(i)$ $\forall j \geq i$ in that order. As mentioned above, $Z^*_n(n)$ gives the optimal cost of CCP on P.

2.2.1. Time Complexity: For a given i and j, $Z_j^*(i)$ takes $O(j-i)$ time to be computed. For computing $Z_j^*(i)$ $\forall j \geq i$, we need $\sum_{j=i}^{n}(j-i)$ operations. For computing $Z_j^*(i)$ $\forall j \geq i$, $1 \leq i \leq n$ we need $\sum_{i=1}^{n}\sum_{j=i}^{n}(j-i) = O(n^3)$ operations. Similarly, to compute each $p(i)$ and $u(i)$ it takes atmost $\sum_{k=1}^{i}(i-k)$ operations. For computing $p(i)$ and $u(i)$, $1 \leq i \leq n$ we need $\sum_{i=1}^{n}\sum_{k=1}^{i}(i-k) = O(n^3)$ operations. Thus, the algorithm takes $O(n^3)$ time.

3. Conditional Covering Problem on Paths when the cost is unity

3.1. Preliminaries

In this section, we consider the variation of CCP when each vertex is assigned a unit cost and arbitrary covering radius and we give an $O(n^2)$ algorithm for the same. Also, we show how this algorithm can be used to solve CCP on interval graphs in *Section* 4. Let $span(i) = \{k | D_{ik} \leq R_i\}$ $\forall i \in V$. Note that $span(i) = \{$The set of vertices that i can cover in $V(G)\} \cup \{i\}$. A vertex i is said to be a *non-maximal* vertex if $span(i) \subset span(j)$ for some $j \in V$. Else, i is said to be a *maximal* vertex. Let M be the set of maximal vertices and M' be the set of non-maximal vertices in $V(G)$ respectively. $(V(G) = M \cup M')$

Let S be a CCS of G and let $v \in S$. A vertex v is said to *uniquely* cover a vertex u if $\not\exists v' \in S$ such that v' covers u. We define a CCS S to be a *proper* CCS if every $v \in S \cap M'$ *uniquely* covers some vertex $i \in S \cap M$ which satisfies the property, $span(v) \subset span(i)$ i.e. every non-maximal vertex v in S uniquely covers some maximal vertex $i \in S$ which satisfies $span(v) \subset span(i)$. The next *Lemma* relates every CCS with a *proper* CCS.

Lemma 3.1. *For every CCS S, on a graph G, there exists a proper CCS S' such that $c(S') \leq c(S)$.*

Proof. Assume S is not proper. Consider every $v \in S \cap M'$ that does not *uniquely* cover any vertex $i \in M \cap S$ such that $span(v) \subset span(i)$. We can remove every such v in S and add a vertex $i \in M$ such that $span(v) \subset span(i)$ and call the resultant set S'. Since i was not uniquely covered by v in S, i remains covered in S'. Any other vertex covered by v in S is now covered by i in S'. So the set S' still remains a CCS of G and since every vertex incurs only a unit cost, $c(S') \leq c(S)$. (Note that if i is already present in S, then we would just end up removing a vertex and no new vertex which is not present in S actually gets added to S'). □

3.2. Algorithm

For path graphs, we can see that, $span(i) = \{k | h(i) \leq k \leq g(i)\}$. First, we make the following remark regarding maximal vertices.

Remark 3.2.
1. *For any $i \in M$ and $j \in M'$ if $span(j) \subset span(i)$, then i can cover all vertices in $span(j)$ except i itself.*
2. *For $i, j \in M$ and $i < j$, $g(i) \leq g(j)$ and $h(i) \leq h(j)$.*

For any CCS S, we define $g(S) = \text{Max}\{g(i) | i \in S\}$ i.e. S cannot cover any vertex j where $j > g(S)$. Given a *proper* CCS S, we say that a vertex i is the *last maximal* vertex of S, if $i \in M \cap S$ and $\nexists j \in M \cap S$ such that $j > i$. The following lemma gives the property of $g(S)$ when S is a *proper* solution.

Lemma 3.3. *If S is a proper CCS, then $g(S) = g(i) = n$, where i is the last maximal vertex of S.*

Proof. Let i be the last maximal vertex of a CCS S. Let us assume on the contrary that $g(i) \neq n$ and that $g(v) = n$ where $v \in S \cap M'$. In any *proper* CCS S, if $v \in S \cap M'$ then there exists a $j \in S \cap M$ such that $span(v) \subset span(j)$ and v uniquely covers j. Clearly, given such a v and j, $g(j) \geq g(v)$. Also, since i is the last maximal vertex, $i \geq j$ and hence $g(i) \geq g(j)$ (by the second point of Remark 3.2). Therefore, $g(i) \geq g(j) \geq g(v)$ which is a contradiction. □

We will now restrict our search for the optimal CCS for the CCP on the path P only to the set of *proper* CCS on the path P. The next *Theorem* states how the optimal solution can be obtained by searching among *proper* solutions. Let Z_i be the minimum cost *proper* CCS for path P such that i is the *last maximal* vertex of the CCS. Note that, this does not mean i is the last vertex in the CCS corresponding to Z_i. There can be a non-maximal vertex after i. Also we observe that Z_i is defined only when i is a maximal vertex and Z_i does not exist for some maximal vertices $j \in M$, when $g(j) < n$ (as a consequence of *Lemma* 3.3).

Theorem 3.4. *The cost of the optimal CCS of P, is given by $z^* = \underset{i \in M, g(i)=n}{Min} c(Z_i)$.*

Proof. We restrict our search for the optimal solution only in the set of *proper* CCS of P due to *Lemma* 3.1. Every *proper* CCS must have some vertex $i \in M$ to be the *last maximal* vertex. By definition, Z_i is the minimum cost *proper* CCS for path P such that i is the *last maximal* vertex of Z_i. Hence, We have to minimize $c(Z_i)$ $\forall i \in V$. Due to *Lemma* 3.3, $g(Z_i) = g(i)$ and Z_i is CCS of P only when $g(i) = n$. Hence we minimize $c(Z_i)$ for all Z_i such that $g(i) = n$ to get z^*. □

3.2.1. Calculation of $c(Z_i)$: We next describe how to find $c(Z_i)$ efficiently for $i \in M$. We define *protected* and *unprotected* costs for all the maximal vertices $i \in M$ of the path P denoted by $p(i)$ and $u(i)$ as follows. Let A_i be the minimum cost *proper* CCS for the vertices 1 through i such that i is the *last maximal* vertex of A_i. The Protected cost $p(i)$ is $c(A_i)$. Let B_i be the minimum cost *proper* CCS for the vertices 1 through $i-1$ such that i is the *last maximal* vertex of B_i.

The unprotected cost $u(i)$ is $c(B_i)$. Similar to the set Z_i, in A_i and B_i too, we can place a non-maximal vertex even to the *right* of i, as long as the solution is *proper*. The vertex i is necessarily covered by A_i (i is protected by A_i) but may not be necessarily covered by B_i(i may remain unprotected by B_i). For each $i \in M$, we define $Q_i = \{v \in M' | v$ covers i and $span(v) \subset span(i)\}$. The next *Lemma* characterizes the vertices which can cover i in A_i.

Algorithm 1 : Algorithm to compute $p(i)$ $\forall i \in M$ - $\{min\}$

```
Require: A Path $P = 123 \ldots n$ along with $R_i$, $\forall i \in V(P)$ .
  /* Preprocessing Step */
  Compute $g(i)$, $h(i)$ $\forall i \in V(P)$.
  Find the sets $M$ and $M'$ such that $V = M \cup M'$.
  Find $GA_1(i)$, $GA_2(i)$, $GB_1(i)$, $GB_2(i)$ and $Q_i$ $\forall i \in M$.
  /* Dynamic Programming Step */
  for $i$=$min$ + 1 to $n$ do
    if $i \in M$ then
      if $(GA_1(i) = GB_1(i)) = \emptyset$ then
        $p'(i) = \infty$
      else
        $p'(i) = Min\{\underset{k \in GA_1(i)}{Min} p(k), \underset{k \in GB_1(i)}{Min} u(k)\} + 1.$
      end if
      if $(GA_2(i) = GB_2(i)) = \emptyset$ and $p'(i) = \infty$ then
        $u(i) = \infty$
      else
        $u(i) = Min\{p'(i), \underset{k \in GA_2(i)}{Min} p(k) + 1, \underset{k \in GB_2(i)}{Min} u(k) + 1\}.$
      end if
      if $u(i) \neq p'(i)$ and $Q_i \neq \emptyset$ then
        $p(i) = \text{Min}\{p'(i), u(i) + 1\}.$
      else
        $p(i) = p'(i).$
      end if
    end if
  end for
```

Lemma 3.5. *In A_i, the vertex i should either be covered by a $j \in M \cap A_i$ such that $j < i$ or be* uniquely *covered by a $v \in Q_i$.*

Proof. In A_i, let i be covered by a maximal vertex j. By definition of A_i, i is the *last maximal* vertex of A_i and hence $j < i$.
On the other hand, if no maximal vertex covers i, then some non-maximal vertex $v \in M' \cap A_i$ must be covering i. If $v \in M' \cap A_i$ is covering i then $v \in Q_i$. (Assume

$v \notin Q_i$. A_i is a *proper* CCS and it would imply the existence of a maximal vertex which is also covering i contradicting our assumption) Such a v must be *uniquely* covering i. To see this, note that A_i is a *proper* CCS. If v is not uniquely covering i, it must be uniquely covering some other maximal vertex in A_i. This maximal vertex will also cover i, contradicting our assumption. Hence the proof. □

We will now give a set of dynamic programming equations for solving $p(i)$ and $u(i)$ $\forall i \in M$. First, the following *Remark* will solve them for the base case.
Let *min* be the least indexed maximal vertex.

Remark 3.6. *Clearly $h(min) = 1$ due to* Remark 3.2. *Hence $B_{min} = \{min\}$ and $u(min) = 1$. To find A_{min}, we pick a vertex v from Q_{min} (if $Q_{min} \neq \emptyset$) and append it to B_{min}. So we have $A_{min} = B_{min} \cup \{v\}$. The cost $p(min) = 2$. If $Q_{min} = \emptyset$, we set $p(min) = \infty$ and A_{min} does not exist for the vertex min. The correctness of this remark is due to* Lemma 3.5.

For all the maximal vertices $i \in M$ - $\{min\}$ we define the following sets. These definitions are borrowed from [5] but unlike the previous section, where the following sets are defined for all the vertices in V, we define them only on maximal vertices and further use them to calculate the $p(i)$ and $u(i)$. If these sets do not exist for any of the maximal vertex, we set them to be null.

1. $GA_1(i) = \{k \in M | k+1 \leq i \leq g(k)$ and $h(i) > k\}$
2. $GB_1(i) = \{k \in M | k+1 \leq i \leq g(k)$ and $h(i) \leq k\}$
3. $GA_2(i) = \{k \in M | k+1 \leq h(i) \leq g(k)+1 \leq i\}$
4. $GB_2(i) = \{k \in M | g(k)+1 \leq i$ and $h(i) \leq k\}$

Let $p'(i)$ be the cost corresponding to the minimum cost *proper* CCS for the vertices 1 through i such that i is the *last maximal* vertex of CCS and i is covered by a maximal vertex k, $k < i$. Now, we use the following *Theorem* to find $p(i)$ and $u(i)$ for $i \in M$ - $\{min\}$. There may be some maximal vertices i for which A_i or B_i may not exist. A_i will not exist when $GA_1(i) = GB_1(i) = \emptyset$ and B_i will not exist when $(GA_2(i) = GB_2(i)) = \emptyset$ and $p'(i) = \infty$. In all such cases we will set $p(i)$ and $u(i)$ to be infinity explicitly in the algorithm. We will now consider only those cases when A_i and B_i exists and state the following *Theorem*.

Theorem 3.7. *The following are the dynamic programming equations to solve for $p(i)$ and $u(i)$ $\forall i \in M$ - $\{min\}$:*

$$p'(i) = Min\{\underset{k \in GA_1(i)}{Min} p(k), \underset{k \in GB_1(i)}{Min} u(k)\} + 1. \tag{3}$$

$$u(i) = Min\{p'(i), \underset{k \in GA_2(i)}{Min} p(k) + 1, \underset{k \in GB_2(i)}{Min} u(k) + 1\}. \tag{4}$$

$$p(i) = \begin{cases} Min\{p'(i), u(i)+1\} & p'(i) \neq u(i) \text{ and } Q_i \neq \emptyset \\ p'(i) & \text{otherwise} \end{cases} \tag{5}$$

Proof. We assume that the above equations correctly calculate $p(j)$ and $u(j)$ $\forall j \in M$ and $j < i$. We should prove that $p(i)$ and $u(i)$ are correctly calculated. By *Lemma* 3.5, We know that i can be covered in two possible ways. For each of the two possible ways, we analyze the terms in all the three equations.

1. i is covered by a $k \in M$ such that $k < i$:
   p′(i): Clearly, by definition $k \in GA_1(i) \cup GB_1(i)$. For a $k \in GA_1(i)$, i cannot cover k and k should be covered by some other vertex. Hence, we take the protected cost of k for calculations in equation (3). For a $k \in GB_1(i)$, i covers k. Hence, we take the unprotected cost of k for calculations in equation (3). We add a cost one due the facility placed at i.
   u(i): A vertex $k \in GA_2(i)$ is not covered by i and hence we use the protected cost $p(k)$ in (4). Similarly, a vertex $k \in GB_2(i)$ is covered by i and hence we use the unprotected cost $u(k)$ in (4). We add a cost one due to the facility placed at i. Sometimes, it may be cheaper to cover vertex i than to leave it uncovered. The optimal cost of doing this is $p'(i)$ and hence we include the term $p'(i)$ in (4).
   p(i): When the i in $p(i)$ is covered by a maximal vertex, the cost is just $p'(i)$.
2. i is uniquely covered by a vertex $v \in Q_i$:
   p′(i): By definition of $p'(i)$, i is covered by a maximal vertex and hence i cannot be uniquely covered by a vertex in $Q(i)$ in this case.
   u(i): Since it is not necessary to cover i in $u(i)$, we will not choose any non-maximal vertex v to cover i. This is because, if the solution has to be a *proper* solution, then v has to uniquely cover i. But since $\text{span}(v) \subset \text{span}(i)$, by removing v, all the vertices covered by v except i are still covered by i (i need not be covered in $u(i)$ anyway). Hence v can be removed.
   p(i): If $Q_i = \emptyset$, this case cannot be considered at all and $p(i) = p'(i)$. We assume Q_i is not empty and we use a $v \in Q_i$ to cover i uniquely. Now, vertices 1 to $i-1$ should be covered using minimum cost and this cost is clearly given by $u(i)$. So, If Q_i is not empty, we can pick a vertex v from it and append it to B_i. The cost of this new CCS $(B_i \cup \{v\})$ of P_i is given by $u(i)+1$. Also, if $u(i) = p'(i)$, then $p(i) = p'(i)$. This is because, $u(i) = p'(i)$ means that in B_i, i is covered by a maximal vertex and a $v \in Q_i$ will no longer cover i *uniquely* and hence v will not cover any maximal vertices *uniquely.* Therefore for the case $u(i) = p'(i)$, we set $p(i) = p'(i)$.

Those vertices i, for which we were able to find a CCS in the second case (when $p'(i) \neq u(i)$ and $Q_i \neq \emptyset$), we finally set $p(i) = Min\{p'(i), u(i)+1\}$ as in equation (5). □

Remark 3.6 is used to find the protected and unprotected costs for *min. Algorithm* 1, uses *Theorem* 3.7 to find $p(i)$, $u(i)$ and hence $z(i)$(if it exists) for all $i \in M$ - $\{min\}$ and hence the CCS is always a *proper* CCS. It first does the preprocessing to calculate all the required values. Then the dynamic programming equations are used to find the protected and unprotected costs. After calculating these values, we use *Theorem* 3.4 to find the cost of the optimal solution for CCP on paths. The calculation of $GA_1(i)$, $GA_2(i)$, $GB_1(i)$ and $GB_2(i)$ for all i takes $O(n^2)$ time. Similarly, the dynamic programming step also takes $O(n^2)$ time. All other calculations take linear time. Thus *Algorithm* 1 runs in $O(n^2)$ time.

4. Conditional Covering Problem on Interval Graphs

A graph G is an interval graph if its vertices can be put in one-to-one correspondence with a family F of intervals on the real line such that two vertices are adjacent in G *iff* their corresponding intervals have nonempty intersection [3]. F is known as the intersection model of the graph. Let $I_i = [a_i, b_i]$ be the interval corresponding to a vertex i of the graph. We can assume that the set of $2n$ left and right end points are distinct for the graph. A graph G is a proper interval graph, *iff* no interval is properly contained within another interval of the graph. Any interval graph is a proper interval graph *iff* $a_1 < a_2 < a_3 < \dots a_n$ and $b_1 < b_2 < b_3 < \dots b_n$.

Consider an interval graph $G = (V, E)$ such that G has unit edge weights. But each vertex i has an arbitrary positive *covering radius* value R_i and unit cost values. We give a $O(n^2)$ algorithm to solve CCP on such a graph. But first, we make the following *Remark* about CCP on proper interval graphs.(When each vertex is assigned positive arbitrary *covering radius* and unit cost.)

Remark 4.1. *Consider a proper interval graph. For any two vertices i and j, if $a_i < a_j$ and $b_i < b_j$ then we say that $i < j$. Such an ordering of vertices of the proper interval graph is called the proper interval graph ordering. We can observe that for any vertex i, span$(i) = \{k | h(i) \leq k \leq g(i)\}$.* Remark 3.2 *and hence* Lemma 3.3 *holds correct for proper interval graphs. As a consequence all the algorithms used for paths hold good in this case also without any modification.*

Now, we turn our attention to interval graphs. The first property stated in *Remark* 3.2 holds true for any graph with any ordering. However we should prove that the second property of the *Remark* 3.2 is valid for the maximal vertices of the interval graph.
We order all the vertices of the interval graph according to their left endpoints a_i (i.e., $i < j$ iff $a_i < a_j$).

Lemma 4.2. *If i, $j \in M$ such that $i < j$, then $g(i) \leq g(j)$ and $h(i) \leq h(j)$.*

Proof. We prove that we will get a contradiction otherwise.
If $g(i) \leq g(j)$ and $h(i) > h(j)$, then *span*$(i) \subset$ *span*(j) and i is not maximal anymore i.e. $i \notin M$.
If $g(i) > g(j)$ and $h(i) \leq h(j)$, then *span*$(j) \subset$ *span*(i) and j is not maximal anymore i.e. $j \notin M$
If $g(i) > g(j)$ and $h(i) > h(j)$, then we consider two cases.
Case 1: $b_i < b_j$. Since $i < j$, $a_i < a_j$. Now, $a_i < a_j$ and $h(i) > h(j)$ imply that $R_j > R_i$. Also $b_i < b_j$ and $g(i) > g(j)$ imply that $R_i > R_j$. This leads to a

contradiction.
Case 2: $b_i > b_j$. Since $i < j$, $a_i < a_j$. Clearly $I_j \subset I_i$. Now, $a_i < a_j$ and $h(i) > h(j)$ imply that $R_j > R_i$. Also since $I_j \subset I_i$, we see that the upper-reach of i can be reached by j in atmost $R_i + 1$ steps. But since $R_j > R_i$, we have $R_j \geq R_i+1$ and hence $g(j) \geq g(i)$ which is a contradiction. Hence the proof. □

As a consequence of *Lemma* 4.2, we see that *Remark* 3.2 and hence *Lemma* 3.3 hold good for interval graphs. Hence, as stated above for proper interval graphs, all the algorithms used for paths hold good for interval graphs without any modification.

5. Conclusion

In this paper, we studied the Conditional Covering Problem on some special classes of graphs such as paths and interval graphs. We showed that Horne and Smith's algorithm[5] for CCP on paths, and its extension to star and trees in [5, 4] is incorrect and gave a correct $O(n^3)$ algorithm for CCP on paths. We also presented a $O(n^2)$ algorithm for CCP on paths when unit cost is assigned to all vertices and further extended the algorithm to interval graphs without increasing the time complexity. It is an interesting open problem to see if our $O(n^3)$ algorithm for paths can be extend to stars and trees. Additionally, CCP can be examined on special classes of graphs such as series-parallel and asteroidal triple free graphs on which the total domination problem can be solved efficiently [9, 6].

References

[1] Jeffrey B, Goldberg Brian, J.Lunday, and J. Cole Smith. Algorithms for solving the conditional covering problem on paths. *Naval Research Logistics*, 52(4):293–301, 2005.

[2] Sohail S. Chaudhry, I. Douglas Moon, and S. Thomas McCormick. Conditional covering: Greedy heuristics and computational results. *Computers & OR*, 14(1):11–18, 1987.

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

[4] Jennifer A. Horne and J. Cole Smith. A dynamic programming algorithm for the conditional covering problem on tree graphs. *Networks*, 46(4):186–197, 2005.

[5] Jennifer A. Horne and J. Cole Smith. Dynamic programming algorithms for the conditional covering problem on path and extended star graphs. *Networks*, 46(4):177–185, 2005.

[6] Dieter Kratsch. Domination and total domination on asteroidal triple-free graphs. *Discrete Applied Mathematics*, 99(1-3):111–123, 2000.

[7] I. Douglas Moon and S.S. Chaudhry. An analysis of network location problems with distance constraints. *Management Science*, 30:290–307, 1984.

[8] I. Douglas Moon and Lee Papayanopoulos. Facility location on a tree with maximum distance constraints. *Computers & OR*, 22(9):905–914, 1995.

[9] J. Pfaff, R. Laskar, and S.T. Hedetniemi. Linear algorithms for independent domination and total domination in series-parallel graphs. *Management Science*, 30:290–307, 1984.

Balasubramanian Sivan
Computer Sciences Department
University of Wisconsin Madison
Wisconsin - 53706, USA
e-mail: `bsivan@wisc.edu`

S. Harini
Department of Computer Science and Engineering
Indian Institute of Technology Madras
Chennai- 600036, India
e-mail: `harini.santhanam@gmail.com`

C. Pandu Rangan
Department of Computer Science and Engineering
Indian Institute of Technology Madras
Chennai - 600036, India
e-mail: `rangan@iitm.ernet.in`