A Project Report

On

DOMINATING SETS IN CIRCULANT GRAPHS

BY

INDRANIL BHAUMIK

2014B4A70924H

Under the supervision of

Prof. A. MICHAEL ALPHONSE

SUBMITTED IN FULLFILLMENT OF THE REQUIREMENTS OF

MATH F376: DESIGN ORIENTED PROJECT (DOP)

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI (RAJASTHAN)

HYDERABAD CAMPUS

(JAN- APR 2018)

ACKNOWLEDGMENTS

One cannot singularly create a piece of work without a shadow of light cast upon by his/her associates who always provide the right directions. Therefore I, Indranil Bhaumik would like to express my sincere gratitude towards my institute and my faculty in-charge Prof. A. Michael Alphonse for the progress and partial completion of my study project on “Dominating sets in Circulant Graphs”**.**

I would even like to thank my project-mates whose constant help and support as partners in the project helped me to meet the deadlines and expectations of the project.

Birla Institute of Technology and Science-Pilani,

Hyderabad Campus

Certificate

This is to certify that the project report entitled "DOMINATING SETS IN CIRCULANT GRAPHS" submitted by Mr. INDRANIL BHAUMIK(ID No. 2014B4A70924H) in fulfillment of the requirements of the course **MATH F376** Design Oriented Project Course, embodies the work done by him under my supervision and guidance.

Date:

(Prof. A. Michael Alphonse)

BITS- Pilani, Hyderabad Campus

ABSTRACT

Circulant graphs are something which have potentially many applications in wider areas like network problems, engineering etc. The project aims at understanding their basic structures and finding patterns such that a generalization can be found for C(n,{1,m}) where both m & n can vary. So far, we have formulations for upto m=5. There will be an effort to calculate for greater values, especially for m=6 and m=7. The idea is to understand the various patterns involved in the construction of the tree so as to reduce the number of comparisons made which can help in reducing the running time of the algorithm. Identification of the optimal paths giving the domination number for a given graph has also been done later.

CONTENTS

INTRODUCTION

Dominating sets for circulant graphs can be used to apply in various network problems having similar structures. We do have an algorithm to convert the graph into a tree with nodes representing the vertices of the same. The edges relate to the closest uncovered vertex. As the number of vertices increase, it directly translate into a tree of longer length and a higher branching factor. We aim to reduce the number of comparisons by using reduction method and further ways as mentioned later. As it's universally known, this problem falls in the category of NP-hard problems which don't have any polynomial running time algorithm and thus we can't devise a method which can run for any given number of vertices in reasonable amount of time, hence the need for some reductions. Due to this limitation with the computational constraints, some mechanical work has been done to experimentally determine the specific paths which covers most number of vertices for a given chord length.

DEFINITIONS

Dominating Set of a Graph:

Let G be a graph and V be the set of all vertices of G. A set $D \subseteq V$ is called a *dominating set* for G if every vertex of G is either in D, or adjacent to some vertex in D. A minimum dominating set in a graph G is a dominating set with minimum number of vertices. The number of vertices in a minimum dominating set is called domination number of a graph G and is denoted by $\gamma(G)$.

Independent Set of a Graph:

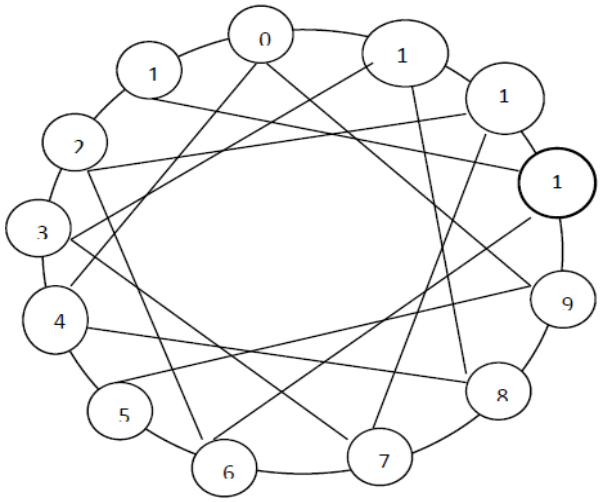
For a graph G = (V, E) , a set $I \subseteq V$ is called independent set of G if for each $u, v \in I$, $N(u) \cap \{v\}$ is empty, where N(u) denotes the set of all adjacent vertices of u.

Independent Dominating Set of a Graph:

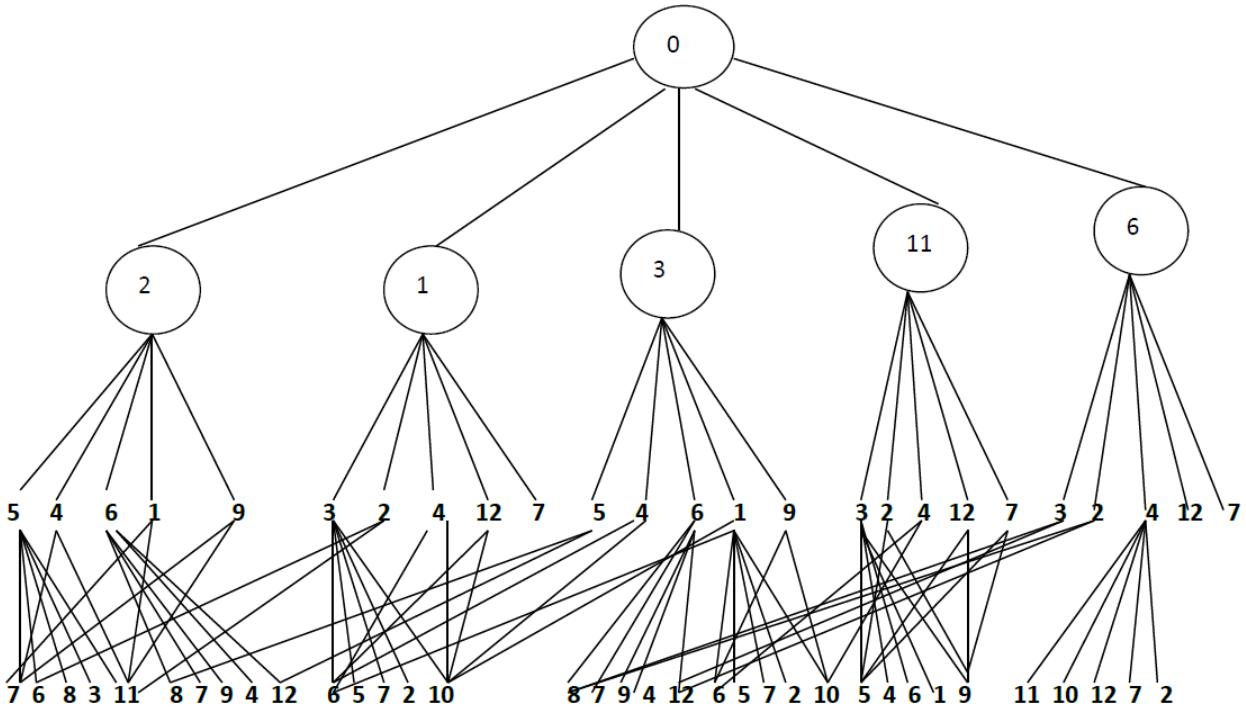
A subset D of V an independent dominating set if D is both an independent and dominating set.

Circulant graphs:

Let's take n to be a positive integer and let S = { s1, s2, …. sk } be a set of chord lengths called the connection set. The Circulant graph Circ (n; S) is a graph with vertex set {0, 1, 2, …. , n-1 } and edge joining i and j whenever $i \equiv j \pm sm \pmod n$, $1 \leq m \leq k$.



C(13,{1,4})



Tree for the given graph.

Finding Domination number of Circulant Graphs:

Nader Jafari Rad has given the following result:

For any integer $n \geq 6$, the domination number is given by :

$$\gamma(Circ(n,\{1,3\}) = \begin{cases} \left\lceil \frac{n}{5} \right\rceil \; if \; n \not\equiv 4 \; mod \; 5 \\ \left\lceil \frac{n}{5} \right\rceil + 1 \; if \; n \equiv 4 \; mod \; 5 \end{cases}$$

The proof is as follows:

If S is any dominating set, since each vertex in S can dominate at most five vertices including itself, the number of vertices in S $\geq \lceil n/5 \rceil$.

Note that the same elementary proof does not work for the circulant graphs of the type $Circ(n,\{1,4\})$.

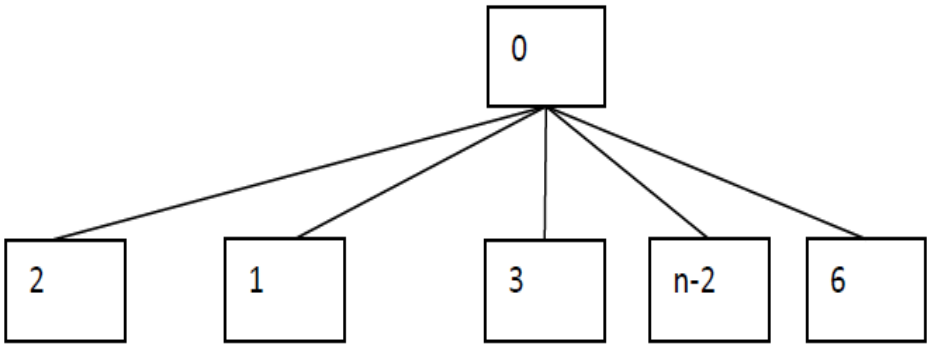
Concepts of Min-Plus Linear Algebra can be used to find the dominating number of circulant graphs Circ(n; S) where S is any subset of {0, 1, 2, 3,….9}.The issue with this approach is that the methodology finds only the dominating number not the minimum dominating set.

Finding Minimum Domination Set of Circulant Graphs:

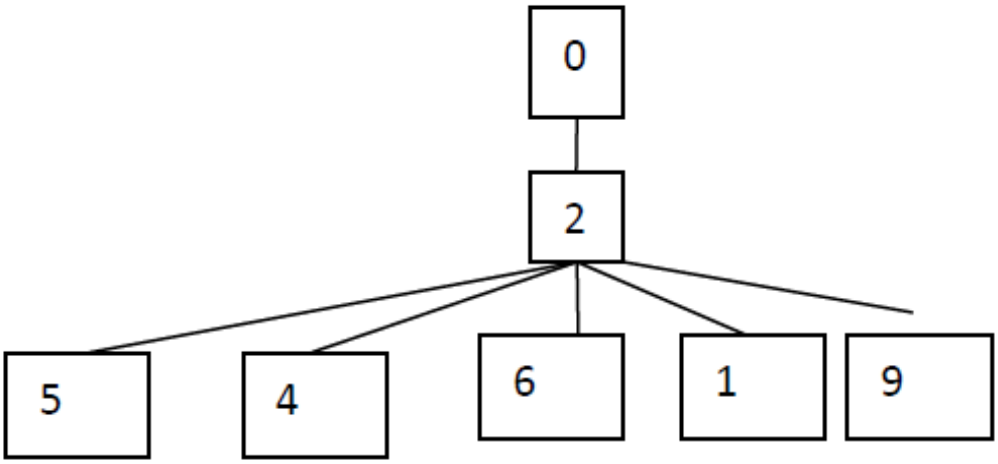
We now present a method to find the minimum dominating set and domination number of circulant graphs Circ(n, {1,4}) where $n \geq 4$ is a positive integer. This method involves constructing a tree which we call a dominating tree that has some repeated patterns in nature which enables us to use the dynamic programming concepts.

Construction of the Dominating tree :

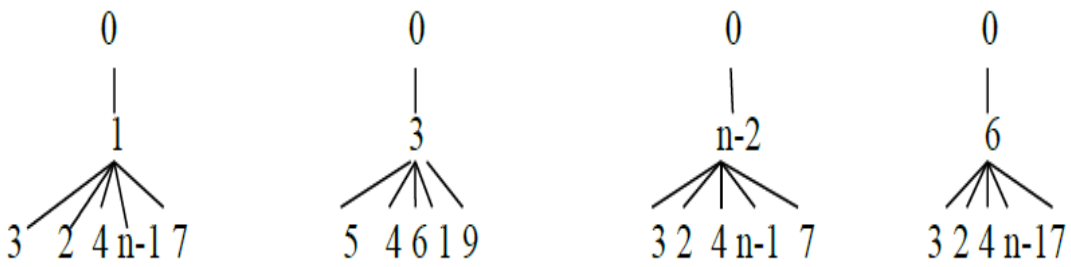
Always name the vertices 0, 1, 2, n-1. We first choose the vertex 0 to be included in the dominating set. The same vertex 0 will be chosen as the root of the tree. We call this root level as 0 th level. Once 0 is chosen the vertices 1, 4, n-1, n-4 are dominated by 0. We traverse the graph in the anti- clock wise direction and search for the first vertex which is not dominated by 0. We observe that 2 is the first vertex which is not dominated by 0. As a result, we include 2 as a part of the dominating set. If 2 is not included in the dominating set then one of the vertices 1, 3, n-2, 6 has to be included in the dominating set, since they are adjacent to 2 and hence will compensate for the absence of 2. So at the next level we'll the vertices 2, 1, 3, n-2, 6 as the children of 0.



The algorithm goes as- at each level we maintain the following order. At any level, if i is the first vertex which is not dominated by any of its ancestors when we traverse in the counter clockwise direction starting from 0, then the children at that level must be ordered from left to right as i, i-1, i+1(mod n), i-4(mod n), i+4(mod n).The siblings of i are found by the method that if i is not included in the dominating set then one of the vertices i-1, i+1(mod n), i-4(mod n), i+4(mod n) must be included in the dominating set since they cover the i^{th} vertex.



By applying the similar rules, the next level can be drawn as follows:



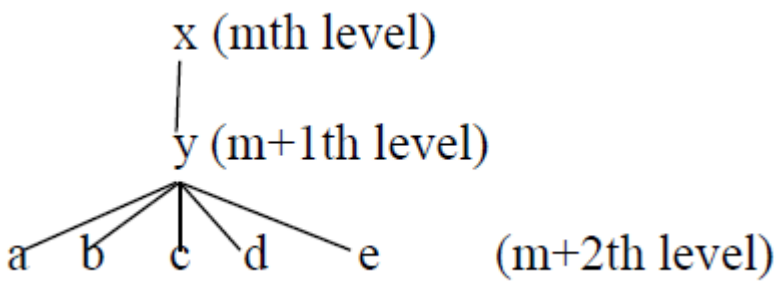
Continuing in this way, when we encounter a leaf for the first time we stop the process. We will show that if we encounter a leaf at the kth level for the first time the domination number of the graph is k+1.The reason for this path being the domination set is that by the time we reach this node, all the remaining vertices would've been covered and hence we'll be left with no more children, which should give us one possilble path.

Theorem: During the construction of the dominating tree associated with Circ(n, {1,4}) if we encounter a leaf at kth level for the first time, then the dominating number is k+1. This basically says that, the leaf node together with its parents form the minimum dominating set for the graph given.

Proof: Let $\gamma(Circ(n,\{1,4\}))=k+1$ and D = {x1, x2....,xk, xk+1} be a minimum dominating set of the graph $Circ(n,\{1,4\})$. Note that we always name the vertices of Circ(n, {1,4}) as 0, 1, 2,...n-1.

In general, we can always assume that x1 = 0. Since each vertex in the dominating set must be at some level j in the dominating tree where $0 \leq j \leq k$ and at only one level without loss of generality we can further assume that for j = 2, 3, 4, k+1, xj is at the j-1th level. By the construction of the dominating tree at level j-1, xj is either a left child of xj-1 or one of the other four children of xj-1. If xj is the left most child of xj-1 then xj is the first vertex say u, that is not dominated by any of its ancestors (including its parent) when we traverse the graph in the counter clock wise direction. If xj is one of the other four children, then xj dominates u. Hence from the root (0 level) to level k the vertices in D form a path of length k in the dominating tree. Since by our assumption, D is a dominating set, there is no need to choose any further child of xk+1 and hence xk+1 must be a leaf. If we encounter a leaf in the dominating tree in the mth level, where m < k+1, then the vertices in the path from the root to that leaf forms a dominating set and hence $(Circ(n,\{1,4\}))\leq m<k+1$ which is a contradiction to the fact that $\gamma(Circ(n,\{1,4\}))=k+1$. This completes the proof.

Representation of a branch or path:



Branch is represented as

{x(m), y(m+1), a(m+2), b(m+2), c(m+2), d(m+2), e(m+2)}

Definition :

Consider any three successive vertices x, y, z along the path from the root to a leaf in the dominating tree. We call such vertices as a pattern and is denoted by <x-y-z>. Let <a-b-c> and <x-y-z> be two patterns. Let a|, b|, c| be the ascending order of a, b, c and x|, y|, z| be ascending order of x, y, z. We say that the patterns <a-b-c> and <x-y-z> are similar if b| - a| = y1 – x| and c| - b| = z| - y|.

Theorem: Consider the paths (of infinite length) starting from the root in the dominating tree : A={0(0), 2(1), 9(2), 11(3), 18(4),….}, B ={0(0), 1(1), 7(2), 13(3)…}. Let $ai\ and\ bi$ be the number of vertices dominated by the vertices in the paths A and B respectively starting from root to the ith level. Let Mi=max{ai,bi}. Then $Mi\geq$ the number of vertices dominated by the vertices in any path starting from root to to the ith level.

By comparing the number of dominated vertices by the vertices in each path starting from the root. But making long-drawn comparisons is pretty difficult since at mth level the number of evaluations increase since we have 5^m comparisons to make. There are some specific patterns repeated in the dominating tree that largely reduce the number of comparisons, which is the very basis for our method so as to optimise for allowing greater value of n (number of vertices).

Implementation Of The Above Algorithm in C++

```cpp
#include <bits/stdc++.h>
using namespace std;

int vertex,k,minval;
vector <int> visited; //A vector defined to keep track of the visited nodes.
/*Definition of tree nodes which store the name of the node and the singular weight as 1,
while also having 5 empty children.*/
struct node
{
    int data;
    int weight;
    struct node *one;
    struct node *two;
    struct node *three;
    struct node *four;
    struct node *five;
};

//Initialisation of all nodes.
struct node *newNode(int data, int w)
{
    struct node *node = new struct node;
    node->data = data;
    node->weight= w+1;
    node->one = NULL;
    node->two = NULL;
    node->three = NULL;
    node->four = NULL;
    node->five = NULL;
    return(node);
}

//Function to create the tree and add nodes to form the paths.
void addNodes(struct node* root)
{
   int j;
   visited[(root->data)]+=1;
   visited[((root->data)+1)%vertex]+=1;
   visited[((root->data)-1 + vertex)%vertex]+=1;
   visited[((root->data)+k)%vertex]+=1;
   visited[((root->data)-k + vertex) %vertex]+=1;
```

```
42
43      for(j=((root->data)+1)%vertex; j!=(root->data);j=(j + 1) % vertex)
44      {
45          if(visited[j]==0)
46          {
47              root->one=newNode(j,(root->weight));
48              root->two=newNode((j-1 + vertex)%vertex,(root->weight));
49              root->three=newNode((j+1)%vertex,(root->weight));
50              root->four=newNode((j-k + vertex)%vertex,(root->weight));
51              root->five=newNode((j+k)%vertex,(root->weight));
52
53              addNodes(root->one);
54
55              visited[(root->one->data)]-=1;
56              visited[((root->one->data)+1)%vertex]-=1;
57              visited[((root->one->data)-1 + vertex)%vertex]-=1;
58              visited[((root->one->data)+k)%vertex]-=1;
59              visited[((root->one->data)-k + vertex) %vertex]-=1;
60
61              addNodes(root->two);
62
63              visited[(root->two->data)]-=1;
64              visited[((root->two->data)+1)%vertex]-=1;
65              visited[((root->two->data)-1 + vertex)%vertex]-=1;
66              visited[((root->two->data)+k)%vertex]-=1;
67              visited[((root->two->data)-k + vertex) %vertex]-=1;
68
69              addNodes(root->three);
70
71              visited[(root->three->data)]-=1;
72              visited[((root->three->data)+1)%vertex]-=1;
73              visited[((root->three->data)-1 + vertex)%vertex]-=1;
74              visited[((root->three->data)+k)%vertex]-=1;
75              visited[((root->three->data)-k + vertex) %vertex]-=1;
76
77              addNodes(root->four);
78
79              visited[(root->four->data)]-=1;
80              visited[((root->four->data)+1)%vertex]-=1;
81              visited[((root->four->data)-1 + vertex)%vertex]-=1;
82              visited[((root->four->data)+k)%vertex]-=1;
```

```
                visited[((root->four->data)-k + vertex) %vertex]-=1;

                addNodes(root->five);

                visited[(root->five->data)]-=1;
                visited[((root->five->data)+1)%vertex]-=1;
                visited[((root->five->data)-1 + vertex)%vertex]-=1;
                visited[((root->five->data)+k)%vertex]-=1;
                visited[((root->five->data)-k + vertex) %vertex]-=1;
                break;
          }
        }
        if(j==(root->data))
            minval= std::min(minval,(root->weight));
}

int main()
{
    cout<<"Enter n:";
    cin>>vertex;
    cout<<"Enter k:";
    cin>>k;
    minval=INT_MAX;
    visited.resize(vertex);
    for(int i=0;i<vertex;i++)
        visited.push_back(0);
    struct node *root = newNode(0,0);
    addNodes(root);
    cout<<"Domination number:"<<minval<<endl;
    return 0;
}
```

Dominating Paths For Chord Lengths of k=6 & k=7

Let us take the case of circulant graphs of type Circ(n, {1,6}) where $n \geq 6$ is a positive integer. To find the dominating path, the same algorithm is used which was done when k=4, previously. The domination numbers for vertex-values upto n=30 were obtained by running the code implemented by using the algorithm. For the calculations the value of n=20 was taken. Let us first see the results obtained after running the code:

N-values	Domination number
(9,10,11,12,13)	3
(14,15,16,17)	4
(18,19,20,21)	5
(22,23,24,25,26)	6
(27,28,29,30)	7

The paths found after every stage of the algorithm are stated below:

Path	Children	Stage	Removed path
0	2,1,3,8,16	1	
0->2	4,3,5,10,18	2	
0->1	3,2,4,9,17		
0->3	5,6,4,11,19		
0->8	2,3,1,8,16		Yes
0->16	2,3,1,8,16		Yes
0->2->4	7,8,6,13,1	3	
0->2->3	5,6,4,11,19		Yes
0->2->5	7,8,6,13,1		Yes
0->2->10	4,5,3,10,18		Yes
0->2->18	4,5,3,10,18		Yes
0->1->3	5,6,4,11,19		Yes

0->1->2	4,5,3,10,18		Yes
0->1->4	8,9,7,14,2		
0->1->9	3,4,2,17,9		Yes
0->1->17	3,4,2,17,9		Yes
0->3->5	7,8,6,13,1		Yes
0->3->6	8,9,7,14,2		
0->3->4	7,8,6,13,1		
0->3->11	5,6,4,11,19		Yes
0->3->19	5,6,4,11,19		Yes
0->2->4->7	9,10,8,15,3	4	Yes
0->2->4->8	11,12,10,17,5		Yes
0->2->4->6	9,10,8,15,3		Yes
0->2->4->13	9,10,8,15,3		Yes
0->2->4->1	9,10,8,15,3		Yes
0->1->4->8	11,12,10,17,5		Yes
0->1->4->9	11,12,10,17,5		Yes
0->1->4->7	9,10,8,15,3		Yes
0->1->4->14	9,10,8,15,3		Yes
0->1->4->2	9,10,8,15,3		Yes
0->3->6->8	10,11,9,16,4		Yes
0->3->6->9	11,12,10,17,5		
0->3->6->7	10,11,9,16,4		Yes
0->3->6->14	10,11,9,16,4		Yes
0->3->6->2	10,11,9,16,4		Yes
0->3->4->7	11,12,10,17,5		Yes
0->3->4->8	11,12,10,17,5		Yes
0->3->4->6	8,9,7,14,2		Yes
0->3->4->13	8,9,7,14,2		Yes
0->3->4->1	8,9,7,14,2		Yes

Now, we'll take the second case of k=7. The following table illustrates the values we get after running the code.

N-values	Domination Number
(9,10)	2
(11,13,15)	3
(12,14,16,18,20)	4
(17,19,22,25)	5
(21,23,24,26,27,30)	6
(28,29)	7

These are the paths found after every stage of the algorithm (n=20):

Path	Children	Stage	Removed Path
0	2,3,1,9,15	1	
0->2	4,5,3,11,17	2	
0->3	5,6,4,12,18		
0->1	3,4,2,10,16		
0->9	2,3,1,9,15		Yes
0->15	2,3,1,9,15		Yes
0->2->4	6,7,5,13,9	3	Yes
0->2->5	8,9,7,15,1		
0->2->3	5,6,4,12,18		Yes
0->2->11	4,5,3,11,17		Yes
0->2->17	4,5,3,11,17		Yes
0->3->5	8,9,7,15,1		
0->3->6	8,9,7,15,1		Yes
0->3->4	6,7,5,13,19		Yes
0->3->12	5,6,4,12,18		Yes
0->3->18	5,6,4,12,18		Yes
0->1->3	5,6,4,12,18		Yes
0->1->4	6,7,5,13,19		Yes
0->1->2	4,5,3,11,17		Yes
0->1->10	3,4,2,10,16		Yes
0->1->16	3,4,2,10,16		Yes
0->2->5->8	10,11,9,17,3	4	Yes
0->2->5->9	11,12,10,18,4		Yes

0->2->5->7	10,11,9,17,3		Yes
0->2->5->15	10,11,9,17,3		Yes
0->2->5->1	10,11,9,17,3		Yes
0->3->5->8	11,12,10,18,4		Yes
0->3->5->9	11,12,10,18,4		Yes
0->3->5->7	9,10,8,16,2		Yes
0->3->5->15	9,10,8,16,2		Yes
0->3->5->1	9,10,8,16,2		Yes

NOTE: At every stage, the paths which were covering the maximum number of vertices were retained while others were eliminated for further evaluation.

OBSERVATIONS:

1. We can afford to do the reductions at the second stage itself as it's been observed that the paths which cover maximum number of vertices at the second stage itself continue to dominate other paths even as the stages increase.

2. In the case of k=6, the optimal path is seen to be 0->3->6->9….We can also see that the difference between the vertex numbers is also a constant at the value of 3.

3. The above path has been verified for all the other possible n-values and has been found to be correct. Further, the given path is unique. The constant difference between the vertex numbers also gives a clue for why we see that the domination numbers are the same for a group of 4 or 5 values of 'n' and increase incrementally.

4. The case of k=7 is a bit different. As we can see from the chart, the domination number alternates between n-values with no clear pattern.

5. While we do get multiple paths (five, actually) which dominate maximum number of vertices, yet they aren't the optimal ones. As for the case of n=20, the domination number is 4, but these five paths don't cover all the vertices by the end of iteration 4, which is in contrast to the algorithm defined, on which the calculations were done.

6. There's one peculiar case of the path 0->5->10->10->15……., which does seem to satisfy the case of n=20 but doesn't exactly fits in with the other n-values. One thing must be noted that this path can't be determined by applying the usual algorithm.

PROPOSITIONS :

1. From the patterns observed for the case of k=6, we can easily correlate the groups of 4 or 5 we found to the reason why we're finding a path with a constant difference in the vertex-number values and hence, this path should uniquely determine the domination numbers for n>30.

2. The unusual case of k=7 produces inconsistent domination numbers with progressive n-values. It can be concluded that the given algorithm determines the upper bound of the domination number for a given 'n' and doesn't provide the path which can give the actual domination number.

3. Furthermore, the algorithm doesn't always provides with the most optimal path in many cases. The domination number value too has a correction of + or – 1. Hence, instead of relying on just the domination number value, there's a need to observe both the paths obtained & the number itself and thus correlate between them so as to find the accurate number, like in the case of n=7, the path we suggested can't be obtained by the domination tree, yet it's the most optimal one.

CONCLUSION

The report illustrates the vast potential of Circulant graphs and their potential applications in various fields. As we progressed with this project, we tried to reduce the number of comparisons by deleting some redundant branches which affect the branching factor and thus increasing the running time of the program which already is a function of the input value n, being an NP-hard problem. Due to the computational constraints, we had to rely on a certain number of data-sets to verify the answers we obtained. We've experimentally found out the domination paths for chord-lengths of 6 & 7. The work to generalize it for other 'k' values is for future consideration. We further aim to include the path nodes along with the domination numbers so as to get an accurate path for a given graph which can coincide with the domination number thus obtained.

REFERENCES

1. **A. Michael Alphonse:** Domination on Circulant Circ(n,{1,4}) graphs.
2. **A. Michael Alphonse:** Domination on circulant graphs with two chord lengths.
3. **Nadar Jafari Rad (2009):** Domination on circulant graphs.