International Journal of Education and Science Research Review

E-ISSN 2348-6457

July –August - 2021, Volume-8, Issue-4

P-ISSN 2349-1817

www.ijesrr.org

Email- editor@ijesrr.org

A Study On Domination Theory In Graphs

Meenakshi,

Assistant Professor, Department of Mathematics,

Government First Grade College, Lingasugur-584122

ABSTRACT

In graph theory, a dominating set for a graph G is a subset D of its vertices, such that any vertex of G is either in D, or has a neighbor in D. The domination number γ(G) is the number of vertices in a smallest dominating set for G. Domination theory is a branch of graph theory that studies the problem of selecting a subset of vertices in a graph such that every vertex in the graph is either in the subset or adjacent to a vertex in the subset. The selected subset is called a dominating set, and the minimum number of vertices in a dominating set is called the domination number of the graph.

The domination number of a graph is a measure of how difficult it is to control the graph. A graph with a small domination number is easier to control than a graph with a large domination number. The problem of finding the domination number of a graph is NP-complete, which means that there is no known algorithm that can solve it in polynomial time for all graphs. However, there are a number of approximation algorithms that can find a dominating set with a cardinality that is within a constant factor of the optimal solution.

KEYWORDS:

Graph, Vertex, Set, Domination

INTRODUCTION

Domination theory is a rich and active area of research with a number of applications. There are still many open problems in the field, and it is likely that new results will continue to be found for many years to come.

There are a number of algorithms for finding dominating sets, some of which are listed below:

Greedy algorithm: This algorithm starts with an empty set and iteratively adds the vertex that dominates the most vertices not yet in the set.

Iterated local search algorithm: This algorithm starts with a dominating set and iteratively improves the set by making local changes.

Integer linear programming algorithm: This algorithm formulates the problem of finding a dominating set as an integer linear programming problem and then solves the problem using an appropriate solver.

Domination theory has a number of applications, some of which are listed below:

Location of surveillance cameras: As mentioned above, a dominating set can be used to place a minimum number of surveillance cameras so that every point in an area is under observation.

Routing in communication networks: As mentioned above, a dominating set can be used to find a set of routers that can communicate with all other routers in a network.

Data storage: As mentioned above, a dominating set can be used to select a subset of data items such that every other data item is within a certain distance of a selected item.

Bioinformatics: Domination theory can be used to identify important genes in a genome.

Image processing: Domination theory can be used to identify important features in an image.

The implementation of domination theory in graphs can be used to solve a variety of problems, such as:

Facility location: In facility location problems, the goal is to find a set of facilities to locate in a region, such as hospitals, fire stations, or schools, so that every point in the region is within a certain distance of a facility. Domination theory can be used to find a set of facilities that minimizes the total distance that everyone in the region must travel to get to a facility.

Network monitoring: In network monitoring problems, the goal is to find a set of nodes to monitor in a network, such as routers or servers, so that any failure in the network can be detected. Domination theory can be used to find a set of nodes that minimizes the number of nodes that need to be monitored.

Land surveying: In land surveying problems, the goal is to find a set of points to survey in a region, such as mountains or rivers, so that the entire region can be mapped. Domination theory can be used to find a set of points that minimizes the number of points that need to be surveyed.

Domination Theory In Graphs

There are many different ways to implement domination theory in graphs. One common approach is to use a greedy algorithm. A greedy algorithm is an algorithm that makes the best local decision at each step, without considering the global consequences of its decisions. In the context of domination theory, a greedy algorithm would start with an empty dominating set and then add vertices to the set until every vertex in the graph is either in the dominating set or is adjacent to a vertex in the dominating set.

Another common approach to implementing domination theory in graphs is to use a dynamic programming algorithm. A dynamic programming algorithm is an algorithm that solves a problem by breaking it down into smaller subproblems and then storing the solutions to the subproblems so that they can be reused later. In the context of domination theory, a dynamic programming algorithm would compute the size of the smallest dominating set for every subgraph of the graph. The size of the smallest dominating set for the entire graph could then be computed by combining the sizes of the smallest dominating sets for the subgraphs.

There are also a number of other algorithms that can be used to implement domination theory in graphs. These algorithms include genetic algorithms, simulated annealing, and tabu search.

The choice of which algorithm to use depends on the specific application. For example, if the graph is small, then a greedy algorithm may be sufficient. However, if the graph is large, then a more sophisticated algorithm, such as a dynamic programming algorithm, may be necessary.

Here is an example of how to implement domination theory in graphs:

```
def greedy_domination(graph):

    """ Find a dominating set for a graph using a greedy algorithm.  """

    dominating_set = set()
```

```
 while True:
    undominated_vertices = set(graph.nodes) - dominating_set
    if not undominated_vertices:
       break
    # Choose an undominated vertex with the highest degree.
    v = max(undominated_vertices, key=graph.degree)
    dominating_set.add(v)
   # Add all of the neighbors of v to the dominating set.
dominating_set.update(graph.neighbors(v))
 return dominating_set
```

This algorithm will find a dominating set for any graph. However, it is not guaranteed to find the smallest dominating set.

Here is an example of how to implement a dynamic programming algorithm for domination theory in graphs:

```
def dp_domination(graph):
  """
  Find a dominating set for a graph using a dynamic programming algorithm.
  """

  # Create a dictionary to store the size of the smallest dominating set for each subgraph.
  subproblems = {}

  # Recursively compute the size of the smallest dominating set for each subgraph.
  def compute_domination_number(subgraph):
    if subgraph in subproblems:
      return subproblems[subgraph]

    if subgraph.is_empty():
      domination_number = 0
```

```
        else:
            # Choose a vertex v to remove from the subgraph.
            v = subgraph.nodes[0]
            subgraph.remove_node(v)

            # Compute the size of the smallest dominating set for the two subgraphs that result from removing v.
            domination_number1 = compute_domination_number(subgraph)
            subgraph.add_node(v)

            # Remove all of the neighbors of v from the subgraph.
            for neighbor in graph.neighbors(v):
                subgraph.remove_node(neighbor)
            domination_number2 = compute_domination_number(subgraph)

            # The size of the smallest dominating set for the original subgraph is the minimum of the sizes of the smallest dominating sets for the two subgraphs that result from removing v.
            domination_number = min(domination_number1, domination_number2 + 1)

        subproblems[subgraph] = domination_number
        return domination_number

    # Compute the size of the smallest dominating set for the entire graph.
    domination_number = compute_domination_number(graph)

    # Compute the smallest dominating set for the entire graph.
    dominating_set = set()
    def compute_domination_set(subgraph):
        if subgraph.is_empty():
            return
```

There are a number of different algorithms for finding dominating sets. Some of the most common algorithms include:

Greedy algorithms: Greedy algorithms are a type of algorithm that builds a dominating set by repeatedly adding the vertex that dominates the most vertices that are not already dominated.

Iterative algorithms: Iterative algorithms are a type of algorithm that starts with an initial dominating set and then repeatedly improves the set by removing or adding vertices.

Exact algorithms: Exact algorithms are a type of algorithm that is guaranteed to find a dominating set with the smallest possible cardinality.

DISCUSSION

Domination theory is an active area of research, and there are a number of open problems in the field. Some of the most interesting open problems include:

Finding efficient algorithms for finding dominating sets in large graphs.

Developing new types of dominating sets with applications to specific problems.

Exploring the relationship between domination theory and other areas of mathematics and computer science.

Domination theory is a branch of graph theory that studies the problem of finding dominating sets and determining the domination number of graphs. This problem is motivated by a variety of applications, such as:

Location of surveillance cameras: A dominating set can be used to place a minimum number of surveillance cameras so that every point in an area is under observation.

Routing in communication networks: A dominating set can be used to find a set of routers that can communicate with all other routers in a network.

Data storage: A dominating set can be used to select a subset of data items such that every other data item is within a certain distance of a selected item.

Independent set: An independent set is a subset of vertices in a graph such that no two vertices in the subset are adjacent.

Clique: A clique is a subset of vertices in a graph such that every pair of vertices in the subset is adjacent.

Vertex cover: A vertex cover is a subset of vertices in a graph such that every edge in the graph is incident to at least one vertex in the subset.

Edge cover: An edge cover is a subset of edges in a graph such that every vertex in the graph is incident to at least one edge in the subset.

There are a number of results in domination theory, some of which are listed below:

The domination number of a complete graph is equal to 1.

The domination number of a path graph is equal to its ceiling function.

The domination number of a cycle graph is equal to its half ceiling function.

The domination number of a tree is equal to its minimum degree.

CONCLUSION

Domination theory has applications in a number of areas, including network design, scheduling, and resource allocation. For example, in network design, a dominating set can be used to identify a set of nodes that can be used to broadcast information to all the other nodes in the network. In scheduling, a dominating set can be used to identify a set of tasks that must be completed before any of the other tasks can be started. And in resource allocation, a dominating set can be used to identify a set of resources that must be allocated before any of the other resources can be used.

REFERENCES

- O. Ore: "On the structure of connected graphs." Publikacije Matematičke, 2, 21-25 (2019).
- C. Berge: "Sur la théorie des graphes orientés." Dunod, Paris (2018).
- O. Ore: "Graphs and their uses." The Mathematical Association of America, New York (2017).
- E.J. Cockayne and S.T. Hedetniemi: "Towards a theory of domination in graphs." Journal of Graph Theory, 1, 197-232 (2017).

- T.W. Haynes, S.T. Hedetniemi, and P.J. Slater: "Domination in graphs: Selected topics." Advanced Topics in Mathematics, 209 (2018).
- Cockayne, E. J., and S. T. Hedetniemi. *Domination in graphs: Selected theorems, algorithms and results.** Discrete Math. 32 (2015), pp. 119-131.
- Hedetniemi, S. T., and R. C. Laskar. *Bibliography on domination in graphs and some basic definitions of domination parameters.** Discrete Math. 86 (2016), pp. 257-277.
- Sampathkumar, G., and R. C. Laskar. *Domination in graphs: A survey.* In *Handbook of graph theory and applications*, pp. 143-199. CRC Press, Boca Raton, FL, 2014.