Vertex Cover Problem Using Brute Force and Heuristic approach

A COURSE PROJECT REPORT

Submitted by

SMRIDH VARMA [RA2111003011530]
BHAVIKA RUSTAGI [RA2111003011556]

Under the Guidance of

Dr. Kishore Anthuvan Sahayaraj K
Assistant Professor,
Department of Computing Technologies

In partial fulfillment for the course of

18CSC204J - Design and Analysis of Algorithms

In CTECH

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE ENGINEERING

SCHOOL OF COMPUTING

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR - 603203

MAY 2023

COLLEGE OF ENGINEERING & TECHNOLOGY
SRM INSTITUTE OF SCIENCE & TECHNOLOGY
S.R.M. NAGAR, KATTANKULATHUR – 603 203
Chengalpattu District

BONAFIDE CERTIFICATE

Certified that this mini project report "**Vertex Cover Problem Using Brute Force and Heuristic approach**" is the bonafide work of **Smridh Varma (RA2111003011530) and Bhavika Rustagi (RA2111003011556)**, who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Faculty In-Charge	**HEAD OF THE DEPARTMENT**
Dr. Kishore Anthuvan Sahayaraj K	Dr. M Pushpalatha
Assistant Professor,	Professor/ Head,
Department of Computing Technologies	Department of Computing Technologies
SRM Institute of Science and Technology	SRM Institute of Science and Technology

ACKNOWLEDGEMENT

We delightfully express our heartfelt thanks to our honourable **Vice Chancellor Dr. C Muthamizhchelvan**, for being the beacon in all our endeavours.

We would like to express my warmth of gratitude to our **Registrar Dr. S. Ponnusamy**, for his encouragement.

We express our profound gratitude to our **Dean (College of Engineering and Technology) Dr. T. V. Gopal**, for bringing out novelty in all executions.

We would like to express our heartfelt thanks to the **Chairperson, School of Computing Dr. Revathi Venkataraman**, for imparting confidence to complete my course project.

We extend my gratitude to our **HoD Dr. M. Pushpalatha, Professor & Head, CTECH**, and my professors for their support.

We are highly thankful to our Course project **Faculty Dr. Kishore Anthuvan Sahayaraj K, Assistant Professor, CTECH**, for his assistance, timely suggestion, and guidance throughout the duration of this course project.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project.

ABSTRACT

In this report, we present two different approaches to solve the vertex cover problem, a well-known problem in graph theory. The vertex cover problem involves finding the smallest set of vertices in an undirected graph that covers all its edges. It is an NP-hard problem, meaning that there is no known algorithm that can solve it in polynomial time.

The first approach presented is a brute-force algorithm that uses recursion to search for the smallest vertex cover of a given graph. The algorithm works by considering all possible combinations of vertices in the graph, making it very slow and impractical for large instances of the problem. The time complexity of the brute-force algorithm is O(2^n), where n is the number of vertices in the graph.

The second approach presented is a heuristic algorithm that uses a greedy strategy to find a near-optimal vertex cover. The algorithm works by iteratively selecting an uncovered edge from the graph and adding its endpoints to the vertex cover, removing all edges incident on these vertices from the graph, and repeating until all edges are covered. The heuristic algorithm is much faster than the brute-force algorithm and can handle larger instances of the problem, with a time complexity of O(m), where m is the number of edges in the graph. However, the heuristic algorithm may not always find the optimal solution, making it less reliable than the brute-force algorithm.

We analyze the time complexity of both algorithms and compare their performance on different instances of the vertex cover problem. We conclude that the choice of algorithm depends on the size of the problem and the desired level of optimality. The brute-force algorithm is more reliable but extremely slow, whereas the heuristic algorithm is faster but may not always find the optimal solution. Therefore, researchers and practitioners must carefully consider the size of the problem and the desired level of accuracy when choosing which algorithm to use.

TABLE OF CONTENTS

CHAPTERS	CONTENTS	PAGE NO.
1	INTRODUCTION	6
2	BRUTE-FORCE ALGORITHM	7
3	HEURISTIC ALGORITHM	9
4	TIME COMPLEXITY ANALYSIS	11
5	PERFORMANCE COMPARISON	12
6	CONCLUSION	14
	REFERENCES	15

1. INTRODUCTION

In this report, we focus on the vertex cover problem, a well-known problem in graph theory. We present two different approaches to solving the problem, and we analyze the time complexity of both algorithms and compare their performance on different instances of the problem.

The vertex cover problem is an NP-hard problem, meaning that there is no known algorithm that can solve it in polynomial time. The problem involves finding the smallest set of vertices in an undirected graph that covers all its edges. The vertex cover problem is important in computer science, as it has applications in fields such as network design, computational biology, and social network analysis.

The brute-force algorithm is the first approach presented in this report. The algorithm uses recursion to search for the smallest vertex cover of a given graph. The algorithm works by considering all possible combinations of vertices in the graph, making it very slow and impractical for large instances of the problem. However, the brute-force algorithm is guaranteed to find the optimal solution.

The heuristic algorithm is the second approach presented in this report. The algorithm uses a greedy strategy to find a near-optimal vertex cover. The algorithm works by iteratively selecting an uncovered edge from the graph and adding its endpoints to the vertex cover, removing all edges incident on these vertices from the graph, and repeating until all edges are covered. The heuristic algorithm is much faster than the brute-force algorithm and can handle larger instances of the problem, but may not always find the optimal solution.

In this report, we analyze the time complexity of both algorithms and compare their performance on different instances of the vertex cover problem. We conclude that the choice of algorithm depends on the size of the problem and the desired level of optimality. The brute-force algorithm is more reliable but extremely slow, whereas the heuristic algorithm is faster but may not always find the optimal solution. Therefore, researchers and practitioners must carefully consider the size of the problem and the desired level of accuracy when choosing which algorithm to use.

2. BRUTE-FORCE ALGORITHM

The brute-force algorithm is a simple and straightforward approach to solve the vertex cover problem. The algorithm works by recursively considering all possible subsets of vertices in the given graph and selecting the smallest subset that covers all its edges. This approach ensures that the algorithm always finds the optimal solution to the problem, but the time complexity grows exponentially with the size of the graph, making it impractical for large instances of the problem.

The algorithm starts by selecting an arbitrary vertex in the graph and considering two cases: either the vertex is in the vertex cover or it is not. If the vertex is in the cover, then all edges incident on the vertex are covered, and the algorithm can proceed recursively to solve the vertex cover problem on the remaining graph. If the vertex is not in the cover, then all edges incident on the vertex are not covered, and the algorithm can proceed recursively to solve the vertex cover problem on the remaining graph with the selected vertex removed.

The algorithm continues this process for all possible subsets of vertices in the graph, keeping track of the size of the smallest subset that covers all edges. Once all possible subsets have been considered, the algorithm returns the smallest subset found as the solution to the vertex cover problem.

The time complexity of the brute-force algorithm is O(2^n), where n is the number of vertices in the graph. This is because there are 2^n possible subsets of vertices in the graph, and the algorithm considers each subset in turn. This exponential growth makes the brute-force algorithm impractical for graphs with even moderately sized n, as the number of possible subsets quickly becomes unmanageable.

In summary, the brute-force algorithm is a simple and reliable approach to solve the vertex cover problem, but its exponential time complexity makes it impractical for large instances of the problem. The algorithm considers all possible subsets of vertices in the graph, ensuring that it always finds the optimal solution, but its running time grows exponentially with the size of the graph.

PSEUDOCODE:

```
function vertex_cover_brute_force(G):
    n = G.number_of_vertices()
    V = G.vertices()
    for i = 0 to 2^n - 1:
    S = set of vertices corresponding to binary representation of i
    if is_vertex_cover(G, S):
    return S
    return None

function is_vertex_cover(G, S):
    for e in G.edges():
    if e[0] not in S and e[1] not in S:
    return False
    return True
```

3. HEURISTIC ALGORITHM

The heuristic algorithm is a faster alternative to the brute-force algorithm for solving the vertex cover problem. While the algorithm may not always find the optimal solution, it can handle larger instances of the problem and still provide near-optimal results. The heuristic algorithm works by iteratively selecting edges in the graph and adding their endpoints to the vertex cover until all edges are covered.

The algorithm starts by initializing an empty set of vertices as the vertex cover. It then selects an arbitrary uncovered edge in the graph and adds its endpoints to the vertex cover. This ensures that both endpoints of the edge are covered, and the edge itself is also covered. The algorithm then removes all edges incident on the selected vertices from the graph and repeats the process with the remaining uncovered edges.

The algorithm continues this process until all edges in the graph are covered. At this point, the set of vertices in the vertex cover is returned as the solution to the vertex cover problem.

The heuristic algorithm is based on a greedy strategy and does not consider all possible subsets of vertices. However, it provides a reasonable approximation of the optimal solution in much less time than the brute-force algorithm. The time complexity of the heuristic algorithm is O(mn), where n is the number of vertices in the graph, and m is the number of edges. This is because the algorithm iteratively selects edges and removes all edges incident on the selected vertices from the graph.

The heuristic algorithm is not guaranteed to find the optimal solution, and the size of the vertex cover returned may be slightly larger than the optimal solution. However, the algorithm can handle larger instances of the problem and still provide reasonably accurate results. Additionally, the algorithm is much faster than the brute-force algorithm, making it a more practical solution for most applications.

In summary, the heuristic algorithm is a faster alternative to the brute-force algorithm for solving the vertex cover problem. The algorithm uses a greedy strategy to iteratively select edges and add their endpoints to the vertex cover until all edges are covered. While the algorithm may not always find the optimal solution, it can handle larger instances of the problem and still provide reasonably accurate results in much less time than the brute-force algorithm.

PSEUDOCODE:

```
function vertex_cover_heuristic(G):
    V = G.vertices()
    E = G.edges()
    C = set()
    while E is not empty:
        e = select_edge(E)
        C = C.union(set(e))
        remove_edges(E, e)
    return C


function select_edge(E):
    return any edge in E


function remove_edges(E, e):
    for edge in E:
        if edge[0] in e or edge[1] in e:
            E.remove(edge)
```

4. TIME COMPLEXITY ANALYSIS

The time complexity of an algorithm refers to the amount of time it takes to solve a problem as a function of the size of the input. The time complexity is an important metric for evaluating the efficiency of an algorithm, and it is often expressed using big O notation. In the case of the vertex cover problem, there are two primary algorithms for solving the problem: the brute-force algorithm and the heuristic algorithm.

The brute-force algorithm has a time complexity of O(2^n), where n is the number of vertices in the graph. This is because the algorithm considers all possible subsets of vertices in the graph, of which there are 2^n. For each subset, the algorithm checks whether it is a valid vertex cover and keeps track of the smallest valid vertex cover found. Because the number of possible subsets grows exponentially with the size of the input, the brute-force algorithm is only practical for small instances of the problem.

The heuristic algorithm has a time complexity of O(mn), where n is the number of vertices in the graph, and m is the number of edges. This is because the algorithm iteratively selects edges and removes all edges incident on the selected vertices from the graph. The algorithm continues until all edges are covered. While the heuristic algorithm does not guarantee an optimal solution, it can often provide a reasonable approximation in much less time than the brute-force algorithm.

In practice, the time complexity of an algorithm is not the only consideration when selecting an algorithm for solving a problem. Other factors such as memory usage, scalability, and ease of implementation are also important. However, the time complexity provides a useful baseline for evaluating the performance of different algorithms and comparing them to each other.

In summary, the brute-force algorithm has a time complexity of O(2^n), while the heuristic algorithm has a time complexity of O(mn). The brute-force algorithm is only practical for small instances of the problem, while the heuristic algorithm can handle larger instances and still provide reasonably accurate results. The time complexity provides a useful metric for evaluating the performance of different algorithms, but other factors should also be considered when selecting an algorithm for solving a problem.

5. PERFORMANCE COMPARISON

To compare the performance of the brute-force and heuristic algorithms for solving the vertex cover problem, we can analyze their runtimes on different instances of the problem. In general, the brute-force algorithm is expected to be slower than the heuristic algorithm, as the brute-force algorithm considers all possible subsets of vertices in the graph, while the heuristic algorithm uses a greedy approach to iteratively select edges and add their endpoints to the vertex cover.

Let's consider an example of a small graph with 4 vertices and 4 edges, shown below:

```
1 -- 2

|     |

3 -- 4
```

Using the brute-force algorithm, we can generate all possible subsets of vertices, as shown below:

{}, {1}, {2}, {3}, {4}, {1,2}, {1,3}, {1,4}, {2,3}, {2,4}, {3,4}, {1,2,3}, {1,2,4}, {1,3,4}, {2,3,4}, {1,2,3,4}

For each subset, we can check whether it is a valid vertex cover and keep track of the smallest valid vertex cover found. In this case, we can see that the optimal solution is {1,2}. However, the brute-force algorithm considers 2^n subsets, where n is the number of vertices, so it needs to check 16 different subsets before finding the optimal solution.

Using the heuristic algorithm, we can start by selecting an arbitrary uncovered edge in the graph, such as the edge between vertices 1 and 2. We then add both vertices 1 and 2 to the vertex cover, remove all edges incident on these vertices, and repeat the process with the remaining uncovered edges. In this case, we would next select the edge between vertices 3 and 4, and add both vertices 3 and 4 to the vertex cover. This completes the vertex cover, and we obtain the solution {1,2,3,4}.

As we can see from this example, the brute-force algorithm considers many more subsets than the heuristic algorithm, leading to longer runtimes. In general, the runtime of the brute-force algorithm grows exponentially with the number of vertices, while the runtime of the heuristic algorithm grows linearly with the number of edges. Thus, the heuristic algorithm can handle larger instances of the problem and still provide reasonably accurate results in much less time than the brute-force algorithm.

In summary, the performance of the brute-force and heuristic algorithms for solving the vertex cover problem can be compared by analyzing their runtimes on different instances of the problem. The brute-force algorithm considers all possible subsets of vertices, leading to longer runtimes for larger instances, while the heuristic algorithm uses a greedy approach to iteratively select edges and add their endpoints to the vertex cover, leading to faster runtimes for larger instances.

6. CONCLUSION

In conclusion, the vertex cover problem is a well-known problem in graph theory that involves finding a minimum set of vertices that covers all edges in a graph. The problem is NP-complete, meaning that no known algorithm can solve it in polynomial time.

Two primary algorithms for solving the vertex cover problem are the brute-force algorithm and the heuristic algorithm. The brute-force algorithm considers all possible subsets of vertices in the graph, while the heuristic algorithm uses a greedy approach to iteratively select edges and add their endpoints to the vertex cover. The brute-force algorithm has a time complexity of O(2^n), while the heuristic algorithm has a time complexity of O(mn), where n is the number of vertices in the graph, and m is the number of edges.

While the brute-force algorithm guarantees an optimal solution, it can only handle small instances of the problem due to its exponential time complexity. In contrast, the heuristic algorithm does not guarantee an optimal solution but can handle larger instances of the problem and still provide reasonably accurate results in much less time than the brute-force algorithm.

In practice, the choice of algorithm for solving the vertex cover problem will depend on the specific problem instance and the requirements of the application. If an optimal solution is required and the size of the input is small, the brute-force algorithm may be a good choice. If a fast solution is required, and an approximation is acceptable, the heuristic algorithm may be a better choice.

In summary, the vertex cover problem is an important problem in graph theory, and the choice of algorithm for solving it will depend on the specific problem instance and requirements of the application. Both the brute-force and heuristic algorithms have their strengths and weaknesses, and selecting the appropriate algorithm will depend on a variety of factors, including runtime, memory usage, and accuracy of the solution.

REFERENCES

1. https://stackoverflow.com/questions/66513948/brute-force-exact-algorithm-for-vertex-cover

2. https://www.geeksforgeeks.org/introduction-and-approximate-solution-for-vertex-cover-problem/

3. https://lcs.ios.ac.cn/~caisw/VC.html#:~:text=Heuristic%20algorithms%20for%20Minimum%20Vertex,least%20one%20endpoint%20in%20it

4. https://chat.openai.com/