Vertex Cover Problem: 2-Approximation Algorithm and Complexity Analysis

1 Introduction

The Vertex Cover problem is a fundamental NP-complete problem in graph theory and computer science. Given an undirected graph $G = (V, E)$, a vertex cover is a subset $C \subseteq V$ such that every edge in E has at least one endpoint in C. The optimization version seeks the minimum vertex cover.

2 2-Approximation Algorithm

2.1 Algorithm Description

The algorithm follows these steps:

1. Initialize an empty vertex cover C
2. While there are uncovered edges in E:
   (a) Select an arbitrary uncovered edge (u, v)
   (b) Add both u and v to C
   (c) Remove all edges incident to u or v from consideration
3. Return C as the approximate vertex cover

2.2 Pseudocode

3 Time Complexity Analysis

3.1 Mathematical Analysis

Let $n = |V|$ be the number of vertices and $m = |E|$ be the number of edges.

1. **Edge Selection:** Each iteration of the while loop selects one edge. In the worst case, we select $\lfloor m/2 \rfloor$ edges.

Algorithm 1 2-Approximation Algorithm for Vertex Cover

procedure ApproxVertexCover($G = (V, E)$)
  $C \leftarrow \emptyset$
  $E' \leftarrow E$ ▷ Copy of edge set
  while $E' \neq \emptyset$ **do**
    Choose an arbitrary edge $(u, v) \in E'$
    $C \leftarrow C \cup \{u, v\}$
    Remove from E' every edge incident to u or v
  end while
  return C
end procedure

2. **Edge Removal:** For each selected edge (u, v), we need to remove all edges incident to u or v. Using an adjacency list representation:
    - Finding all edges incident to a vertex takes $O(\deg(u))$ time
    - Removing edges from E' can be done in $O(1)$ per edge if using appropriate data structures
3. **Total Complexity:** Each edge is examined at most once (when it's either selected or removed). Therefore, the total time complexity is:
$$T(n, m) = O(m)$$
4. **Space Complexity:** The algorithm requires $O(n + m)$ space to store the graph and additional $O(n)$ space for the vertex cover.

3.2 Detailed Step-by-Step Analysis

1. **Initialization:** $O(1)$
2. **Creating edge list:** $O(m)$
3. **Main while loop:**
    - Each iteration processes one edge (u, v)
    - Number of iterations: at most $\lfloor m/2 \rfloor$
    - Per iteration:
        - Edge selection: $O(1)$ with proper tracking
        - Adding vertices to cover: $O(1)$
        - Marking incident edges as covered: $O(\deg(u) + \deg(v))$
4. **Total:** Since each edge is processed at most once in the marking step, total work is:
$$\sum_{i=1}^{k} (\deg(u_i) + \deg(v_i)) \leq 2m = O(m)$$
where k is the number of selected edges.

4 Approximation Ratio Proof

4.1 Theorem

The algorithm achieves a 2-approximation ratio.

4.2 Proof

Let A be the set of edges selected by the algorithm, and C be the vertex cover produced.

1. No two edges in A share an endpoint (by construction).
2. Any vertex cover must include at least one endpoint from each edge in A.
3. Therefore, any vertex cover (including the optimal C^*) must have size at least $|A|$.
4. The algorithm adds 2 vertices for each edge in A, so $|C| = 2|A|$.
5. Thus:
$$|C| = 2|A| \leq 2|C^*|$$

5 Conclusion

The 2-approximation algorithm for Vertex Cover provides a practical solution with guaranteed performance bounds. Its linear time complexity makes it suitable for large-scale applications, while the 2-approximation ratio ensures the solution is never more than twice the optimal size. The algorithm's simplicity and efficiency make it a valuable tool in practical applications where exact solutions are computationally prohibitive.