Approximation Algorithm for Vertex Cover

Supervisor: Prof. Vishnu Narayanan (IE 609)

Authors:

Guramrit Singh: 210050061

Rahul Choudhary: 200070065

Indian Institute of Technology, Bombay

Table of Contents

Table of Contents

Vertex Cover

Vertex Cover: A set of vertices where each edge in the graph has at least one endpoint in the set.

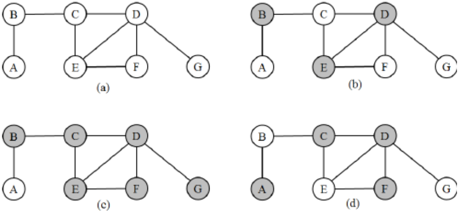


Figure: (a) An unweighted graph G. (b) The optimal vertex cover for G. (c),(d) Other non-optimal vertex covers for G.

Table of Contents

Integer Linear Program Formulation

- **Variables:**
  - Let w_v denote the weight of vertex v, where $w_v \geq 0$.
  - Let x_v be a binary variable associated with vertex v, indicating its inclusion in the vertex cover. Thus, $x_v \in \{0, 1\}$.
- **Constraints:**
  - For every edge $(u, v) \in E$, the constraint $x_u + x_v \geq 1$ ensures that at least one of the vertices u or v is included in the vertex cover to cover the edge (u, v).
- **Description:**

$$
\begin{aligned}
\text{ILP:} \quad & \min \sum_{v \in V} w_v x_v \\
\text{s.t.} \quad & x_u + x_v \geq 1, \quad \forall (u, v) \in E \\
& x_v \in \{0, 1\}, \quad \forall v \in V
\end{aligned}
$$

- **Integer Linear Program** is one of the **NP-complete** problems and so no efficient algorithms are known for it. Next, we apply **LP relaxation** on ILP to get an approximate formulation.

Table of Contents

LP Relaxation of the ILP

$$\begin{aligned}
\text{LP:} \quad \min \sum_{v \in V} w_v x_v \\
\text{s.t.} \quad x_u + x_v \geq 1, \quad \forall (u, v) \in E, \\
x_v \geq 0, \quad \forall v \in V.
\end{aligned}$$

Explanation:

- The binary constraint $x_v \in \{0, 1\}$ in the ILP is replaced by the continuous relaxation $0 \leq x_v \leq 1$.
- Since $w_v \geq 0$, we can as well remove $x_v \leq 1$ constraint since any optimal solution will always satisfy it.
- This relaxation converts the Integer Linear Program (ILP) into a Linear Program (LP), which can be solved efficiently using techniques like the simplex algorithm or interior-point methods as seen in class.

Dual of LP (DP) Formulation

Dual (LP):

- Define a dual variable $y_{uv} \geq 0$ for each constraint $x_u + x_v \geq 1$ corresponding to edge $(u, v) \in E$.
- Description:

$$
\begin{aligned}
\text{DP:} \quad & \max \sum_{(u,v)\in E} y_{uv} \\
& \text{s.t.} \quad \sum_{u \text{ such that } (u,v)\in E} y_{uv} \leq w_v, \quad \forall v \in V \\
& \qquad y_{uv} \geq 0, \quad \forall (u, v) \in E.
\end{aligned}
$$

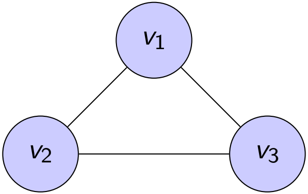
Complementary Slackness:

- The complementary slackness conditions for the primal and dual solutions are:

$$
\begin{aligned}
&\text{(CS1)} \quad y_{uv} > 0 \implies x_u + x_v = 1, \quad \forall (u,v) \in E, \\
&\text{(CS2)} \quad \sum_{u \text{ such that } (u,v)\in E} y_{uv} < w_v \implies x_v = 0, \quad \forall v \in V.
\end{aligned}
$$

LP-OPT is not always equal to VC-OPT

- **Example:** Consider a complete graph of size 3 where each vertex has a weight of 1.



- **VC-OPT:** The minimum vertex cover includes any two vertices (e.g., $\{v_1, v_2\}$). Therefore:

$$\text{VC-OPT} = w_{v_1} + w_{v_2} = 1 + 1 = 2.$$

- **LP-OPT:** The LP relaxation allows fractional values. Assign $x_{v_1} = x_{v_2} = x_{v_3} = 0.5$, satisfying all constraints Objective value:

$$\text{LP-OPT} = 0.5 + 0.5 + 0.5 = 1.5 < \text{VC-OPT}$$

Table of Contents

Algorithm Overview

- As noted in the previous example, the optimal solution to the LP may not correspond to a valid vertex cover. Thus, it is unlikely that an integral primal solution (i.e. a vertex cover) will satisfy the complementary slackness conditions with its dual solution.
- Since requiring both complementary slackness conditions to hold may be too stringent, our algorithm ensures that at each step, the primal and dual solutions satisfy (CS2), gradually improving the dual objective. The algorithm terminates when certain conditions are met.
- We will later demonstrate that this is a **2-approximation algorithm**.

Algorithm for Finding a Vertex Cover

Following are the steps involved in the algorithm:

1. **Initialize** an empty set S for the vertex cover:

$$S = \emptyset$$

   where S is the set representing the vertex cover.

2. Begin with a **feasible dual** solution, i.e., set all $y_{uv} = 0$.
3. Check if a valid vertex cover is formed:
   - If yes, output S.
   - If no, select an uncovered edge uv and **increment its dual variable** y_{uv} until one of the dual constraint becomes tight.
4. For any vertex u with a **tight dual constraint**:
   - Set $x_u = 1$.
   - **Include** vertex u in the set S.
5. Goto step 3.

Proof of 2-Approximation

Claim

The algorithm provides a 2-approximation for the vertex cover, i.e.

$$ALG\text{-}VC \leq 2 \cdot OPT\text{-}VC$$

where $ALG\text{-}VC$ is the vertex cover returned by the algorithm and $OPT\text{-}VC$ is the optimal vertex cover.

Proof: Using complementary slackness, we have the following:

$$ALG\text{-}VC = \sum_{u \in V} w_u x_u = \sum_{u \in S} w_u = \sum_{u \in S} \sum_{v \text{ such that } (u,v) \in E} y_{uv}$$

$$\leq 2 \sum_{e \in E} y_e = 2 \cdot \text{Dual Objective}$$

Thus,

$$ALG\text{-}VC \leq 2 \cdot \text{Dual Objective} \leq 2 \cdot OPT\text{-}LP \leq 2 \cdot OPT\text{-}VC$$

Table of Contents

Graph Model for Algorithm simulation

- **The Erdős–Rényi (ER) graph model** is a random graph model used to study the structure of graphs.
- It is defined by **n vertices** and **edge probability** p between any two vertices.
- Each of the $\binom{n}{2}$ edges in the graph is **independently** included with probability p.
- This model is used to analyze **random networks**, especially in **social networks and computer networks**.
- The ER model is **simple** but forms the basis for understanding complex network behaviors in large-scale networks.

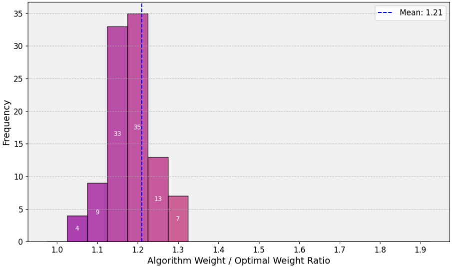
Sparse Graphs using ER

- A graph is considered sparse if the number of edges are of $\mathbf{O(n)}$, i.e. number of edges grows linearly with the number of vertices.
- For the ER model, the graph is sparse when the edge probability p is small, specifically when $p = \frac{k}{n}$, where k is a constant.
- In this case, the expected number of edges is $\frac{k(n-1)}{2}$ which is $\mathbf{O(n)}$.
- Sparse graphs exhibit **low average connectivity**, meaning most vertices are only connected to a few others.
- Sparse graphs are often used in **real-world applications**, such as social networks and computer networks, where most nodes have a limited number of connections.

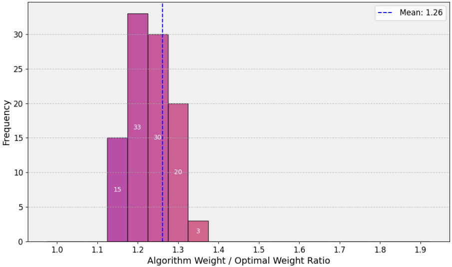
Algorithm Simulations

- Developed a Python program to generate graphs based on the Erdős–Rényi (ER) model.
- Constructed a dataset with $n = 200$ and $k \in \{1, 2, 3, 4, 5, 10\}$, where $p = \frac{k}{n}$ as defined earlier. For each value of p, 100 graph samples were generated.
- Additionally, created a dataset with a larger number of vertices, $n = 2000$ and $k = 3$, with 1000 samples.
- Implemented a separate Python program to simulate the algorithm as outlined previously.
- Used the `pulp` library to compute the optimal vertex cover.
- Compared the vertex cover produced by our algorithm with the optimal vertex cover.

Results (1/4)

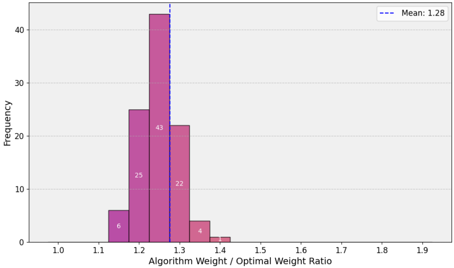


(a) $n = 200,\ k = 1$

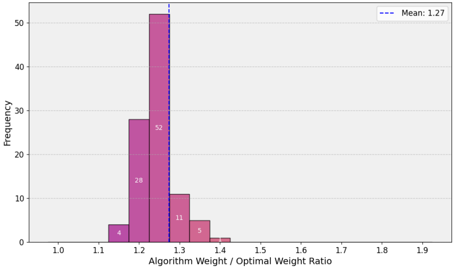


(b) $n = 200,\ k = 2$

Results (2/4)

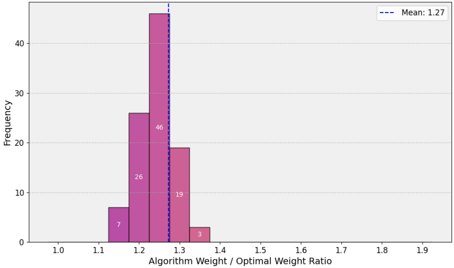


(a) $n = 200,\ k = 3$

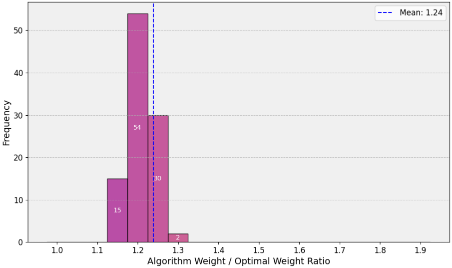


(b) $n = 200,\ k = 4$

Results (3/4)



(a) $n = 200, \ k = 5$



(b) $n = 200, \ k = 10$

Results (4/4)

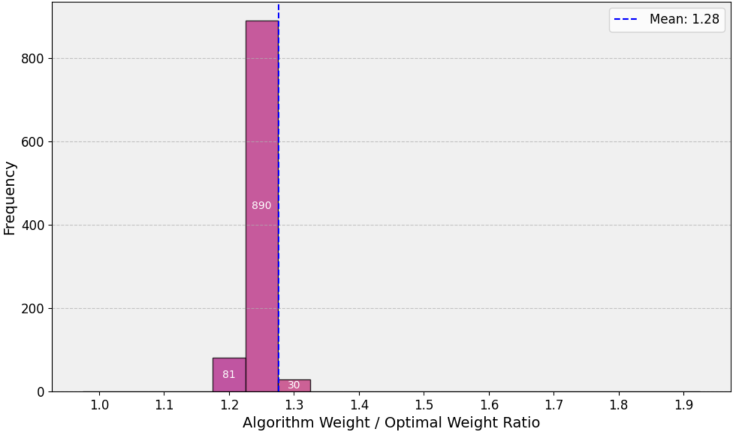


Figure: $n = 2000,\ k = 3$

Observations

- From the simulations and visualizations above, it is evident that for sparse graphs, even on larger graphs, the ratio of the algorithm's weight to the optimal vertex cover weight generally lies within the range $[1.21, 1.28]$ on average. This is substantially better than the theoretical upper bound of 2 established earlier.
- As a result, the proposed algorithm demonstrates excellent effectiveness in finding a low-weight vertex cover for sparse graphs.

Thank You!!!