

**Randomized and Approximation Algorithms for Vertex Cover**

**Instructor: Dr. Syed Hassan Shah**

**CPSC 535: Advanced Algorithms (Fall 2023)**

**Group Project 4**

**GROUP 5**

**Team Members:**

**Pallavi Khedle**

**Raja Ravindra Kathi**

**Gayathri Nagulavancha**

**Tejaskumar Pareshbhai Patel**

**Mani Krishna Sanganabhatla**

**Sri Datta Venga Sampath Venga S**

**Sravya Naragam**

**Submission date: 11/10/2023**

**Table of Contents**

[**1. Introduction 3**](#_zfkwi09akgvi)

[**2. Background 3**](#_e35ccsv4rhjo)

[**3. Randomized Algorithms for vertex cover**](#_t0fmg7d5kgfo) **4**

**3.1 Simple Randomized Algorithm****5**

**3.2 Weighted Randomized Algorithm****6**

**3.3 Local Search Randomized Algorithm****8**

[**4. Approximation Algorithms for vertex cover**](#_pssyocxrwfas) **9  
4.1 Approximation Ratio 10**

**4.2 Vertex Greedy Approximation Algorithm****10**

**4.3 Minimum Vertex Cover by Matchings****11**

[**5. Comparison between Randomzed and Approximation Algorithms 13**](#_3q90xfkwgfm2)

[**6. Current Research and Conclusion**](#_g6pfs8symw01) **14**

[**7. References**](#_aiqoatx6xkmk) **15**

### **1. Introduction**

Vertex cover is a fundamental problem in graph theory and computer science. Given a graph, the goal is to find the smallest set of vertices such that every edge in the graph is incident to at least one vertex in the set. This problem is NP-complete, meaning that there is no known polynomial-time algorithm for solving it to optimality.

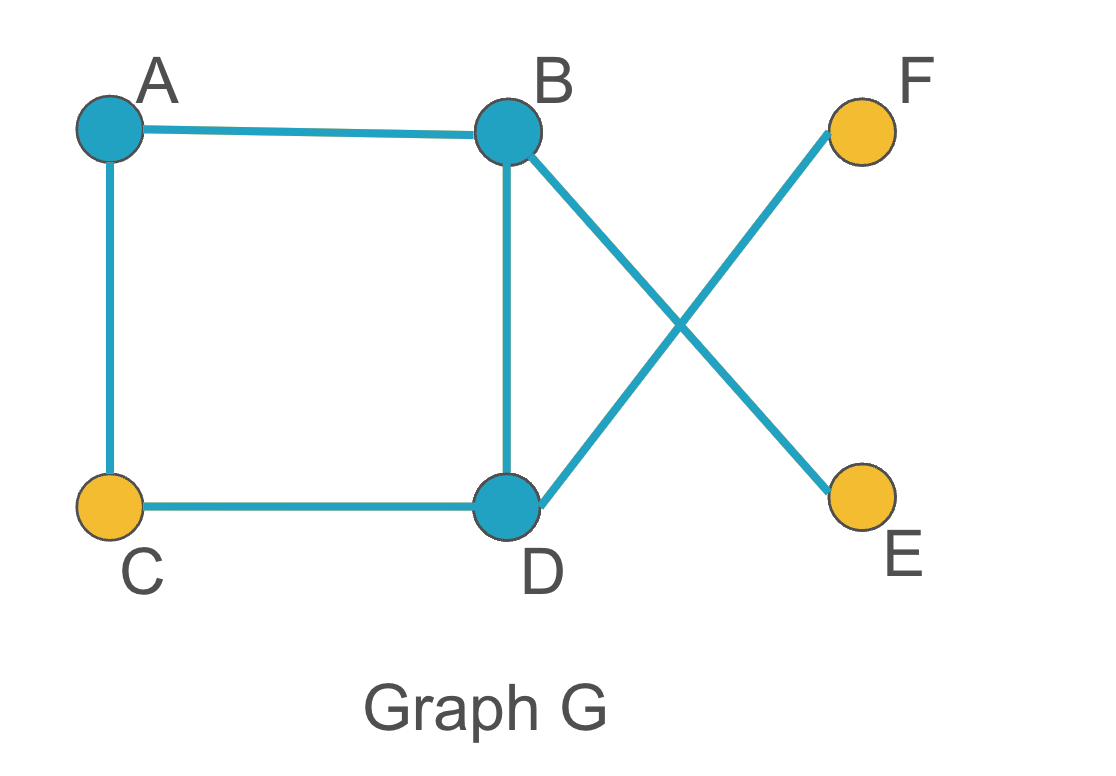


Figure 1: Graph G, to demonstrate vector cover

In Figure1, Vertices of Graph G, V = {A, B, C, D, E, F} and Vertex Cover, V’ = {A,B,D}. A, B and D vertices together cover all the edges of the graph.

Randomized algorithms and approximation algorithms are two techniques that can be used to find good solutions to NP-complete problems. Randomized algorithms use randomness to generate and improve solutions, while approximation algorithms guarantee to find a solution that is within a certain factor of the optimal solution.

### **2. Background**

Vertex cover is a well-studied problem in graph theory and computer science. It was first introduced by Richard Karp in his 1972 paper "Reducibility Among Combinatorial Problems". Karp showed that the vertex cover problem is NP-complete, meaning that it is one of the hardest problems in computer science.

Randomized algorithms for vertex cover were first developed in the early 1980s. One of the first randomized algorithms was the greedy randomized algorithm, which was introduced by Luby and Raghavan in 1986. The greedy randomized algorithm is a simple and efficient algorithm that works well in practice.

The local search randomized algorithm was introduced by Johnson and Trick in 1993. The local search randomized algorithm is more efficient than the greedy randomized algorithm, but it is also more complex to implement.

Approximation algorithms for vertex cover were first developed in the early 1990s. One of the first approximation algorithms was the greedy approximation algorithm, which was introduced by Christofides in 1975. The greedy approximation algorithm is a simple and efficient algorithm that has a good approximation ratio.

The matchings approximation algorithm was introduced by Hochbaum and Shmoys in 1986. The matchings approximation algorithm is also simple and efficient, and it has a better approximation ratio than the greedy approximation algorithm.

The following is a brief history of vertex cover algorithms:

* 1972: Richard Karp introduces the vertex cover problem.
* 1980s: Randomized algorithms for vertex cover are developed.
* 1990s: Approximation algorithms for vertex cover are developed.
* 2000s: New randomized and approximation algorithms for vertex cover are developed.

### **3. Randomized Algorithms for the Vertex Cover**

### Randomized Algorithms for the vertex cover problems are like clever explorers. These algorithms make random choices to find a good solution to the problem. Randomized algorithms finds solution to the problem of picking the smallest group of points to cover all lines in the graph. This kind of solutions might not be the perfect, but it is a smart way to handle the big problems.

*What is a Randomized Algorithm?*

Algorithms making decisions with a bit of randomness like someone rolling a fair die. In this case the algorithm uses the randomness to decide which points to include in the group. Using randomness helps the algorithm explore different possibilities and not get stuck in one way of doing things. It's like trying different paths to see which one works out better.

How does the algorithm work?

For each edge in the graph, the algorithm randomly picks one of the two edges at the end of the edge and adds it to the group. It does this for all the lines in the graph. Randomized algorithms for vertex cover aim to provide a solution that may not be optimal but is likely to be close to the optimal solution with high probability.These algorithms are particularly useful for large graphs where finding an exact solution to the vertex cover problem is computationally expensive.

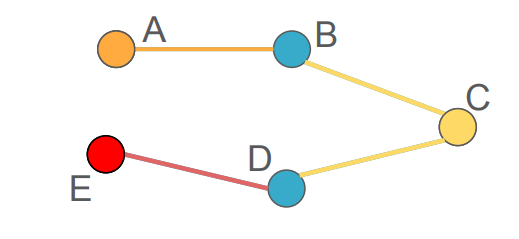
**3.1 Simple Randomized Algorithm**

*Input*: Graph G = (V, E).

*Output:* Vertex cover C.

*Pseudo code*:

* Initialize an empty set C.
* For each edge (u, v) in E:
* Add u or v to C with equal probability (0.5 for each).
* Return C as the vertex cover.

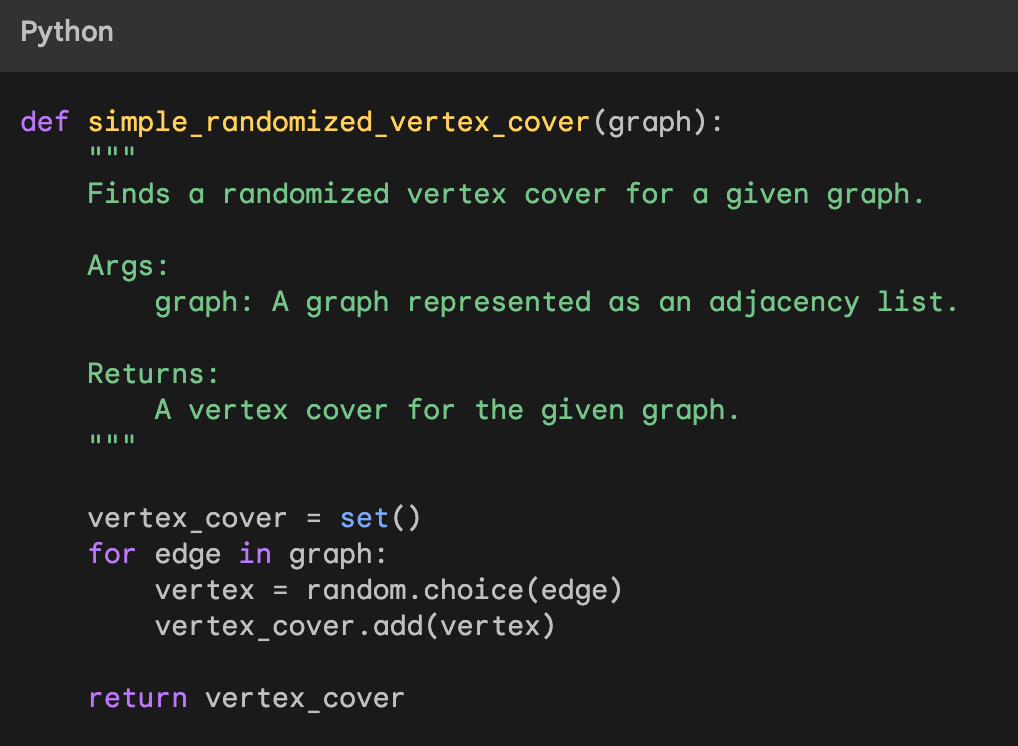
  
 Figure 2: Graph showing implementation of Simple Randomized Algorithm

In the above figure, Vertex cover is given as V’={A, C, E}by using the above algorithm.

This algorithm is simple and easy to implement. The algorithm runs in linear time, making it efficient for large graphs. The solution is not guaranteed to be optimal. The quality of the solution may vary across different runs due to its random nature.

Randomized algorithms for the vertex cover problem introduce a balance between solution quality and computational efficiency. Although these algorithms do not guarantee optimal solutions, they present a practical alternative, particularly suitable for addressing the complexities posed by large graphs.

*Python Code for Simple Randomized Algorithm:*

**Figure3: Python Code showing implementation of Simple Randomized Algorithm

The above figure shows code implementation of this algorithm with time complexity as O(E) where E is the number of Edges in the graph.

**3.2 Weighted Randomized Algorithm**

The Weighted Randomized Algorithm for the Vertex Cover problem involves assigning random weights to vertices and using these weights to probabilistically select vertices to form a vertex cover. The algorithm proceeds as follows:

*Weight Assignment***:**Give each vertex a random weight, here random.

*Probability of Selection***:**   
Make sure the probabilities for each vertex add up to one by normalizing the weights.

*Vertex Selection and Cover Evaluation***:**Choose vertices based on the likelihood that they will cover edges iteratively. Now add the chosen vertices to the vertex cover. And then Continue covering all of the edges.

The quality of weight assignment and the amount of randomness added to the vertex selection process determine how effective the method is. The predicted size of the vertex cover and its approximation ratio in relation to the ideal solution can be used to evaluate the performance. This method shows how randomization can be used to approximate solutions to the Vertex Cover problem using weighted probability.

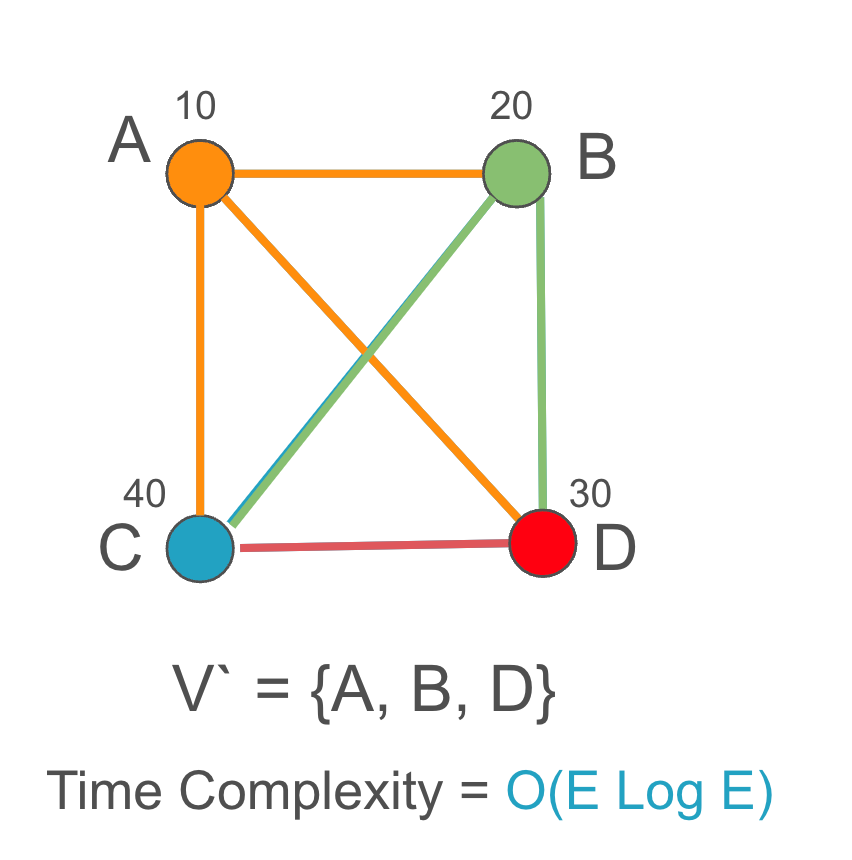
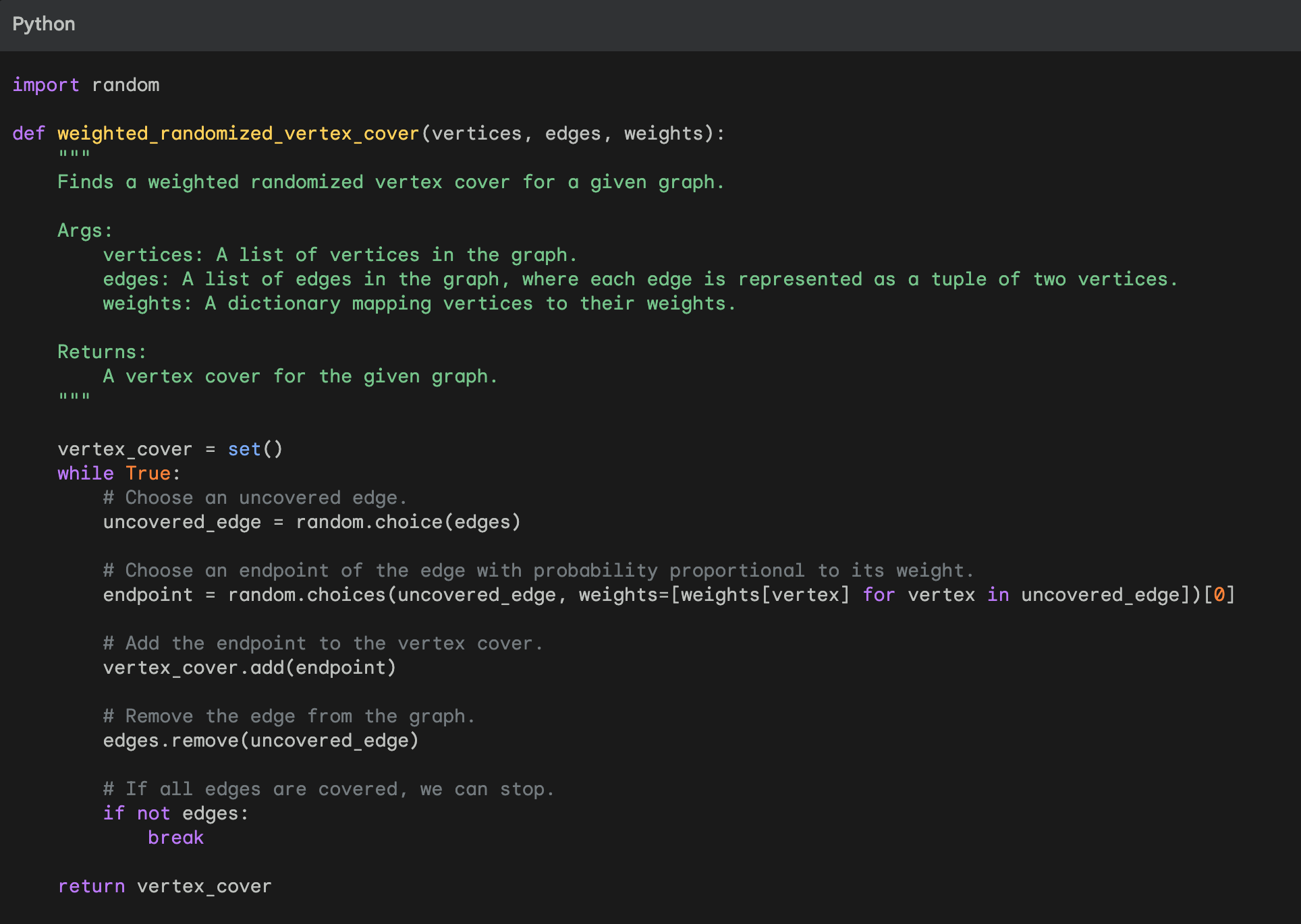


Figure 4: Graph depicting the Vertex cover V’ using Weighted Randomized Algorithm

  
Figure 5: Pseudo code in python for weighted randomized algorithm

**3.3 Local Search Randomized Algorithm :**

Local Search Randomized Algorithms are optimization algorithms that iteratively refine solutions by exploring the neighborhood of the current solution through random moves. These algorithms are particularly useful for solving combinatorial optimization problems, such as the Vertex Cover problem. Here's a concise summary:

*Objective:*

Iteratively improve a given solution to a combinatorial optimization problem (e.g., Vertex Cover) by exploring the local neighborhood through random moves.

*Initialization***:**

Start with an initial solution, which can be obtained through a heuristic or random assignment.

*Local Search Iterations***:**

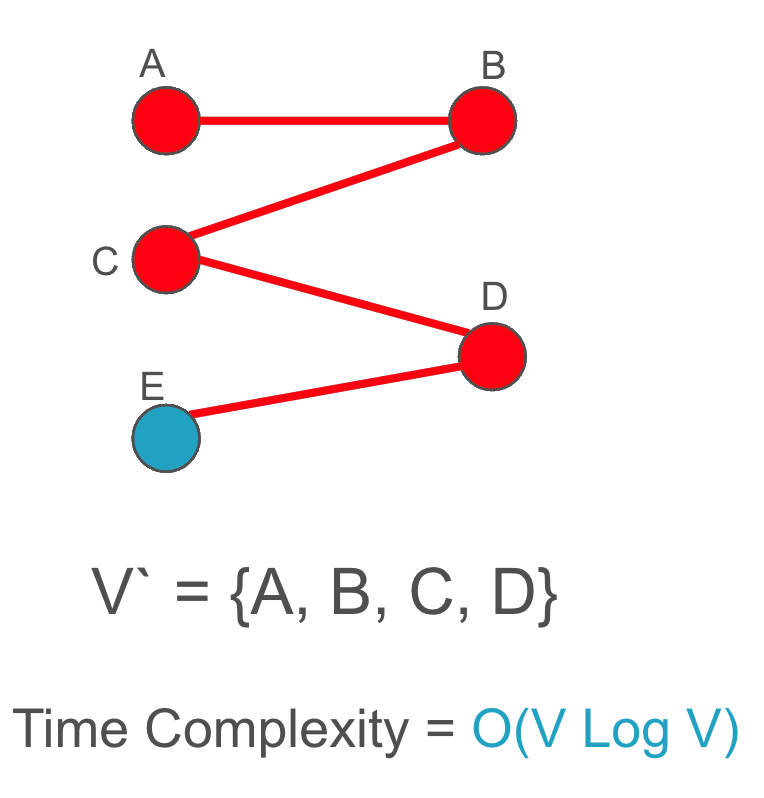
Repeatedly modify the current solution by making small, random changes. Evaluate the quality of the modified solution.

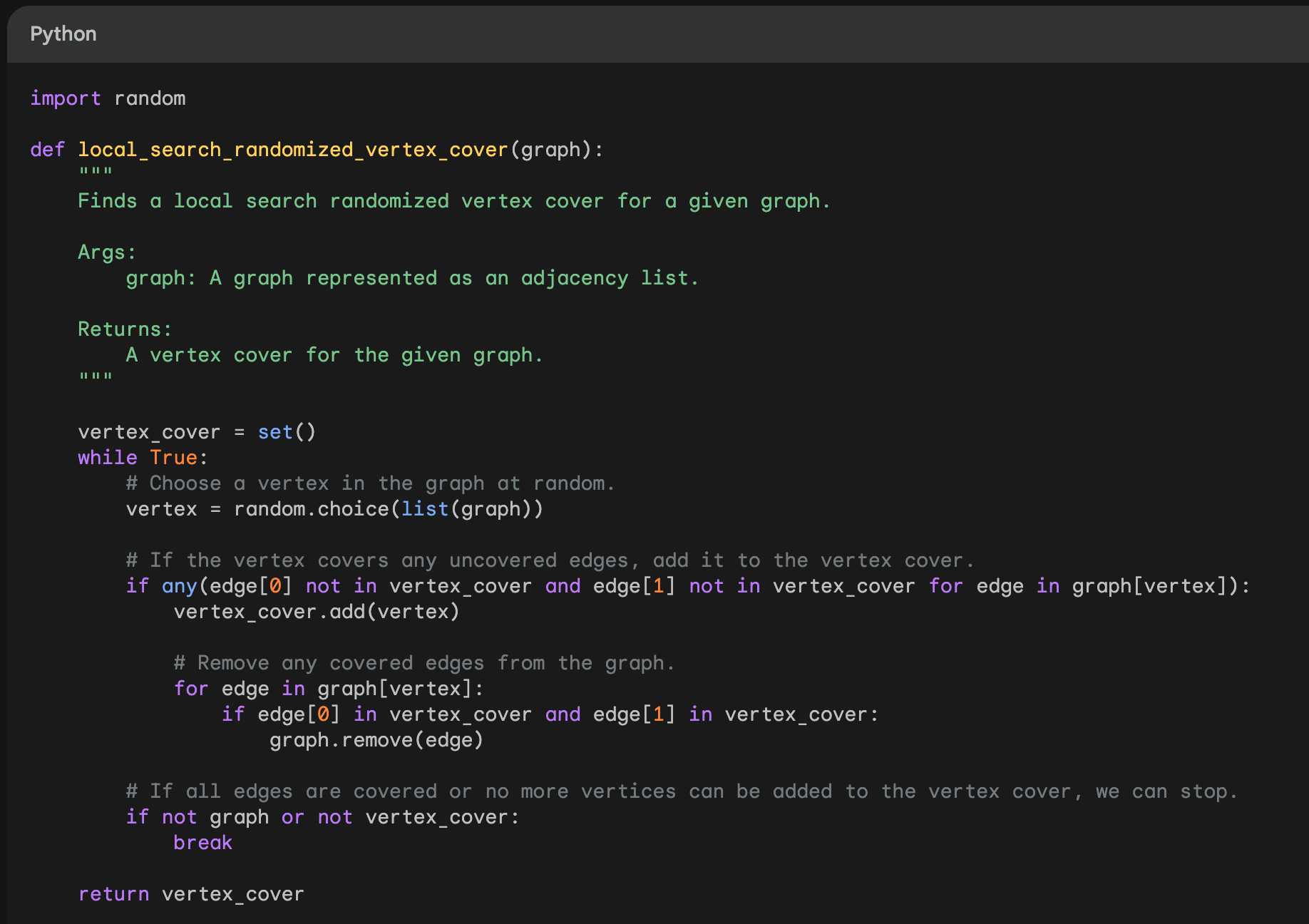
*Termination***:**

Stop the algorithm after a specified number of iterations or when a termination criterion is met.

*Evaluation***:**

Explore the solution space by making random local modifications.

  
Figure6: Graph showing implementation of Local Search Randomized Algorithm

  
Figure 7: Python Code for implementation of Local Search Randomized Algorithm

### **4. Approximation Algorithm for vertex cover**

In computational theory, problems are classified based on the factors but mainly on resources and time the algorithm takes to find the solution. Based on this, there are P, NP-Hard, and NP-complete categories of problems. P class states the existence of a solution in polynomial time, while NP-Hard and NP-complete do not guarantee the solution in polynomial time. Hence, the approximation algorithms are used to find the near-optimal solution for these problems. Approximation algorithms are the optimization technique to find the solution that is close to (but not necessarily exactly equal) to the optimal solution. These algorithms are for the NP-hard and NP-complete category of the problems, meaning there is no known polynomial time algorithm to find the exact solution in a general scenario. Hence, instead of finding the exact solution, approximation provides the solutions within a factor of optimal solution.

#### *Common Approaches:-*

* Greedy Approach
* Linear Programming Relaxations
* Randomized Algorithms

#### **4.1 Approximation Ratio**

These algorithms are measured against the approximation ratio, representing how close the solution provided by the algorithm is to the optimal solution. Also known as the approximation factor, it is a measure of how well the solution of approximation algorithm solves an optimization problem in comparison to the optimal solution. It quantifies the relation between the solution by approximation algorithm and the possible optimal solution.

Mathematically, let OPT represent the value of the optimal solution (maximum / minimum), and ALG represents the approximation algorithm solution value. The approximation ratio can be defined as:

P = ALG / OPT, representing how close the solution provided by the algorithm is to the optimal solution.

For example,

If P =1, the algorithm exactly produces the optimal solution.

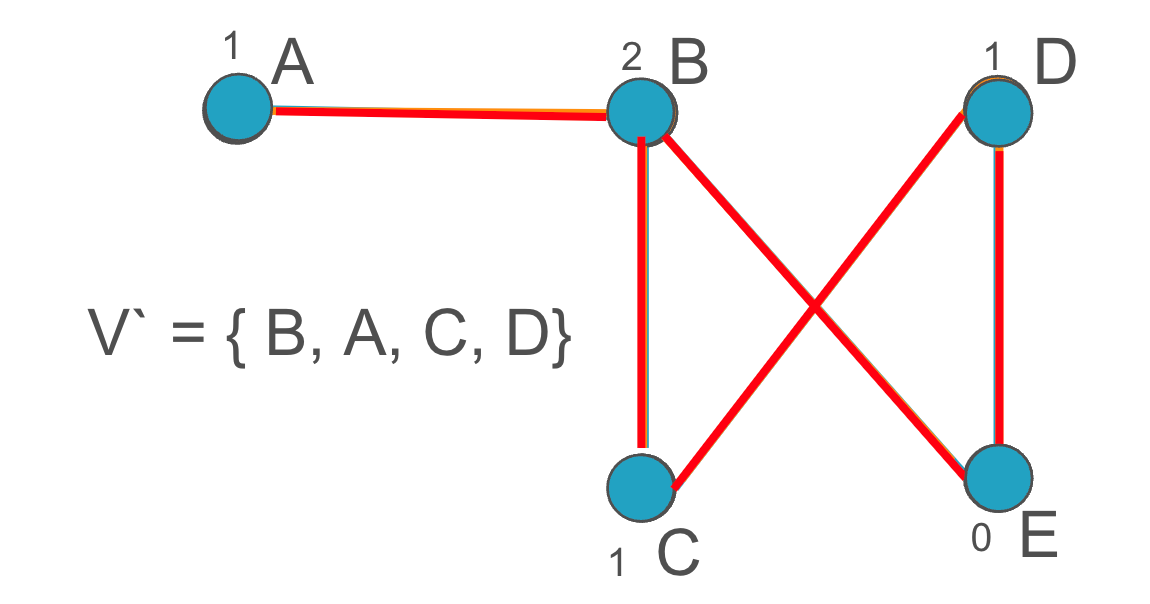
If P =2, the algorithm produces a solution that is at most twice the optimal solution

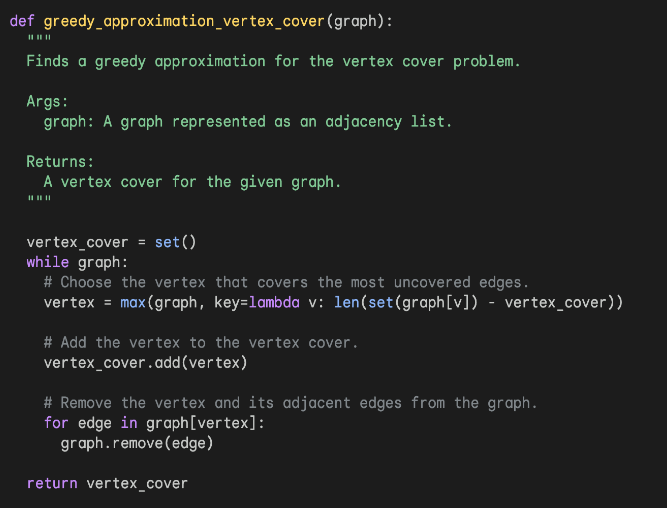
**4.2 Vertex Greedy Approximation Algorithm**

Vertex greedy approximation algorithm is a simple and commonly used approach. The greedy principle generally picks locally optimal choices at each step for overall optimal solution.

*Algorithm:*

1. *Initialization:* Initialize with an empty set C (the vertex cover).
2. *Greedy Choice:* At each step, choose a vertex that covers the maximum number of uncovered edges. I.e., Pick a vertex with the highest degree in the remaining subgraph.
3. *Update:* Add the current vertex to C and remove the current vertex and its incident edges from the graph.
4. *Repeat:* Repeat steps 2 and 3 until all edges are covered.

  
Figure 8: Graph showing implementation of Vertex Greedy Approximation Algorithm

****Figure 9: Python code implementation for Vertex Greedy Algorithm

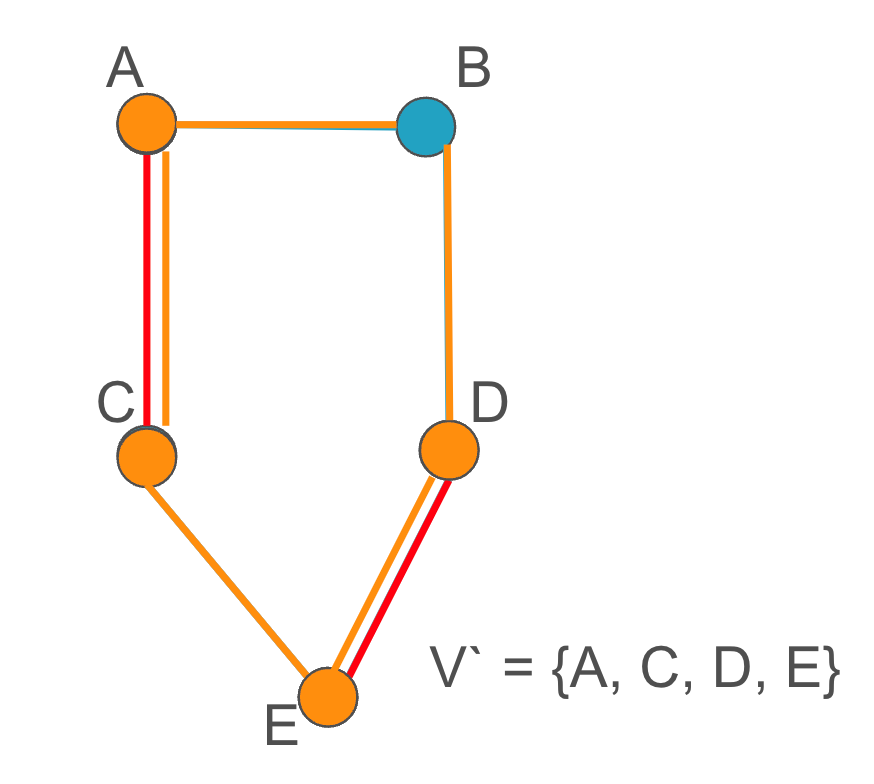
The analysis of the Vertex Cover greedy algorithm shows that its performance is bounded by a factor of 2. In other words, the size of the vertex cover produced by the algorithm is guaranteed to be at most twice the size of the optimal vertex cover. This 2-approximation ratio makes it a useful and efficient algorithm for practical purposes.

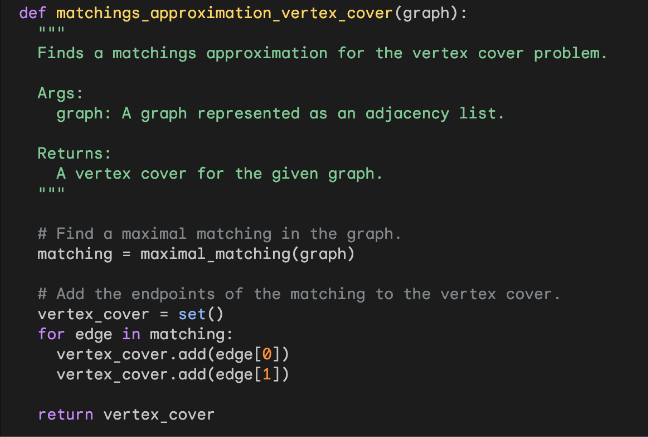
#### **4.3 Minimum Vertex Cover by Matchings**

One common approximation algorithm for the minimum vertex cover problem uses matchings. A matching is a set of edges in which no two edges share a common vertex.

*Algorithm*:

1. *Initialization:*Start with an empty set C (the vertex cover).
2. *Construct a Matching:*Find a maximum cardinality matching in the graph. This can be done using algorithms like the augmenting path algorithm.
3. *Update Vertex Cover:* Add all the vertices in the matching to C.
4. *Remove Covered Edges:*Remove all edges covered by the vertices added to C.
5. *Repeat:*Repeat steps 2-4 until all edges are covered.

  
Figure 10: Image showing minimum vertex cover by matching approximation

  
Figure 11: Python code implementation for minimum vertex cover by matchings approximation

The analysis of this algorithm shows that the size of the vertex cover produced by the algorithm is at most twice the size of the optimal vertex cover. This gives it a 2-approximation ratio for the minimum vertex cover problem.

### **5. Comparison between Randomized and Approximation Algorithms**

| **Aspects** | **Randomized algorithms** | **Approximation algorithms** |
| --- | --- | --- |
| **Guarantee** | Do not guarantee to find the optimal vertex cover | Guarantee to find a vertex cover that is at most some factor larger than the optimal vertex cover |
| **Efficiency** | Typically very efficient | May be slower than randomized algorithms |
| **Use of randomness** | Use randomness to generate and improve vertex covers | Do not use randomness |
| **Use cases** | Network routing, Resource scheduling, Load balancing  (Google Maps, Netflix CDN) | Circuit Design, Network Design, Task Scheduling |

Figure 12: A table showing comparable differences between both the discussed approaches  
  
Randomized Algorithms are generally used when speed is more important than accuracy and when vertex cover is part of a larger optimization problem. Approximation Algorithms on other hand are generally used when accuracy is more important than speed and when problem is being solved in isolation.  
  
*Examples:*

*Routing*: Randomized algorithms can be used to find efficient routes for vehicles or packets in a network. For example, the Google Maps app uses a randomized algorithm to find the fastest route between two locations.

*Scheduling*:   
Randomized algorithms can be used to find efficient schedules for tasks or resources. For example, the operating system on your computer uses a randomized algorithm to schedule tasks such as running programs and processing input/output.

*Load balancing*:   
Randomized algorithms can be used to distribute load evenly across multiple servers or processors. For example, the content delivery network (CDN) that delivers Netflix videos uses randomized algorithms to distribute load across its servers around the world.

*Circuit design*:   
Approximation algorithms can be used to design efficient circuits for electronic devices. For example, the chips in your computer and smartphone use approximation algorithms to design efficient circuits for processing data and graphics.

*Network design*:   
Approximation algorithms can be used to design efficient networks for communication or transportation. For example, the internet service provider (ISP) that provides you with internet access uses approximation algorithms to design its network of routers and switches.

*Scheduling*: Approximation algorithms can be used to find good schedules for tasks or resources, even if the schedules are not optimal. For example, the airline industry uses approximation algorithms to schedule flights and crews.

Ultimately, the decision of whether to use a randomized algorithm or an approximation algorithm for vertex cover depends on the specific needs of the application.

### **6. Current Research and Conclusion**

Current research on randomized and approximation algorithms for vertex cover is focused on developing algorithms with better approximation ratios and improved running times. Researchers are also developing new algorithms for specific variants of the vertex cover problem, such as the weighted vertex cover problem and the capacitated vertex cover problem.

One promising area of research is the development of parallel algorithms for vertex cover. Parallel algorithms can be used to solve large and complex vertex cover problems more efficiently than sequential algorithms.

Another promising area of research is the development of approximation algorithms for vertex cover with side constraints. Side constraints are additional requirements that must be satisfied by the solution to the vertex cover problem. For example, a side constraint might specify that the vertex cover must contain a certain number of vertices from a particular set of vertices.

Randomized and approximation algorithms for vertex cover have been used to solve a wide variety of real-world problems, such as routing in networks, scheduling tasks and resources, load balancing in distributed systems, circuit design, network design, and fraud detection.

Current research on randomized and approximation algorithms for vertex cover is focused on developing algorithms with better approximation ratios and improved running times, as well as developing new algorithms for specific variants of the vertex cover problem and parallel algorithms.

### 

### **7. References**

### <https://cgm.cs.mcgill.ca/~athens/cs507/Projects/2001/CW/npproof.html>

* <https://www.x-mol.net/paper/article/1524862929378959360>
* <https://hochbaum.ieor.berkeley.edu/html/pub/VC1983.pdf>
* <https://people.eecs.berkeley.edu/~nika/pubs/stochasticVC.pdf>
* <https://www.youtube.com/watch?v=eHZifpgyH_4&t=1960s&ab_channel=MITOpenCourseWare>  
  16. Complexity: P, NP, NP-completeness, Reductions  
  17. Complexity: Approximation Algorithms