CS-E400204: Approximation Algorithms **Spring (Fifth Period) 2022**

Lecture 1 (April 20): Introduction and Vertex Cover

Lecturer: Kamyar Khodamoradi *Scribe: Kamyar Khodamoradi*

1.1 Introduction

Many optimization problems are NP-hard. Therefore, one cannot find optimal solutions to such problems efficiently unless NP = P. In this course we will be studying the design approximation algorithms for such problems.

Recall that P is the class of problems solvable in polynomial time. On the other hand, NP is, roughly speaking, the class of *decision* problems whose solutions can be verified by a polynomial time algorithm. Finally, NP-hard informally refers to the class of problems that are "at least as hard as the hardest problem in NP". For these problems, we would like to:

(1) find the optimal solution

(2) find it efficiently (by that, we mean in polynomial time)

(3) find it for all instances

Unfortunately, all three are not possible simultaneously under the widely believed assumption that NP $\neq$ P. This means that we have to make a compromise:

- If we relax (3), usually it means we are studying special instances of the problem, or dealing with a special type of randomized algorithms.
- If we relax (2), we are in the field of exact algorithms (which can take exponential time). For example, this is common in the field of Integer Programming via using Branch and Bound methods.
- If we relax (1), we are in the realm of approximation algorithms and heuristics.

In this course, we relax (1) to some extent. That is,

- We seek (near) optimal solutions to hard optimization problems
- We still prove a guarantee on the *approximation factor* of our solutions, i.e., the ratio between the value of the objective function of our solution to the value of the objective function of the best possible solution.
- We find such solutions using efficient algorithms.

Example 1 (Vertex Cover Problem)

- *Input:*
  - *a graph* $G = (V, E)$

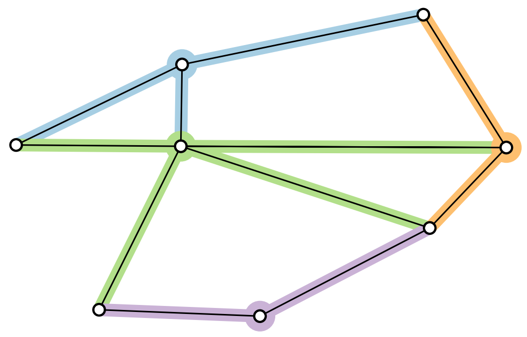

Figure 1.1: Example graph and a vertex cover of size 4

  - *a cost function* $c : V \to \mathbb{Q}_{\geq 0}$

- *Output:*
  - *A minimum* vertex cover*: a subset of vertices* $V' \subseteq V$ *of the minimum total cost*[1] *such that every edge is* covered *by* V'*. We say that an edge is covered by* V' *if at least one end-point of it is in* V'*.*

A special case of the problem is the CARDINALITY VERTEX COVER *for which* $c(v) = 1$ *for all vertices* $v \in V$*.*

1.2 NP **Optimization Problems**

Let Π indicate a minimization (maximization) problem. Π is characterized by:

- A set D_Π of *instances*. For every $I \in D_\Pi$, let $|I|$ be the size of I, which is the number of bits in binary needed to represent I.
- For each instance $I \in D_\Pi$, a set $S_\Pi(I) \neq \varnothing$ of *feasible solutions* to I such that:
  - for each solution $s \in S_\Pi(I)$, it size $|s|$ is polynomial in $|I|$, and
  - for each pair (I, s), there is a polynomial algorithm to decide whether $s \in S_\Pi(I)$.
- a polynomial-time computable *objective function* $obj_\Pi(I, s)$ that assigns a non-negative rational value to each feasible solution $s \in S_\Pi(I)$ for I.
- Π is either a minimization or maximization problem.

Back to our Example 1, for $\Pi =$ CARDINALITY VERTEX COVER we have:

- $D_\Pi =$ the set of all graphs
- $I \in D_\Pi$ would be a graph $G(V, E)$
  - What is the size of an instance $|I|$?

[1]The cost function c is *additive* or *modular*, meaning that $c(V') = \sum_{v \in V'} c(v)$ for all $V' \subseteq V$.

- $S_\Pi(I)$ = the set of all vertex covers of the instance $I = G(V, E)$
  - Why is $|s|$ polynomial in $|I|$?
  - How to detect, in polynomial time, that $s \in S_\Pi(I)$?
- $obj_\Pi(I, s) = |s|$
- Π is a minimization problem.

When dealing with optimization problems, we either want to minimize or maximize $obj_\Pi(I)$. In any case, we usually call the best solution for an instance I the *optimum* solution and denote by $\mathrm{OPT}_\Pi(I)$ (or simply OPT when Π and I are clear from the context).

1.3 Approximation Algorithms

An *α-factor approximation algorithm* (or simply, and α-approximation) is a polynomial time algorithm whose solution is always within a factor α of OPT.

Definition 1 *For a minimization problem Π, we say that algorithm A has approximation factor α if it runs in polynomial time and for any instance $I \in D_\Pi$, it produces a solution $s \in S_\Pi(I)$ such that $\dfrac{obj_\Pi(I,s)}{\mathrm{OPT}(I)} \le \alpha(|I|)$. α can be a constant or a function of the size of the instance.*

Remark 1 The convention is that for minimization problems, $\alpha(I) \ge 1$ and for maximization, $\alpha(I) \le 1$. Therefore, for a maximization problem, the only difference is that we require $\dfrac{obj_\Pi(I,s)}{\mathrm{OPT}(I)} \ge \alpha(|I|)$ instead.

Just as there are randomized algorithms that compute exact solutions, there are randomized algorithms that compute approximate solutions.

Definition 2 *An algorithm A is a randomized factor α-approximation for problem Π if for any instance $I \in D_\Pi$, A produces a feasible solution $s \in S_\Pi(I)$ such that $Pr\left[obj_\Pi(I,s) \le \alpha(|I|) \cdot \mathrm{OPT}\right] \ge 1/2$.*

This probability can be increased to $(1 - 1/2^x)$ by repeating the same algorithm A for x times.

1.4 A 2-approximation for Cardinality Vertex Cover

First attempt:

Algorithm 1 An edge-greedy approximation for CARDINALITY VERTEX COVER

```
Input: Graph $G = (V, E)$
Output: A set $S \subseteq V$
  $S \leftarrow \varnothing$
  while $E \neq \varnothing$ do
    let $e = (u, v)$ be an edge in $E$
    $S \leftarrow S \cup \{u\}$
    remove $u$ and all its incident edges from $G$
  end while
  return S
```

What is the approximation factor of this algorithm?

Second attempt:

Algorithm 2 A degree-greedy approximation for CARDINALITY VERTEX COVER

```
Input: Graph $G = (V, E)$
Output: A set $S \subseteq V$
  $S \leftarrow \varnothing$
  while $E \neq \varnothing$ do
    $v \leftarrow$ vertex with the maximum degree in $G$
    $S \leftarrow S \cup \{v\}$
    remove $v$ and all its incident edges from $G$
  end while
  return S
```

Can this algorithm achieve a constant factor approximation?

Final attempt:

One main challenge in analyzing the approximation ratio, or $\frac{obj_\Pi(I,s)}{\text{OPT}}$ is that OPT is unknown. The main idea is to find a "close-enough" lower bound L on OPT. Since $\frac{obj_\Pi(I,s)}{\text{OPT}} \leq \frac{obj_\Pi(I,s)}{L}$, this lower bound can give us the approximation factor α.

1.4.1 Lower Bounding Vertex Cover via Maximal Matching

Our main observation in this section is that for CARDINALITY VERTEX COVER, the size of any (maximal) matching provides a lower bound on OPT.

Definition 3 *An edge set $M \subseteq E$ of a graph $G = (V, E)$ is a* matching *if no two edges of M are incident. A matching M is called a* maximal matching *if there is no matching M' such that $M' \supsetneq M$.*

Algorithm 3 A 2-approximation for CARDINALITY VERTEX COVER via maximal matching

```
Input: Graph $G = (V, E)$
Output: A set $S \subseteq V$
  $S \leftarrow \varnothing$
  while $E \neq \varnothing$ do
    let $e = (u, v)$ be an edge in $E$
    $S \leftarrow S \cup \{u, v\}$
    remove $u$, $v$, and all their incident edges from $G$
  end while
  return S
```

First, notice that the algorithm returns a feasible solution.

Lemma 1 *The set S returned by Algorithm 3 is a vertex cover.*

Proof. Every edge $e \in E$ is deleted only after at least one of its endpoints is added to S. So, every edge is covered by S in the end. ■

Then, we formalize our observation about maximal matching and its relation to the approximation factor via the following two lemmas.

Lemma 2 *Let M be any matching and S be any vertex cover. Then, $|M| \leq |S|$.*

Proof. Every edge $e \in M$ must have at least one of its endpoints in S in order to be covered. Since M is a matching, no two edges in M share an endpoint. Therefore, an edge $e \in M$ is covered by a vertex of S that covers no other edge $e' \in M$. So, $|S| \geq |M|$. ■

Lemma 3 *Algorithm 3 is a 2-approximation.*

Proof. Let M be the set of all edges that the algorithm considers in the first line of the **while** loop. These edges form a matching since all other edges incident to their endpoints are deleted in the third line of the **while** loop. As the set S is only made up of the endpoints of M, then $|S| = 2\,|M|$. Let OPT be the size of the smallest vertex cover. By Lemma 2, $|M| \leq \text{OPT}$. Putting everything together, we get $|S| = 2\,|M| \leq 2\text{OPT}$. ■

Some final remarks about the VERTEX COVER problem:

- Its best known approximation factor is $2 - \Theta\left(1/\sqrt{\log |V|}\right)$ [Karakostas, 04].
- Assuming NP $\neq$ P, it cannot be approximated better than $10\sqrt{5} - 21 > 1.3606$ in polynomial time [Dinur and Safra, 02].
- Under an additional complexity assumption known as the *Unique Game Conjecture*, it cannot be efficiently approximated within factor $2 - \varepsilon$ for any constant $\varepsilon > 0$ [Khot and Regev, 03].