

Course Logistics

- CLRS Chapter 29 and 35
- Last homework due Friday

1 Approximation Algorithms for Optimization Problems

There are so many hard problems out there without known polynomial time solutions!
Is all hope lost? What do we do?

Approximation algorithms “quickly” find an answer that is “close” to the solution.

We are now focused on optimization problems, rather than just their *decision* versions.

Definition Let Q be a computational minimization problem (e.g., find a minimum vertex cover, find a minimum s - t cut) and assume that C^* is the optimal (minimum) solution to the problem. An _____ for Q with approximation factor p is an algorithm that

- Runs in
- Outputs a solution value C that is guaranteed to satisfy

Question 1. *What is a lower bound we must assume holds for the value of p when defining an approximation algorithm?*

A No lower bound is needed, p can be any real number

B $p \geq -1$

C $p \geq 0$

D $p \geq 1$

E $p \geq C^*$

We can also have approximation algorithms for maximization problems, but in this case C^* represents the *maximum* value (i.e., the solution) for the problem and _____

Wait, is this even possible? We want to guarantee that $\frac{C}{C^*}$ is small in polynomial time, but finding C^* is NP-hard. How can we do that?

To design an approximation algorithm, we need two pieces:

1. _____ A procedure for finding a value \hat{C} that satisfies _____
 - It should take polynomial time
 - This method _____ solve the original problem, but is often related
2. _____ An algorithm for the original problem that returns
a suboptimal solution _____
 - It should take polynomial time
 - This method often:

We then must prove that _____ and this provides a p -approximation!

Caveat. Often, the lower bounding procedure is *explicit*: there is an actual algorithm that computes the lower bound, and then the upper bounding algorithm explicitly uses the lower bound. However, in some cases no explicit lower bound is computed, and instead one shows implicitly that the upper bounding algorithm has to provide a solution that is better than some lower bound, even though the lower bound isn't computed explicitly. This can be tricky, but it is often possible!

2 Matchings and Vertex Covers

Let $G = (V, E)$ be an undirected and unweighted graph.

A matching $\mathcal{M} \subseteq E$ is a set of edges such that no two edges share the same node.

A vertex cover $\mathcal{C} \subseteq V$ is a set of nodes such that each edge $(u, v) \in E$ has at least one node in $\mathcal{C} \subseteq V$.

Lemma 2.1. *Let \mathcal{M} be a matching of G and \mathcal{C} be a vertex cover. Then $|\mathcal{M}| \leq |\mathcal{C}|$.*

3 The approximation algorithm for vertex cover

A matching $\mathcal{M} \subseteq E$ is a *maximal* matching if for every edge $e \in E - \mathcal{M}$, $\mathcal{M} \cup \{e\}$ is no longer a matching.

The algorithm

VERTEXCOVERAPPROX($G = (V, E)$)

1. Compute a maximal matching \mathcal{M} of G :
 - Set $F = E$, $\mathcal{M} = \emptyset$
 - While $|F| > 0$
 - Add any edge $e \in F$ to \mathcal{M}
 - For each remaining $f \in F$, if $|e \cap f| > 0$, remove f from F
2. Let S be the set of nodes adjacent to an edge in \mathcal{M}
3. Return S

Theorem 3.1. *Let C^* be the minimum sized vertex cover of G . The algorithm VERTEXCOVERAPPROX runs in polynomial time in terms of the size of G and outputs a vertex cover $S \subseteq V$ satisfying $|S| \leq 2C^*$. Thus, this is a 2-approximation algorithm for vertex cover.*

4 Graph problems as mathematical optimization problem

We can write the maximum s - t flow problem as the following optimization problem:

The weighted vertex cover problem can be written as the following optimization problem:

Question 2. *Which of the above two problems is a linear program?*

- A** The first
- B** The second
- C** Both
- D** Neither

5 Linear Programming Relaxation for Weighted Vertex Cover

This is the integer program for the weighted vertex cover problem:

Question 3. *Let \hat{C} be the optimal solution value to the linear programming relaxation of the weighted vertex cover integer program, and let C^* be the optimal solution to the weighted vertex cover problem. Which of the following is always true?*

- A** $\hat{C} \leq C^*$
- B** $\hat{C} \geq C^*$
- C** $\hat{C} < C^*$
- D** $\hat{C} > C^*$
- E** $\hat{C} = C^*$

6 The Algorithm

WEIGHTEDVERTEXCOVERAPPROX($G = (V, E)$)

1. Solve the linear programming relaxation of the weighted vertex cover problem
2. For each node $v \in V$, if $x_v \geq 1/2$, add v to a node set S . In other words, define

$$S = \{v \in V : x_v \geq 1/2\}$$

3. Return S as a vertex cover

Theorem 6.1. *Let C^* be the weight of the minimum weighted vertex cover of G . The algorithm WEIGHTEDVERTEXCOVERAPPROX runs in polynomial time in terms of the size of G and outputs a vertex cover $S \subseteq V$ satisfying $\sum_{v \in S} w_v \leq 2C^*$.*