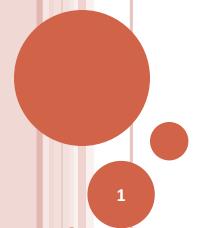
CS F364 Design & Analysis of Algorithms

COMPLEXITY – OPTIMIZATION PROBLEMS

Approximation Algorithms

- Introduction
- Example: Vertex Cover
- Lower Bounding
- Approach



APPROXIMATION ALGORITHMS

- Requirement:
 - Designing a tractable algorithm for a hard problem such that
 - solutions obtained may be sub-optimal within a known factor (need not be a constant)
 - Proving upper-bounds on the sub-optimality of solutions obtainable within polynomial running time
- Several algorithm design techniques are applicable
- Proofs are (often) fairly involved.

APPROXIMATION ALGORITHMS

- Lower Bounding Problem:
 - Designing an approximation algorithm includes :
 - oproviding guarantees on the measure of the solution output by the algorithm
 - This requires <u>measure of the solution</u> to be compared with the <u>optimal measure</u>
 - But for an NP-hard problem:
 - ofinding the optimal measure is as hard as finding the optimal solution.
 - Prove this! (see slide set on relative complexity of different forms of optimization problems).
- O How do we get out of the vicious loop?
 - Estimate a <u>bound on the optimal measure</u> as opposed to the optimal measure itself.

EXAMPLE: VERTEX COVER

- Problem Definition
 - I = { G | G=(V,E) is an undirected graph }
 - F(G) = { S | G= (V,E) and S ⊆ V, s.t.
 ∀ ((u1,u2)) ∈ E : (u1 in S) OR (u2 in S) }
 for G in I
 - m(G,S) = |S| for S in F(G)
 - goal = min
- Referred to as *Cardinality Vertex Cover*
 - as opposed to Min Vertex Cover where
 - o instances are vertex-weighted (i.e. G=(V,E,w), w:V→N) and
 - o m(G,S) = $\Sigma_{u \in S}$ w(u)

APPROXIMATION - EXAMPLE: CARDINALITY VERTEX COVER

Openitions:

- Given a graph H = (U,F) a subset M of edges is said to be a matching if no two edges in M share an endpoint.
- A matching of maximum size is said to be a maximum matching
- A matching that is maximal under inclusion is said to be a maximal matching

o Claim:

- <u>Size of a maximal matching</u> provides <u>a lower bound</u> for the <u>size of the vertex cover</u>.
- Proof:
 - oAny vertex cover must include at least one of the endpoints of each edge in a maximal matching.

ALGORITHM: MAXIMAL MATCH

- Algorithm Greedy_Maximal_Match(G):
 - Let G be (V,E);
 - 2. M = { }
 - 3. repeat
 - pick an edge (u1, u2) in E // Greedy Choice
 - remove vertices u1 and u2 from V, and edges incident on either of these vertices from E
 - 3. M = M U { (u1, u2) } until (G becomes empty)
 - 4. return M

```
• Algorithm Greedy_Maximal_Match(G):
```

ALGORITHM: MAXIMAL MATCH

- Let G be (V,E);
- 2. M = { }
- 3. repeat
 - pick an edge (u1, u2) in E // Greedy Choice
 - remove vertices u1 and u2 from V, and edges incident on either of these vertices from E
 - 3. $M = M U \{ (u1, u2) \}$

until (G becomes empty)

- 4. return M
- o Claim:
 - Greedy_Maximal_Match is a polynomial time algorithm:
- Proof:
 - The loop executes |E| times
 - o Cost per step:
 - cost of insertion of one element in a set i.e. O(|E|)

APPROXIMATION ALGORITHM: CARDINALITY VERTEX COVER

- Algorithm Greedy_Vertex_Cover(G):
 - M = Greedy_Maximal_Match(G)
 - 2. $S = \{ u \mid (u,v) \text{ in M OR } (v,u) \text{ in M } \}$
 - 3. return S

o Claim:

 Greedy_Vertex_Cover(G) returns a vertex cover for G that is at most 2 times the optimal size.

o Proof:

- m*(G) > |M|
 See Claim reg. Vertex Cover and Maximal Matching
- |S| = 2*|M| < 2*m*(G)

EXAMPLE: CARDINALITY VERTEX COVER

• Question:

• Can we improve the approximation guarantee of the above algorithm by better analysis?

o Answer:

- Consider bipartite graphs K_{n,n}
 Infinite family of instances
- This is referred to as a tight example.

EXAMPLE: CARDINALITY VERTEX COVER

• Question:

 Can we design a better algorithm using the same lower bound (of the size of a maximal matching)?

o Answer:

- Consider the family of complete graphs K_n, for odd n.
 - oSize of any maximal matching = (n-1)/2
 - oSize of a minimum vertex cover = n-1

• Question:

- Is there a better lower-bound for this problem?
- Answer?
 - This is still an open problem!

APPROXIMATION ALGORITHM — DESIGN STEPS

- 1. <u>Estimate a Lower-Bound (Upper-Bound) L(x)</u> for the minimum (maximum) measure m*(x) for any input instance x
- Essential steps

- Design an algorithm A that produces a feasible solution y in SOL(x)
- 3. Prove that the measure m(x,y) is upper-bounded (lower-bounded) by a multiple (or an additive) of L(x).
- 4. Verify whether the ratio of m(x,y) to L(x) or difference between m(x,y) and L(x) is obtained by tight analysis.
- 5. Verify whether A is the best algorithm given L.
- 6. Verify whether there is a better estimate than L.