CS F364
Design & Analysis of Algorithms

COMPLEXITY – OPTIMIZATION PROBLEMS

Approximation Algorithms

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

APPROXIMATION ALGORITHMS

- Requirement:
  1. 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*)
  2. 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 providing <u>guarantees on the measure</u> 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, finding 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*).
- 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 $S \subseteq V$, s.t.

    $\forall$ ((u1,u2)) $\in$ 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
    - instances are vertex-weighted (i.e. G=(V,E,w), w:V→N) and
    - $m(G,S) = \Sigma_{u \in S} w(u)$

Approximation - Example: Cardinality Vertex Cover

- Definitions:
  - 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*
- Claim:
  - *Size of a maximal matching* provides **a lower bound** for *the size of the vertex cover*.
  - Proof:
    - Any vertex cover must include at least one of the endpoints of each edge in a maximal matching.

Approximation Example: Cardinality Vertex Cover

- Algorithm Greedy_Maximal_Match(G):
  1. Let G be (V,E);
  2. M = { }
  3. repeat
     1. pick an edge (u1, u2) in E // Greedy Choice
     2. 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
- Claim:
  - Greedy_Maximal_Match is a polynomial time algorithm:
- Proof:
  - The loop executes |E| times
    - Cost per step:
      - cost of insertion of one element in a set i.e. O(|E|)

Example: Cardinality Vertex Cover

- Algorithm Greedy_Vertex_Cover(G):
  1. M = Greedy_Maximal_Match(G)
  2. S = { u | (u,v) in M OR (v,u) in M }
  3. return S

- Claim:
  - Greedy_Vertex_Cover(G) returns a vertex cover for G that is at most 2 times the optimal size.
- 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?
- 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)?
- Answer:
  - Consider the family of complete graphs K_n , for odd n.
    - Size of any maximal matching = (n-1)/2
    - Size of a minimum vertex cover = n-1

- Question:
  - Is there a better lower-bounding technique?
- Answer?
  - This is still an open problem!

Approximation Algorithm – Design Steps

1. *Estimate a Lower-Bound (Upper-Bound) L(x)* for the minimum (maximum) measure m*(x) for any input instance x
2. 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.

Essential steps