Approximation Algorithms

COMP 215 Lecture 21

Handling NP-Hard Problems

- We have already handled several NP-hard problems with the following approach:
  - Design an algorithm that is guaranteed to find the correct solution.
  - Hope it runs in a reasonable amount of time.
- This is not such a dumb idea.
- Recall that NP-hard problems are (probably) intractable *in the worst case*.
- It may be that the worst case is avoidable or rare.
  - E.g. we may need to solve the 0-1 knapsack problem, and know that the weight limit will be within some polynomial bound of the number of items.

Approximation Algorithms

- Another alternative is to give up on getting the exact answer, and settle for a good approximation.
- An example is the approximation for TSP using minimum spanning trees.
- This version of the TSP:
  - Assumes a completely connected graph.
  - Assumes that the triangle inequality holds,
    - $W(u, v) \leq W(u, y) + W(y, v)$

TSP Approximation Algorithm

- First find the minimum spanning tree of the graph.
- By starting at an arbitrary node and traversing each edge twice we have a path (not a tour) that visits every vertex.
- Convert the path to a tour by taking a “shortcut” every time our path would revisit a node.
- Let's look at an example...

Analysis of TSP Approximation

- Notice that the weight of the minimum spanning tree must be less than the weight of the optimal tour.
- The path that follows the MST is therefore less than twice the length of the optimal tour.
- Because of the triangle inequality, every time we take a shortcut in our MST path, we can guarantee that the total length will not increase.
- Therefore approximation distance < 2 x min-distance.

Can We Always Find an Approximation Algorithm in P?

- What if we need an approximation algorithm that gives us a bound of approx < c * mindist for the *general* version of the TSP?
- Such an algorithm is possible only if P=NP.
- Proof sketch:
  - Assume that we have a polynomial time algorithm for approximating TSP.
  - Such an algorithm could be used to solve the Hamiltonian circuit problem in polynomial time.
  - The Hamiltonian circuit problem is known to be NP-Complete.
  - Completing the proof requires a reduction from Hamiltonian circuit to TSP.

Proof can be found in: Complexity and approximation
By G. Ausiello, 1999

Reducing Hamiltonian Circuit to TSP

- Input to the Hamiltonian circuit problem is an unweighted graph $G = (V,E)$.
- Convert this to a completely connected weighted graph G' with the same set of vertices.
- Set the weights as follows:
  - W(E') = 1 if the corresponding edge exists in G.
  - W(E') = c * #vertices in G otherwise.
- What happens when we run the hypothesized TSP approximation algorithm on G' with a weight limit of c?
  - If G had a tour, then TSP approximation must answer yes.
  - Any approximate tour that uses an edge not in G is automatically too large.