Massachusetts Institute of Technology
6.854J/18.415J: Advanced Algorithms
David Karger

Handout 17
Wednesday, October 31, 2012

Problem Set 9

Due: Wednesday, November 7, 2012.

Collaboration policy: collaboration is *strongly encouraged.* However, remember that

1. You must write up your own solutions, independently.
2. You must record the name of every collaborator.
3. You must actually participate in solving all the problems. This is difficult in very large groups, so you should keep your collaboration groups limited to 3 people in a given week.
4. **No bibles. This includes solutions posted to problems in previous years.**

NONCOLLABORATIVE Problem 1. In telephone networks, a popular network architecture is the SONET ring: a set of n nodes (routers) arranged in a cycle, with each connected by wiring to its predecessor and successor on the cycle. A SONET ring is the minimal redundant network; it can tolerate any one edge failure without losing connectivity.

The SONET ring loading problem posits a set of m *calls* that need to be carried on the network: each call requires connecting a specific pair of nodes i and j with a single path in the ring (note there are exactly 2 possible paths for each pair). These paths impose *load*; the load on an edge is the number of paths traversing that edge. The goal is to choose routes that minimize the maximum demand on any edge.

Give a polynomial time 2-approximation algorithm for the SONET-ring loading problem.

Problem 2. $1 \mid prec \mid \sum w_j C_j$. In this scheduling problem there are *precedence constraints*: job j cannot be started until after all jobs in its *precedence set* $A(j)$ have been completed. Each job also gets a *weight* w_j. Our goal is to minimize the *weighted average completion time* $\sum w_j C_j$ (where C_j is the time job j completes). Assuming that the p_j are polynomially-bounded integers, we will give a constant-factor approximation for this problem via a linear programming relaxation. Define variables x_{jt} for each (integer) time step t, denoting the "indicator" that job j completed at time t.

(a) Write down constraints forcing the ILP to solve the problem. In particular, enforce that every job completes, that a job is not processed before its predecesors, and (most subtly) that the total processing time of jobs completed before time t is at most t.

(b) The LP relaxation of this ILP is a kind of "timesharing" schedule for jobs. Define the *fractional completion time* of job j to be $\overline{C}_j = \sum_t t x_{jt}$. To turn it into an actual order, consider the *halfway point* h_j of each job: this is the time at which half the job is completed. Prove that $\overline{C}_j \geq h_j/2$.

(c) Consider the schedule that processes jobs in order of their halfway points. Prove that no job runs before its predecessors.

(d) Prove that for the given order, the actual completion time for job j is at most $4\overline{C}_j$.

(e) Conclude that you have a constant-factor approximation for $1 \mid prec \mid \sum w_j C_j$.

OPTIONAL (f) Suppose that the processing times are not polynomially bounded integers. Show how rounding can reduce the problem to this case while adding a negligible additional error.

OPTIONAL (g) Suppose that each job comes with a *release date* r_j before which it cannot be processed. Generalize your algorithm to handle this case (with a slightly worse constant).

Problem 3. Directed TSP. The Metric Traveling Salesman Problem can be studied on *directed* graphs. Edges now have a direction as well as a weight. We restrict to the case where the edge lengths satisfy the triangle inequality (or equivalently, we allow you to visit a vertex more than once if you wish) but we do not require that an edge have the same length in both directions. Thus there are always two opposite directed edges between any two nodes. No constant-factor approximation is known for this problem. A logarithmic approximation is achieved using a *cycle cover* relaxation: to find a minimum cost *set* of vertex-disjoint cycles that contains all the vertices of the graph.

(a) Show that a minimum cycle cover can be found in polynomial time. **Hint:** Consider each vertex as two: an "entry node" and an "exit node". To what structure on the entry and exit nodes does a cycle cover correspond?

(b) Suppose that given a cycle cover, you choose one *representative node* from each cycle. Prove that this set of representative nodes is at most half the total nodes, and that the optimum tour traversing only these representative nodes costs less than the original optimum.

(c) It follows that if you repeat this finding and selecting representatives from minimum cycle covers $O(\log n)$ times, you will get a one-vertex graph. Prove that you can unravel all the selections you've done, patching together the various cycles you found to produce a tour of the whole graph whose cost is $O(\log n)$ time the optimum.

Problem 4. You have reached a river (modelled as a straight line) and must find a bridge to cross it. The bridge is at some integer coordinate upstream or downstream.

(a) Give a 9-competitive deterministic algorithm for optimizing the total distance travelled up and downstream before you find the bridge. This is optimal for deterministic strategies.

(b) Give a randomized 7-competitive algorithm for the problem

OPTIONAL (c) Give a randomized algorithm with competitive ratio strictly better than 7 (one can actually do better than 5-competitive).

Problem 5. Consider the *MTF-every-other* online linear-search algorithm which moves an accessed item to the front on every odd request for that item (i.e. the 1st, 3rd, 5th, etc. times the item is accessed). Prove that MTF-every-other is $O(1)$-competitive.

OPTIONAL Problem 6. Given a graph G, an *independent set* is a set of vertices such that no two are neighbors. The Maximum Independent Set (MIS) problem is a famous NP-complete problem. Interestingly, the complement of an independent set is a vertex cover, so the complement of the MIS is the minimum vertex cover. Thus, solving one exactly also solves the other exactly. We've seen (twice) how to get a two-approximation for vertex cover. Even though it is complementary, the situation for MIS is much worse.

Early in the study of approximation hardness, MIS was shown to be MAX-SNP-hard, meaning there is some constant to within which it *cannot* be approximated (unless $P = NP$). This is also true of vertex cover, but the situation for MIS is much worse as the following problem shows. In class we used a "linear scaling" argument to prove you couldn't find absolute approximations, but linear scaling does nothing to relative approximation. Here we will use a "quadratic scaling" argument to draw a similar conclusion about relative approximation.

Suppose one has an α-(relative) approximation algorithm for MIS. Consider the following "graph product" operation for a graph G. Create a distinct copy G_v of G for each vertex v of G. Then connect up the copies as follows: if (u, v) is an edge of G, then connect every vertex in G_u to every vertex in G_v.

(a) Prove that if there is an independent set of size k in G, then there is an independent set of size k^2 in the product graph.

(b) Prove that given an independent set of size s in the product graph, one can find an independent set of size $\sqrt{s}$ in G.

(c) Prove that if there is an α-approximation for MIS for *some* fixed α, then there is a polynomial approximation scheme for MIS.

(d) Conclude from the MAX-SNP-hardness of MIS that MIS has *no* constant-factor relative approximation (unless $P = NP$)

Note that this problem demonstrates that the approximation ratios achievable for complementary problems are quite different.

OPTIONAL Problem 7. We're going to look a little closer at some of the vertex cover approximations we studied. Recall the linear program we used for vertex cover:

$$\begin{array}{rc} \min & \sum y_v \\ y_u + y_v & \geq 1 \forall (u,v) \in E \\ y & \geq 0 \end{array}$$

Let y^* be an extreme poiont of this polyhedron.

(a) Show that $y^*_v \in \{0, 1/2, 1\}$ for any vertex v.

problempart Prove that if $y^*_v = 0$ then there is some optimal vertex cover that does not contain v, while if $y^*_v = 1$ then there is some optimal vertex cover that does contain v. Thus, you can reduce your vertex cover problem to one in which the LP relaxation above assigns weight 1/2 to every vertex. Conclude a 2-approximation for the minimum weighted vertex cover problem.

(b) Show that $y^*_v \in \{0, 1\}$ for any vertex v if G is bipartite. **Hint:** Use part (a). Hence, conclude a polynomial time algorithm for the minimum weighted vertex cover problem in bipartite graphs.

(c) Give a combinatorial 2-approximation algorithm for the minimum weighted vertex cover problem.

(d) In class we used a greedy "maximal matching" construction in a 2-approximation for vertex cover. To tighten this connection, take the dual of the vertex cover LP. How does it relate to matching?

(e) Using the previous part (and some proofs of integrality) show that in a *bipartite* graph, the size of the minimum vertex cover equals the size of the maximum matching.

Problem 8. How long did you spend on this problem set? Please answer this question using the Google form that is sent to you via a separate email. This problem is mandatory, and thus counts towards your final grade. It is due by the Monday 2:30pm after the pset due date. You can find the link to the form on the course website.