Khoury College of Computer Sciences
Northeastern University
CS 7870: Seminar in Theoretical Computer Science

Fall 2023
30 September 2023
Rajmohan Rajaraman

Problem Set 2 (due Friday, October 20)

- Since the background of the students in the class varies, some may find these problems easy while others may find some problems challenging. The main purpose of these problem sets is for you to learn the course material better by applying ideas learnt in class and/or exploring related problems.
- Please work on these problems on your own, or in collaboration with fellow students in class. Please try not to seek out solutions or solution approaches from the web. If you need hints, post on Piazza; discussions there may be helpful for all.
- Please typeset your solution. Latex, plain text, pdf, and Word are all acceptable formats. For figures, feel free to draw by hand or use any convenient tool.

Problem 1 (3 points) Multicommodity flow

Consider the following multicommodity flow problem. We are given a directed graph $G = (V, E)$ with edge capacities and a set of source-sink pairs $\{(s_i, t_i) : s_i, t_i \in V, 1 \leq i \leq k\}$. We would like to find the largest amount d such that a flow of d can be routed along a single directed path from s_i to t_i, for each i, subject to edge capacity constraints. Note that multiple commodities flowing through an edge have to share its capacity. This problem arises in many networking applications, and also forms a foundation for many network algorithms.

Intriguingly, the problem is NP-complete even when there are only two commodities and all capacities are one! For simplicity, we will focus on the case where all capacities are one; so, note that the optimal value for the flow value d for each commodity can be less than 1.

1. Solve the LP relaxation of the problem (which removes the constraint that every s_i-t_i flow should be along a single path).
2. Decompose the flow of each commodity i into a set P_i of at most m flow paths, where m is the number of edges. For each commodity i, select a path at random from P_i with probability equal to the flow of i along the path. Send all the flow for commodity i (as determined by the LP) along this path. Call the resulting flow f.
3. Return the flow obtained by scaling f down by the largest amount of flow going across any edge.

(a) Give a polynomial time algorithm for the flow decomposition computed in step 2 of the algorithm.

(b) Let n denote the maximum of k and m. Prove that after step 2 of the above algorithm, the flow along any edge has an expected value of at most one and is at most $O(\log n)$ with probability at least $1 - 1/n^3$.

Hint: Use Chernoff bounds. See the theorem on relative error in the following wikipedia entry.

http://en.wikipedia.org/wiki/Chernoff_bounds

(Fix an edge e. For each commodity i, the LP solution gives the probability that in the randomized algorithm e will be used by commodity i. Using these probabilities, you can bound the expected flow on e, and then use Chernoff to obtain a high probability upper bound on the flow on e. You can adjust the hidden constant in $O(\log n)$ to achieve the desired probability.)

(c) Conclude that the above algorithm yields an unsplittable multicommodity flow whose value is at most an $O(\log n)$ factor less than that of an optimal unsplittable multicommodity flow, with probability at least $1 - 1/n$.

Problem 2 (2 points) Vertex Multicover in Bipartite Graphs

Let us define the vertex multicover problem in bipartite graphs as follows. We are given a bipartite graph $G = (V_1 \cup V_2, E)$, a weight w_v for each vertex v, and a coverage requirement r_e for each edge e. We are asked to determine a multiset M of vertices such that for each (u, v) the total number of occurrences of u or v in M is at least r_e. (A multiset is a collection in which elements may repeat. We can represent the multiset by a list which includes, for each element that appears at least once, the number of occurrences of the element.) The objective is to minimize the total weight of M, which is given by the sum, over all v, of w_v times the number of occurrences of v in M.

Write an integer linear program for the problem. Present an optimal algorithm for the problem by iteratively rounding the associated linear programming relaxation. (*Hint:* Generalize the approach for vertex cover in bipartite graphs. Specifically, first consider the weighted vertex cover problem in bipartite graphs. Here r_e is 1 for every edge. In class, we studied an iterative rounding algorithm for maximum weighted matching problem in bipartite graphs. Show how to extend that approach to weighted vertex cover. Then, extend to the vertex multicover problem.)

Problem 3 (5 points) Minimum cost aggregation tree in directed graphs

Consider the following linear program for undirected MST.

$$\begin{array}{lll} \min & \sum_{e \in E} w_e x_e & \\ \displaystyle\sum_{e \in S} x_e & \geq 1 & \forall \text{ nonempty cuts } S \\ x_e & \in \{0, 1\} & e \in E \end{array}$$

(a) Show that the linear relaxation of the above program has an integrality gap of at least $2 - 2/n$.

In the *aggregation tree problem*, we are given a directed graph G with a root r and are asked to find a minimum cost set T of edges so that every vertex has a path to r in T; so, in effect, the minimum-cost "spanning tree" directed toward the root.

(b) Note that given G and r, it is possible that no aggregation tree exists. Give a simple algorithm to check if a rooted aggregation tree exists.

In the remainder of this problem, we assume that for the given instance, an aggregation tree with root r exists. We are going to analyze a primal-dual algorithm for finding an optimal aggregation tree.

(c) Modify the LP written above for MST to obtain an LP P for directed graphs, which has a variable x_e for each edge $e \in E$ and a constraint for each set S not containing r. Using a variable y_S for each S not containing r, compute dual D of P.

The following recursive algorithm sketch computes (x, y) with x and y being optimal solutions for P and D, respectively.

1. Find subgraph G' of G consisting of all the vertices in G and only of edges with zero weight. (Note that if there are no edges of zero weight, then G' will have no edges.) If G' is strongly connected or has an aggregation tree directed to r, then return any aggregation tree directed toward r. Otherwise, compute any strongly connected component C of G'.
2. Let w^* denote the weight of the least weight edge crossing out of C. Decrease the weight of every edge crossing out of C by w^*. Contract C to form a single node $\widehat{C}$. Replace every edge (u, v), $u \notin C$ and $v \in C$, by edge $(u, \widehat{C})$ (parallel edges are ok). Recursively solve the problem for the contracted graph to obtain solution (x', y').
3. Derive x by taking the union of the edges from x' with edges from a suitable aggregation tree in C. Derive y from y' accordingly.

(d) Provide details for the final step of the algorithm. Show that the algorithm ensures that x and y are primal and dual feasible, respectively.

(e) Show that $\sum_e w_e x_e = \sum_S y_S$ and conclude that the above algorithm returns an optimal aggregation tree.