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

Handout 11
Wednesday, October 10, 2012

Problem Set 6

Due: Wednesday, October 17, 2011.

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.**

Problem 1. Every April, the MIT admissions office hosts several thousand "prefrosh" admitted students. They need to assign each prefrosh to an MIT-student host. Students and prefrosh fill out a multiple-choice "profile" that the admissions office uses to automatically compute a "suitability" function $f(x, y)$. Sometimes $f(x, y) = -\infty$ due to allergies or other issues. We were asked to find an optimal (maximum total suitability) assignment of prefrosh to student hosts, subject to the following constraints:

- No prefrosh can be assigned to a student if their suitability is $-\infty$.
- Each student can be assigned to at most one prefrosh, and each prefrosh must be assigned to exactly one student.
- Students occupy suites of possibly multiple students (each student in exactly one), and each suite s has room for only some maximum number m_s of prefrosh.
- Suites are spread among floors of dormitories, and fire code limits the total number of prefrosh on floor g to some number M_g.
- Similarly, there is a total fire-code limit $\mathcal{M}_d$ on the number of prefrosh who should be assigned to each dormitory d.

(a) Assuming there is a feasible solution, show how the problem can be solved by an application of min-cost flow.

(b) If there is no feasible solution, it may be necessary to break the limits on suites (but the fire-code is inflexible). Give an efficient algorithm that finds the solution that minimizes the total "overage" (sum of amounts by which individual suite limits m_s are broken) and, among such solutions, maximizes the total suitability. Your algorithm should detect if there is no such solution.

NONCOLLABORATIVE Problem 2. Consider a unit-capacities network where each edge has cost 1. Does the shortest augmenting path algorithm (for max-flow, that just minimizes the number of edges on the path) or Dinic's normal blocking-flow algorithm compute a min-cost max-flow in the graph? Why or why not?

Problem 3. Cost-scaling algorithm for minimum-cost flow.

(a) Suppose that you have an optimal solution to some minimum-cost circulation problem with integer costs and you then change one edge cost by one unit. Show how you can re-optimize the solution in $\tilde{O}(mn)$ time. Note that this is faster than solving the minimum-cost circulation problem from scratch. **Hint:** which edges can now be involved in negative-cost cycles?

(b) Deduce a cost-scaling algorithm for minimum-cost flow (with integer costs) that makes $O(m \log C)$ (where C is the maximum cost) calls to your solution to part (a) and prove its correctness.

OPTIONAL (c) Design a cost-scaling algorithm for minimum-cost flow that makes $O(n \log C)$ calls to your solution to part (a).

Note: The resulting time bounds are worse than the capacity-scaling algorithm we saw in lecture, but a more careful algorithm along these lines attains a better time bound of $O(mn \log\log U \log_n C)$.

NONCOLLABORATIVE Problem 4. Linear Programming By Hand. Consider the following linear programming problem:

$$\begin{aligned} &\text{minimize } cx \text{ subject to} \\ x_1 + x_2 &\geq 1 \\ x_1 + 2x_2 &\leq 3 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \\ x_3 &\geq 0 \end{aligned}$$

For each of the following objectives c, give the optimum value and the set of optimum solutions:

(a) $c = (-1, 0, 0)$

(b) $c = (0, 1, 0)$

(c) $c = (0, 0, -1)$

Problem 5. You work for the Short-Term Capital Management company and start the day with D dollars. Your goal is to convert them to Yen through a series of currency trades involving assorted currencies, so as to maximize the amount of Yen you end up with. You are given a list of pending orders: client i is willing to convert up to u_i units of currency a_i into currency b_i at a rate of r_i (that is, he will give you r_i units of currency b_i for each unit of currency a_i). You may also borrow an arbitrary amount of any currency, with zero interest, provided you pay it back in the same currency by the end of the day. Assume that going around any directed cycle of trades, $\prod r_i < 1$—that is, there is no opportunity to make a profit by arbitrage.

(a) Formulate a linear program for maximizing the amount of Yen you have at the end of trading.

(b) Show that it is possible to carry out trades to achieve the objective of the linear program, without ever borrowing currency. (**Hint:** there is a sense in which your solution can be made acyclic.)

(c) Show that there is a sequence of trades that will let you end the day with the optimum amount of Yen and no other currency except dollars.

Problem 6. An internet switch is a transfer point where packets arriving on certain *input lines* are switched over to certain *output lines* where they proceed to the next switch on their journey. More precisely, each incoming packet can be assumed to be labeled with the desired output line to which it must be delivered. In each time step, the switch is typically constrained to transfer a set of packets that form a *matching:* in other words, at most one packet can be drawn from each input line, and at most one packet can be delivered to each output line. Let λ be the maximum of (i) the largest rate at which packets arrive on an input line and (ii) the maximum rate at which packets want to depart from an output line. We wish to prove that so long as the switch can deliver matchings at rate λ, it can deliver the specified traffic.

This is quite similar to the graduate student scheduling problem we studied before, but this week we will tackle it using linear programming and the language of fractional bipartite matchings. The following linear program reflects the minimum cost bipartite matching problem. Given a graph with n vertices per side and m edges, define a variable x_{ij} for each edge ij, let $N(v)$ denote the vertices neighboring vertex v (on the side opposite v), and consider the polytope defined by the constraints

$$\begin{aligned} \sum_{j \in N(i)} x_{ij} &= 1 \\ \sum_{i \in N(j)} x_{ij} &= 1 \\ x_{ij} &\geq 0. \end{aligned}$$

Any solution in which the x_{ij} are integers defines a perfect matching. But there can be other solutions, yielding so called *doubly stochastic matrices* in which all the row and column sums are 1, but need not be integers. We will show, however, that every such matrix is actually a *convex combination* of perfect matchings: in other words, a doubly stochastic matrix M can be written as a sum $\sum \lambda_i M_i$, where each M_i is an integer double stochastic matrix (representing a perfect matching) and $\sum \lambda_i = 1$.

(a) Argue that at any vertex M of the polytope described above, at least $m - 2n + 1$ of the x_{ij} must be equal to 0. **Hint:** the equality constraints in the polytope are not linearly independent.

(b) Conclude that at least one row and at least one column of the matrix M contains a single 1 with all other entries 0, if M represents a vertex.

(c) Use induction to conclude the claim (about convex combinations) and deduce the original claim (about internet switches).

OPTIONAL Problem 7. This problem investigates a more sophisticated approach to scaling min-cost flows. Instead of bit scaling, we scale a parameter ϵ allowing a small amount of negative cost on residual edges (forbidden in a fully optimal flow). We define a *ϵ-optimal* flow to be one where (for some given price function) every residual arc has cost at least $-\epsilon$ for some positive scaling parameter ϵ. Note if $\epsilon = 0$ we have an optimum min-cost flow.

The goal here is a cost scaling step that transforms an ϵ-optimal flow to an $\epsilon/2$-optimal flow. We start by saturating all the negative reduced-cost arcs, creating supplies and demands. We encompass the supplies into single virtual source and demands in a single virtual sink as usual. Then, we find a flow to route the supply at the source back to the demand at the sink while adjusting prices to maintain $\epsilon/2$-optimality. At any time, define the *admissible arc graph* to be the set of arcs that have negative cost. We will maintain that this graph is acyclic. Note that initially the admissible arc graph has no edges!

(a) The admissible arcs do not form a layered graph. Nevertheless explain how, assuming the admissible graph is acyclic, a blocking flow algorithm can be used to eliminate all admissible supply-demand paths, without creating any new admissible arcs.

(b) Show that if there is no admissible path from any excess to any deficit, we can "relabel" the vertices reachable from the source in the admissible graph, decreasing their prices by $\epsilon/2$, without violating $\epsilon/2$-optimality. Note that this shortens every excess-deficit path by $\epsilon/2$.

(c) The relabelling step may create new admissible arcs. Argue that even so the admissible graph remains acyclic.

(d) Show that after $3n$ relabeling steps, all supply will have been sent back to the demand. **Hint:** Immediately after saturating the negative arcs, what is the

shortest path distance (minimum cost path) from source to sink? So what is it after $3n$ relabelings? Is this possible, given that the graph is kept $\epsilon/2$-optimal at all times?

(e) Conclude that a single scaling phase can be completed in $O(mn \log C)$ time using a capacity-scaling blocking flow, or $O(mn \log n)$ time using Sleator-Tarjan dynamic trees. Deduce an $O(mn^2 \log C)$ bound for min-cost flow.

(f) In graphs with unit edge capacities, the unit-capacity blocking flow bounds can be extended to min-cost flows. Show that after $O(\sqrt{m})$ block/relabel steps, only $O(\sqrt{m})$ units of excess will still be in the graph (**Hint:** of the original flow paths used to saturate negative arcs, most will have had their reverses made admissible by the time $O(\sqrt{m})$ blocking steps occur, so supply can return to where it came from). Conclude that cost-scaling blocking flow completes in $(m^{3/2} \log C)$ time and the min-cost flow algorithm in $O(m^{5/2} \log C)$ time.

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.