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

Handout 13
Wednesday, October 15, 2003

Problem Set 7

Due: Monday, October 20, 2003.

NONCOLLABORATIVE Problem 1. 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). 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 2. Consider a graph in which edges have costs (possibly negative, representing profits). Suppose you want to find a *minimum mean cycle* in this graph: one with the minimum ratio of cost to length (number of edges). Going around such a cycle repeatedly (assuming it is negative) provides you with the maximum possible profit per unit length/time, so is the fastest way to earn money if you are, for example, a delivery service. Let's assume the costs c_{ij} are integers. Consider the following linear program:

$$\begin{aligned} w &= \min \sum c_{ij} f_{ij} \\ \sum_j f_{ij} - f_{ji} &= 0 \quad (\forall i) \\ \sum f_{ij} &= 1 \\ f_{ij} &\geq 0 \end{aligned}$$

(a) Explain why this captures the minimum mean-cost cycle problem. (**Hint:** f_{ij} is a circulation so can be decomposed into cycles.)

(b) Give the dual of this linear program—it will involve maximizing a certain variable λ

(c) Give an explanation (in terms of min-cost-flow reduced costs) for why this dual formulation also captures minimum mean-cost cycles. (**Hint:** how much is added to the cost of a k-edge cycle?)

(d) Suggest a combinatorial algorithm (not based on linear programming) that uses binary search to find the right λ to solve the dual problem. Can you use this to find a minimum mean-cost cycle? **Note:** to know when you can terminate the search, you will need to lower bound the difference between the smallest and next smallest mean cost of a cycle.

Problem 3. Markov chains. An $n \times n$ matrix P is **stochastic** if all entries are nonnegative and every row sums to 1, that is $\sum_j p_{ij} = 1$ (so each row can be thought of as taking a convex combination). Stochastic matrices are used to represent the **transition matrices** of **Markov chains**—random walks through a series of states. The term p_{ij} represents the probability, if you are in a current state i, that your next state will be j (thus the sum to one rule). If you have a probability distribution π over your current state, where π_i dentoes the probability you are in state i, then after a transition with probability defined by P, your new probability distribution is πP.

Use duality (or Farkas' Lemma) to prove that for any stochastic matrix P, there is a **nonzero** vector $\pi \geq 0$ such that $\pi P = \pi$. The vector π can be normalized to 1, in which case it represents a probability distribution that is **stationary** under the action of the transition matrix—that is, if π is the probability distribution on what state you are in before a transition, it is also the probability distribution after the transition. This proves that every Markov chain has a stationary probability distribution.

Hint: you must somehow express the constraint $\pi > 0$ (a strict inequality). Consider the constraint $\sum \pi_i = 1$.

OPTIONAL Problem 4. Game theory (this problem is neat, and not really hard, but you are working too hard so it is optional). In a *0-sum 2-player game*, Alice has a choice of n so-called *pure* strategies and Bob has a choice of m *pure* strategies. If Alice picks strategy i and Bob picks strategy j, then the *payoff* is a_{ij}, meaning a_{ij} dollars are transfered from Alice to Bob. So Bob makes money if a_{ij} is positive, but Alice makes money if a_{ij} is negative. Thus, Alice wants to pick a strategy that minimizes the payoff while Bob wants a strategy that maximizes the payoff. The matrix $A = (a_{ij})$ is called the *payoff matrix*.

It is well known that to play these games well, you need to use a *mixed* strategy—a random choice from among pure strategies. A mixed strategy is just a particular probability distribution over pure strategies: you flip coins and then play the selected pure strategy. If Alice has mixed strategy x, meaning he plays strategy i with probability x_i, and Bob has mixed strategy y, then it is easy to prove that the expected payoff in the resulting game is xAy. Alice wants to minimize this expected payoff while Bob wants to maximize it. Our goal is to understand what strategies each player should play.

We'll start by making the pessimal assumption for Alice that whichever strategy she picks, Bob will play best possible strategy against her. In other words, given Alice's strategy x, Bob will pick a strategy y that achieves $\max_y xAy$. Thus, Alice wants to find a distribution x that minimizes $\max_y xAy$. Similarly, Bob wants a y to maximize $\min_x xAy$.

So we are interested in solving the following 2 problems:

$$\min_{\sum x_i = 1} \max_{\sum y_j = 1} xAy$$

$$\max_{\sum y_j = 1} \min_{\sum x_i = 1} xAy$$

Unfortunately, these are nonlinear programs!

(a) Show how to convert each program above into a linear program, and thus find an optimal strategy for both players in polynomial time.

(b) Give a plausible explanation for the meaning of your linear program (why does it give the optimum?)

(c) Use strong duality (applied to the LP you built in the previous part) to argue that the above two quantities are *equal*.

The second statement shows that the strategies x and y, besides being optimal, are in *Nash Equilibrium*: even if each player knows the other's strategy, there is no point in changing strategies. This was proven by Von Neumann and was actually one of the ideas that led to the discovery of strong duality.