CS 670 (Spring 2025) — Assignment 3

Due: 03/12/2025

(1) [10 points]
Problem 7.22 from the textbook.

(2) [10 points]
Problem 7.33 from the textbook.

(3) [10 points]
Problem 7.40 from the textbook.

(4) [10 points]
Suppose that you have two companies (think Microsoft, Apple), each trying to sell their own copy of n different products (think Word Processor, Operating System, Spread Sheet, ...). For each product $i = 1, \ldots, n$, we know the following:

(a) the price $p_i \geq 0$ that company 1 charges and the price $p'_i \geq 0$ that company 2 charges.

(b) the quality $q_i \geq 0$ of company 1's version, and the quality $q'_i \geq 0$ of company 2's version.

For instance, Apple may make a better operating system, but Microsoft a better Word Processor. A user would like to assemble a system, consisting of exactly one copy each of the n products, e.g., the user wants to buy one Word Processor, one Operating System, etc. The goal is to maximize total quality minus total price.

In principle, the user can combine the products of the two companies. But there's a catch: the products of different companies may not be perfectly compatible. For each pair (i, j), we have a penalty term $\pi_{ij} \geq 0$, the loss in utility from combining company 1's version of product i with company 2's version of product j. Notice that it is possible that $\pi_{ij} \neq \pi_{ji}$: just because Apple's Word Processor doesn't run well on Microsoft Windows doesn't mean that MS-Word doesn't run well on Mac-OS. We assume that products are always compatible internally, i.e., there's no penalty for combining two products from the same company. All pairwise penalty terms are subtracted from the a priori utility of the system.

Give a polynomial-time algorithm for computing which components to purchase from which of the two companies to maximize the total system utility (including penalties) minus the total price.

(5) [0 points]
Chocolate Problem (2 chocolate bars): In the previous problem, you were solving the user's problem of finding the right system to purchase. Now, let's turn the tables and maximize the profit for company 1. Company 1 knows (and cannot change) all the qualities of the products (both its own and its competitor's), as well as the competitor's prices and all incompatibilities. The only thing that the company can change are the prices of its own products. Once company 1 sets its prices, we assume that the user chooses a subset to optimize exactly the objective function from the previous problem.

Company 1 then gets paid exactly the prices for the products it sells to the user. Thus, higher prices means more profits from selling that particular product, but also, it might mean that the user buys the competitor's version instead, meaning no profit at all. In fact, not only for that particular product, but there might be chain effects of other products being bought from the competitor as well. So the right choice of prices is important.

Give a polynomial-time algorithm for finding optimal prices to charge to maximize overall revenue. (Hint: You might want to prove first that the set of products that company 1 sells at the optimum prices is the same as the set of products it would sell if it priced all products at price 0.)

(6) [0 points]
Research Question: If you look carefully in the chocolate problem, you will notice that we assume that there is only one user (or, equivalently, that all users have the same quality values for all products). Here is an interesting questions, which — as far as I know — is open. Suppose that there are just two users. For both users, the incompatibility penalties π_{ij} are the same, and of course, they have to be charged the same prices. But the two users may have completely different q_i, q'_i values, because they value different features in their products. Your goal is to choose prices p_i to maximize the joint revenue from products sold to both users.

If you make any progress on this — either by finding a polynomial-time algorithm or proving some hardness result — I'd be interested to hear about it.