

SOLUTIONS: Homework 5, CS520, Spring '09

1. Give another proof that integer linear programming is NP-hard by describing a polynomial-time reduction of SUBSET-SUM to it. Let $\{i_1, i_2, \dots, i_n\}$ be the set of integers and let t be the target sum. Describe the integer linear program; you don't need to prove that it's correct.

Solution: Maximize $\sum_{j=1}^n x_j i_j$ subject to $0 \leq x_1, x_2, \dots, x_n \leq 1$ and $\sum_{j=1}^n x_j i_j \leq t$.

2. Simplex requires that you start out with a feasible solution. This raised the question of how we find an initial feasible solution.

We showed that the problem of finding a feasible solution can itself be reduced to linear programming. The trick is illustrated by inequalities 29.105-29.108 and 29.112-29.114 on pages 811 and 813.

Show that the converse is true, that linear programming reduces to the problem of finding a feasible solution. That is, show that if you have a program P that inputs the constraints of a linear program and gives a feasible solution if one exists, you can use it to find an optimum solution to an arbitrary linear program.

Instead of explaining the reduction, illustrate it on the following example:

$$\text{Maximize } 5x_1 + 4x_2 + 3x_3$$

Subject to:

$$2x_1 + 3x_2 + x_3 \leq 5$$

$$4x_1 + x_2 + 2x_3 \leq 11$$

$$3x_1 + 4x_2 + 2x_3 \leq 8$$

$$x_1, x_2, x_3 \geq 0.$$

Solution:

The dual is:

$$\text{Minimize } 5y_1 + 11y_2 + 8y_3$$

Subject to:

$$2y_1 + 4y_2 + 3y_3 \geq 5$$

$$3y_1 + y_2 + 4y_3 \geq 11$$

$$y_1 + 2y_2 + 2y_3 \geq 8$$

$$y_1, y_2, y_3 \geq 0$$

By Lemma 29.8, $5x_1 + 4x_2 + 3x_3 \leq 5y_1 + 11y_2 + 8y_3$ for any feasible solutions (x_1, x_2, x_3) and (y_1, y_2, y_3) for these problems. By Theorem 29.10, $5x_1 + 4x_2 + 3x_3 \geq 5y_1 + 11y_2 + 8y_3$ if and only if they are optimum solutions. We can add this as a constraint to force any feasible solution to be an optimum one:

Constraints:

$$5x_1 + 4x_2 + 3x_3 \geq 5y_1 + 11y_2 + 8y_3$$

$$2x_1 + 3x_2 + x_3 \leq 5$$

$$4x_1 + x_2 + 2x_3 \leq 11$$

$$3x_1 + 4x_2 + 2x_3 \leq 8$$

$$2y_1 + 4y_2 + 3y_3 \geq 5$$

$$\begin{aligned}
3y_1 + y_2 + 4y_3 &\geq 11 \\
y_1 + 2y_2 + 2y_3 &\geq 8 \\
x_1, x_2, x_3, y_1, y_2, y_3 &\geq 0
\end{aligned}$$

3. Problem 26-3, page 693. Let us call a solution *feasible* if, for each experiment brought along, the required instruments are also brought along. It's feasible even if some instruments are brought along that aren't used in any of the experiments that were brought along.

- (a) Think about part a. Describe a bijection it implies from the set of feasible solutions to the set of finite $s - t$ cuts in the graph the book proposes. Interpret all the edges the book proposes to be directed edges.

Solution: *The experiments and instruments left behind are on s 's side of the cut; the ones taken along are on the t side of the cut. Two different feasible solutions map to two different $s - t$ cuts. If a solution is feasible, there is no unbounded edge from s 's side to t 's side. If, in an $s - t$ cut, there is no unbounded edge from s 's side to t 's side, the corresponding solution is feasible. It's a bijection.*

- (b) Give a formula that describes the profit of a solution in terms of capacity of the corresponding cut. *Hint: Let $P = \sum_{i=1}^m p_i$ be Spock's "dream profit." (He majored in logic, and this is his first introduction to the disappointing world of economics.) Consider including P and his degree of disappointment as elements of your formula.*

Solution: *From P , he must deduct the costs of instruments taken along and the profits of experiments left behind. In other words, he must deduct the capacities of the edges crossing the cut. Minimizing the capacity of the cut maximizes the profit.*

4. In a communication network, two paths from s to t are *vertex-disjoint* if the only vertices they share are s and t , and *edge-disjoint* if they share no edges. Assume that the network is an undirected graph; data can travel either direction on an edge.

- (a) A bad person armed with a cable cutter wants to disconnect s from t . A cable cutter can take out an edge. Every time she cuts a cable, she has a chance $p > 0$ of getting caught; p is the same for all cables. Describe an algorithm for identifying a set of cables that maximizes the bad person's probability of getting away with it.

Your algorithm should be based on a reduction to max-flow. It suffices to describe the reduction.

Solution: *Give each edge a capacity of 1. The terrorist is looking for a minimum $s - t$ cut, which can be found with a max-flow algorithm.*

- (b) Now suppose you're a good person at s . Your goal is to maximize the bad person's chance of getting caught. Network administrators insist that you have to decide on a set of edge-disjoint paths to communicate with t on. Once established, you can't change your mind about which paths to use. What's worse, the bad person can find out which paths you chose. Now the bad person doesn't have to sever all paths between s and t , just the chosen ones.

The administrators' rule seems to work against you. Give an algorithm that chooses the paths in a way that prevents the bad person's knowledge of the

chosen paths from helping her. That is, prove that the bad person has to cut just as many cables as if she didn't know which paths you chose, and as many cables as if you were free to change your mind about the communication paths after the sabotage got underway.

Your algorithm should be based on a reduction to max-flow. It suffices to describe the reduction.

Solution: *The edges with flow on them in the above solution are a max flow. If the size of the min cut is k , the flow is k . Greedily follow edges, starting at s and stopping when you reach t , which gives you a first "path." Remove these edges. This reduces the flow to $k - 1$. Recurse on what remains to find $k - 1$ other paths. They are all edge-disjoint, since you remove an edge once you include it in a path. The terrorist must cut k edges whether her goal is to cut all paths or just your selected paths.*

- (c) Now consider the variant where the bad person is armed not with cable cutters, but with bombs. Each bomb can blow up a node. Because they are heavily guarded, she can't blow up s or t . Each time she plants one at some other node, she has a probability $p' > 0$ of getting caught. This probability is the same for all nodes other than s or t . Describe an algorithm for identifying a set of nodes that maximizes the bad person's chance of getting away with it.

It suffices to describe a reduction to max-flow.

Solution: *Each edge of an undirected graph consists of two oppositely-directed edges. Divide each node v into two nodes v_{in} and v_{out} . Reroute all edges directed into v to v_{in} , and all edges directed out of v to be directed out of v_{out} . Make all edges have infinite capacity. For each v , add an edge of capacity 1 from v_{in} to v_{out} . A min cut consists of edges of the last variety, and tells a minimum set of vertices that can be removed to cut all $s - t$ paths.*

- (d) Suppose you're a good person at s . This time, you have to decide in advance on a set of vertex-disjoint paths between s and t , and the bad person can find out about them. Give an algorithm for selecting them that keeps this knowledge from helping the bad person get away with it.

It suffices to describe a reduction to max-flow.

Solution: *The edges with flow in the above reduction should be selected as the paths, as in (b). Because the capacity of each edge of the form (v_{in}, v_{out}) is 1, they map to vertex-disjoint paths in the original graph. The number of these paths is the size of the flow, which equals the min vertex cut described above. If this has size k , the terrorist must take out k nodes to cut all paths. Since you have k disjoint paths, the terrorist must take out k nodes just to cut your paths.*

- (e) A secondary goal of the good person is to avoid ridicule. Unless you've taken steps in part b to prevent this, you might be in for some, despite having maximized the terrorist's chance of getting caught. Why is this, and what can you do to fix the problem?

Solution: *Some of your communication paths from s to t can have cycles if you used the strategy I described above for finding the edge-disjoint paths. Since your goal is to get a message from s to t , that would be ridiculous. Cut any cycles out of your paths.*

- (f) Apply a similar analysis to part d.

Solution: *There is no danger of a cycle, since a cycle would route two units of flow through some vertex. In this case, the capacities prevent the development of cycles.*