CSC373 A3 solutions

Instructor: Milad Eftekhar

Summer 2013

This document contains the key ideas to solve A3 questions (not the complete solutions). Note that many details may be ignored in several places.

Q1.

Let x_i denote the quantity of material i.

Maximize $1000x_1 + 1200x_2 + 12000x_3$

Subject to:

$$2x_1 + x_2 + 3x_3 \leq 100$$

$$x_1 + x_2 + x_3 \leq 60$$

$$x_1 \leq 40$$

$$x_2 \leq 30$$

$$x_3 \leq 20$$

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

Q2.

(a) **"It is always feasible"** since for any choice of a and b, values $x = 0, y = 0$ satisfy the constraints.

(b) **"$a \leq 0$ or $b \leq 0$".** If $a \leq 0$ then we can increase x (and therefore the objective function) arbitrarily while the constraint is still satisfied. Similar argument holds for b and y. Conversely, if a and b are both positive, the LP cannot be unbounded. Reason: Let $m = \min(a, b)$. Then, $m(x + y) \leq ax + by \leq 1$ (note that $m > 0$). Thus, $x + y \leq 1/m$.

(c) **"$a \neq b$ and $a > 0$ and $b > 0$".** (a) and (b) show that the LP has a finite optimal value when both a and b are positive. If $a > b$, the optimal is uniquely achieved at $x = 0, y = 1/b$. If $b > a$, the optimal is uniquely achieved at $x = 1/a, y = 0$. If $a = b$, any pair (x, y) such that $x + y = 1/a$ achieves the optimum. Therefore, the optimum is unique if both a and b are positive and not equal.

Q3.

Variables are $f_e^{(i)}$ for each edge e and $1 \leq i \leq k$. The LP is as follows:
Maximize $\sum_{i=1}^{k} \sum_{e=(s_i,u);u\in V} f_e^{(i)}$
Subject to:

$\sum_{e=(u,v);u\in V} f_e^{(i)} = \sum_{e=(v,u);u\in V} f_e^{(i)}$ for any $v \in V$ except s_i, t_i and $1 \leq i \leq k$ (*conservation constraints*)

$\sum_{e=(u,s_i);u\in V} f_e^{(i)} = \sum_{e=(t_i,u);u\in V} f_e^{(i)} = 0$ for $1 \leq i \leq k$

$\sum_{i=1}^{k} f_e^{(i)} \leq c_e$ for any edge $e \in E$ (*capacity constraints*)

$\sum_{e=(s_i,u);u\in V} f_e^{(i)} \geq d_i$ for $1 \leq i \leq k$ (*demand constraints*)

$f_e^{(i)} \geq 0$ for $1 \leq i \leq k$ and any edge $e \in E$

Q4.

(a) Let $z = \max_{1\leq i\leq n} |ax_i + by_i - c|$. Then the LP is:
Minimize z
subject to:

$z \geq ax_i + by_i - c$ for $1 \leq i \leq n$

$z \geq c - ax_i - by_i$ for $1 \leq i \leq n$

$z \geq 0$

(b) Let $z_i = |ax_i + by_i - c|$. Then the LP is:
Minimize $\sum_{i=1}^{n} z_i$
subject to:

$z_i \geq ax_i + by_i - c$ for $1 \leq i \leq n$

$z_i \geq c - ax_i - by_i$ for $1 \leq i \leq n$

$z_i \geq 0$ for $1 \leq i \leq n$

Q5.

(a) TRUE. According to the theorem proved in lecture, if $A \leq_p B$ and B is in P, then A is in P. Therefore, since $D_2 \leq_p D_3$ and D_3 is in P, then D_2 is in P. Since $D1 \leq_p D_2$ and D_2 is in P, then D_1 is in P.

(b) FALSE. There are many problems that are p-reducible to each other. For example 3SAT$\leq_p$3-coloring and 3-coloring$\leq_p$3SAT. Moreover, all problems in P are p-reducible to each other.

(c) TRUE. Since each literal appears at most once, each variable can satisfy at most 1 clause. Construct the following bipartite graph. For each variable, put a node on the left side. For each clause, put a node on the right side. Connect each variable to the clauses it appears in. Connect all variables to a source s and all clauses to a sink t. Set the capacity of all edges to 1. The problem is equivalent to deciding if there is a flow of m units (m: the number of clauses) from s to t. Reason: To satisfy F, each clause must pick one (at least) variable it is connected to. Moreover, each variable should be picked by at most one of the clauses it is connected to.

Q6.

(a) This problem is called Hitting-set.

i. Hitting-set is in NP.

Consider the following verifier:

Given an input X and a certificate c:

1. Examine if c has k items.
2. Examine if c is a subset of A.
3. For each subset S_i: examine if $S_i \cap c$ is non-empty.

The run time of this verifier is polynomial (calculate it exactly!).

If X is a yes-instance, then there exists a certificate c such that the verifier outputs Yes for this certificate.

If X is a no-instance, for any certificate c, the verifier outputs No.

ii. Vertex-Cover $\leq_p$ Hitting-Set.

Given a graph G and number k, create an instance (A, S_i, k') of Hitting-Set as follows:

For each vertex $v \in V$, create an element v in the set A.

For each edge $e = (u, v) \in E$, create a set S_e containing the elements u and v.

Set $k' = k$.

Clearly this transformation is polynomial.

Finding a vertex cover of size at most k in G is equivalent to finding a set $B \subseteq A$ with size at most k in the created instance of Hitting-Set.

(b) i. This variation of Scheduling problem is in NP.

Consider the following verifier:

Given an input X and a certificate c:

1. Examine if the number of scheduled jobs is at least k.
2. Examine if $a_i \leq t_i \leq d_i - p_i$ for each scheduled job i.
3. For any pair of scheduled jobs, examine if their intersection is empty.

The run time of this verifier is polynomial (calculate it!) ...

ii. Subset-Sum $\leq_p$ Scheduling.

Given a set of numbers A and a target number t, create an instance Y of scheduling problem as follows:

For each number $a_i \in A$, create a job $(1, 1 + \sum_{a \in A} a, a_i)$ where n is the size of set A.
Create a job $\mathcal{J} = (t, t+1, 1)$
Set $k = n + 1$.

Clearly this transformation can be done in polynomial time.
If X is a yes-instance in subset-sum, it means that the numbers in S can be divided into two subsets S_1 (with a sum of t) and S_2 (with a sum of $\sum_{a \in A} a - t$). Therefore, all jobs corresponding to numbers in S_1 can be scheduled from time 1 to t, all jobs corresponding to S_2 can be scheduled from $t+1$ to $1 + \sum_{a \in A} a$, and $\mathcal{J}$ can be scheduled from t to $t+1$. Hence all $n+1$ jobs can be scheduled without conflicting. So the answer to the scheduling problem for input Y is Yes.
If the answer to the scheduling problem for input Y is yes, then it means all jobs including $\mathcal{J}$ are scheduled; $\mathcal{J}$ is scheduled from t to $t+1$ (its only possibility). Put the numbers corresponding to all jobs that are scheduled before $\mathcal{J}$ in S_1 and the numbers corresponding to all jobs scheduled after $\mathcal{J}$ in S_2. The sum of numbers in subset S_1 is t. Hence the answer to subset-sum for input X is Yes.