CS5800 Algorithms	*Out: November 19, 2019*
Solutions for Problem Set 7	
Name: Instructor's Solutions	*Due: December 4, 2019*

Linear Programming

Problem 1 (Manipulating Linear Programs) 5+5+5

Consider the following LP:

$$\begin{aligned} \min \quad & x + y \\ \text{s. t.} \quad & 3x + y \leq 6 \\ & x + 3y \leq 4 \\ & 2x + y \geq 5 \\ & x \geq 0 \end{aligned}$$

1. Write the LP in Standard Form.

Solution:

We need to use the $y = (y^+ - y^-)$ trick for both the Standard and the Canonical forms since both those forms need all variables to be non-negative. For the standard form we also need to use slack variables s_1, s_2, s_3.

$$\begin{aligned} \max \quad & -(x + (y^+ - y^-)) \\ \text{s. t.} \quad & 3x + (y^+ - y^-) + s_1 = 6 \\ & x + 3(y^+ - y^-) + s_1 = 4 \\ & 2x + (y^+ - y^-) - s_3 = 5 \\ & x, y^+, y^- \geq 0 \end{aligned}$$

■

2. Write the LP in Canonical Form.

Solution:

Here we only need the $(y^+ - y^-)$ trick, but not the slack variable.

$$\begin{aligned} \max \quad & -x - (y^+ - y^-) \\ \text{s. t.} \quad & 3x + (y^+ - y^-) \leq 6 \\ & x + 3(y^+ - y^-) \leq 4 \\ & -2x - (y^+ - y^-) \leq -5 \\ & x, y^+, y^- \geq 0 \end{aligned}$$

■

3. Write the dual of the LP.

Solution:

We do five things starting from the canonical form.

(a) Swap min and max.
(b) Introduce as many non-negative variables as there are constraints.
(c) Swap b and c, the coefficients in the objective function and the right hand side of the constraints.
(d) Change the $\leq$ signs in the constraints to $\geq$ signs.
(e) Take the transpose of the coefficient matrix of the constraints.

$$\begin{aligned} \text{min} \quad & 6a + 4b - 5c \\ \text{s. t.} \quad & 3a + b - 2c \geq -1 \\ & a + 3b - c \geq -1 \\ & -a - 3b + c \geq 1 \\ & a, b, c \geq 0 \end{aligned}$$

■

Note: If you need to use slack variables, call them $s_1, s_2, \ldots$ and if you need to enforce a non-negativity condition on an unconstrained variable a use the a^+ and a^- superscript notation.

Problem 2 (Reductions to Linear Programs) 25+20

For each of the following problems, write Linear Programs (LPs) whose optimal solution will answer the question in the problem:

1. A store sells two types of toys, A and B. The store owner pays 8 and 14 for each one unit of toy A and B respectively. One unit of toys A yields a profit of 2 while a unit of toys B yields a profit of 3. The store owner estimates that no more than 2000 toys will be sold every month and he does not plan to invest more than 20,000 in inventory of these toys. How many units of each type of toys should be stocked in order to maximize his monthly total profit?

   (a) Write the LP to solve this. 10

   Solution:

   The objective is to maximize profit, which is given by $2a + 3b$ where a is the number of toys of type A and b is the number of toys of type B. The total number of toys you can sell is $a + b$ which is less than 2000. The total amount spent on the toys will be $8a + 14b$ which has to be less than 20000. These give all the constraints (along with the non-negativity constraints on a and b).

$$\begin{aligned} \text{max} \quad & 2a + 3b \\ \text{s. t.} \quad & a + b \leq 2000 \\ & 8a + 14b \leq 20000 \\ & a, b \geq 0 \end{aligned}$$

   ■

   (b) Write the dual of the LP. 5

Solution:

The above LP is already in Canonical form so the dual is straightforward.

$$\begin{aligned} \min \quad & 2000x + 20000y \\ \text{s. t.} \quad & x + 8y \geq 2 \\ & x + 14y \geq 3 \\ & x, y \geq 0 \end{aligned}$$

■

(c) Solve the LP geometrically. 10

Solution:

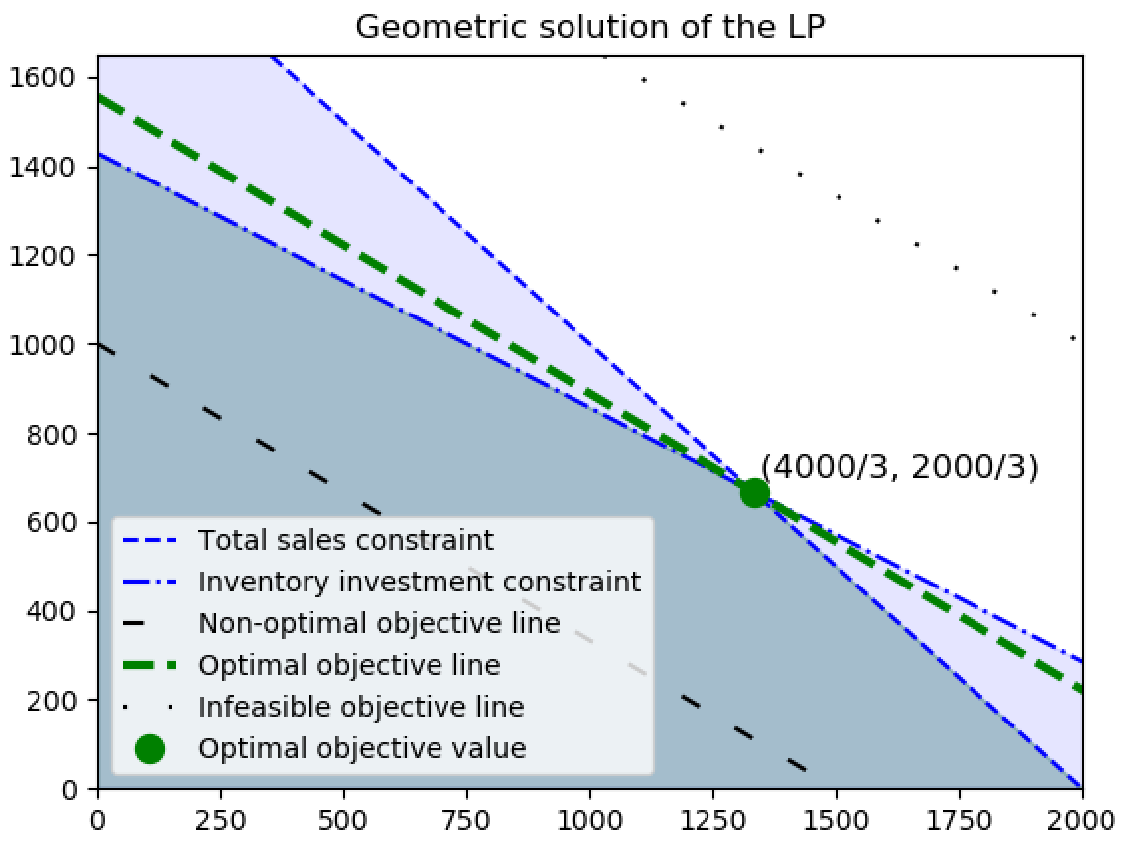

Figure 1: Geometric solution

We plot all the lines corresponding to the constraints, and find the feasible regions and their intersection (the shaded region in Figure 1). Then we move the objective function line until it just touches the feasible region (moving it further would move it out of the feasible region). This point will be the optimal. Note that this point is always going to be on one of the vertices of the feasible region.

When we solve geometrically we get that the optimal is $(4000/3, 2000/3) = (1333.3, 666.67)$ for a maximum profit of 4666.66. However we cannot use a fractional amount of toys, so we try rounding this to the nearest integers (by taking floor and ceiling) and choosing the best option that is still feasible. This gives us the value $(1334, 666)$ for a maximum profit of 4666. ■

2. Formulate the Stock Traders problem as an Integer Linear Program.

   You know the prices of stocks for the next n days. The price of the stock on day i is given by p_i, you can make at most k transactions (buy-sell pairs) such that at each point you hold at most 1 stock and at the end of the n days you do are not holding any stock in hand. You need to find the maximum profit that can be made given the prices $[p_1, p_2, \ldots, p_n]$.

   Write an Integer Linear Program (ILP) whose optimal solution will give the answer to this problem. An ILP is an LP with the additional constraint that the variables are forced to be integers. Explain briefly why the ILP will give the correct solution. You do not need to solve the LP or write a dual.

Solution:

There are two ways to reduce to ILP. The first is to have $0-1$ indicator variables for transactions t_{ij} starting on day i and ending on day j, that is, you buy on day i and sell on day j.

$$\begin{aligned}
\max \quad & \sum_{i,j\in[n]} (p_j - p_i)t_{ij} && \text{; Maximize profits over all transactions} \\
\text{s. t.} \quad & \sum_{i,j\in[n]} t_{ij} \le k && \text{; At most } t \text{ transactions} \\
& t_{ij} \le 0 \quad \forall i \ge j && \text{; Cannot sell before buying} \\
& t_{ij} + t_{ab} \le 1 \quad \forall i \le a < b < j && \text{; No new transactions while another is ongoing} \\
& t_{ij} + t_{ab} \le 1 \quad \forall i \le a < j < b && \text{; No new transactions before current finishes} \\
& t_{ij} \in \{0,1\}
\end{aligned}$$

The second is to have $0-1$ indicator variables b_i for when you buy on day i and indicator variables s_i for when you sell on day j.

$$\begin{aligned}
\max \quad & \sum_{i\in[n]} (s_i - b_i)p_i && \text{; Maximize profits} \\
\text{s. t.} \quad & \sum_{i\in[n]} s_i = \sum_{i\in[n]} b_i && \text{; Buy as many times as you sell} \\
& \sum_{i\in[n]} s_i \le k && \text{; At most } t \text{ transactions} \\
& 0 \le \sum_{i\in[j]} b_i - \sum_{i\in[j]} s_i \le 1 \quad \forall j \in [n] && \text{; On any day } j\text{, hold either 0 or 1 stock} \\
& s_i, b_i \in \{0,1\}, \forall i
\end{aligned}$$

■

NP-Completeness

Please use the following template for proving NP-Completeness of a decision problem A via polynomial-time reduction from a decision problem B:

- Show that A is in fact in NP.
- Clearly specify the reduction. Make sure to clearly specify the notation you use for the problem instance of A and the problem instance of B. Make sure that the transformation of the problem B instance into an instance of problem A is also clear.
- Prove that this reduction is correct: Show that the instance of problem A returns TRUE if and only if the corresponding instance of problem B returns TRUE.

Problem 3 (Reductions to simple variants) 10+10

We will look at some very simple reductions, where instead of reducing from one problem to another, we will need reductions between very close variants of the same problem. Note that for this problem, you do not need to show that the problems are in NP, the rest of the template should be followed.

a. The original Cook-Levin Theorem showed that every NP-Complete problem can be reduced to the SAT problem. However, we generally use the restricted 3-SAT problem in our reductions. You need to show that SAT can be polynomial-time reduced to 3-SAT.

Recall that in both SAT and 3-SAT, we have some variables and some clauses, with each clause being an OR of literals (variables or negations of variables). There is a satisfying assignment if you can set each variable to TRUE or FALSE such that *all* of the clauses are satisfied. The difference between SAT and 3-SAT is that in 3-SAT each clause will have exactly 3 literals, while in SAT each clause may have any number of literals.

Solution:

Showing that 3-SAT is in NP is straightforward. An assignment acts as a certificate that we can verify in polynomial time: Given a satisfying assignment we can verify in polynomial time that the assignment would satisfy the formula by just ensuring that at least one literal in each clause is satisfied based on the assignment.

Now we specify the reduction. We start with a SAT formula $\phi = C_1 \vee \dots C_m$ over variables $x_1, \dots, x_n$ and create a 3-SAT formula ψ. We will be replacing every clause C_i in ϕ with a set of equivalent clauses C_i' such that C_i is satisfiable iff all clauses in C_i' are satisfiable. Construction of these equivalent set of clauses varies depending on how many the number of literals k in the clause C_i. The easiest case is when $k = 3$ and we can keep the clause C_i as is and the equivalent set of clauses will just be the original clause so when $k = 3$, C_i is true iff all clauses in C_i' (which is simply equal to C_i) are true.. We now describe the construction for other values of k.

If $k = 1$, that is, $C_i = (z_1)$ then we introduce two new variables y_i^1, y_i^2 and create four clauses: $(z_1 \vee y_i^1 \vee y_i^2), (z_1 \vee y_i^1 \vee \overline{y_i^2}), (z_1 \vee \overline{y_i^1} \vee y_i^2), (z_1 \vee \overline{y_i^1} \vee \overline{y_i^2})$. Note that by construction, the only way for all of these clause to be satisfied is for z_1 to be True. So C_i is true iff all clauses in C_i' are true.

If $k = 2$, that is, $C_i = (z_1 \vee z_2)$ then we introduce one new variables y_i^1 and create two clauses: $(z_1 \vee z_2 \vee y_i^1), (z_1 \vee z_2 \vee \overline{y_i^1})$. Note that by construction, the only way for both of these clause to be satisfied is for either of z_1 or z_2 to be True, which would satisfy the clause C_i. So once again C_i is true iff all clauses in C'_i are true.

If $k > 3$, that is, $C_i = (z_1 \vee z_2 \vee \cdots \vee z_k)$ then we introduce $k-3$ new variables $y_i^1, y_i^2, \ldots, y_i^{k-3}$ and create $k-2$ clauses: $(z_1 \vee z_2 \vee y_i^1)$, a clause of the form $(\overline{y_i^j}, z_{j+2}, y_i^{j+1})$ for all $1 \leq j \leq k-4$, and $(\overline{y_i^{k-3}} \vee z_{k-1} \vee z_k)$.

Hence in our reduction we used all the original variables and added the extra y variables. Now we need to show that this reduction is correct. That is if there is a satisfying assignment of ϕ there is a assignment for ψ and vice versa. We have already shown that the reduction is correct for the cases when $k \leq 3$, that is, the original clause is satisfied iff its set of equivalent clauses are satisfied. Thus, we will focus on the case where $k > 3$.

First suppose there is an assignment of x_j such that ϕ and hence C_i is satisfied. We keep the assignments for the original variables as is and show how to assign truth values to the y variables. If either of z_1, z_2 are True, then we can set all the y variables to be False. The negated literals in the later clauses in C'_i will all become true making all of those clause true hence satisfying everything in C'_i. If either of z_{k-1}, z_k are True, then we can set all the y variables to be True. The corresponding literals in all the earlier clauses in C'_i will all become true making all of those clause true hence satisfying everything in C'_i. If the first literal in C_i that is true is z_l for some $3 \leq l \leq k-2$ then we set y_i^j to True for all $1 \leq j \leq l-2$ and y_i^j to False for all $l-1 \leq j \leq k-3$. The clause in C'_i that contains z_l is satisfied by z_l and the clauses before are satisfied by the literals y_i^j for $j \leq l-2$ while the clauses after the one with z_l are satisfied by the negations of literals y_i^j for $j \geq l-1$.

The other direction of the reduction is easily verified. If the 3-SAT formula ψ is satisfied then we can use the same assignment to the variables x_j and it will satisfy ϕ because at least one of the original literals from C_i must be true for all clauses in C'_i to be true.

Thus we have shown both directions and proved that this is a correct reduction. ■

b. Show a polynomial-time reduction from the Hamiltonian Cycle (HC) problem to the Traveling Salesman Problem (TSP). The following definitions below would be helpful. You can assume that all graphs here are directed.

Definition (Hamiltonian Cycle). *A* Hamiltonian cycle *in a graph $G = (V, E)$ is a cycle in G that visits every vertex in G exactly once. That is, an ordering of all vertices such that every vertex occurs once in the ordering; the first and last vertices are the same; and every adjacent pair of vertices has an edge between them.*

Problem (Hamiltonian Cycle). *Given a graph $G = (V, E)$ is there a Hamiltonian Cycle in G or not?*

Definition (Traveling Salesman Problem). *A* tour *in a graph $G = (V, E)$ is an ordering of all vertices such that; the first and last vertices are the same; and every adjacent pair of vertices has an edge between them. If G is a weighted graph, the cost of the tour is the sum of weights on each edge.*

Problem (Traveling Salesman Problem). *Given a graph $G = (V, E)$, with every pair of nodes (u, v) having an edge between them with an associated weight of $d(u, v) \in \mathbb{Z}^+$ is there a tour in G with total cost less than a given bound B?*

Solution:

Showing that TSP is in NP is straightforward: A tour acts as a certificate. To verify one would simply add the weights of the edges and check that it is less than the bound B, make sure that it is indeed a tour that visits all vertices, starts and ends at the same vertex.

To reduce from an HC instance of a graph $G = (V, E)$ to a TSP instance, we need a graph G', costs $c(u, v)$ for each pair of vertices in $u, v \in V$, and a bound B. The new graph $G' = (V, E \cup \overline{E})$ is made using the same vertices as in G. The cost from u to v is 1 if $(u, v) \in E$ and is 2 otherwise. The bound B is set to the number of vertices $|V| = n$.

If there is a Hamiltonian Circuit, then the ordering of vertices will be a tour visiting every vertex in G'. Since the Hamiltonian Circuit used edges in E, the costs incurred along each edge in that tour is 1. Since there are n edges in a cycle of n vertices, the total cost is $n \leq B$ as required.

If there is a TSP tour, then the total cost is at most B. But since a tour is a cycle visiting each vertex at least once, it has n edges and since each edge has a cost at least 1, if the total cost is at most $B = n$ then the tour must have only used edges in the source graph G and must have only visited n vertices. That gives us a Hamiltonian Path as required. ■

Problem 4 (Independent Set to Vertex Cover and Clique) 10+10

We have seen reductions from 3-SAT, and we know that any NP-Complete problem can be reduced to any other NP-Complete problem. However, often times it is much easier to reduce from certain problems. We will now look at two problems, Vertex Cover and Clique where reducing from Independent Set (IS) is much easier.
The following definitions and lemma will be helpful. You can assume that all graphs are undirected.

Definition (Independent Set). *An* independent set *in a graph $G = (V, E)$ is a subset $V' \subseteq V$ such that, for all $u, v \in V'$, the edge $\{u, v\} \notin E$.*

Problem (Independent Set). *Given a graph $G = (V, E)$ and a positive integer $k \leq |V|$, does G contains an independent set V' having $|V'| \geq k$?*

Definition (Vertex Cover). *A* vertex cover *in a graph $G = (V, E)$ is a subset $V' \subseteq V$ such that, for all edges $\{u, v\} \in E$, at least one of u and v belong to V'.*

Problem (Vertex Cover). *Given a graph $G = (V, E)$ and a positive integer $k \leq |V|$, does G contains a vertex cover V' having $|V'| \leq k$?*

Definition (Clique). *A* clique *in a graph $G = (V, E)$ is a subset $V' \subseteq V$ such that, for every pair of vertices $u, v \in V'$, the edge $\{u, v\} \in E$.*

Problem (Clique). *Given a graph $G = (V, E)$ and a positive integer $k \leq |V|$, does G contains a clique V' having $|V'| \geq k$?*

Lemma. *For any graph $G = (V, E)$ and subset $V' \subseteq V$, the following statements are equivalent:*

a. V' is a vertex cover for G.

b. $V \setminus V'$ is an independent set for G.

c. $V \setminus V'$ is a clique in the complement G^c of G, where $G^c = (V, E^c)$ with $E^c = \{\{u, v\} : u, v \in V$ and $\{u, v\} \notin E\}$.

a. Give a polynomial-time reduction from Independent Set to Vertex Cover to show that if Independent Set is NP-Complete then Vertex Cover will also be NP-Complete.

Solution:

Clearly VERTEX COVER is in NP. A subset of vertices acts as a certificate. Given a subset of vertices we can easily verify that its size is at most k and that for all edges $\{u, v\} \in E$, either $u \in V'$ or $v \in V'$.

We are given a graph $G = (V, E)$ and a number k and are asked if there is an INDEPENDENT SET of size at least k. By the lemma we know that $V' \subseteq V$ is an INDEPENDENT SET in G if and only if $V \setminus V'$ is a VERTEX COVER in G. Further, note that if V is of size k then $V \setminus V'$ is of size $n - k$.

Thus, by the lemma the reduction is to simple use the same graph and ask for a vertex cover of size at most $n - k$. That is, $G' = G$ and $k' = n - k$. ■

b. Give a polynomial-time reduction from INDEPENDENT SET to CLIQUE to show that if INDEPENDENT SET is NP-Complete then CLIQUE will also be NP-Complete.

Solution:

Clearly CLIQUE is in NP. A subset of vertices acts as a certificate. Given a subset of vertices we can easily verify that its size is at least k and that for all $u, v \in V'$ the edge $u, v \in E$.

We are given a graph $G = (V, E)$ and a number k and are asked if there is an INDEPENDENT SET of size at least k. By the lemma we know that $V' \subseteq V$ is an INDEPENDENT SET in G if and only if V' is a CLIQUE in the graph $G^c = (V, E^c)$, where E^c is the complement of the edge set as defined in the lemma.

Thus, by the lemma the reduction is to simple use the complement graph and ask for a Clique of size at least k. That is, $G' = G^c$ and $k' = k$. ■