1. Find the dual of the following linear program:

Maximize \( z = x_1 - 4x_2 + 3x_4 + 2x_5 \)

Subject to:

\[ \begin{align*}
3x_1 - x_2 + x_3 + x_6 & \leq 2 \\
-x_2 + x_3 + x_4 - x_5 + 3x_6 & = 5 \\
8x_1 - x_2 - 5x_3 + x_6 & \geq 0 \\
x_2 & \geq 0 \\
x_3 & \geq 0 \\
x_5 & \geq 0
\end{align*} \]

Solution:

Minimize \( w = 2y_1 + 5y_2 \)

subject to

\[ \begin{align*}
3y_1 - 8y_2 - y_1 & = 1 \\
-y_1 + y_2 + y_3 & \geq -4 \\
y_1 + y_2 + 5y_3 & \geq 0 \\
y_2 & = 3 \\
-y_2 & \geq 2 \\
y_1 + 3y_2 - y_3 & = 0
\end{align*} \]

\( y_1 \geq 0 \quad y_2 \text{ free} \quad y_3 \geq 0 \)

There are many variants that are also correct. For example you can convert the primal to a minimization problem and then take the dual. Also the free variable can be changed in sign. If you convert the maximization form, you need the inequality constraints as less than or equal. That means you have to do something about the last inequality in the primal. Note that the two constraints involving only \( y_2 \) are inconsistent; this implies that the primal has no optimal solution.

2. Consider the network flow problem shown below. The problem is to send as much flow as possible through the network, from node \( S \) to node \( T \). The numbers on the arcs represent the capacity of the arcs. Flow must be conserved at each node. So your model should have upper bounds on arc flows to represent the arc capacities (all flows should be non-negative), and equations expressing that total flow that comes into a node should equal the total flow that leaves it.
(a) Model the problem as a generalized dictionary. Indicate the type for each of the variables you use (that is: fixed, free, lower bounded, upper bounded, bounded). Are all the rows of your dictionary necessary? Explain.

(b) Start with all your variables at zero, and perform one iteration of the simplex method for generalized dictionaries. What are the new values of your variables? Do you need to keep all the elements of the new dictionary?

Solution:
(a)

Minimize \(-f = -x_1 - x_2\)

\[
\begin{align*}
y_A &= x_1 + x_3 - x_4 - x_5 \\
y_B &= x_2 + x_3 - x_4 + x_5 - x_6 \\
\end{align*}
\]

\(x_j, j=1, \ldots, 6\) are bounded variables with \(0 \leq x_j \leq c_j\); \(f\) is a free variable; \(y_A\) and \(y_B\) are fixed at 0. The balance equation for node T is redundant and is not needed.

(b) With all the variables initialized to zero, the dictionary is feasible, so no Phase I is needed. You want to increase \(x_1 \) or \(x_2\), but you can't because the basic variables are both fixed. The first iteration is a degenerate one that replaces a fixed variable say \(y_A\), with one of the bounded variables, say \(x_1\). The variables do not change values. The only change is to bring \(x_1\) into the basis. After this is done you can increase it. Similarly, at some point, \(y_B\) will have to be brought out of the basis too. After the fixed variables are out of the basis, they can be eliminated because they are fixed at 0 and cannot reenter the basis. They play no further role in the optimization.

3. The following linear program in canonical form is almost optimal, except that two of the basic variables, \(x_1\) and \(x_2\), are not feasible. How would you improve the situation by making a pivot? That is, how would you keep the cost coefficients positive, and reduce the infeasibility? What is the pivot column? What is the pivot row? What are the current basic variables and their values in the basic solution? What are the new basic variables and their values? What is the current value of \(z\) in the basic solution corresponding to the canonical form. What is the value of \(z\) in the new basis? Is one pivot enough to get to an optimal canonical form? Note: you do not have to carry out the complete pivot operation to answer this question. Explain the logic behind your approach. In particular, how would you generalize this to an arbitrary canonical form with non-negative objective coefficients and some infeasible basic variables?

\[
\begin{align*}
\text{Min. } z \\
\text{s.t. } &\quad 5x_6 + 2x_7 + 3/2x_8 = z \\
&\quad -10x_6 - 2x_7 + 1/4x_8 = -1 \\
&\quad +2/3x_6 - 2x_7 - x_8 + x_9 = -2 \\
&\quad +x_6 - x_8 - x_9 = 2 \\
&\quad +3/5x_7 + 1/8x_8 + 3x_9 = 1 \\
&\quad +x_6 + x_7 + 1/5x_8 + x_9 = 1 \\
&\quad x_j \geq 0, j=1, \ldots, 9
\end{align*}
\]

Solution:
You need to increase the values of \(x_1\) and/or \(x_2\) to get them feasible (\(\geq 0\)). Traditionally, you work on the most negative basic variable, in this case, \(x_2\) (you could start with \(x_1\), though). In this case, we can increase the basic variables, \(x_7\) or \(x_8\); that is, we bring
either \( x_7 \) or \( x_8 \) into the basis. If we bring in \( x_8 \), then in the pivot process when we make the coefficient in the cost row zero for \( x_8 \), the cost coefficient for \( x_7 \) will be -1. We don’t want this because we want to keep the cost coefficients non-negative. So we try bringing in \( x_7 \). This works. So we pivot in the \( x_2 \) row and the \( x_7 \) column. \( z \) was 0, and is now 2. \( x_2 \), \( x_8 \), and \( x_9 \) remain at 0. \( x_7 \) becomes non-basic (and feasible) at 0. We are also fortunate in that, \( x_2 \), which was infeasible at -1, now is basic at +1; moreover the other basic variables remain feasible: \( x_4 \) that was 2 remains at 2; \( x_6 \) was 2, and is now, 2/5; \( x_5 \), which was 1 is now 0. So the new dictionary will be both feasible, and optimal and we are done.

This process is called the **dual simplex** method. It can be interpreted as the simplex method applied to the dual. The general procedure is:

**Step 1 (Row Choice):** Pick a row, \( r \), in which the basic value is negative (often you pick the most negative).

**Step 2 (Choose Column):** Pick a column, \( s \), so that the new cost row will remain non-negative. The rule is 

\[ s = \arg\min_j \left\{ \frac{c_j}{-\alpha_{rs}} \mid \alpha_{rs} < 0 \right\} \]

which is similar to the row choice rule for the simplex method.

**Step 3 (Pivot):** This is exactly the same as the pivot used in the simplex method.

There is no guarantee that the pivot will result in an optimal solution. In general, other basic variables with negative values will not become feasible; in fact they may get worse. Additionally, basic variable that were feasible may become infeasible! What provides a finiteness for the algorithm is that the basic value for the objective will increase each iteration (actually it will not decrease, there is an issue of dual degeneracy analogous to that of the simplex method).

Suppose you have a set of \( m \) linear less than or equal constraints. You want to know if there is a non-negative vector \( x \) that satisfies all the constraints with a specific \( k \) of them holding with equality, the rest may or may not hold with equality. System I represents what you want. Do systems I and II make up a theorem of alternatives? If it does, prove it. If not, give a counter example. Assume all the constants on the right are non-negative.

**I** There exists \( x \geq 0 \) such that \( Ax = b \geq 0 \), and \( Dx \leq f \geq 0 \), or

**II** There exists vectors, \( u \) and \( v \), with \( v \geq 0 \) such that:

\[ uA \leq vD, \quad ub > vf \]

But not both.

\( A^{kn}, b \) represents the constraints that you want to hold with equality, and \( D^{(m-k)n} \), \( f \) represents the constraints that may or may not be strict.

**Solution:**

Consider the auxiliary problem:
Min. \( ey \)
\[
\text{s.t.} \quad A x + I y = b \\
D x \leq f \\
x \geq 0, y \geq 0
\]

It has a feasible solution: \( x=0, y=b \). The objective has a lower bound of 0; therefore, the auxiliary problem has an optimal solution by the fundamental theorem of linear programming. Either the optimal value of \( ey \) is 0 or it is positive. If the value is 0, the solution satisfies System I. If the value is positive, look at the dual. It is:

\[
\begin{align*}
\text{Max} & \quad ub - vf \\
\text{s.t.} & \quad uA - vD \leq 0 \\
& \quad v \geq 0
\end{align*}
\]

By strong duality \( ub-vf \) at optimality equals the positive optimal value of the auxiliary problem. Thus, \( u \) and \( v \) provide a solution to system II. Since the dual of the dual is the primal, if we take the dual of the dual we get back to the auxiliary problem, and its optimal value is still positive, so both systems cannot have a solution.

5. A very useful tool for integer, mixed integer, large scale, and non-linear programming is lagrangian relaxation. The general situation is as follows. Consider the problem:

\[
\begin{align*}
\text{Min} & \quad f(x) \\
\left( P \right) & \text{s.t.} \quad g(x) \geq 0 \\
& \quad x \in \mathcal{D}
\end{align*}
\]

where \( x \in \mathbb{R}^n \), and \( g \) is an vector of \( m \) constraints. We suppose that the problem without the inequality constraints is relatively simple to solve. The inequality constraints complicate things. To get around the problem it is sometime useful to put the offending constraints into the objective:

\[
\begin{align*}
\left( P_n \right) & \quad \text{Min} \quad L(x, \pi) = f(x) - \pi g(x) \\
& \quad \text{s.t.} \quad x \in \mathcal{D}
\end{align*}
\]

where \( \pi \) is a non-negative vector. \( \pi g \) is nothing but a non-negatively weighted sum of the constraints. \( \left( P_n \right) \) is called a lagrangian relaxation.

(a) Show that the “lagrangian relaxation” is a relaxation in the sense of our relaxation lemma (see attachment); that is, it satisfies the conditions of the lemma.

Solution:

Let \( \mathcal{D}^0 = \{ x \mid g(x) \geq 0, x \in \mathcal{D} \} \) then the domain, \( \mathcal{D}^0 \), of \( P \) is a subset of, \( \mathcal{D} \), the domain of \( \left( P_n \right) \). Moreover for any point in \( \mathcal{D}^0 \) we have \( L(x, \pi) = f(x) - \pi g(x) \leq f(x) \)
since $\pi \geq 0$, and $g(x) \geq 0$. Thus the conditions for the relaxation lemma are satisfied for any non-negative $\pi$.

(b) What does the lemma tell us specifically about the relations between (P) and $(P_\pi)$?

Solution:

Any optimal solution of $(P_\pi)$ is a lower bound on the optimal solution of (P); moreover, if we have a solution, $x^*$, of $(P_\pi)$, so that $x^* \in D^0$, and $f(x^*) = L(x^*; \pi)$, then $x^*$ optimizes (P). Note that for this to happen $\pi g$ must be zero, which is a complementary slackness condition.

(c) The lagrangian dual of (P) is:

\[
LD \quad v_d = \max_{\pi} \min_{x \in D} L(x; \pi)
\]

Why is this of interest? What is the relationship between $v_d$ and the optimum $P$ value of (P), $v_p$? Prove that what you say is correct.

Solution:

Since the optimal value of $L(x; \pi)$ in $(P_\pi)$ is a lower bound for the objective function in (P) for any non-negative $\pi$, the maximum of these is the best lower bound of this sort we can get. Of course $v_d \leq v_p$. The difference is the "duality gap." In general, it is positive; however, for linear programs it is 0 as we see in the next part.

(d) Show that, in the case of linear programming we have $v_{d^*} = v_p$.

\[
\begin{align*}
\text{Min } & cx \\
\text{s.t. } & Ax \geq b \\
\text{(PLP) } & Dx \geq d \\
& x \geq 0
\end{align*}
\]

where $Ax \geq b$ is the set of difficult constraints, and $D = \{x \mid Dx \geq d, x \geq 0\}$.

Solution:

We can assume that (PLP) has an optimal solution $x^*$; then it also has an optimal dual solution in the $k$-vector $\pi^*$, and $(m-k)$-vector $a^*$. Then use the same $\pi^*$ in $(P_\pi)$. Then use the same $x^*$, and $a^*$ as primal and dual solutions for $(P_\pi)$. They satisfy the complementary slackness conditions for $(P_\pi)$, and the objectives have the same value, so in this case $v_{d^*} = v_p$. In other applications such as to integer programming and other non-convex programs, the gap will generally be positive.

R. Van Slyke, November 26, 2004