(e) Describe the dual problem in the language of the original project scheduling model.

15.8 *Continuation*. Here is an algorithm for computing optimal start times t_j:

1. List the jobs so that the predecessors of each job come before it in the list.
2. Put $t_0 = 0$.
3. Go down the list of jobs and for job j put $t_j = \max\{t_i + d_i : i \text{ is a predecessor of } j\}$.

(a) Apply this algorithm to the specific instance from Problem 15.4. What are the start times of each of the jobs? What is the project duration?

(b) Prove that the solution found in Part (a) is optimal by exhibiting a corresponding dual solution and checking the usual conditions for optimality (*Hint: The complementary slackness conditions may help you find a dual solution.*).

15.9 *Currency Arbitrage*. Consider the world's currency market. Given two currencies, say the Japanese Yen and the US Dollar, there is an exchange rate between them (currently about 110 Yen to the Dollar). It is always true that, if you convert money from one currency to another and then back, you will end up with less than you started with. That is, the product of the exchange rates between any pair of countries is always less than one. However, it sometimes happens that a longer chain of conversions results in a gain. Such a lucky situation is called an *arbitrage*. One can use a linear programming model to find such situations when they exist.

Consider the following table of exchange rates (which is actual data from the Wall Street Journal on Nov 10, 1996):

```
param rate:
      USD     Yen    Mark    Franc   :=
USD   .       111.52 1.4987  5.0852
Yen   .008966 .      .013493 .045593
Mark  .6659   73.964 .       3.3823
Franc .1966   21.933 .29507  .
;
```

It is not obvious, but the USD→Yen→Mark→USD conversion actually makes $0.002 on each initial dollar.

To look for arbitrage possibilities, one can make a *generalized network model*, which is a network flow model with the unusual twist that a unit of flow that leaves one node arrives at the next node multiplied by a scale factor—in our example, the currency conversion rate. For us, each currency is represented by a node. There is an arc from each node to every other node. A flow of one unit out of one node becomes a flow of a different magnitude at the head node. For example, one dollar flowing out of the USD node arrives at the Franc node as 5.0852 Francs.

Let x_{ij} denote the flow from node (i.e. currency) i to node j. This flow is measured in the currency of node i.

One node is special; it is the *home* node, say the US Dollars (USD) node. At all other nodes, there must be flow balance.

(a) Write down the flow balance constraints at the 3 non-home nodes (Franc, Yen, and Mark).

At the home node, we assume that there is a supply of one unit (to get things started). Furthermore, at this node, flow balance will not be satisfied. Instead one expects a net inflow. If it is possible to make this inflow greater than one, then an arbitrage has been found. Let f be a variable that represents this inflow.

(b) Using variable f to represent net inflow to the home node, write a flow balance equation for the home node.

Of course, the primal objective is to maximize f.

(c) Using y_i to represent the dual variable associated with the primal constraint for currency i, write down the dual linear program. (Regard the primal variable f as a free variable.)

Now consider the general case, which might involve hundreds of currencies worldwide.

(d) Write down the model mathematically using x_{ij} for the flow leaving node i heading for node j (measured in the currency of node i), r_{ij} for the exchange rate when converting from currency i to currency j, and f for the net inflow at the home node i^*.
(e) Write down the dual problem.

(f) Can you give an interpretation for the dual variables? Hint: It might be helpful to think about the case where $r_{ji} = 1/r_{ij}$ for all i, j.
(g) Comment on the conditions under which your model will be unbounded and/or infeasible.

Notes

The Hitchcock problem was introduced by Hitchcock (1941). Dijkstra's algorithm was discovered by Dijkstra (1959).

The Max-Flow Min-Cut Theorem was proved independently by Elias et al. (1956), by Ford & Fulkerson (1956) and, in the restricted case where the upper bounds are all integers, by Kotzig (1956). Fulkerson & Dantzig (1955) also proved the Max-Flow Min-Cut Theorem. Their proof uses duality, which is particularly relevant to this chapter.

The classic references for dynamic programming are the books by Bellman (1957) and Howard (1960). Further discussion of label-setting and label-correcting algorithms can be found in the book by Ahuja et al. (1993).