

1. *Project Scheduling*. This problem deals with the creation of a project schedule; specifically, the project of building a house. The project has been divided into a set of jobs. The problem is to schedule the time at which each of these jobs should start and also to predict how long the project will take. Naturally, the objective is to complete the project as quickly as possible (time is money!). Over the duration of the project, some of the jobs can be done concurrently. But, as the following table shows, certain jobs definitely can't start until others are completed.

<b>Job</b>	<b>Duration (weeks)</b>	<b>Must be Preceded by</b>
0. Sign Contract with Buyer	0	–
1. Framing	2	0
2. Roofing	1	1
3. Siding	3	1
4. Windows	2.5	3
5. Plumbing	1.5	3
6. Electrical	2	2,4
7. Inside Finishing	4	5,6
8. Outside Painting	3	2,4
9. Complete the Sale to Buyer	0	7,8

One possible schedule is the following:

Job	Start Time
0. Sign Contract with Buyer	0
1. Framing	1
2. Roofing	4
3. Siding	6
4. Windows	10
5. Plumbing	9
6. Electrical	13
7. Inside Finishing	16
8. Outside Painting	14
9. Complete the Sale to Buyer	21

With this schedule, the project duration is 21 weeks (*the difference between the start times of jobs 9 and 0*).

To model the problem as a linear program, introduce the following decision variables:

$t_j$  = the start time of job  $j$ .

- Write an expression for the objective function, which is to minimize the project duration.
- For each job  $j$ , write a constraint for each job  $i$  that must precede  $j$ ; the constraint should ensure that job  $j$  doesn't start until job  $i$  is finished. These are called *precedence constraints*.

- Continuation.* This problem generalizes the specific example of the previous problem. A project consists of a set of jobs  $\mathcal{J}$ . For each job  $j \in \mathcal{J}$  there is a certain set  $\mathcal{P}_j$  of other jobs that must be completed before job  $j$  can be started. (This is called the set of *predecessors* of job  $j$ .) One of the jobs, say  $s$ , is the starting job; it has no predecessors. Another job, say  $t$ , is the final (or terminal) job; it is not the predecessor of any other job. The time it will take to do job  $j$  is denoted  $d_j$  (the *duration* of the job).

The problem is to decide what time each job should begin so that no job begins before its predecessors are finished, and the duration of the entire project is minimized. Using the notations introduced above, write out a complete description of this linear programming problem.

3. *Continuation.* Let  $x_{ij}$  denote the dual variable corresponding to the precedence constraint that ensures job  $j$  doesn't start until job  $i$  finishes.
- Write out the dual to the specific linear program in Problem 1.
  - Write out the dual to the general linear program in Problem 2.
  - Describe how the optimal value of the dual variable  $x_{ij}$  can be interpreted.
4. *Continuation.* The project scheduling problem can be represented on a directed graph with arc weights as follows. The nodes of the graph correspond to the jobs. The arcs correspond to the precedence relations. That is, if job  $i$  must be completed before job  $j$ , then there is an arc pointing from node  $i$  to node  $j$ . The weight on this arc is  $d_i$ .
- Draw the directed graph associated with the example in Problem 1, being sure to label the nodes and write the weights beside the arcs.
  - Return to the formulation of the dual from Problem 3(a). Give an interpretation of that dual problem in terms of the directed graph drawn in Part (a).
  - Explain why there is always an optimal solution to the dual problem in which each variable  $x_{ij}$  is either 0 or 1.
  - Write out the complementary slackness condition corresponding to dual variable  $x_{26}$ .
  - Describe the dual problem in the language of the original project scheduling model.

5. *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 1. 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.*).

6. *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  $i, j$ .

6

- (g) Comment on the conditions under which your model will be unbounded and/or infeasible.