ORIE 4820: Spreadsheet-Based Modeling and Data Analysis
Solving Linear Programs With Excel
Spring 2013

During this lab exercise, you will examine three problems that can be formulated as linear and/or integer linear programs. The primary purpose is to give you practice ***formulating and constructing linear programming models in Excel***, and ***solving LPs/ILPs using Excel's Solver.***

The template file for this exercise is ***Solving-LPs-with-Excel.xlsm***. Download a copy of the file from the course Blackboard site and *save it on your computer*.

Topics/Tools we will cover:

- ***Formulating decision problems as linear programs*** and/or ***integer linear programs***
- ***Representing LP/ILP models*** in Excel
- ***Using Excel's Solver*** to find optimal solutions and ***interpreting the solutions***
- ***Automating Solver*** using VBA

If you have problems or questions at any point during the session, please raise your hand.

Background:

The workbook ***Solving-LPs-with-Excel.xlsm*** contains several worksheets:

- The *Product Mix* worksheet contains relevant information for the product mix problem described in class. (*Product Mix – Graph* is a supplementary worksheet that simply drives the graphical visualization of this problem.)
- The *Box Packing* worksheet contains relevant information for a slightly smaller version of the box packing problem we built a greedy algorithm for last week.
- The *Shift Scheduling* worksheet contains relevant information for a staff scheduling problem to be examined.

We are interested in formulating these decision problems as mathematical programs and in finding solutions using Excel's Solver as an optimization tool.

Section 1: Product Mix Problem

Consider the product mix decision problem we discussed in class, where we want to determine ***how much of each product (A and B) to make to maximize next month's profit***. With x_A and x_B representing the number of units of each product to produce, we formulated the problem as follows:

$$\begin{aligned}
&\text{Maximize } 450\,x_A + 420\,x_B \\
&\text{subject to:} \\
&2\,x_A \le 336 \\
&2.5\,x_B \le 336 \\
&2\,x_A + 1.5\,x_B \le 336 \\
&x_A \ge 75 \\
&x_A \le 140 \\
&x_B \ge 0 \\
&x_B \le 140
\end{aligned}$$

For this model to be entered into Solver, you must organize the data so that your spreadsheet contains one cell representing each "piece" of the LP. ***There should be a cell for each decision variable, a cell that captures the objective function, a cell that captures the "left-hand side" of each constraint, and a cell that captures the "right-hand side" of each constraint***.

(1) Enter the above ***linear program*** into Solver (Data->Analysis->Solver) and solve. Under "Select a Solving Method", be sure to select the "Simplex LP" option.

(2) ***Save*** the model to a specific area on the worksheet:
   a. From the ribbon, select Data->Analysis->Solver.
   b. Click "Load/Save", then select a blank area of the appropriate size starting with cell G22, and click "Save".

(3) ***Record a macro*** to re-solve the model at the click of a button:
   a. From the ribbon, select Developer->Code->Record Macro.
   b. Type in a descriptive name for the macro, like "Solve_Product_Mix", and press OK.
   c. From the ribbon, select Data->Analysis->Solver.
   d. Click "Load/Save", then select the entire area where you saved the model, and click "Load". (If you get a dialog box that asks whether you want to "merge" or "replace" the existing model, select "replace".)
   e. Click "Solve", then OK to keep the Solver solution.
   f. From the ribbon, select Developer->Code->Stop Recording. Your macro has been recorded.
   g. Create a button and attach to it the macro you just created (Developer->Controls->Insert, then Form Controls). You can edit the text on the button or move the button around as you would an object in Powerpoint.

Before running your macro, be sure that Visual Basic recognizes the calls to the Solver tool:

   h. Press Alt-F11 to access the Visual Basic Editor.
   i. From the Visual Basic Editor main menu, select Tools->References…
   j. Make sure the SOLVER box is checked, and press OK.

Section 2: Box Packing Revisited

Recall the Box Packing problem from last week:

Given a set of items having different weights and a set of boxes with weight capacities, pack all items into boxes using as few boxes as possible.

This week, we will consider a smaller version of the same problem (with 7 possible boxes instead of 10), but instead of using an iterative greedy algorithm to construct a solution, you will formulate and (attempt to) solve the problem as an ***integer linear program*** (ILP) using Excel's Solver.

(1) Write down an ILP formulation of the problem in standard form:

Hint: Use decision variables x_{ij}, for $i = 1, \ldots, 20$ and $j = 1, \ldots, 7$, where $x_{ij} = 1$ if item i is placed in box j, and 0 otherwise. In the formulation, you will need to link the x_{ij} to the additional decision variables y_j, for $j = 1, \ldots, 7$, where $y_j = 1$ if box j is used, and 0 otherwise.

(2) Organize the information on the worksheet so that Solver can be used to solve the problem. You may have to create additional cells in order to do this.

(3) Answer the following questions:

a. Enter and solve the problem that you formulated above. Was Solver able to find an optimal solution within 2 minutes? If not, what does the best found solution look like after 2 minutes?

b. Assuming that items can be split among boxes, re-formulate and solve the relaxed problem. Which constraint(s) did you change, and how? Was it easier for Solver to find a solution this time? How does the solution compare with part (a)?

Section 3: Shift Scheduling Problem

This problem was adapted from: Operations Management, Reid, R.D. and N.R. Sanders, John Wiley & Sons, 2002 (Supplement B: Linear Programming, p.B40, #5).

The manager of a large discount store needs to schedule cashiers so that he has an adequate number to serve customers throughout the day without having too many. The store is open from 9:00 am to 9:00 pm every day of the week. Depending on the day and on the time of day, customer volume changes markedly. Based on historical data, the manager has found that the intra-day customer volumes are essentially the same on Monday-Thursday, but those for Friday, Saturday, and Sunday are all different. For planning purposes, he has divided each day into three 4-hour segments and has estimated the minimum of cashiers needed for each time period:

Monday-Thursday			Friday			Saturday			Sunday		
9am-1pm	1pm-5pm	5pm-9pm	9am-1pm	1pm-5pm	5pm-9pm	9am-1pm	1pm-5pm	5pm-9pm	9am-1pm	1pm-5pm	5pm-9pm
6	5	8	3	8	4	10	14	7	4	12	6

In constructing a weekly master schedule, there are certain rules regarding *individual employee work schedules* that must be respected:

- The weekly work schedule for each employee must consist of 5 consecutive work days with 2 consecutive days off. An employee's "start" day, however, can be any day of the week.
- Each employee must be assigned to work either the 9am-5pm "Morning" shift for 5 consecutive days or the 1pm-9pm "Evening" shift for 5 consecutive days.

(1) Write down an ILP formulation of the problem in standard form:

Hint: There are 14 possible work schedules, so let the decision variables be:

x_{ij} = The number of employees whose 5-day work schedule begins on day i (i = M, T, W, R, F, S, N) and covers shift j, where j = 1 denotes the 9am-5pm shift and j = 2 denotes the 1pm-9pm shift.

(2) Organize the information on the worksheet so that Solver can be used to solve the problem. You may have to create additional cells in order to do this.

Hint: Create a 0-1 matrix that identifies the 4-hour time segments that are covered by each of the 14 schedules.

(3) Answer the following questions:

a. Find a feasible weekly schedule that ***minimizes the total number of cashiers scheduled***.

b. Suppose that the minimum cashier requirements for all of the 1-5pm timeslots are removed (i.e., all RHSs for these constraints are zero). ***Re-solve the problem, and explain the results.***

c. Suppose that you wanted to solve this problem using an ***iterative greedy approach*** instead of solving it as an ILP. What would be your "greedy" selection mechanism, and how would you implement it?