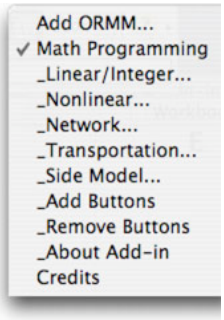
Mathematical Programming

The Mathematical Programming Add-in constructs models that can be solved using the Solver Add-in or one of the solution add-ins provided in the collection.



When the Math Programming add-in is installed, several new command lines are added to the OR_MM menu. The menu items under the title Math Programming create models of the different types.

Selecting an item from this list causes a dialog box to be presented which constructs a mathematical programming model. The models created by the add-in are solved with the Excel Solver, the Jensen Network Solver or the Jensen LP/IP Solver. All are Excel add-ins. Documentation for these programs can be reached by clicking the links on the lower left.

The Solver add-in comes with Excel, and it can solve linear programming, integer programming and nonlinear programming models. The Math Programming add-in automatically builds Solver models and calls the computational procedures that solve the problems. All four model types can be can be solved in this way.

The Jensen LP/IP Solver solves linear or integer programming problems. It is available for the Linear/Integer Programming and Network Flow Programming model types.

The Jensen Network Solver can solve pure or generalized network flow models. Both linear and integer problems can be solved. It is available for the Network Flow Programming or Transportation model types.

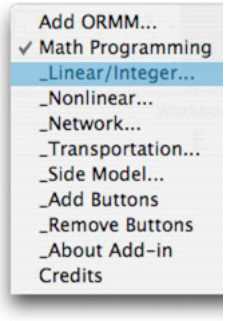
Parametric analysis can be applied to any of the math programming models. Here one parameter is allowed to vary within a specified range and the model is solved for each value. The results are provided by a table and a chart.

Side Models provide an additional form on which math programming models may be constructed. They are much more compact representation than the forms provided earlier.

A model worksheet includes *buttons* that change the model or call the solution algorithms. When opening a workbook on a different computer than the computer that created the model *Excel* will present a dialog indicating missing links. It is best to cancel this dialog. Then choose the *Add Buttons* command to replace all the buttons on math programming models in the workbook. Before saving a workbook that will be transferred to another computer, choose the *Remove Buttons* command. This removes all the buttons in the workbook so the link message does not appear. The buttons can subsequently be added with the *Add Buttons* command. The demonstration workbook for this add-in has its buttons removed, so the first thing to do is add the buttons.

To build and solve models, the Math Programming add-in and one of the solver add-ins must be installed. The Math Programming add-in can build models, but not solve them. The solver add-ins can solve models, but not build them.

Linear/Integer Programming



This option is used for constructing a model whose objective function and constraints have the linear form. The model may have real, integer or mixed variables.

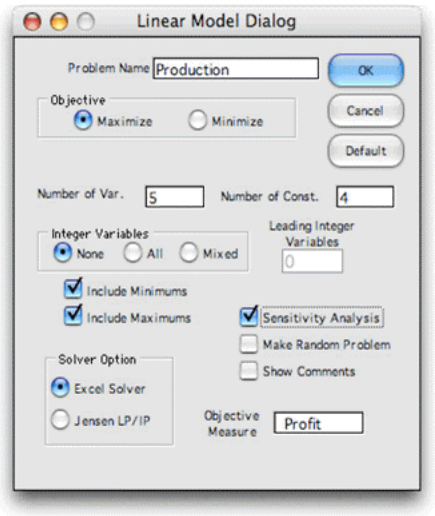
Example Problem: Finding the Optimum Product Mix

A type of problem most often identified with the linear program is the problem of distributing scarce resources among alternative activities. In this example the scarce resources are times available on four machines and the alternative activities are the production volumes of five products. The machine requirements in hours per unit are shown for each product in the table below. With the exception of product 4, that does not require machine 1, each unit of product must pass through all four machines. The unit profits for each product are also shown in the table. There are four machines of type 1, five of type 2, three of type 3 and seven of type 4. Each machine operates 40 hours per week. The linear programming model is to determine the optimum weekly production quantities for the products. The goal is to maximize total profit.

	Prod 1	*Prod 2*	*Prod 3*	*Prod 4*	*Prod 5*
Machine 1	1.2	1.3	0.7	0.0	0.5
Machine 2	0.7	2.2	1.6	0.5	1.0
Machine 3	0.9	0.7	1.3	1.0	0.8
Machine 4	1.4	2.8	0.5	1.2	0.6
Profit/Unit	$18	25	10	12	15

To enter the model for this situation, select the Linear/Integer item from the OR_MM menu. The dialog defining the structure of the linear model is presented. Fields are available to define the Name, number of variables and number of constraints for the model. Buttons determine whether the problem is a maximization or a minimization, and the integer character of the variables of the problem. The variables may be specified as no integer, all integer, or mixed. In the latter case the field to the right of the buttons determines the number of leading integer variables. For a mixed problem, the leading integer variables have the smallest indices. Additional integer variables may be specified by placing the letter I before their indices on the model worksheet.

One important point concerns the *Problem Name*. The entry here is used to provide Excel names to many ranges on the worksheet. The name must satisfy Excel's restrictions for naming ranges, that is: the names must not contain spaces, they must start with a letter and they may not include punctuation marks. If an error occurs when this dialog closes, try a different name for the problem. The program automatically suggests names like LP_1, LP_2 and so on. The user may prefer a more descriptive name.



The fields below the integer definitions determine some features of the model that will affect the data entry. When the *Include Minimums* box is checked, a row is provided to hold values for the minimums of the decision variables. If no row is provided, each variable is restricted from below by 00 restricts from below each variable. When the *Include Maximums* box is checked, a row is provided to hold values for the maximum values or upper bounds of the decision variables. If no row is provided, the upper bounds are assumed to be infinitely large.

The *Sensitivity Analysis* checkbox determines whether the solution procedure will create a sensitivity analysis worksheet after performing the optimization. With the *Random Problem* option selected, the program generates data for the model using random numbers. Otherwise, all data items will be 0, except the constraint bounds, which will be given large values. The *Show Comments* checkbox will install Excel comments on important cells of the worksheet. This helps to understand the purpose of the various cells and ranges.

The objective measure field specifies the measure to be maximized or minimized. Here we are maximizing the profit.

The *Solver* option specifies the solution add-in to be used. Either choose the *Excel Solver* or the Jensen LP/IP Solver add-in. When using the *Excel Solver* it is important that the *Solver* dialog be opened *before* trying to construct a model. Then the model is automatically loaded into the Excel Solver when the model structure is placed on the worksheet.

Most of the options on the original model dialog can be changed by using the *Change button* that is placed on the model worksheet.

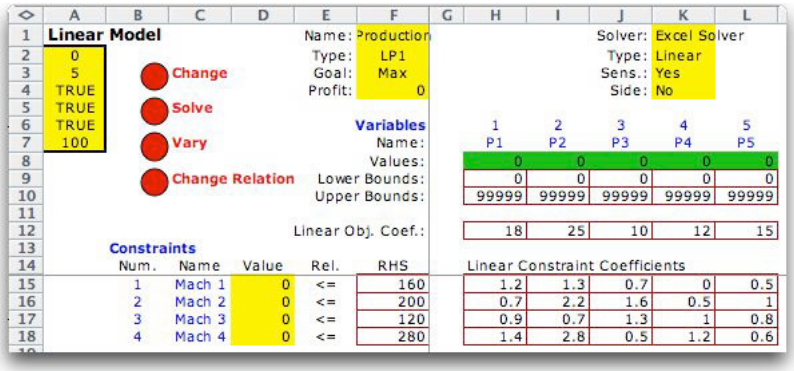
Constructing the Worksheet

Pressing OK initiates the Macro that constructs the worksheet below. Generally the outlined cells may be changed on the worksheet to enter data describing the model. Cells colored yellow should not be changed. They hold formulae and data required by the program. Cells outside the region defining the model may be used for any purpose. The cells defining the variable values, H8:L8, may be set by the user to experiment with various solutions, however, the Solver Add-in replaces the contents of this range with the optimum solution. Generally, green cells hold numbers filled in by algorithms implemented with VBA subroutines.

The range A2:A7 holds the Excel Solver model. That range is left blank when the Jensen Solver is used. When using the Excel Solver it is important that the user establish contact with that program before a model is created. To do this, select the Solver item from the Tools menu. After the Solver dialog opens, simply close the dialog. This establishes the link to the Excel Solver. The Math-Programming add-in automatically loads and calls the Solver as necessary.

	A	B	C	D	E	F	G	H	I	J	K	L
1	**Linear Model**				Name:	Production				Solver:	Excel Solver	
2	0				Type:	LP1				Type:	Linear	
3	5		Change		Goal:	Max				Sens.:	No	
4	TRUE				Profit:	0				Side:	No	
5	TRUE		Solve									
6	TRUE					Variables		1	2	3	4	5
7	100		Vary			Name:		X1	X2	X3	X4	X5
8						Values:		0	0	0	0	0
9			Change Relation			Lower Bounds:		0	0	0	0	0
10						Upper Bounds:		10000	10000	10000	10000	10000
11												
12						Linear Obj. Coef.:		0	0	0	0	0
13		Constraints										
14		Num.	Name	Value	Rel.	RHS		Linear Constraint Coefficients				
15		1	Con1	0	<=	10000		0	0	0	0	0
16		2	Con2	0	<=	10000		0	0	0	0	0
17		3	Con3	0	<=	10000		0	0	0	0	0
18		4	Con4	0	<=	10000		0	0	0	0	0

The user adds data describing the coefficients for his or her problem. Variable names are placed in row 7, the objective coefficients are in row 12, and the lower and upper bounds for variables in rows 9 and 10. The upper bounds for the constraints are placed in column F starting at row 15. Constraint relations are in column E. Initially the relations are <=, however, they may changed to >= or = by placing the cursor on a cell and clicking on the *Change Relation* button. Several relations can be changed simultaneously by selecting a range of cells before clicking the button. The constraint coefficients start in cell H15 and continue downward and to the right. After entering the data for the example problem the worksheet appears as below.



	A	B	C	D	E	F	G	H	I	J	K	L
1	**Linear Model**				Name:	Production				Solver:	Excel Solver	
2	0				Type:	LP1				Type:	Linear	
3	5		Change		Goal:	Max				Sens.:	Yes	
4	TRUE				Profit:	0				Side:	No	
5	TRUE		Solve									
6	TRUE					Variables		1	2	3	4	5
7	100		Vary			Name:		P1	P2	P3	P4	P5
8						Values:		0	0	0	0	0
9			Change Relation			Lower Bounds:		0	0	0	0	0
10						Upper Bounds:		99999	99999	99999	99999	99999
11												
12						Linear Obj. Coef.:		18	25	10	12	15
13		Constraints										
14		Num.	Name	Value	Rel.	RHS		Linear Constraint Coefficients				
15		1	Mach 1	0	<=	160		1.2	1.3	0.7	0	0.5
16		2	Mach 2	0	<=	200		0.7	2.2	1.6	0.5	1
17		3	Mach 3	0	<=	120		0.9	0.7	1.3	1	0.8
18		4	Mach 4	0	<=	280		1.4	2.8	0.5	1.2	0.6

The worksheet shows only the coefficients that describe the objective and constraints of the model. The algebraic model is below.

$$
\begin{array}{lrrrrrrl}
\text{Maximize} & \Sigma = & 18P_1 & +25P_2 & +10P_3 & +12P_4 & +15P_5 & \\
M_1: & 0 \le & 1.2P_1 & +1.3P_2 & +0.7P_3 & +0.0P_4 & +0.5P_5 & \le 160 \\
M_2: & 0 \le & 0.7P_1 & +2.2P_2 & +1.6P_3 & +0.5P_4 & +1.0P_5 & \le 200 \\
M_3: & 0 \le & 0.9P_1 & +0.7P_2 & +1.3P_3 & +1.0P_4 & +0.8P_5 & \le 120 \\
M_4: & 0 \le & 1.4P_1 & +2.8P_2 & +0.5P_3 & +1.2P_4 & +0.6P_5 & \le 280
\end{array}
$$

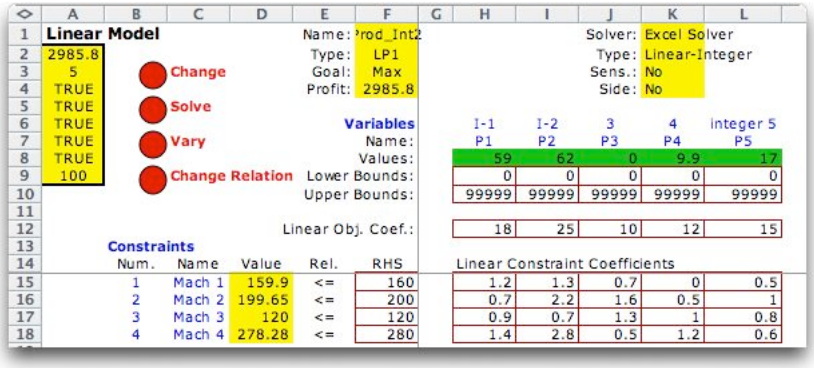
$$0 \le P_j \le 99999 \text{ for } j = 1, \ldots, 5.$$

For the example only the relations on the right of the algebraic constraints are relevant, so we used single bound constraints. For some problems it might be more convenient to specify two bounds for each constraint. In that case the constraint lower bounds would have been 0.

The large numbers used for the upper bounds on the variables indicate that there are no effective upper limits to the variables. Constraint and variable bounds may be either positive or negative.

Solving the Problem

The problem is Solved by clicking on the Solve buttonClicking on the Solve button solves the problem. When using the Excel Solver, the model is created, loaded into the Solver program and the algorithmic portion of Solver is called to obtain a solution. The solution for the example is shown below. The optimum decision variable values are in the range H8:L8. The objective function value is in F4. The constraint values give the evaluation of the constraint expressions for the optimum solution.



	A	B	C	D	E	F	G	H	I	J	K	L
1	**Linear Model**				Name:	Prod_Int2				Solver:	Excel Solver	
2	2985.8				Type:	LP1				Type:	Linear-Integer	
3	5		Change		Goal:	Max				Sens.:	No	
4	TRUE		Solve		Profit:	2985.8				Side:	No	
5	TRUE											
6	TRUE					**Variables**		I-1	I-2	3	4	integer 5
7	TRUE		Vary			Name:		P1	P2	P3	P4	P5
8	TRUE					Values:		59	62	0	9.9	17
9	100		Change Relation			Lower Bounds:		0	0	0	0	0
10						Upper Bounds:		99999	99999	99999	99999	99999
11												
12						Linear Obj. Coef.:		18	25	10	12	15
13		**Constraints**										
14		Num.	Name	Value	Rel.	RHS		Linear Constraint Coefficients				
15		1	Mach 1	159.9	<=	160		1.2	1.3	0.7	0	0.5
16		2	Mach 2	199.65	<=	200		0.7	2.2	1.6	0.5	1
17		3	Mach 3	120	<=	120		0.9	0.7	1.3	1	0.8
18		4	Mach 4	278.28	<=	280		1.4	2.8	0.5	1.2	0.6

The change button is used to change the structure of the problem by changing the numbers of variables and/or constraints or by changing the integer nature of the variables. The solver option, the Goal (max or min) and the Sensitivity choice can also be changed. All changes are automatically incorporated into the Solver model.

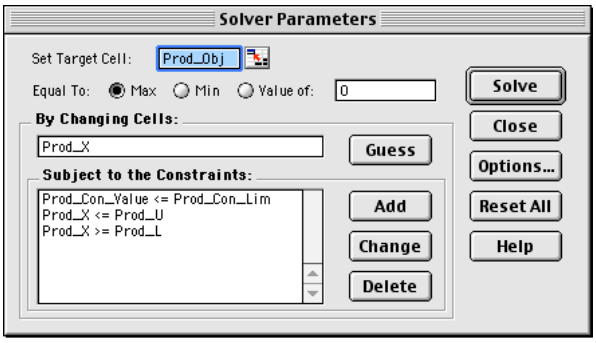
Jensen Solver

The Jensen LP/IP Solver is called automatically when the Solve button is clicked. Jensen solvers are available for linear programming problems and network flow programming problems with or without integer variables. Generally, it takes much longer to solve problems when some or all the variables are required to be integer. The network solver can also solve some nonlinear problems.

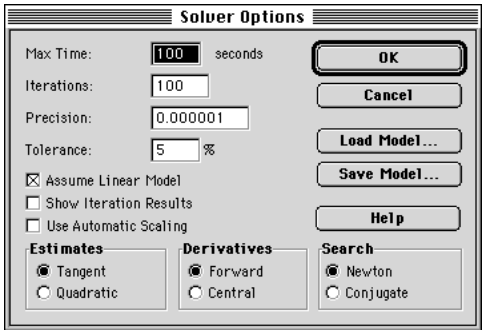
Excel Solver

When using the Excel Solver, you must establish connection to the Solver by selecting the Solver option from the Tools menu. This opens a dialog. Just close the dialog. The add-in will control subsequent activities with regard to the Excel Solver.

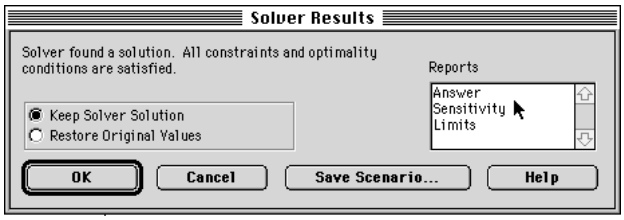
After the Math Programming add-in has created a model that uses the Excel Solver, the student can interact with the Solver via the Tools menu. The Solver dialog for the example problem is below. The fields refer to named ranges on the worksheet. Clicking the Solver button initiates the Solver algorithm.



The Options button provides access to Solver options that control the solution procedure. The most important option for linear programming is the *Assume Linear Model* button. When checked, the solution algorithm is a simplex procedure, otherwise a nonlinear programming algorithm is used even when the model is linear. Generally, the simplex procedure is faster and more accurate than nonlinear programming for linear problems.



When Solver has found the optimum, the Results dialog appears. Several optional reports may be selected. The Sensitivity report provides reduced costs and sensitivity ranges for the variables and dual values and sensitivity ranges for the constraint bounds. If the add-in has called the Solver, this dialog is skipped. A sensitivity report is generated if that option had been selected during the problem definition.



It is usually only necessary to load a particular model only once. If the characteristics of the model such as coefficients, bounds, number of variables and number of constraints are changed, the model is automatically adjusted and need not be reloaded. To solve a modified problem, simply click on the Solve button in the Solver dialog. On the other hand if the integrality restrictions for some variables or the direction of the objective function is changed (from max to min or from min to max) then the model must be reloaded into solver. The add-in does all this automatically, however, the changes can also be accomplished by interacting directly with the Excel Solver.

Sensitivity Analysis

The sensitivity analysis provided by the Excel Solver for the example is shown below. We assume that the student is familiar with the meaning of the various terms, so they will not be discussed here. Note however that the ranges associated with the objective coefficients are given by allowable increases and decreases. For example, the row for P1 indicates that the objective coefficient value (18 in the data) can range between 13.26 and 26.81 (to two decimal places) while the current solution remains optimal. The ranges for the R.H. Side values are similarly specified.

Microsoft Excel 9.0 Sensitivity Report
Worksheet: [mpdem.xls]Prod
Report Created: 5/6/2002 4:37:33 PM

Adjustable Cells

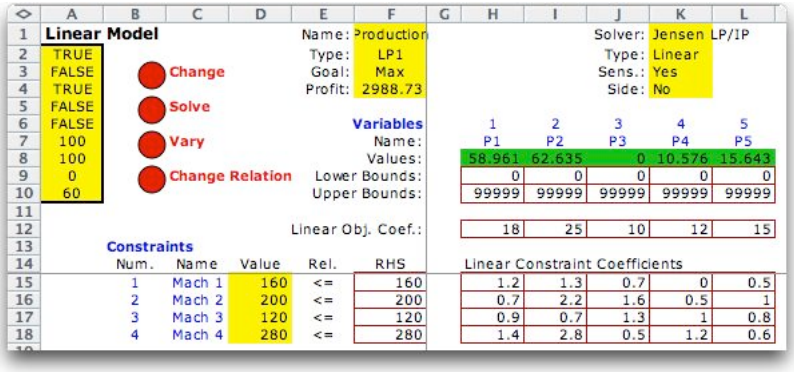
Cell	Name	Final Value	Reduced Cost	Objective Coefficient	Allowable Increase	Allowable Decrease
$H$8	Values: P1	58.96136795	0	18	6.806629834	4.744827586
$I$8	Values: P2	62.63457885	0	25	16.81584158	1.156302521
$J$8	Values: P3	0	-13.53029343	10	13.53029343	1E+30
$K$8	Values: P4	10.57631412	0	12	5.922178988	0.71001032
$L$8	Values: P5	15.64281191	0	15	0.363060686	5.090909091

Constraints

Cell	Name	Final Value	Shadow Price	Constraint R.H. Side	Allowable Increase	Allowable Decrease
$D$15	Mach 1 Value	160	4.819506016	160	12.99610895	60.6514658
$D$16	Mach 2 Value	200	5.201604391	200	30.36363636	15.30991736
$D$17	Mach 3 Value	120	8.963478995	120	117.5049505	18.8700565
$D$18	Mach 4 Value	280	0.363099008	280	15.64116095	20.68111455

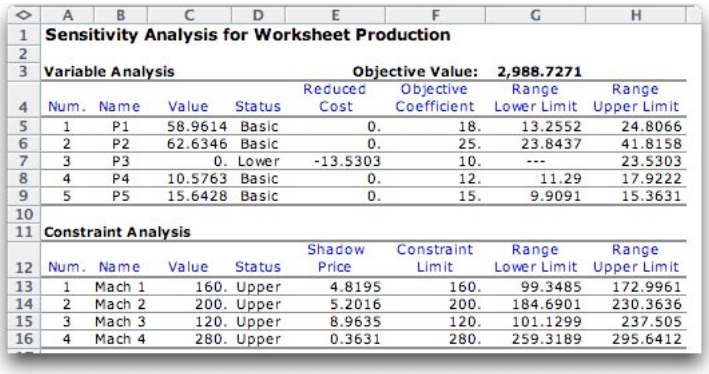
Jensen Solver

Using the Change Button, we have changed the Solver to the Jensen LP/IP Solver. To use this solver the LP/IP Solver add-in (lpip_solver.xla) must be installed. When it is installed the LP/IP Solver item appears on the OR_MM menu. The worksheet for appears almost identical for this solver except the model has been deleted from the range starting in cell A2. The Jensen Solver reads the data directly from the ranges on the worksheet. Of course, one should expect the same solution from both solvers, but they may be different if alternative optimums are present.



	A	B	C	D	E	F	G	H	I	J	K	L
1	**Linear Model**				Name:	Production				Solver:	Jensen LP/IP	
2	TRUE				Type:	LP1				Type:	Linear	
3	FALSE		Change		Goal:	Max				Sens.:	Yes	
4	TRUE				Profit:	2988.73				Side:	No	
5	FALSE		Solve									
6	FALSE					**Variables**		1	2	3	4	5
7	100		Vary			Name:		P1	P2	P3	P4	P5
8	100					Values:		58.961	62.635	0	10.576	15.643
9	0		Change Relation			Lower Bounds:		0	0	0	0	0
10	60					Upper Bounds:		99999	99999	99999	99999	99999
11												
12						Linear Obj. Coef.:		18	25	10	12	15
13		**Constraints**										
14		Num.	Name	Value	Rel.	RHS		Linear Constraint Coefficients				
15		1	Mach 1	160	<=	160		1.2	1.3	0.7	0	0.5
16		2	Mach 2	200	<=	200		0.7	2.2	1.6	0.5	1
17		3	Mach 3	120	<=	120		0.9	0.7	1.3	1	0.8
18		4	Mach 4	280	<=	280		1.4	2.8	0.5	1.2	0.6

The sensitivity analysis given by the Jensen LP/IP Solver is shown below. Some columns have been widened from the original to show more decimal accuracy. In most cases Excel results have many decimals of accuracy that can be observed by widening the columns and choosing the *General* format to display numbers. The Jensen display differs from the Excel display in that lower and upper limits of ranges are shown rather than amounts of increase and decrease.



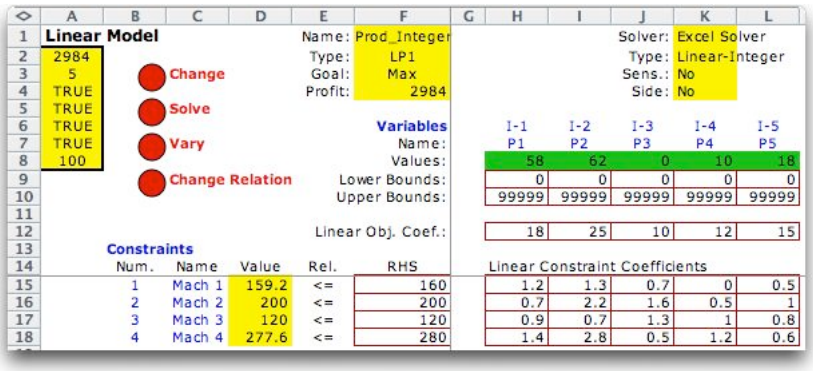
	A	B	C	D	E	F	G	H
1	**Sensitivity Analysis for Worksheet Production**							
2								
3	**Variable Analysis**					**Objective Value:**	**2,988.7271**	
4	Num.	Name	Value	Status	Reduced Cost	Objective Coefficient	Range Lower Limit	Range Upper Limit
5	1	P1	58.9614	Basic	0.	18.	13.2552	24.8066
6	2	P2	62.6346	Basic	0.	25.	23.8437	41.8158
7	3	P3	0.	Lower	-13.5303	10.	---	23.5303
8	4	P4	10.5763	Basic	0.	12.	11.29	17.9222
9	5	P5	15.6428	Basic	0.	15.	9.9091	15.3631
10								
11	**Constraint Analysis**							
12	Num.	Name	Value	Status	Shadow Price	Constraint Limit	Range Lower Limit	Range Upper Limit
13	1	Mach 1	160.	Upper	4.8195	160.	99.3485	172.9961
14	2	Mach 2	200.	Upper	5.2016	200.	184.6901	230.3636
15	3	Mach 3	120.	Upper	8.9635	120.	101.1299	237.505
16	4	Mach 4	280.	Upper	0.3631	280.	259.3189	295.6412

The student can use either the Excel Solver or the Jensen LP/IP Solver to find solutions to linear models. Both can solve problems that include integer variables. The Excel Solver is faster for larger problems and can also solve nonlinear problems. More information about the Excel Solver can be found in the Excel documentation or from the Frontline Systems web site. The Jensen Solver can provide additional information about the algorithmic process used to find optimum solutions. For more information see the LP/IP Solver page.

Integer Variables

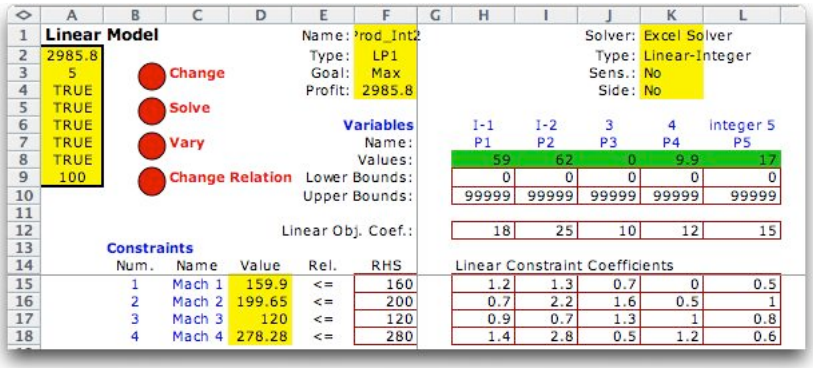
The change button is used to change the structure of the problem by changing the numbers of variables and/or constraints or by changing the integer nature of the variables. The solver option can also be changed.

The following result is obtained when all the variables are required to be integer. Note that the letter I precedes the index of each integer variable. For this case they are all integer.



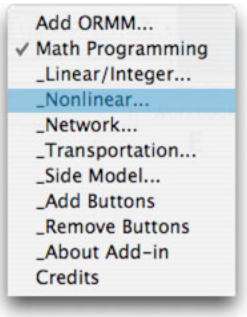
	A	B	C	D	E	F	G	H	I	J	K	L
1	**Linear Model**				Name:	Prod_Integer				Solver:	Excel Solver	
2	2984				Type:	LP1				Type:	Linear-Integer	
3	5	Change			Goal:	Max				Sens.:	No	
4	TRUE				Profit:	2984				Side:	No	
5	TRUE	Solve										
6	TRUE					**Variables**		I-1	I-2	I-3	I-4	I-5
7	TRUE	Vary				Name:		P1	P2	P3	P4	P5
8	100					Values:		58	62	0	10	18
9		Change Relation				Lower Bounds:		0	0	0	0	0
10						Upper Bounds:		99999	99999	99999	99999	99999
11												
12						Linear Obj. Coef.:		18	25	10	12	15
13		**Constraints**										
14		Num.	Name	Value	Rel.	RHS		Linear Constraint Coefficients				
15		1	Mach 1	159.2	<=	160		1.2	1.3	0.7	0	0.5
16		2	Mach 2	200	<=	200		0.7	2.2	1.6	0.5	1
17		3	Mach 3	120	<=	120		0.9	0.7	1.3	1	0.8
18		4	Mach 4	277.6	<=	280		1.4	2.8	0.5	1.2	0.6

The LP/IP dialog has an entry that indicates the number of leading variables that are to be integer. The leading variables are those with the lowest indices. Variables other that the leading ones can be made integer by placing the letter I or any word beginning with I in the cells normally holding the variable indices. The image below shows the solution when the two leading variables are integer and P5 is made integer by placing the word "integer" in its index field.



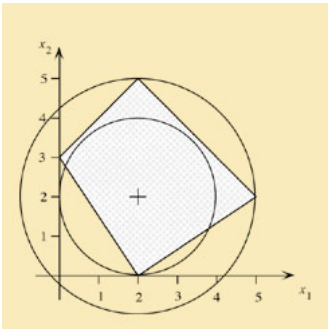
	A	B	C	D	E	F	G	H	I	J	K	L
1	**Linear Model**				Name:	rod_Int2				Solver:	Excel Solver	
2	2985.8				Type:	LP1				Type:	Linear-Integer	
3	5	Change			Goal:	Max				Sens.:	No	
4	TRUE				Profit:	2985.8				Side:	No	
5	TRUE	Solve										
6	TRUE					**Variables**		I-1	I-2	3	4	integer 5
7	TRUE	Vary				Name:		P1	P2	P3	P4	P5
8	TRUE					Values:		59	62	0	9.9	17
9	100	Change Relation				Lower Bounds:		0	0	0	0	0
10						Upper Bounds:		99999	99999	99999	99999	99999
11												
12						Linear Obj. Coef.:		18	25	10	12	15
13		**Constraints**										
14		Num.	Name	Value	Rel.	RHS		Linear Constraint Coefficients				
15		1	Mach 1	159.9	<=	160		1.2	1.3	0.7	0	0.5
16		2	Mach 2	199.65	<=	200		0.7	2.2	1.6	0.5	1
17		3	Mach 3	120	<=	120		0.9	0.7	1.3	1	0.8
18		4	Mach 4	278.28	<=	280		1.4	2.8	0.5	1.2	0.6

Nonlinear Programming



For nonlinear programming models some terms of the objective function or constraints involve nonlinear functions of the decision variables. Although the statement of the nonlinear model is similar to the linear programming model, the process of solving such problems is significantly more complicated.

The figure shows a linearly constrained region and contours of a nonlinear objective function.



The Excel Solver finds solutions for nonlinear programming models, but the nature of the solution obtained usually requires additional analysis. For some problems, the solution can be identified as the global optimum solution, that is, the solution that maximizes or minimizes the objective function over all feasible solutions. For other problems, the solution can only be identified as a local optimum, that is, a solution better than all feasible points in the close neighborhood of the solution. Because of numerical difficulties associated with the procedure, it is even possible that the Solver may fail to find a local optimum. We leave these questions to more advanced texts on nonlinear programming. Here we describe the operation of the add-in when the Nonlinear item on the OR_MM menu is selected.

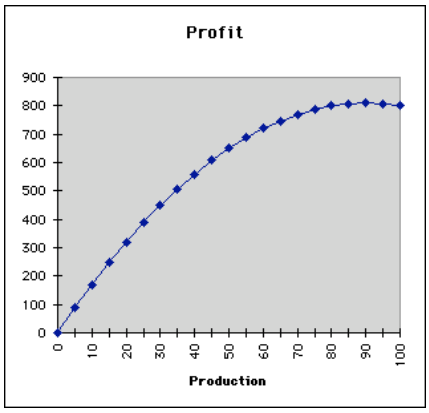
Example Problem - Product Mix with Decreasing Marginal Profit

We modify the product mix example used to illustrate the linear programming add-in to incorporate nonlinear terms in the objective. Management has determined that the marginal profit for each product decreases as the number produced increases. A study determines that the profit for each product is approximated by the quadratic function given below. The table shows the coefficients for each product. With the same constraints on machine time, our goal is to find the product mix that maximizes profit.

Profit for product i with production p_i: $c_i p_i - d_i p_i^2$

Product	1	2	3	4	5
Linear Coefficient	18	25	10	12	15
Nonlinear Coefficient	-0.1	-0.15	-0.01	-0.01	-0.04

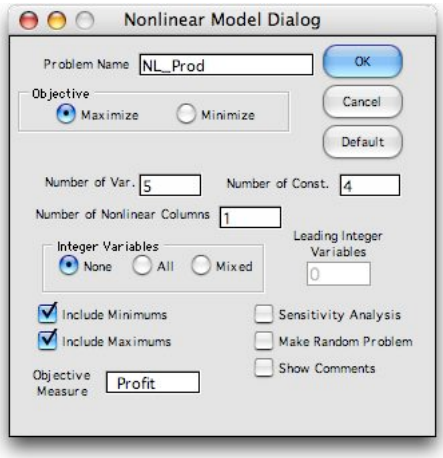
The graph below shows the profit for the first product as a function of production. The function starts at zero, rises to a maximum and then decreases. This is a concave function of production. The marginal profit, or the derivative of the profit function, is decreasing with production volume.



The mathematical programming model is the same as in the linear programming example except the objective function is now the sum of nonlinear terms. This is called a *separable* objective function because each term is a function of only one decision variable.

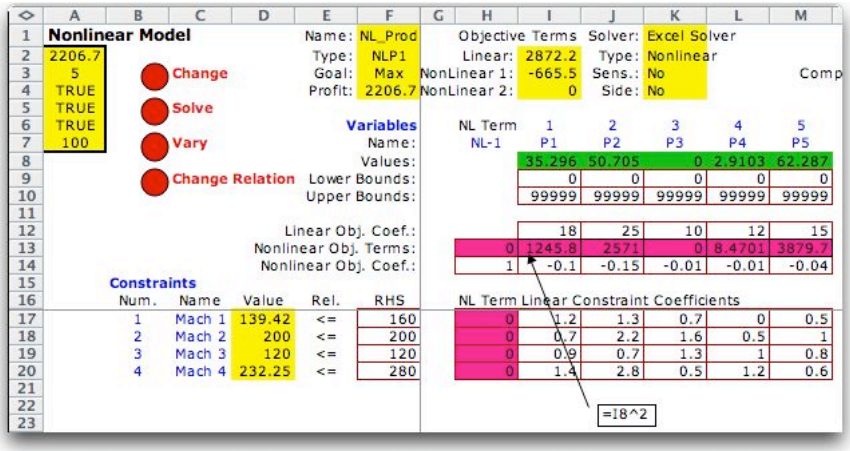
$$Z = \text{Maximize} \sum_{i=1}^{5} \left(c_i p_i - d_i p_i^2 \right)$$

Excel Model



To model this problem with the Mathematical Programming Add-in, select Nonlinear from the OR_MM menu. The *Nonlinear Model Dialog* presents the same options as the *Linear Model Dialog* except that it also asks for the number of extra columns. Although not necessary for this problem, we enter 1 as the number of extra columns to illustrate the effect. The remainder of the options on this dialog should be clear from the discussion of the *Linear Model Dialog*. There are no solver options because only the Excel Solver can handle nonlinear models. Although the Excel Solver does solve problems with integer variables, great care must be exercised when interpreting these answers. Theoretically, all nonlinear solutions should be judged carefully with respect to optimality. Problems may have multiple local optimums, and simple algorithms may reach one of these rather than the global optimum.

The worksheet obtained after data entry and solution is below. The solution was found using the Excel Solver program.



As for linear models, the linear objective coefficient row (row 12) models the linear terms of the objective. These coefficients are multiplied by the vector of variable values by the Excel formula in cell I2. The cell is labeled Linear to indicate that this value is the linear part of the objective function.

The worksheet has two additional rows in the portion of the model describing the variables, one row for nonlinear objective coefficients and the other for nonlinear terms. The space for nonlinear objective coefficients in row 14 is useful for nonlinear terms that have only a single coefficient. In this case it holds the values of $-d$. The equations for the quadratic terms are entered into row 13. We color the cells in this row from column H through M magenta. This color indicates that the student is to enter Excel formulas in these cells. For example, the equation in I13 is to be the square of the value of P1. That value is in I8, so the formula in I13 is.

=I8^2

This describes the nonlinear effect of the variable in P1 on profit. The rest of the terms in row 13 are similar, but each nonlinear term refers to the variable above it. For example J13 holds the formula (=J8^2). The terms from J13 through M13 are easily copied form I13 using the Excel *Fill Right* command.

The nonlinear coefficients multiply the nonlinear terms and are summed to obtain the entry called the *Nonlinear 1* component of the objective function. The formula for this contribution is in cell I3. In general, row 13 may hold any nonlinear, differentiable functions of the variable values. It is important that the functions be well defined for all values of the variables. For example the function (=1/I13) would fail if I13 were allowed to take on the value 0.

Initially all nonlinear terms are set to zero. Both the nonlinear expressions in row 13 and the nonlinear coefficients in row 14 must be nonzero for the nonlinear terms to affect the objective function.

A column (H) has been created to hold other nonlinear terms that may appear in the objective function and constraints. Although not used for this example, these cells can hold any continuous, differentiable functions of the decision variables. Where problems have several identifiable sources of nonlinear terms, it may be convenient to have more than one column to hold these terms. The nonlinear terms in the extra columns are summed and are shown as the *Nonlinear 2* component of the objective function in cell I4. The *Linear*, *Nonlinear 1*, and *Nonlinear 2* values are summed in cell F4 to obtain the objective function. This quantity is optimized by the Solver.

Nonlinear terms in the constraints can be entered as explicit functions of the decision variable in the range H17:H20. The nonlinear function values are accumulated with the linear constraint functions in the Value column of the constraints.

Solving the Problem

Given a starting vector for the decision variables, the Excel Solver program uses a search procedure to move to a local optimum. That is, a point in the feasible region such that no improvement can be made by small perturbations from the point. It is a known result that when maximizing a concave objective function with linear constraints every local optimum is a global optimum, the solution that is the best of all feasible solutions. Since the objective function terms for this example are concave, when the Solver program terminates at a point with an indication of optimality, the point should be the global optimum.

In general, we cannot be sure that the Excel Solver or any nonlinear programming algorithm will terminate at an optimal point unless some rather stringent conditions are satisfied. These conditions are the subject of nonlinear programming textbooks. For example if the objective function terms were convex, rather than concave, the algorithm will always terminate at an extreme point of the feasible region (when the region is defined by linear constraints). In many cases the extreme points will not be globally optimum solutions. The situation is even further complicated for integer-nonlinear models. An extensive coverage of these features would take many pages, but the user should beware of the problem of assuring optimality for nonlinear models.

There are limitations on the functions that can appear in nonlinear terms. The functions should be defined, continuous and differentiable in and near the feasible region. This eliminates the use of IF, integer, and absolute value functions in the model. Ratios with denominators that might go to zero or square root terms that have negative arguments are similarly not allowed. These restrictions must be obeyed at every point that the Solver program might try to evaluate. Even if the initial solution is feasible, the Solver program might make small incursions outside the feasible region during the search process. If the program encounters a point for which a function cannot be evaluated it stops with an error message.

The problem of finding the global optimum from perhaps a large set of local optima is been the subject of much theoretical as well as practical effort. This field of research and practice is called *Global Optimization*. *Frontline Systems*, the distributor of the Excel Solver has premium systems that implement highly advanced methods for finding global optimum solutions.

Microsoft Excel 9.0 Sensitivity Report
Worksheet: [mp.xls]NLProd
Report Created: 1/18/2001 3:42:30 PM

Adjustable Cells

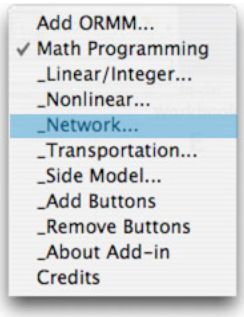
Cell	Name	Final Value	Reduced Gradient
$I$8	Values: P1	35.2961412	0
$J$8	Values: P2	50.70462038	0
$K$8	Values: P3	0	-6.258322317
$L$8	Values: P4	2.910349303	0
$M$8	Values: P5	62.28736164	0

Constraints

Cell	Name	Final Value	Lagrange Multiplier
$D$17	M1 Value	139.4150568	0
$D$18	M2 Value	200	0.772622704
$D$19	M3 Value	120	11.55548114
$D$20	M4 Value	232.2523709	0

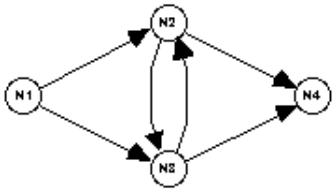
The sensitivity analysis provided by the Excel Solver for nonlinear programming problems has a slightly different format than for linear programming problems as illustrated. The reduced gradient for a variable is the partial derivative of the objective function with respect to that variable, evaluated at the stopping point of the algorithm. The Lagrange multipliers are similar to the shadow prices associated with a linear problem.

Network Flow Programming



This important class of linear programming models has the advantage that elements of a problem can be described by a picture rather than a series of algebraic expressions as for the general linear programming model. Even if a particular problem cannot entirely be expressed as a network flow problem, very often major components of the problem can be expressed as a network. For an "almost" network model, the problem can be described using this model construct, and the worksheet or the Solver model modified to incorporate nonstandard features.

A network is a collection of nodes and arcs. Flow enters and leaves at the nodes and passes through the arcs. Generally there are lower and upper bounds on arc flows. Each arc has a unit cost and the arc cost is the flow multiplied by the unit cost. The goal is to find the flow that minimizes total cost.



Each arc has a gain factor that multiplies the flow entering the arc to obtain the flow that leaves the arc. A network with all gains equal to 1 is called a pure network. If some gains are other than 1, the network is called a generalized network.

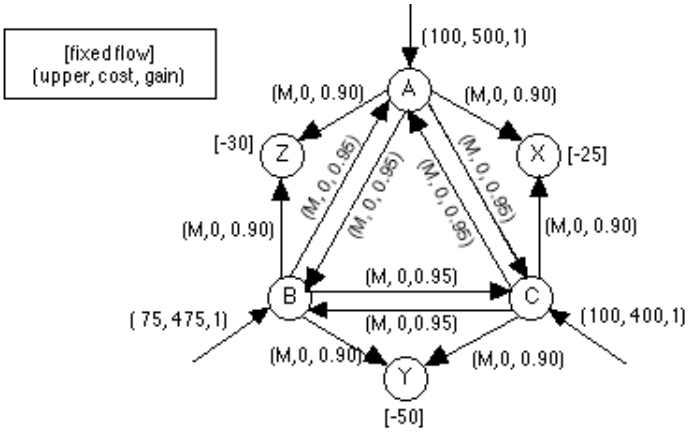
Example Problem - Power Distribution

Consider a regional power system with three generating stations: A, B, and C. Each station serves its own local area. Three outlying areas are also served by the system: X, Y, and Z. The power demanded at areas X, Y, and Z is 25 MW (megawatts), 50 MW, and 30 MW, respectively. The maximum generating capacity beyond local requirements and the cost of generation at the three stations is shown in the table.

Power can be transmitted between any pair of generating stations, but 5% of the amount is lost. Power can be transmitted from some of the generating stations to the outlying areas, but 10% of the amount is lost. Lines exist from stations A and C to X, from B and C to Y, and from A and B to Z. Our goal is find the minimum cost power distribution plan between generating stations and to outlying areas.

Generating Station	*Excess capacity (MW)*	*Cost of generation per MW ($)*
A	100	500
B	75	475
C	100	400

The network model for this problem consists of nodes and arcs as shown in the figure. The nodes are the circles and represent the generating stations and cities. The arcs are the directed line segments between nodes. They represent the transmission lines between generating stations and/or cities.

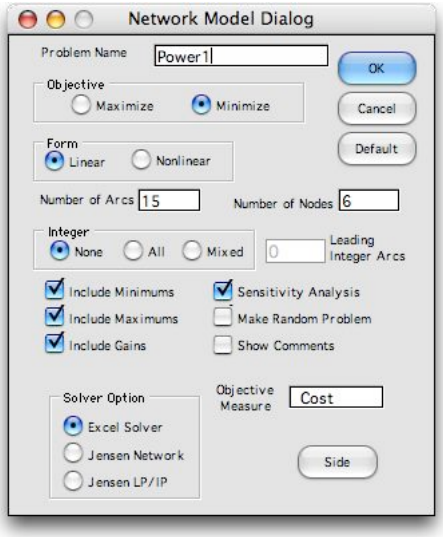


Network Flow Model of Power Distribution Problem

The numbers adjacent to the nodes in the square brackets represent flows entering the network or flows leaving the network, positive numbers for flows entering and negative numbers for flows leaving. Power enters at the generation stations, nodes A, B, and C through arcs that enter these nodes. Power leaves the network at nodes X, Y, and Z, where the negative numbers at the nodes indicates the amounts to be withdrawn at the nodes. The arcs for this case have three parameters, the upper bound, cost and gain. The upper bound for each transmission arc is M, indicating a large number. For an arc that passes from node i to node j, the gain multiplies the flow leaving node i to obtain the flow that enters node j. For this problem the gain factors represent the losses of power in the transmission lines.

A solution to the network model is an assignment of flows to the arcs that satisfies the flow requirements at the nodes. Flow is conserved at each node in that the total flow entering must equal the total flow leaving. An arc that touches only one node, such as those entering A, B and C, contributes only to the conservation equation for that node. The optimum solution minimizes cost.

Constructing the Network

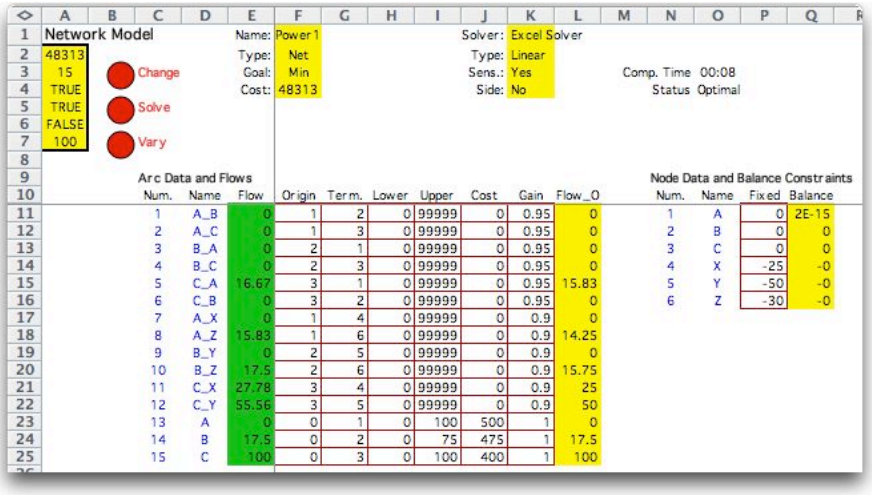


To enter the model in the form of a network, select Network from the OR_MM menu. The Network Model Dialog allows the specification of a name, indication of maximization or minimization for the optimization, indication of whether the problem is linear or nonlinear, specification of the numbers of arcs and nodes and identification of integer flows on the arcs. Data is entered for the example problem.

The checkboxes near the middle of the dialog determine the set of parameters that will appear on the worksheet. We have unchecked the *Include Minimum* box because all the arcs have 0 arc lower bounds. When data for minimum flows are not shown, the 0 lower bounds are assumed. With the *Include Maximums* box checked, a column will be included for the maximum values of flows. If it were not checked, the default is that arc flows are unbounded from above. With the *Include Gains* box checked a column will be included for arc gains. The default assumption is that arc gains are all 1. The *Sensitivity Analysis* box tells the selected solver to perform a sensitivity analysis on the optimum solution. The *Make Random Problem* box generates a random network. This is convenient for demonstrating the network modeling and solving options.

We see three solver options at the bottom of the dialog: Excel Solver, Jensen Network Solver and the Jensen LP Solver. Any one of these can solve the linear network flow problem with or without integer requirements. The Excel Solver and the Jensen Network Solver can solve nonlinear problems. The Jensen Network Solver does not provide sensitivity analyses.

The worksheet showing the model for the Power problem is shown below. The data has been entered for the arc and node parameters and after the problem has been solved. Column D holds arc names. In this case we have chosen to name each arc by the node names at its ends. Columns F through J hold the arc parameters. Columns F and G contain the indices of the nodes that originate and terminate the arcs. Columns H through J hold the remaining parameters of the arcs. Lower bounds on arc flows are assumed to be zero since that column was not included in the display. The Upper column specifies the upper bound. The transmission arcs for this case have bounds of 99999, the large upper bound not restricting the flow. The Cost column contains the unit cost of arc flow, and the Gain column holds the multipliers for arc flows. The variables for this problem are contained in the Flow column, column E. The flow in an arc is the flow leaving the origin node of the arc. The numbers shown in the illustration are the optimum values obtained by the Solver program. The column labeled Flow_O, column K, contains the flows entering the terminal nodes of the arcs. This is provided by an equation that multiplies the flow and the gain. The numbers in this column are used elsewhere on the worksheet and the formulas should not be changed.



1	Network Model				Name:	Power1				Solver:	Excel Solver									
2	48313				Type:	Net				Type:	Linear									
3	15	Change			Goal:	Min				Sens.:	Yes			Comp. Time	00:08					
4	TRUE				Cost:	48313				Side:	No			Status	Optimal					
5	TRUE	Solve																		
6	FALSE																			
7	100	Vary																		
8																				
9			Arc Data and Flows													Node Data and Balance Constraints				
10			Num.	Name	Flow	Origin	Term.	Lower	Upper	Cost	Gain	Flow_O			Num.	Name	Fixed	Balance		
11			1	A_B	0	1	2	0	99999	0	0.95	0			1	A	0	2E-15		
12			2	A_C	0	1	3	0	99999	0	0.95	0			2	B	0	0		
13			3	B_A	0	2	1	0	99999	0	0.95	0			3	C	0	0		
14			4	B_C	0	2	3	0	99999	0	0.95	0			4	X	-25	-0		
15			5	C_A	16.67	3	1	0	99999	0	0.95	15.83			5	Y	-50	-0		
16			6	C_B	0	3	2	0	99999	0	0.95	0			6	Z	-30	-0		
17			7	A_X	0	1	4	0	99999	0	0.9	0								
18			8	A_Z	15.83	1	6	0	99999	0	0.9	14.25								
19			9	B_Y	0	2	5	0	99999	0	0.9	0								
20			10	B_Z	17.5	2	6	0	99999	0	0.9	15.75								
21			11	C_X	27.78	3	4	0	99999	0	0.9	25								
22			12	C_Y	55.56	3	5	0	99999	0	0.9	50								
23			13	A	0	0	1	0	100	500	1	0								
24			14	B	17.5	0	2	0	75	475	1	17.5								
25			15	C	100	0	3	0	100	400	1	100								

Notice that arcs 13, 14 and 15 do not have origin nodes in the figure defining the problem. Rather, these arcs bring flow into the network at the nodes representing the power plants. This is shown in column F by specifying 0 as the origin nodes of these arcs. Similarly, an arc that starts at some network node and leaves the network has 0 as its terminal node.

Information associated with the nodes is shown in the portion of the worksheet starting in column M. Node indices appear in column M, and names are provided by the user in column N. Fixed flows entering or leaving the node are in Column O. A positive number in this column is a flow that enters at the node, while a negative number is a flow that leaves. In the example, the negative numbers for nodes X, Y and Z are the power demands at the cities.

Column P of the node display, Balance, holds the equations that define conservation of flow for each node. Conservation of flow requires that total flow out of each node equal the total flow in. Formulas in these cells compute the net flow in or out of the cells. Feasible solutions require that the numbers in the Balance column all be zero. The solution shown here has very small values for the balance equations indicating that the solution is feasible within the numerical accuracy associated with the Solver program.

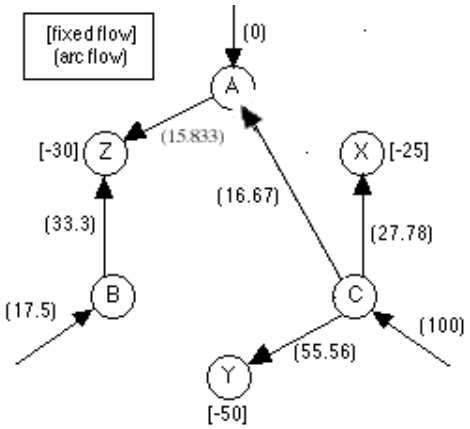
Solving the Problem

With the Excel Solver, clicking the Solve button on the worksheet loads the model into the

Solver and initiates the solution algorithm. The optimum solution is placed in column D of the arc display.

With the Jensen Network Solver or the Jensen LP Solver option, clicking the Solve button causes the data to be read by the chosen program and the initiation of the solution algorithm. Again, the optimum values of the flow placed in column D. Both Jensen solvers can be used when all or some of the arcs are required to have integer flows.

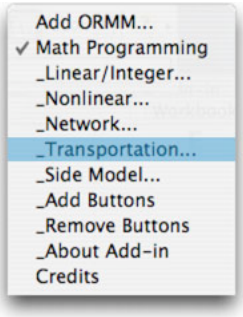
The optimum solution obtained by the Solver is shown graphically in the figure below. The optimum plan has generators B and C providing all the power with the plant C at its maximum capacity. Generator A provides no power the optimum solution. Flows on the transmission lines are indicated on the arcs.



Optimum flows for power distribution problem, Z = 48,313.

Transportation Model

The transportation model is a special case of the linear programming model. The Mathematical Programming Add-in allows the specification of models having this form with several variations.



The figure shows a classic transportation model expressed in the matrix format. Three suppliers with 15 units each of some commodity must ship to three demanders with specified demand. The numbers in the table are the unit shipping cost. The goal is to find a distribution schedule that minimizes the total cost of shipments.

	D1	D2	D3	*Supply*
S1	9	12	10	15
S2	8	15	12	15
S3	13	17	19	15
Demand	10	20	15	

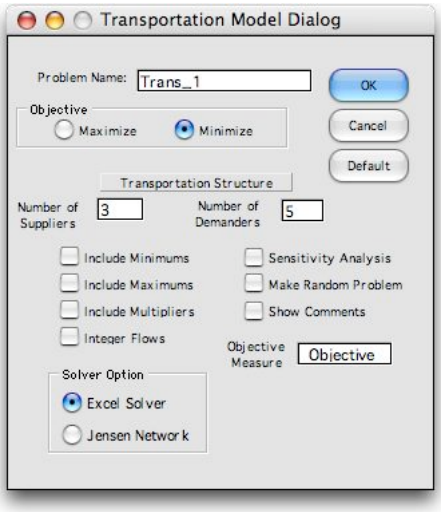
Example - Product Distribution

A manufacturer has three warehouses and five customers. Each warehouse has fifteen units of product available and each customer has a specified demand for the product. The cost of shipping from each warehouse to each customer is known and it is desired to find a shipping schedule that minimizes total shipping cost. Data concerning unit shipping cost and customer demand are given in the table.

Shipping Costs per Unit

	Customer				
Warehouse	D1	D2	D3	D4	D5
S1	15	15	16	11	11
S2	13	11	15	9	6
S3	8	12	11	7	8

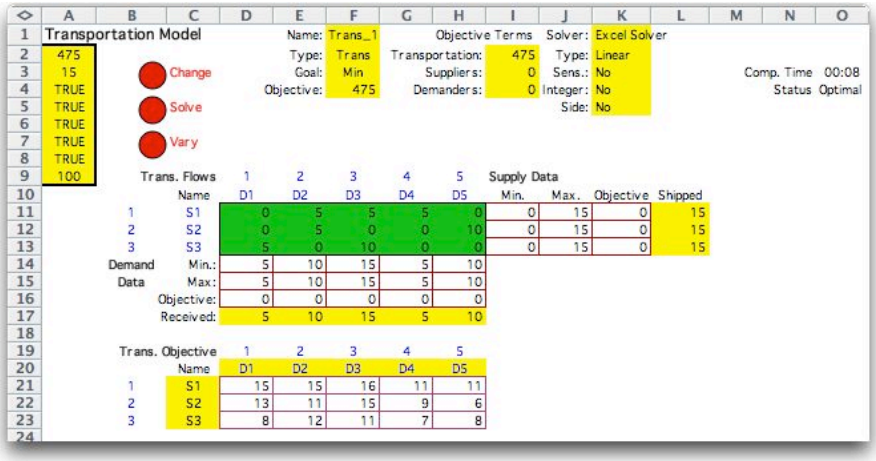
	Customer Demand				
Customer	D1	D2	D3	D4	D5
Demand	5	10	15	5	10



Constructing the Worksheet

Selecting Transportation from the OR_MM menu presents the Transportation Model Dialog.

With the data from the dialog, the add-in constructs a model with the number of suppliers and demanders specified by the example. The dialog provides an opportunity to select the worksheet name, numbers of suppliers and demanders and direction of optimization. The checkboxes allow several nonstandard features of the model that are illustrated later. The worksheet for the example problem after parameters have been entered and after solution by the Solver program is shown below.



The Add-in provides ranges for names of suppliers (C9:C11) and Demanders (D8:H8). The table contained in the range D9:H11 holds the transportation flows. These are variables of the problem and will be determined by the Solver. The row labeled Received (D15:H15) contains a sum formula that computes the total flows received by the demanders. These numbers are used in the constraints and the expressions should not be disturbed. Similarly the column labeled Shipped (L9:L12), contains the total flows shipped by the suppliers and the expressions should not be disturbed.

The rows labeled Min Received, Max Received and Revenue hold parameters associated with the demanders. The first two rows place lower and upper bounds on the amounts received by each demander. Since in the present case these numbers are the same, the model is forcing all demands to be met. The row labeled Revenue specifies the unit revenue for each demander. This number is irrelevant for the example, and we have set the row to zero. The Columns labeled Min Ship, Max Ship and Unit Cost hold parameters for the suppliers. We have specified the maximum shipments as the amounts available at each supplier. We have not constrained the minimum shipments. The unit cost column is provided for cases in which different suppliers may have different costs for the products.

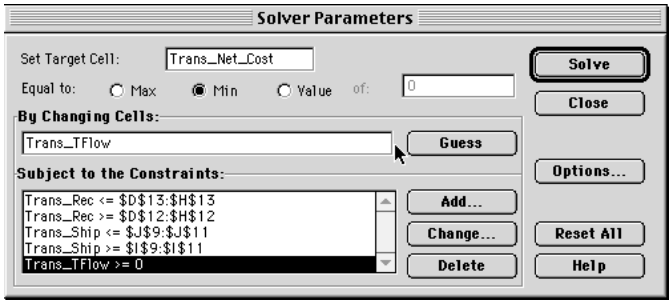
The shipping costs are given in the second table of the model in the range D19:H21. The names heading this table are provided by formulae. Any names defined for the flow table will be automatically transferred to this and other tables.

Problems may be solved with either the Jensen Network Solver or the Excel Solver. Sensitivity analysis is only provided by the Excel Solver. Clicking the Solve button initiates the solution algorithm. The optimum flows are placed in the flow array.

Solving the Problem

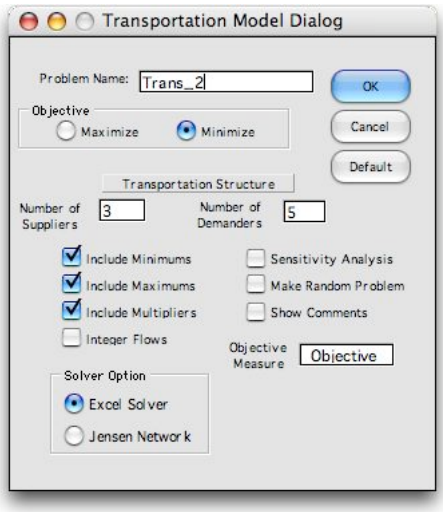
If the Jensen Network Solver is selected the problem is solved automatically when the Solve button is pressed. Similarly, the Excel Solver is automatically called if the Automatic option has been chosen.

With the manual option, selecting the Solve button and following the instructions will load the linear programming model for the transportation problem for the Solver. The dialog below shows the model after it is loaded. Pressing the Solve button in the Solver dialog initiates the solution procedure. The optimum solution is shown in the Transportation Flow array.

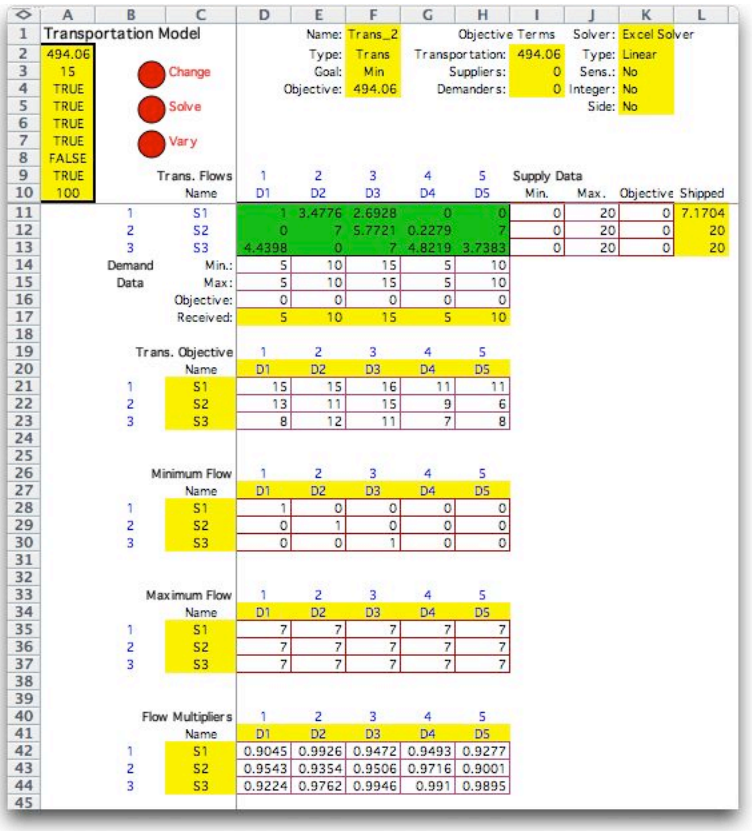


Model Variations

The Transportation Model Dialog allows several variations on the simple transportation model. The checkboxes *Include Minimums*, *Include Maximums* and *Include Multipliers* cause the Add-in to build tables that allow the entry of minimum transportation flows, maximum transportation flows, and arc gain factors respectively. The *Integer Flows* box, adds the requirement that the solution maintain integer flows.



The matrices for the example with arbitrary values of lower bounds, upper bounds and gains are shown below. Lower bounds represent minimum shipments along certain routes, while upper bounds represent shipping capacities. When the gain factors are other than one, the amount shipped on a link is not the same as the amount received. The multipliers shown below represent a loss, perhaps due to spoilage.



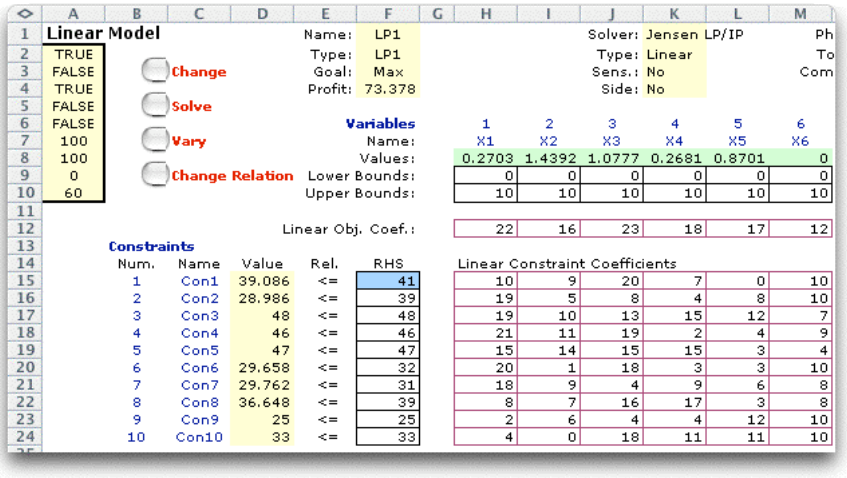
Transportation Model												
				Name:	Trans_2		Objective Terms		Solver:	Excel Solver		
494.06				Type:	Trans		Transportation:	494.06	Type:	Linear		
15	Change			Goal:	Min		Suppliers:	0	Sens.:	No		
TRUE	Solve			Objective:	494.06		Demanders:	0	Integer:	No		
TRUE									Side:	No		
TRUE	Vary											
FALSE												
TRUE		Trans. Flows		1	2	3	4	5	Supply Data			
100		Name		D1	D2	D3	D4	D5	Min.	Max.	Objective	Shipped
	1	S1		1	3.4776	2.6928	0	0	0	20	0	7.1704
	2	S2		0	7	5.7721	0.2279	7	0	20	0	20
	3	S3		4.4398	0	7	4.8219	3.7383	0	20	0	20
	Demand	Min.:		5	10	15	5	10				
	Data	Max:		5	10	15	5	10				
		Objective:		0	0	0	0	0				
		Received:		5	10	15	5	10				
		Trans. Objective		1	2	3	4	5				
		Name		D1	D2	D3	D4	D5				
	1	S1		15	15	16	11	11				
	2	S2		13	11	15	9	6				
	3	S3		8	12	11	7	8				
		Minimum Flow		1	2	3	4	5				
		Name		D1	D2	D3	D4	D5				
	1	S1		1	0	0	0	0				
	2	S2		0	1	0	0	0				
	3	S3		0	0	1	0	0				
		Maximum Flow		1	2	3	4	5				
		Name		D1	D2	D3	D4	D5				
	1	S1		7	7	7	7	7				
	2	S2		7	7	7	7	7				
	3	S3		7	7	7	7	7				
		Flow Multipliers		1	2	3	4	5				
		Name		D1	D2	D3	D4	D5				
	1	S1		0.9045	0.9926	0.9472	0.9493	0.9277				
	2	S2		0.9543	0.9354	0.9506	0.9716	0.9001				
	3	S3		0.9224	0.9762	0.9946	0.991	0.9895				

When the shipping and receiving bounds are integer and the flow multipliers are all 1, we can be sure that the solution to the transportation problem will be integer. When this is not true, however, the flow solution may not be integer. The flow solution shown in the green area is certainly not integer. The Integer check box is provided to add the constraint that the flows be integer. The integer problem is much more challenging for the computer than when integrality is not required. The Excel Solver require several minutes to determine that this problem has no feasible integer solution, while the solution algorithm only requires a second or two when integrality is not required.

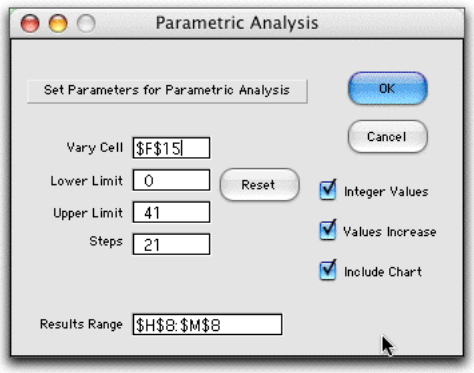
Parametric Analysis

This section describes a recent addition to the Math Programming add-in. To obtain this feature, you must download a version of the add-in dated 6/14/04 or later.

Parametric analysis may be applied to any of the models constructed by the math programming add-in. A linear programming model with six variables and ten constraints is shown below. The data is random. The parametric analysis is performed by clicking the *Vary* button on the worksheet. Note that cell F15 is colored blue. For this example we vary the contents of this cell.



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	**Linear Model**				Name:	LP1				Solver:	Jensen LP/IP		Ph
2	TRUE				Type:	LP1				Type:	Linear		To
3	FALSE		Change		Goal:	Max				Sens.:	No		Com
4	TRUE				Profit:	73.378				Side:	No		
5	FALSE		Solve										
6	FALSE				**Variables**			1	2	3	4	5	6
7	100		Vary		Name:			X1	X2	X3	X4	X5	X6
8	100				Values:			0.2703	1.4392	1.0777	0.2681	0.8701	0
9	0		Change Relation		Lower Bounds:			0	0	0	0	0	0
10	60				Upper Bounds:			10	10	10	10	10	10
11													
12					Linear Obj. Coef.:			22	16	23	18	17	12
13		**Constraints**											
14		Num.	Name	Value	Rel.	RHS		Linear Constraint Coefficients					
15		1	Con1	39.086	<=	41		10	9	20	7	0	10
16		2	Con2	28.986	<=	39		19	5	8	4	8	10
17		3	Con3	48	<=	48		19	10	13	15	12	7
18		4	Con4	46	<=	46		21	11	19	2	4	9
19		5	Con5	47	<=	47		15	14	15	15	3	4
20		6	Con6	29.658	<=	32		20	1	18	3	3	10
21		7	Con7	29.762	<=	31		18	9	4	9	6	8
22		8	Con8	36.648	<=	39		8	7	16	17	3	8
23		9	Con9	25	<=	25		2	6	4	4	12	10
24		10	Con10	33	<=	33		4	0	18	11	11	10



A dialog is presented for entering the data concerning the analysis. The *Vary Cell* is the address of a single cell on the worksheet. For the example we choose cell F15, however any cell whose value is relevant to the solution may be chosen. The example is the right-side value for the first constraint. The dollar signs in the cell reference are not necessary.

The next two fields hold the *Lower* and *Upper* limits on the range over which the parameter will vary. The *Steps* entry indicates the number of intervals into which the range will be divided. The *Results Range* specifies a range on the worksheet that will change with the parameter. The default value is the solution range, however, the range may be changed to show other values of interest. The range must be a single row or column of cells.

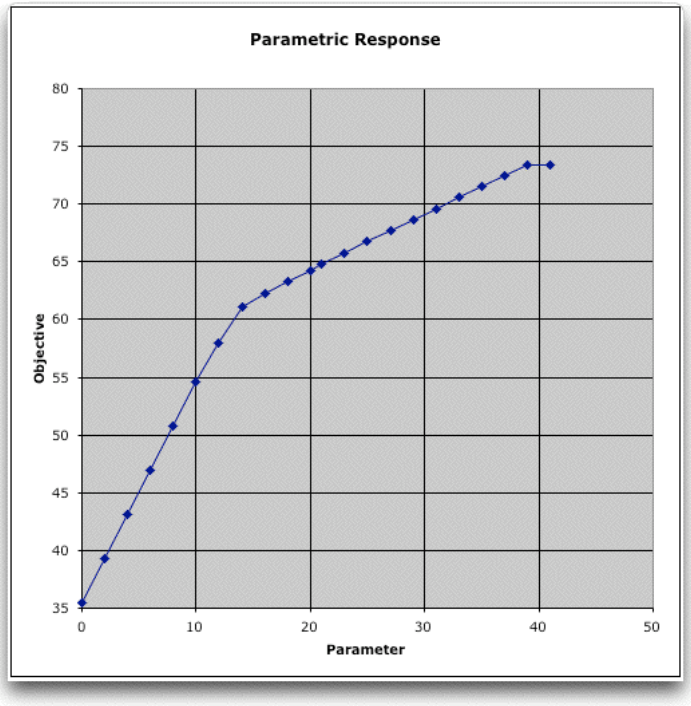
The check boxes at the right indicate respectively whether the parameter is to be restricted to integer values, whether the values will increase or decrease as the parameter is varied, and whether the add-in is to construct a chart of the objective function value as a function of the parameter.

The table below shows the results obtained when the right-side of the first constraint is varied. The first column (E) holds the parameter values. Note that the differences between adjacent values are not equal. This is due to the rounding necessary to obtain integer values. The second column (F) gives the objective function values for each parameter setting. The remaining columns (H through M) give the solution values. Nonzero values are highlighted in blue. For the example, we can clearly see that the linear programming basis changes as the parameter increases.

	D	E	F	G	H	I	J	K	L	M
26										
27		Parametric Analysis			Vary: $F$15	Results: $H$8:$M$8				
28		Param.	Obj.		X1	X2	X3	X4	X5	X6
29		0	35.417		0	0	0	0	2.0833	0
30		2	39.25		0.2	0	0	0	2.05	0
31		4	43.083		0.4	0	0	0	2.0167	0
32		6	46.917		0.6	0	0	0	1.9833	0
33		8	50.75		0.8	0	0	0	1.95	0
34		10	54.583		1	0	0	0	1.9167	0
35		12	57.997		0.9286	0	0	0.3878	1.7993	0
36		14	61.078		0.6429	0	0	1.0816	1.6156	0
37		16	62.265		0.707	0	0.1361	0.8869	1.6245	0
38		18	63.276		0.7346	0.0696	0.2463	0.7288	1.601	0
39		20	64.272		0.7433	0.1611	0.3455	0.6009	1.5634	0
40		21	64.77		0.7476	0.2069	0.3952	0.5369	1.5446	0
41		23	65.765		0.7564	0.2984	0.4944	0.409	1.507	0
42		25	66.761		0.7651	0.3899	0.5937	0.281	1.4693	0
43		27	67.72		0.7229	0.5185	0.6732	0.2344	1.4011	0
44		29	68.668		0.6646	0.6587	0.7466	0.2134	1.3232	0
45		31	69.616		0.6064	0.7989	0.82	0.1923	1.2454	0
46		33	70.564		0.5482	0.9391	0.8934	0.1713	1.1675	0
47		35	71.512		0.49	1.0793	0.9667	0.1502	1.0897	0
48		37	72.44		0.4036	1.24	1.0292	0.1741	0.9949	0
49		39	73.339		0.2758	1.4309	1.0757	0.2642	0.8753	0
50		41	73.378		0.2703	1.4392	1.0777	0.2681	0.8701	0
51										

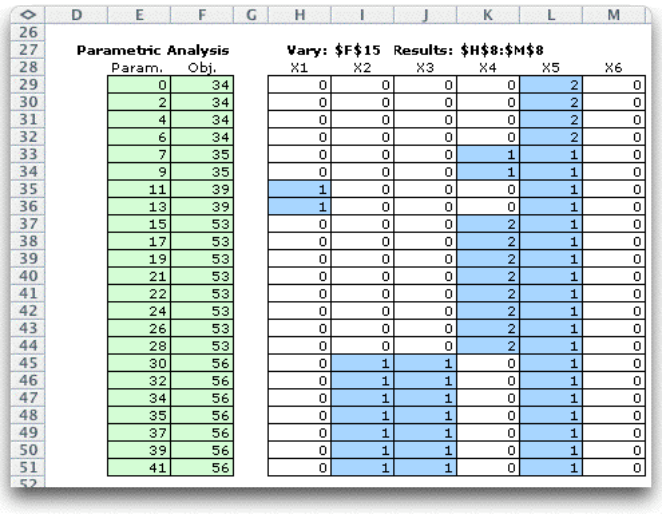
In the example we find a feasible solution for all parameter values within the range, but for nonlinear cases it is possible that there may be no feasible solution for some values of the parameter. Starting from the initial value (0 in this case), the program will solve the model. If no feasible solution is found, but the parameter increases until a feasible solution is found. Once a feasible solution is found, the parameter continues to change until an infeasible solution is again found. Then the process stops. Feasible solutions are shown, but values of the parameter that find no feasible solution are skipped.

A chart showing the objective value as a function of the parameter value is constructed below the table. It should not be surprising that the curve is convex with respect to the parameter value.



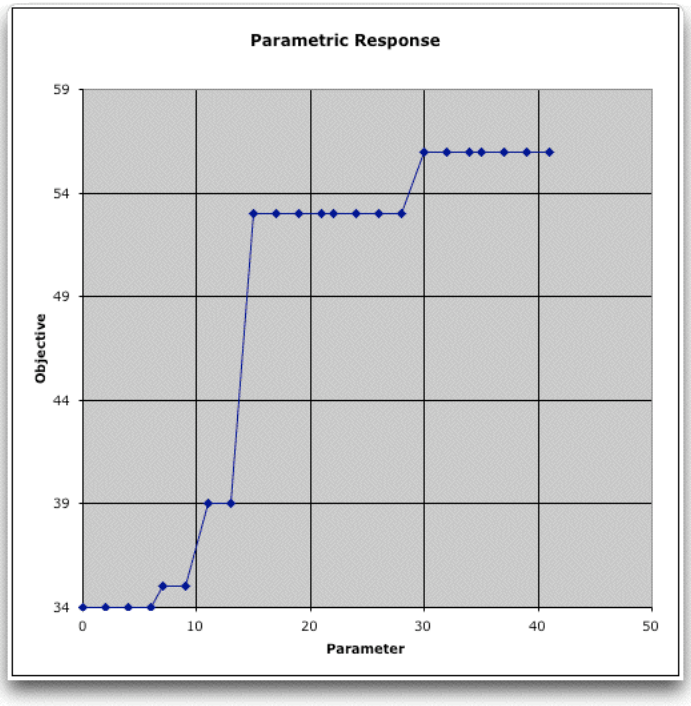
Integer Programming

We solved the same model after requiring that all variables have integer values. Now a branch and bound procedure must be used to find the solution for each parameter value. Of course the process takes quite a bit longer. The tabular results are below.



Parametric Analysis		Vary: $F$15	Results: $H$8:$M$8					
Param.	Obj.	X1	X2	X3	X4	X5	X6	
0	34	0	0	0	0	2	0	
2	34	0	0	0	0	2	0	
4	34	0	0	0	0	2	0	
6	34	0	0	0	0	2	0	
7	35	0	0	0	1	1	0	
9	35	0	0	0	1	1	0	
11	39	1	0	0	0	1	0	
13	39	1	0	0	0	1	0	
15	53	0	0	0	2	1	0	
17	53	0	0	0	2	1	0	
19	53	0	0	0	2	1	0	
21	53	0	0	0	2	1	0	
22	53	0	0	0	2	1	0	
24	53	0	0	0	2	1	0	
26	53	0	0	0	2	1	0	
28	53	0	0	0	2	1	0	
30	56	0	1	1	0	1	0	
32	56	0	1	1	0	1	0	
34	56	0	1	1	0	1	0	
35	56	0	1	1	0	1	0	
37	56	0	1	1	0	1	0	
39	56	0	1	1	0	1	0	
41	56	0	1	1	0	1	0	

The objective function chart shows the steps in the objective value caused by the discrete nature of the solutions.



Other Models

Parametric analysis is available for the other math programming models. For nonlinear models the Excel Solver must be used. For network and transportation models the Jensen Network Solver may be used. For network and transportation models the table of results and the corresponding chart are to the right of the data.

Nonlinear models may have trouble converging to a solution at every parameter value. When the first try at finding a feasible solution fails the program randomly generates a second initial solution. Hopefully the second try will find a feasible solution if one exists.

The parametric feature of the add-in is very general and can be used for a number of interesting problems. One possibility is to place the varying cell outside the model. Then several

model features may be linked by Excel formula to the varying cell. The parametric analysis will affect many features of the model simultaneously.

The results range may too lay outside the model. For example perhaps only a few of the variables are of interest in a very large model. A row or column region of the worksheet can hold equations that link to the cells holding the important variables. Then, only these are shown in the results table. An example would be the integer variables in a large and complex mixed integer programming model.

If the *Results Range* box of the dialog is left empty, only the parameter and objective values are shown in the table.

Side Model

The form of the math programming model provided by the *Math Programming* add-in is not very efficient for a problem with a large set of variables where each variable in the set appears in one or a very few constraints. For this kind of problem, we provide a second model structure that is placed on the same worksheet as the original model structure. Traditional math programming gives the name *side constraints* to constraints that are *in addition* to those of a master model structure. In this spirit we call the model described on this page the *side model*. The side model can identify new variables and new constraints.

The side model can only be used if one of the other model forms, linear, nonlinear, network flow or transportation, has already been constructed on a worksheet using the procedures described earlier in the Math Programming section. We call the problem described by the original model the *master problem*.

The *side model* is a series of one-line models. The one-line models are called *subproblems*. Each subproblem occupies cells on a single row of the Excel worksheet. Cells on the row contain all the variables and constraints for the subproblem as well as parameters and coefficients sufficient to define the model. The side model can have a large number of rows to represent a whole series of similar subproblems. The subproblems are linked to the original model by variable values transferred by equation from the master problem. We will see that a very large model can be constructed with a relatively small number of cells on the worksheet. We can use the *Excel Solver* to solve these problems, but only models of very limited size can be solved with the free Solver available with Excel. The *Jensen LP/IP* Solver has recently been modified to implement the *L-Shaped* method so that quite large models may be addressed. The *L-shaped method* is a decomposition method for solving problems with side constraints.

The side models are very useful for stochastic programming models, models that involve piece-wise linear approximations and models that include fixed charge variables. Several examples are presented in the first three pages. The *L-shaped method* is on the two remaining pages.

- **Multiperiod Model**
- **Stochastic Programming**
- **Regression**
- **L-Shaped Method**
- **L-Shaped Theory**

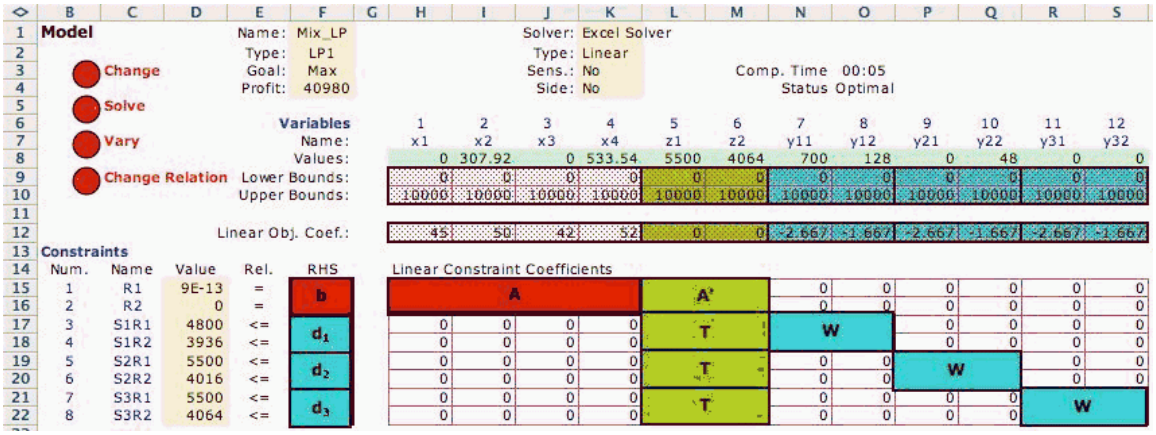
The Block Diagonal Form

An LP is presented below with the form that suggests that a side model might be useful. In the figure below, the colored regions in the constraint coefficients may contain nonzero numbers, while the cells outside these regions contain all zeros. The constraints of this model have the *block diagonal form*. There is a set of variables $\mathbf{x}$ that are related by the first two constraints through the matrix $\mathbf{A}$. The variables in $\mathbf{z}$ are also included in this set of constraints via the matrix $\mathbf{A}'$. The following six constraints contain $\mathbf{z}$ and the variables labeled y11, y12, etc. In the following we call these variables $\mathbf{y}k$ where k=1, 2 or 3. These constraints contain nonzero coefficients in the matrices $\mathbf{T}$ and $\mathbf{W}$. The constraints can be divided into sets so that $\mathbf{y}1$ appears in the first, $\mathbf{y}2$ appears in the second, and so on. The matrix $\mathbf{T}$ multiplies $\mathbf{z}$ in each set, and the matrix $\mathbf{W}$ multiplies $\mathbf{y}k$. The lower bounds, upper bounds and objective coefficients are similarly partitioned. The RHS values of the constraints are also partitioned in this fashion into $\mathbf{b}$, $\mathbf{d}1$, $\mathbf{d}2$ and $\mathbf{d}3$.

The full matrix representation of a problem with the block diagonal form is not efficient because of the large number of zero's in the matrix. Although 0's do not affect the efficiency of the Solvers significantly, the large matrix consumes many cells of the Excel worksheet. The number of cells grows as the product of the number of variables and the number of constraints. Entering

the nonzero data is difficult because it is hard to keep track of rows and columns. In the form presented, the number of variables is limited to the number of columns of the worksheet (the maximum is about 250). Numbers associated with data lists outside the form are not identified with regard to source thus making the model difficult to check and maintain. These difficulties magnify as the number of blocks in the diagonal form increases.

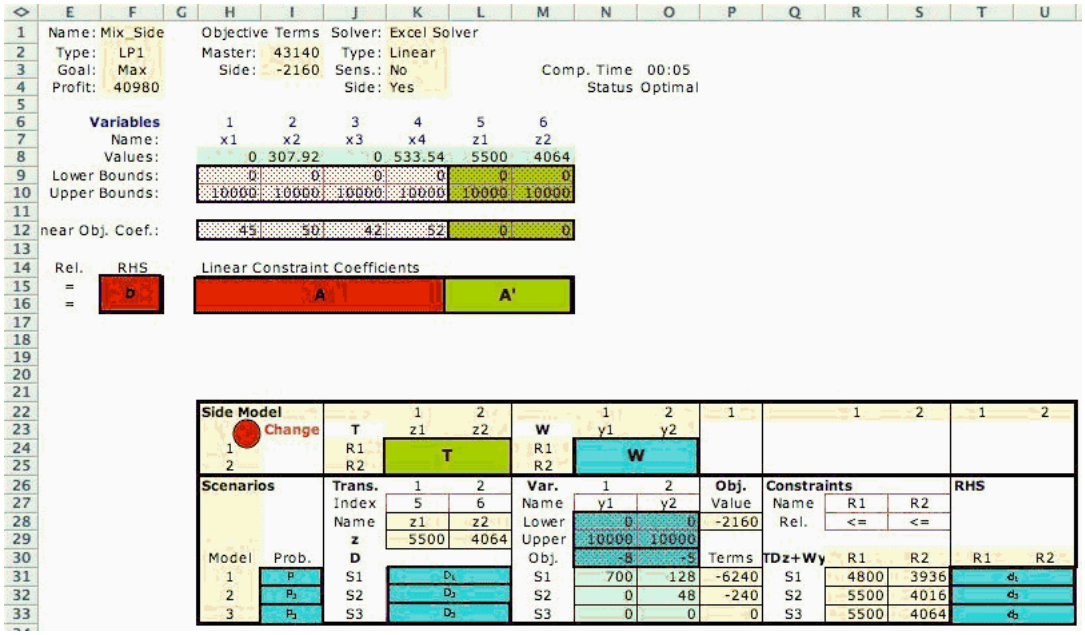
The form is also called the *L-Shaped* form because the nonzero entries roughly resemble an inverted L. The matrix **A** comprises one bar or the L, while the matrix to the right of **A** comprises the other bar.



Side Model

The side model shown below provides a more efficient representation that does not include all the 0's. The model at the top of the page (through row 16) is *master problem*. It describes the first two constraints with the variables **x** and **z**.

The side model starts in row 22 and contains the information describing the *subproblems*. The linking variables, **z**, and the associated matrix **T** is in columns J through L. The linking variables are identified by the indices in cell K27 and L27. The names and values of z1 and z2 are transferred through equations in the range K28:L29. The subproblem variables, bounds and objective coefficients and the matrix **W** appear in columns N and O. The variable values are in the green range N31:N33. These are determined by the Solver. The objective function for the side model is computed in column P. The objective terms are weighted by the numbers in column I. The constraint values, names and relations are in columns R and S. The RHS values are in columns T and U.



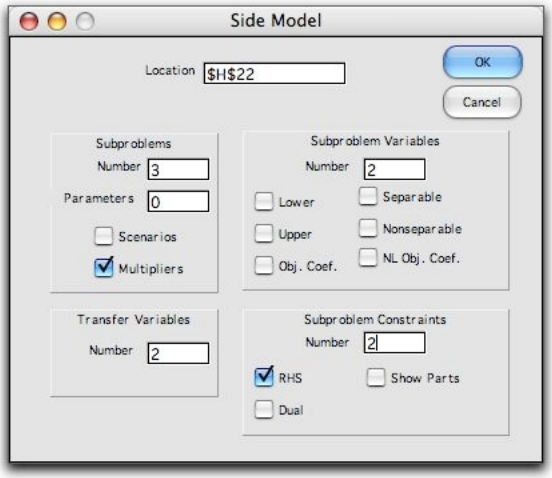
At the top of the page we see the objective function contributions of the master and subproblems in cells I2 and I3 respectively. These are totaled in cell F4. The value of cell F4 is maximized by the Solver.

Notation

The options are more easily discussed when we define notation for problems with the block diagonal form. The general problem is at the left. The model expressed with subproblems is at the right. For simplicity, the forms are shown with equality constraints, however, they can be expressed as inequalities as well. We show all lower bounds to be zero, but the forms allow nonzero or negative lower bounds. We show a linear programming model, but the side model can also be attached to nonlinear master problems, network flow models or transportation models. The side models may include nonlinear terms in the objective function.

General Problem	**Master Problem**
Min. $z = \mathbf{cx} + \mathbf{c}'\mathbf{z} + \sum_{k=1}^{n_s} p_k \mathbf{q}_k \mathbf{y}_k$	Min. $Z = \mathbf{cx} + \mathbf{c}'\mathbf{z} + \sum_{k=1}^{n_s} p_k Z_k$
Subject to:	Subject to:
$\mathbf{Ax} + \mathbf{A}'\mathbf{z} = \mathbf{b}$	$\mathbf{Ax} + \mathbf{A}'\mathbf{z} = \mathbf{b}$
$\mathbf{TD}_k\mathbf{z} + \mathbf{Wy}_k = \mathbf{d}_k$ for $k = 1 \ldots n_s$	$\mathbf{0} \le \mathbf{x} \le \mathbf{u}_x, \mathbf{0} \le \mathbf{z} \le \mathbf{u}_z$
$\mathbf{0} \le \mathbf{x} \le \mathbf{u}_x,\ \mathbf{0} \le \mathbf{z} \le \mathbf{u}_z,\ \mathbf{0} \le \mathbf{y}_k \le \mathbf{u}_{yk}$ for $k = 1 \ldots n_s$	**Subproblems**
	$Z_k = \mathbf{q}_k \mathbf{y}_k$
	$\mathbf{TD}_k\mathbf{z} + \mathbf{Wy}_k = \mathbf{d}_k$ for $k = 1 \ldots n_s$
	$\mathbf{0} \le \mathbf{y}_k \le \mathbf{u}_{yk}$ for $k = 1 \ldots n_s$

After a master problem has been placed on a worksheet, a side model is constructed by selecting the *Side Model* item from the *Math Programming* menu. The dialog below is presented.



The dialog presents options that determine where the side model is placed on the worksheet and the features that are to be included as specific data columns. The cell at the upper left corner of the side model display is indicated in the *Cell* field. The *Subproblems* field holds the number of subproblem rows to be included in the side model. *Parameters* indicate how many columns are to be included to hold parameters of the subproblems. It is often useful to have all relevant parameters entered in these columns, but the parameters can be entered directly in the data columns describing the subproblems. *Transfer* is the dimension of the linking variables $\mathbf{z}$. *Variables* is the dimension of the subproblem variables $\mathbf{y}_k$. The *Constraints* field holds the number of constraints in each side problem. The buttons *Lower*, *Upper*, and *Obj* indicate whether specific columns are to be dedicated to these values. If not checked the values are provided by single cells that pertain to all the subproblems. *Separable Nonlinear* includes columns for separable objective function terms and their coefficients. *Nonseparable Col*. includes a column for nonseparable nonlinear terms. *RHS* button includes a columns to hold a different RHS value for each subproblem. Otherwise a single value is provided to pertain to all subproblems. The *Show Parts* button provides a column for the value of $\mathbf{TD}_k\mathbf{z}$ another for $\mathbf{W}\mathbf{y}_k$. We illustrate a variety of options in the example problems that follow.

Although the example shows a common matrix $\mathbf{T}$ for all subproblems, it is often useful to have different matrices for the different subproblems. This is provided by the matrix $\mathbf{D}_k$ that varies by subproblem. In matrix notation, $\mathbf{D}_k$ is a diagonal matrix that multiplies $\mathbf{T}$. The effect is that the transfer matrix can then be varied by subproblem. Since only the diagonal elements are relevant they are stored on our form in each subproblem row. Although not every situation can be handled using these matrices, we will find that they are very useful for some applications. For applications that have a constant transfer matrix $\mathbf{T}$, the values in the cells are all set to 1. Then $\mathbf{D}$ is an identity matrix.

$$\mathbf{TD}_k\mathbf{z} + \mathbf{W}\mathbf{y}_k = \mathbf{d}_k \text{ for } k = 1 \ldots n_s$$

$$\mathbf{TD}_k\mathbf{z} = \begin{bmatrix} t_{11} & t_{12} \\ t_{21} & t_{22} \end{bmatrix} \begin{bmatrix} D_{i1} & 0 \\ 0 & D_{i2} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$$

$$\mathbf{T}_k\mathbf{z} = \begin{bmatrix} D_{i1}t_{11} & D_{i2}t_{12} \\ D_{i1}t_{21} & D_{i2}t_{22} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$$

See the ORMM Web Site for the Examples of the Side Model

Stochastic Programming

Stochastic Program

$$z = \text{Min. } \tilde{\mathbf{c}}\mathbf{x}$$

Subject to:

$$\tilde{\mathbf{A}}\mathbf{x} = \tilde{\mathbf{b}}$$

$$\mathbf{0} \le \mathbf{x} \le \tilde{\mathbf{u}}$$

$$\tilde{\boldsymbol{\xi}} = vec\left(\tilde{\mathbf{c}}, \tilde{\mathbf{A}}, \tilde{\mathbf{b}}, \tilde{\mathbf{u}}\right)$$

When we recognize the possibility of uncertainty or risk within the context of mathematical programming, the decision problem might be written as below. The model is shown in the typical matrix format for an LP except tildes appear over all the parameter matrices. This implies that all may be affected by a vector of random variables. In practical instances the model could be complicated by nonlinear terms and integrality restrictions. In any event, the model as stated, is not well defined because: (1) the timing of the decisions and observations of the randomness is ambiguous and (2) what is meant by an optimal solution and a feasible solution is unclear.

Stochastic programming addresses the first issue by explicitly defining the sequence of decisions in relation to the realization of the random variables. Given the sequence, an objective function is defined that reflects a rational criterion for evaluating the decisions at the time they must be made. Feasibility conditions must be adapted to the fact that decisions made before the realization of randomness may have feasibility consequences after the realization. How the issues are resolved leads to the several different problems considered in this section. No single problem formulation is sufficient.

For this section we assume that the probability distribution of is known. More properly we should say that stochastic programming is *decision making under risk*, reserving the phrase *decision making under uncertainty* for those situations for which probability distributions are unavailable. We will, however, use the more popular term, uncertainty, to refer to situations in which distributions are known.

For stochastic programming, some variables are to be set by a decision maker, these are the decision variables, while some model parameters are determined by chance, and these are the random variables. There are a variety of situations one might consider in this context. One differentiation is based on when the decision maker must make decisions relative to the time when the random variables are realized. There are several possibilities. Links on the titles go to pages in this section. Some of these descriptions are simplified from more general situations described in more advanced sources.

- *Wait and See*: The decision maker makes no decisions until all random variables are realized.
- *No Recourse*: The decision maker must choose values for the decision variables before any of the random variables are realized. There is a risk of violating the constraints.
- *Chance Constraints*: This is the no-recourse situation when only the RHS vector is random. Solutions are found with specified risks of constraint infeasibility.
- *Simple Recourse*: This topic considers the problem with a random RHS vector. The decision maker must choose values for the decision variables before the RHS values are known, but variables adjusting to the RHS variation are set after the realization. Penalties for constraint violation are specified. A section on *Approximations* indicates how continuous probability distributions can be approximated for this analysis. An *Aircraft Allocation* example describes one of the first problems addressed by stochastic programming.
- *Recourse*: Some of the decision variables must be set before the random variables are realized, while others may wait until after they are realized. Models explicitly represent the initial decisions and all recourse decisions. Although the models can be very large, optimum solutions solve for all possible circumstances. A *Capacity Expansion* example shows that small problems sometimes result in very large models.
- *Multistage Recourse*: Decisions and random variables are determined in stages. The first

set of decision variables must be fixed before all realizations. Then the first set of random variables are realized and the second set of decision variables are fixed. The process continues through a series of decisions and realizations. Typically the stages represent intervals of time. We do not consider this situation.

This section addresses some of the relatively simple problems of stochastic programming, those that we might be able to approach with our Excel add-ins. Along the way we provide examples of situations involving uncertainty. There is a great deal of high-level research in this field attempting to extend the bounds on the variety of problems that can be practically solved. Most solutions require high-power computation. The research and the computation are beyond the capabilities of this author and also of Excel. The reader interested in pursuing this subject further should visit the *Stochastic Programming Community Home Page*.

Uncertainty, or risk, is modeled using random variables and probability distributions. In this section, we often use the features of the *Random Variables* add-in, particularly the *Functions* feature. A page is provided in the random variables section summarizing some of the results of this section. We assume in the following some familiarity with probability theory and the Random Variables add-in. Many of the models constructed in this section use the *Side Model* feature of the Math Programming add-in.

The materials here are partially based on the unpublished manuscript *An Optimization Primer, An Introduction to Linear, Nonlinear, Large-Scale, Stochastic Programming and Variational Analysis*, by Roger J-B Wets, January 11, 2005. Suggestions and notes from University of Texas colleagues David Morton and Lisa Korf were very helpful.

See the ORMM Web Site for types of Stochastic Programming

Excel Solver

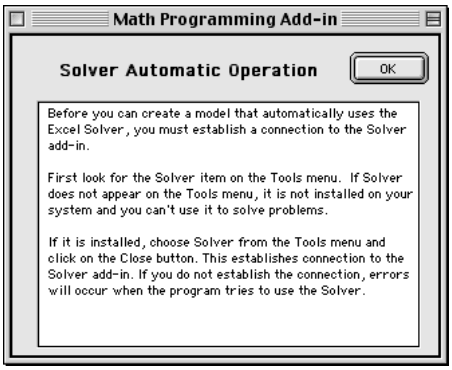
All models formed through the Math Programming add-in can be solved with the Solver add-in that comes with Excel. The add-in comes free with Excel. More capable versions can be purchased from Frontline Systems.

To use this add-in it must be installed on your computer. If it is installed the item "Solver" appears on the Tools menu of Excel. If the item does not appear on the Tools menu, use the custom installation procedure on the Microsoft Office CD to install it.

The Math Programming add-in automatically creates and loads the Solver model every time a model is created and every time the Solve button is clicked. Before this automatic operation is possible, the student must establish contact with the Solver. This is accomplished by simply selecting the Solver item from the Tools menu. The empty Solver dialog box will appear. Simply close the dialog. This action creates the desired link.

If the link is not established before the add-in tries to use Solver, the message from Excel below appears. This indicates that the attempt to use Solver was unsuccessful.

After the OK button is pressed the dialog box from the add-in provides instructions for linking to Solver. Once the connection is made, all interactions with the Solver are automatic.

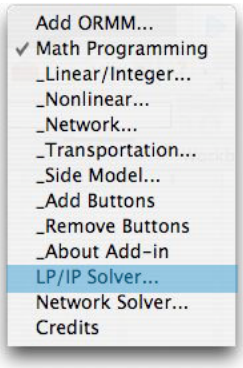


We should note, that the Solver add-in is very useful even without the Jensen add-ins. It can used in many contexts and every student of Excel should learn to use its features.

Linear/Integer Programming Solver

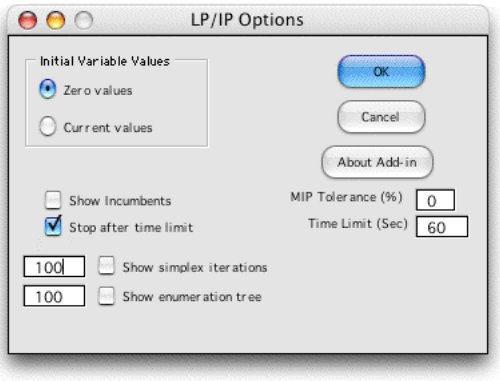
The LP/IP Solver add-in provides an algorithm that solves Linear and Integer Programming problems. It can be used instead of the Excel solver for the linear models created by the Mathematical Programming add-in. There are no built-in limits for model size. Arrays are dimensioned automatically. For large problems, excessive memory requirements may cause the program to crash or computation time may be large.

When this add-in is installed, a new item appears on the OR_MM menu, LP/IP Solver. This item does not solve models. Rather, models are solved when the user clicks on the *Solve* button on the linear, integer, or mixed integer model sheets. The LP/IP Solver menu item presents a dialog that sets several parameters related to the solution process and displays. The dialog will appear only when a mathematical programming model constructed with the Math Programming add-in is present on the active worksheet. Otherwise the message below indicates that it is not available.



"This is not a valid worksheet for the LP/IP Solver"

When a valid worksheet is active, choosing the LP/IP Solver menu item presents the dialog below. These options only affect the Jensen LP/IP Solver, and do not affect the Excel Solver.



Solution Display Options

These options create a separate worksheet for algorithm details that shows iteration information from the solution algorithms. The reports are useful for the student who is learning about the solution methods used. They may also be useful for practitioners trying to discover why certain IP models are so hard to solve.

Simplex Iterations

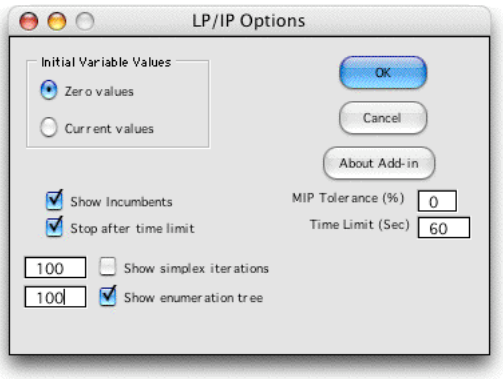
This option creates a new worksheet with the suffix *Details*. Information about the iterations of the primal simplex is printed on this worksheet. The number field to the left on the dialog holds the number of iterations to be displayed. Since some problems may require thousands of iterations, this number should be set to a reasonable value or the memory requirement for Excel will be excessive.

The figure below shows the detail worksheet for the Production problem described earlier. The display shows the slack variables added for the four constraints. No artificial variables are required. Five simplex iterations are necessary to find the optimum solution. The display shows details about the iterations.

	A	B	C	D	E	F
1	Algorithmic Details for problem Production.					
2	Jensen LP Solution Algorithm					
3	Start Solution					
4	Add Artificial and Slack Variables					
5		Slack Variable: 6 added for constraint 1.				
6		Slack Variable: 7 added for constraint 2.				
7		Slack Variable: 8 added for constraint 3.				
8		Slack Variable: 9 added for constraint 4.				
9	Start Phase 2					
10	Iterations	Iter.	Enters	Leaves	Red. Cost	Var. Chg
11		1	2	7	-25	90.9090889
12		2	1	9	-10.045455	50.0000168
13		3	3	6	-22.133927	1.17498464
14		4	4	8	-7.326898	11.6551463
15		5	5	3	-6.5412697	15.6428074
16	Finish	Optimal: z = 2988.72705029472				

Enumeration Tree

When the production variables are required to be integer, the LP/IP add-in can be used to show the details of the enumeration process. Using the LP/IP Solver dialog we click the *Show Enumeration Tree* checkbox. The number field to the left on the dialog is the number of enumeration tree vertices to be displayed. We have also checked the *Show Incumbents* checkbox.



While the problem is solved a record is kept of the enumeration process and displayed on the *Details* worksheet. The production problem with integer variables requires an enumeration tree with 42 nodes.

Clicking both display checkboxes will show both the enumeration tree and the corresponding steps of the simplex method used by the bounding procedure.

Node	Level	Variable	Value	Up/Down	Visit	Relax	
0	0	0	0	0	0	2988	Branch X(4) up at 11
1	1	4	11	1	1	2986	Branch X(1) down at 58
2	2	1	58	-1	1	2983	Branch X(2) up at 63
3	3	2	63	1	1	2980	Branch X(5) down at 15
4	4	5	15	-1	1	2978	Branch X(3) down at 0
5	5	3	0	-1	1	2978	Branch X(4) down at 11
6	6	4	11	-1	1	2977	Branch X(2) down at 63
7	7	2	63	-1	1	2976	Integer: Replace incumbent: Backtrack Level 7 : Branch X(2) up
8	7	2	64	1	2	2961	Fathom: Backtrack Level 7 6 : Branch X(4) up at 12
9	6	4	12	1	2	2975	Fathom: Backtrack Level 6 5 : Branch X(3) up at 1
10	5	3	1	1	2	2968	Fathom: Backtrack Level 5 4 : Branch X(5) up at 16
11	4	5	16	1	2	2973	Fathom: Backtrack Level 4 3 : Branch X(2) down at 62
12	3	2	62	-1	2	2977	Branch X(5) up at 17
13	4	5	17	1	1	2977	Branch X(1) up at 58
14	5	1	58	1	1	2973	Fathom: Backtrack Level 5 : Branch X(1) down at 57
15	5	1	57	-1	2	2976	Fathom: Backtrack Level 5 4 : Branch X(5) down at 16
16	4	5	16	-1	2	2973	Fathom: Backtrack Level 4 3 2 : Branch X(1) up at 59
17	2	1	59	1	2	2984	Branch X(2) up at 63
18	3	2	63	1	1	2964	Fathom: Backtrack Level 3 : Branch X(2) down at 62
19	3	2	62	-1	2	2979	Branch X(5) down at 14
20	4	5	14	-1	1	2979	Branch X(1) down at 60
21	5	1	60	-1	1	2976	Fathom: Backtrack Level 5 : Branch X(1) up at 61
22	5	1	61	1	2	2974	Fathom: Backtrack Level 5 4 : Branch X(5) up at 15

Node	Level	Variable	Value	Up/Down	Visit	Relax	
0	0	0	0	0	0	2988	Branch X(4) up at 11
1	1	4	11	1	1	2986	Branch X(1) down at 58
2	2	1	58	-1	1	2983	Branch X(2) up at 63
3	3	2	63	1	1	2980	Branch X(5) down at 15
4	4	5	15	-1	1	2978	Branch X(3) down at 0
5	5	3	0	-1	1	2978	Branch X(4) down at 11
6	6	4	11	-1	1	2977	Branch X(2) down at 63
7	7	2	63	-1	1	2976	Integer: Replace incumbent: Backtrack Level 7 : Branch X(2) up
8	7	2	64	1	2	2961	Fathom: Backtrack Level 7 6 : Branch X(4) up at 12
9	6	4	12	1	2	2975	Fathom: Backtrack Level 6 5 : Branch X(3) up at 1
10	5	3	1	1	2	2968	Fathom: Backtrack Level 5 4 : Branch X(5) up at 16
11	4	5	16	1	2	2973	Fathom: Backtrack Level 4 3 : Branch X(2) down at 62
12	3	2	62	-1	2	2977	Branch X(5) up at 17
13	4	5	17	1	1	2977	Branch X(1) up at 58
14	5	1	58	1	1	2973	Fathom: Backtrack Level 5 : Branch X(1) down at 57
15	5	1	57	-1	2	2976	Fathom: Backtrack Level 5 4 : Branch X(5) down at 16
16	4	5	16	-1	2	2973	Fathom: Backtrack Level 4 3 2 : Branch X(1) up at 59
17	2	1	59	1	2	2984	Branch X(2) up at 63
18	3	2	63	1	1	2964	Fathom: Backtrack Level 3 : Branch X(2) down at 62
19	3	2	62	-1	2	2979	Branch X(5) down at 14
20	4	5	14	-1	1	2979	Branch X(1) down at 60
21	5	1	60	-1	1	2976	Fathom: Backtrack Level 5 : Branch X(1) up at 61
22	5	1	61	1	2	2974	Fathom: Backtrack Level 5 4 : Branch X(5) up at 15

The solution of the problem is below. Cell F4 shows the profit associated with the best solution found (the final *Incumbent*). Cell F5 shows the upper bound on the profit. In this case, the solution is optimum and the upper bound has the same value as the incumbent.

Below the model we show the incumbents discovered during the procedure. The first feasible solution was found at node 7 of the enumeration tree. The second, and final, feasible solution was found at node 35. It is often useful to view the incumbents in this manner. Integer and mixed-integer problems sometimes have many similar solutions and viewing the incumbents may show this.

Algorithm Control Options

Initial Variable Values

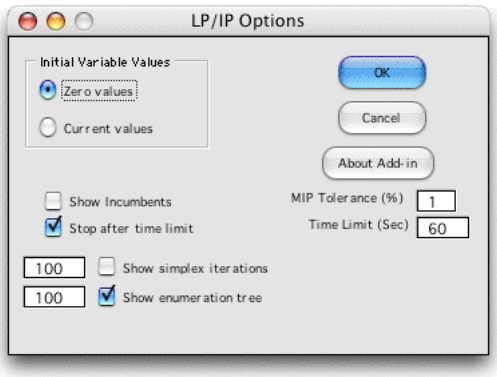
Two different starting strategies are available. The first option starts with all variable values equal to 0. The second strategy gathers the initial variable values from the values shown on the worksheet. The algorithm starts with these values. This advanced start option reduces the number of iterations required when only slight modifications are made to the model. When the model is an IP or MIP and the initial solution is feasible, the solution add-in uses this solution as the initial incumbent. A good initial solution often reduces the size of the enumeration tree and the time required to solve the problem. When the example uses the optimal solution for the initial solution, the enumeration tree has only 16 nodes. The tree is shown below.

Node	Level	Variable	Value	Up/Down	Visit	Relax	
0	0	0	0	0	0	2984	Initital Incumbent
0	0	0	0	0	0	2988	Branch X(4) up at 11
1	1	4	11	1	1	2986	Branch X(1) down at 58
2	2	1	58	-1	1	2983	Fathom: Backtrack Level 2 : Branch X(1) up at 59
3	2	1	59	1	2	2984	Fathom: Backtrack Level 2 1 : Branch X(4) down at 10
4	1	4	10	-1	2	2988	Branch X(2) down at 62
5	2	2	62	-1	1	2987	Branch X(4) up at 10
6	3	4	10	1	1	2985	Branch X(5) up at 17
7	4	5	17	1	1	2985	Branch X(1) up at 59
8	5	1	59	1	1	2983	Fathom: Backtrack Level 5 : Branch X(1) down at 58
9	5	1	58	-1	2	2984	Fathom: Backtrack Level 5 4 : Branch X(5) down at 16
10	4	5	16	-1	2	2981	Fathom: Backtrack Level 4 3 : Branch X(4) down at 9
11	3	4	9	-1	2	2987	Branch X(2) up at 62
12	4	2	62	1	1	2984	Fathom: Backtrack Level 4 : Branch X(2) down at 61
13	4	2	61	-1	2	2986	Branch X(1) up at 59
14	5	1	59	1	1	2984	Fathom: Backtrack Level 5 : Branch X(1) down at 58
15	5	1	58	-1	2	2984	Fathom: Backtrack Level 5 4 3 2 : Branch X(2) up at 63
16	2	2	63	1	2	2981	Fathom: Backtrack Level 2 1 0 :Finished

MIP Tolerance

When solving integer or mixed integer programming problems, vertices of the enumeration tree are fathomed when the relaxed solution objective is less than (for a maximization problem) than the incumbent solution. A nonzero tolerance makes the fathoming test a little easier by fathoming the vertex if its relaxed objective is within x% of the incumbent solution, where x is the number entered in this field. The solution may be found with fewer iterations and less time when this value is greater than 0. When the tolerance is other than 0, however, there is no guarantee that the optimum solution is obtained. It is guaranteed that the solution is with x% of the optimum. The Excel Solver has a similar tolerance specified in the Options dialog of the Solver.

In the dialog below we specify a 1% tolerance level. Clicking the Solve button provides the following solution. The solution returned is not optimum. Rather the solution has the profit of 2976. The optimum has the profit of 2984. The upper bound returned in cell F5 is 2988. One of the primary features of the solution procedure is that it returns both lower and upper bounds on the solution value. Although the selection of 1% in for the example does not result in a guaranteed optimum, we are guaranteed that the solution is 1% of the optimum. The objective for the incumbent and the Upper Bound value provides a range in which the optimum value must reside.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	**Linear Model**				Name:	Prod_Int				Solver:	Jensen LP/IP		Tree Nodes:	14		
2	TRUE				Type:	LP1				Type:	Linear-Integer		Simplex Iter.:	44		
3	FALSE		Change		Goal:	Max				Sens.:	No		Comp. Time	00:02		
4	TRUE				Profit:	2976		Incumbent		Side:	No		Status	1% Solution		
5	FALSE		Solve		Upper Bound:	2988										
6	TRUE					Variables		I-1	I-2	I-3	I-4	I-5				
7	100		Vary			Name:		P1	P2	P3	P4	P5				
8	100					Values:		58	63	0	11	15				
9	0.01		Change Relation			Lower Bounds:		0	0	0	0	0				
10	60					Upper Bounds:		9999	9999	9999	9999	9999				
11																
12						Linear Obj. Coef.:		18	25	10	12	15				
13		Constraints														
14		Num.	Name	Value	Rel.	RHS		Linear Constraint Coefficients								
15		1	Mach 1	159	<=	160		1.2	1.3	0.7	0	0.5				
16		2	Mach 2	199.7	<=	200		0.7	2.2	1.6	0.5	1				
17		3	Mach 3	119.3	<=	120		0.9	0.7	1.3	1	0.8				
18		4	Mach 4	279.8	<=	280		1.4	2.8	0.5	1.2	0.6				

The advantage of choosing a nonzero tolerance is that the size of the tree is significantly reduced. For the example the tree is reduced from 42 to 14 nodes. For larger problems, the reduction will often be a much greater proportion.

Time Limit

When solving all integer or mixed integer programming problems, the process may take a long time. The program will stop when this time limit is reached. The user may give up at this point or continue the optimization. The Excel Solver has a similar time limit specified in the Options dialog of the Solver.

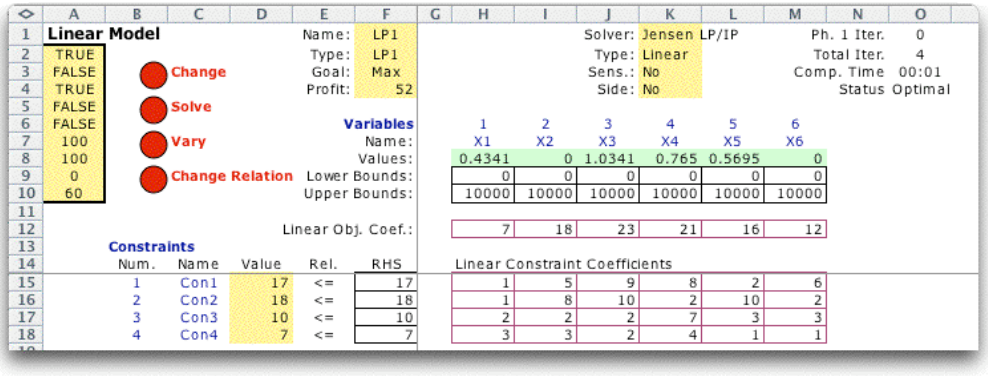
Show Incumbents

When solving all integer or mixed integer programming problems, the algorithm may find feasible solutions during the enumeration process. During the process, the feasible solution with the best value is called the *incumbent* solution. When the process is complete, the final incumbent is placed on the worksheet as the solution. When this checkbox is selected, all incumbents discovered are shown below the model. This is illustrated in a figure above.

Enumeration Tree

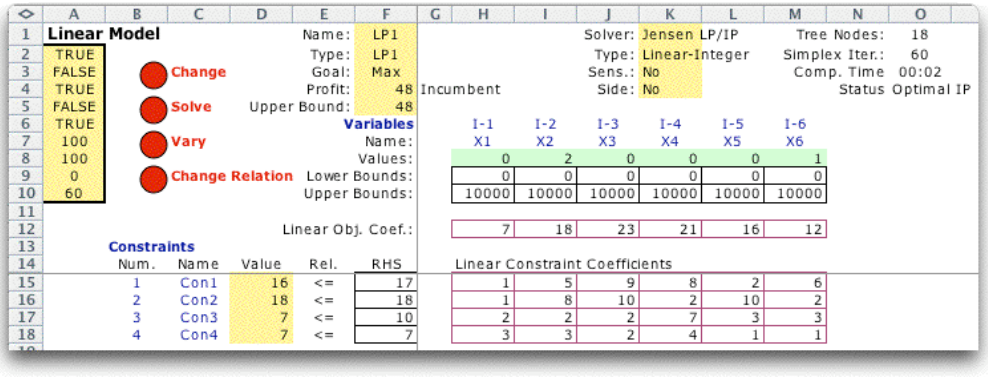
The LP/IP add-in uses a branch-and-bound procedure for finding solutions. The procedure uses an implicit enumeration tree to search for the optimum or for a solution within a specified percentage of the optimum. This page illustrates the tree for an example from Chapter 8 of the book *Operations Research Models and Methods*. A complete discussion of the tree structure is in that text and other texts describing the method.

The example problem is shown below with the problem solved as an LP. For the remainder of this page we enforce integrality on all the variables.



1	**Linear Model**				Name:	LP1					Solver:	Jensen LP/IP			Ph. 1 Iter.	0	
2	TRUE				Type:	LP1					Type:	Linear			Total Iter.	4	
3	FALSE	Change			Goal:	Max					Sens.:	No			Comp. Time	00:01	
4	TRUE				Profit:	52					Side:	No			Status	Optimal	
5	FALSE	Solve															
6	FALSE					**Variables**		1	2	3	4	5	6				
7	100	Vary				Name:		X1	X2	X3	X4	X5	X6				
8	100					Values:		0.4341	0	1.0341	0.765	0.5695	0				
9	0	Change Relation				Lower Bounds:		0	0	0	0	0	0				
10	60					Upper Bounds:		10000	10000	10000	10000	10000	10000				
11																	
12						Linear Obj. Coef.:		7	18	23	21	16	12				
13		**Constraints**															
14		Num.	Name	Value	Rel.	RHS		Linear Constraint Coefficients									
15		1	Con1	17	<=	17		1	5	9	8	2	6				
16		2	Con2	18	<=	18		1	8	10	2	10	2				
17		3	Con3	10	<=	10		2	2	2	7	3	3				
18		4	Con4	7	<=	7		3	3	2	4	1	1				

When we use the *Change* button to express all variables as integer, we see the model below. All the variable indices in row 6 have a predecessor of "I-". This identifies the integer variables. The figure shows the solution to the integer problem. Note that it is much different than the LP solution. In fact the LP and IP solutions are entirely reversed with respect to nonzero values. Both X2 and X6 are 0 for the LP and nonzero for the IP.

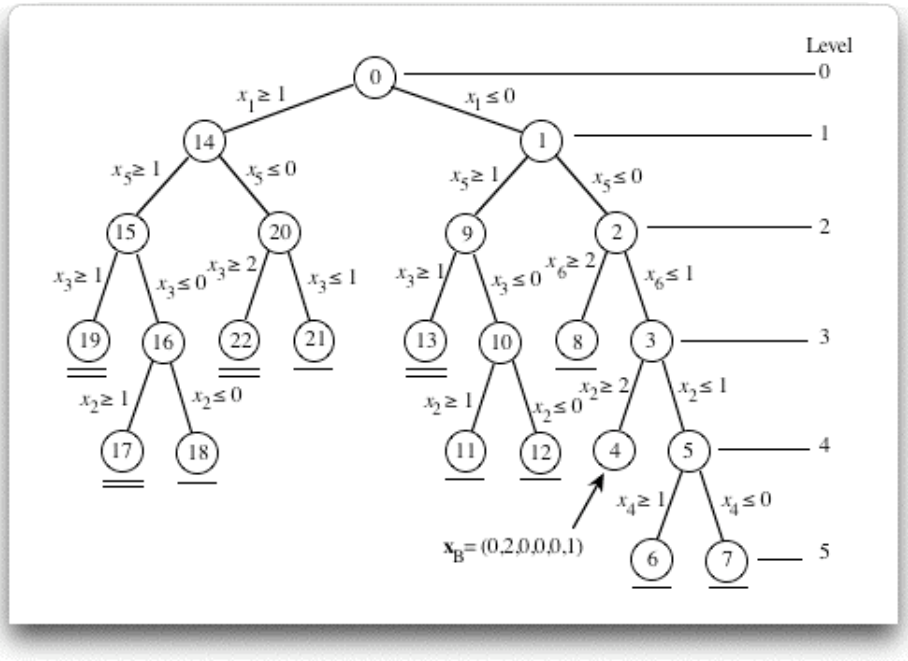


1	**Linear Model**				Name:	LP1					Solver:	Jensen LP/IP			Tree Nodes:	18	
2	TRUE				Type:	LP1					Type:	Linear-Integer			Simplex Iter.:	60	
3	FALSE	Change			Goal:	Max					Sens.:	No			Comp. Time	00:02	
4	TRUE				Profit:	48	Incumbent				Side:	No			Status	Optimal IP	
5	FALSE	Solve			Upper Bound:	48											
6	TRUE					**Variables**		I-1	I-2	I-3	I-4	I-5	I-6				
7	100	Vary				Name:		X1	X2	X3	X4	X5	X6				
8	100					Values:		0	2	0	0	0	1				
9	0	Change Relation				Lower Bounds:		0	0	0	0	0	0				
10	60					Upper Bounds:		10000	10000	10000	10000	10000	10000				
11																	
12						Linear Obj. Coef.:		7	18	23	21	16	12				
13		**Constraints**															
14		Num.	Name	Value	Rel.	RHS		Linear Constraint Coefficients									
15		1	Con1	16	<=	17		1	5	9	8	2	6				
16		2	Con2	18	<=	18		1	8	10	2	10	2				
17		3	Con3	7	<=	10		2	2	2	7	3	3				
18		4	Con4	7	<=	7		3	3	2	4	1	1				

The enumeration tree created by the add-in is shown below. Each line of the display shows one node of the tree.

Node	Level	Variable	Value	Up/ Down	Visit	Relax	
0	0	0	0	0	0	52	Branch X(1) down at 0
1	1	1	0	-1	1	52	Branch X(5) down at 0
2	2	5	0	-1	1	49	Branch X(6) down at 1
3	3	6	1	-1	1	49	Branch X(2) up at 2
4	4	2	2	1	1	48	Integer: Replace incumbent: Backtrack Level 4 : Branch X(2) down at 1
5	4	2	1	-1	2	49	Branch X(4) up at 1
6	5	4	1	1	1	46	Fathom: Backtrack Level 5 : Branch X(4) down at 0
7	5	4	0	-1	2	46	Fathom: Backtrack Level 5 4 3 : Branch X(6) up at 2
8	3	6	2	1	2	42	Fathom: Backtrack Level 3 2 : Branch X(5) up at 1
9	2	5	1	1	2	49	Branch X(3) down at 0
10	3	3	0	-1	1	49	Branch X(2) up at 1
11	4	2	1	1	1	34	Fathom: Backtrack Level 4 : Branch X(2) down at 0
12	4	2	0	-1	2	45	Fathom: Backtrack Level 4 3 : Branch X(3) up at 1
13	3	3	1	1	2	Infeasible	Infeasible: Backtrack Level 3 2 1 : Branch X(1) up at 1
14	1	1	1	1	2	52	Branch X(5) up at 1
15	2	5	1	1	1	49	Branch X(3) down at 0
16	3	3	0	-1	1	48	Fathom: Backtrack Level 3 : Branch X(3) up at 1
17	3	3	1	1	2	Infeasible	Infeasible: Backtrack Level 3 2 : Branch X(5) down at 0
18	2	5	0	-1	2	48	Fathom: Backtrack Level 2 1 0 :Finished

The list is a description of the more commonly represented tree structure below. Although this figure more clearly represents the process, it is difficult to draw for anything but the smallest of problems. Nodes in the figure have one underline if they are fathomed because of bounds and two underlines if they are fathomed because of infeasibility.

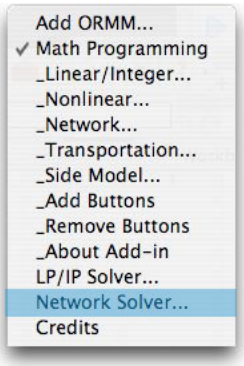


Network Flow Programming Solver

A network flow solution algorithm is provided by the Network Solver add-in (net_solver.xla). The Excel Solver actually solves network problems by solving the underlying linear programming problem. Network algorithms are generally faster than linear programming algorithms for solving problems that can be modeled entirely as networks. This algorithm is used when the user chooses the Jensen Network Solver for either the Network or Transportation models of the Mathematical Programming add-in. The add-in must be installed prior to clicking on the Solve button for either case. The add-in works for linear problems that may or may not have integer variables.

There are no built-in limits for model size. Arrays are dimensioned automatically. For large problems, excessive memory requirements may cause the program to crash or computation time may be large.

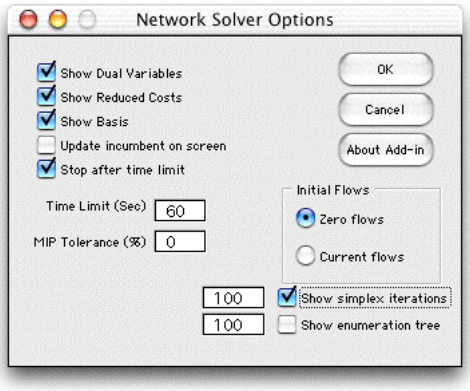
The Jensen Network Solver add-in puts an item on the OR_MM menu: *Network Solver***.**



Choosing this item presents the dialog form shown below. The options on this dialog control the Jensen Network Solver. These options are particularly useful for the student interested in the procedures used in network optimization algorithms. The following describes the several categories of options appearing on this dialog.

Note that the Network Solver add-in only works for models constructed with the Math Programming add-in. A valid network model must be on the active worksheet when the *Network Solver* menu command is selected.

The solution parameters specified on this dialog are stored on the worksheet in column A starting at row 2.



Solution Display Options

These options print information on the problem worksheet or on a separate worksheet. The displays are useful for debugging algorithms, learning the principles of the algorithms and sensitivity analysis.

Show Dual Variables: Prints on the worksheet the node dual variables at optimality. Each node has a dual variable determined by the basis tree. The dual variables are shown to the right of the node external flow data.

Show Reduced Cost: Prints on the worksheet the arc reduced costs at optimality. Arcs are in three categories: basic arcs have a blue background color, nonbasic arcs with flow at the lower bound on a white background and arcs with upper bound flow with gray background. The reduced cost information is shown to the right of the arc data.

Show Basis: For a network flow problem, a basis is defined by a collection of arcs with one arc entering each node of the network. For a pure network problem the collection defines a tree. For the generalized problem it is a tree with one additional arc. When checked, the program displays the tree information to the right of the dual variables.

Simplex Iterations: This option creates a new worksheet with the suffix *Details*. Detailed information showing the iterations of the primal simplex are printed on this worksheet. The number field to the left on the dialog is the number of iterations to be displayed. Since some problems may require thousands of iterations, this number should be set to a reasonable value or the memory requirement for Excel will be excessive.

Enumeration Tree: This option displays on the *Details* worksheet information about the enumeration tree for integer problems. The number field to the left on the dialog is the number of branch and bound vertices to be displayed.

The figure below shows the worksheet for the Power example. The reduced costs are shown on the arc display in column M.

Arc Data and Flows

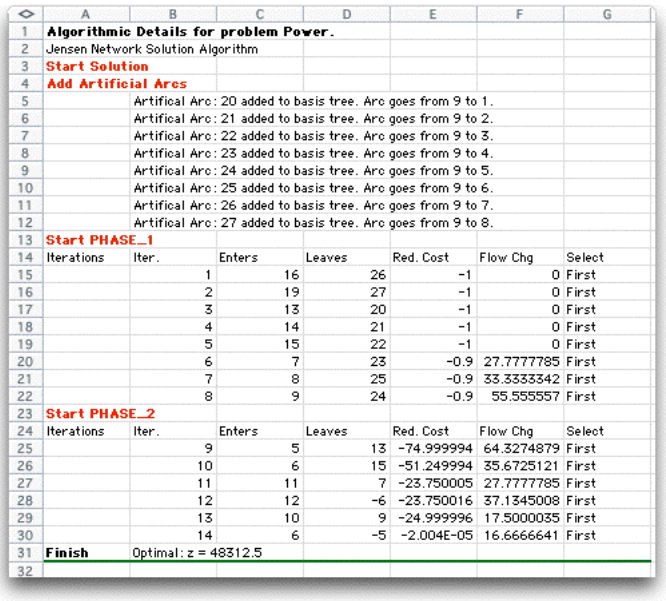
Num.	Name	Flow	Origin	Term.	Lower	Upper	Cost	Gain	Flow_0	Red. Cost
1	A_B	0	1	2	0	99999	0	0.95	0	23.75
2	A_C	0	1	3	0	99999	0	0.95	0	46.3125
3	B_A	0	2	1	0	99999	0	0.95	0	23.75
4	B_C	0	2	3	0	99999	0	0.95	0	46.3125
5	C_A	0	3	1	0	99999	0	0.95	0	-9E-07
6	C_B	16.667	3	2	0	99999	0	0.95	15.833	-1E-05
7	A_X	0	1	4	0	99999	0	0.9	0	23.75
8	A_Z	4E-07	1	6	0	99999	0	0.9	3E-07	-1E-05
9	B_Y	0	2	5	0	99999	0	0.9	0	23.75
10	B_Z	33.333	2	6	0	99999	0	0.9	30	-6E-14
11	C_X	27.778	3	4	0	99999	0	0.9	25	0
12	C_Y	55.556	3	5	0	99999	0	0.9	50	0
13	A	0	0	1	0	100	500	1	0	25
14	B	17.5	0	2	0	75	475	1	17.5	0
15	C	100	0	3	0	100	400	1	100	-51.25

The dual variables basis tree information is shown on the node display.

Node Data and Balance Constraints

Num.	Name	Fixed	Balance	Dual Values	Basis
1	A	0	4E-07	475	-8
2	B	0	-1E-07	475	14
3	C	0	1E-14	451.25	-6
4	X	-25	-7E-07	501.389	11
5	Y	-50	-2E-06	501.389	12
6	Z	-30	-5E-07	527.778	10

The figure below shows the artificial arcs added prior to the beginning of the network simplex iterations. There is one artificial arc for each node. The computer adds three nodes to the original 6 nodes specified in the model. Node 7 is a super source node, node 8 is a super sink node, and node 9 is called the slack node. Four new arcs, 16-19, are added to connect the slack node to the super source and supper sink. The artificial arcs begin at arc 20.



	A	B	C	D	E	F	G
1	**Algorithmic Details for problem Power.**						
2	Jensen Network Solution Algorithm						
3	**Start Solution**						
4	**Add Artificial Arcs**						
5		Artifical Arc: 20 added to basis tree. Arc goes from 9 to 1.					
6		Artifical Arc: 21 added to basis tree. Arc goes from 9 to 2.					
7		Artifical Arc: 22 added to basis tree. Arc goes from 9 to 3.					
8		Artifical Arc: 23 added to basis tree. Arc goes from 9 to 4.					
9		Artifical Arc: 24 added to basis tree. Arc goes from 9 to 5.					
10		Artifical Arc: 25 added to basis tree. Arc goes from 9 to 6.					
11		Artifical Arc: 26 added to basis tree. Arc goes from 9 to 7.					
12		Artifical Arc: 27 added to basis tree. Arc goes from 9 to 8.					
13	**Start PHASE_1**						
14	Iterations	Iter.	Enters	Leaves	Red. Cost	Flow Chg	Select
15		1	16	26	-1	0	First
16		2	19	27	-1	0	First
17		3	13	20	-1	0	First
18		4	14	21	-1	0	First
19		5	15	22	-1	0	First
20		6	7	23	-0.9	27.7777785	First
21		7	8	25	-0.9	33.3333342	First
22		8	9	24	-0.9	55.555557	First
23	**Start PHASE_2**						
24	Iterations	Iter.	Enters	Leaves	Red. Cost	Flow Chg	Select
25		9	5	13	-74.999994	64.3274879	First
26		10	6	15	-51.249994	35.6725121	First
27		11	11	7	-23.750005	27.7777785	First
28		12	12	-6	-23.750016	37.1345008	First
29		13	10	9	-24.999996	17.5000035	First
30		14	6	-5	-2.004E-05	16.6666641	First
31	**Finish**	Optimal: z = 48312.5					
32							

The simplex iterations first drive out the artificial arcs from the basis during phase 1. The phase 2 iterations move toward the optimum basis obtained at iteration 14.

The enumeration tree is obtained for generalized problems (all gains not 1) when some arcs are required to have integer flows. The output is similar to that obtained with the Jensen LP/IP solver.

Algorithm Control Options

Initial Solution

Two different starting strategies are available. The first option starts with all arc flows equal to zero and an all artificial initial basis. The second strategy gathers the initial flows from the values shown on the worksheet. If a basis is displayed, the arcs listed are used as the initial basis. The algorithm starts with these values for the flows and the basis. The advanced start option will result in fewer iterations when only slight modifications are made to the network model.

MIP Tolerance

When solving all integer or mixed integer programming problems, vertices of the enumeration tree are fathomed when the relaxed solution objective is less than (for a maximization problem) than the incumbent solution. This tolerance makes the fathoming test a little easier by fathoming the vertex if its relaxed objective is within x% of the incumbent solution, where x is the number entered in this field. The solution may be found with fewer iterations and less time when this value is greater than 0. When the tolerance is other than 0, however, there is no guarantee that the optimum solution is obtained. It is a guaranteed that the solution is with x% of the optimum. The Excel Solver has a similar tolerance specified in the Options dialog of the Solver.

Time Limit

When solving all integer or mixed integer programming problems the process may take a long time. The program will stop when this time limit is reached. The user may give up at this point or continue the optimization. The Excel Solver has a similar time limit specified in the Options dialog of the Solver.

Update Incumbent on the Screen

When solving all integer or mixed integer programming problems, the algorithm may find feasible solutions during the enumeration process. The feasible solution with the best value is called the incumbent solution. Although intermediate steps are not displayed on the model worksheet while the process continues, when this checkbox is selected, the display will be changed whenever a new incumbent is discovered. For later versions of this add-in the incumbents are displayed automatically.