

# **An Introduction to linear programming with *LinPro 2.0***

# Introduction

## Linear programming

Linear programming was primarily developed by George B. Dantzig in 1947 as a technique for studying the diversified activities of the U. S. Airforce. Dantzig developed the simplex method for solving large linear programs while working on Airforce research projects in the late 1940's. Linear programming is concerned with finding feasible plans that are optimal with respect to a given linear objective function. Linear programming is now widely used in many disciplines. The mathematical definition of linear programming is the analysis of a problem in which a linear function of a number of variables is to be maximized (or minimized) when those variables are subject to a number of constraints in the form of linear inequalities and/or linear equalities. The details of linear programming have appeared in numerous publications for example, Heady and Chandler (1958), Hadley (1962), Murtagh (1981), Dorfman et al., (1958), Pfaffenberger and Walker (1976), Saaty (1988), and Sposito (1989).

## Definition of a linear programming problem

The linear programming problem may be stated mathematically as: Find the values  $(x_1, \dots, x_n)$  which maximize the linear equation

$$z = c_1x_1 + \dots + c_nx_n \quad [1.1]$$

such that the conditions

$$a_{1j}x_1 + \dots + a_{nj}x_n \{ \leq, =, \geq \} b_j \quad j = 1, m \quad [1.2]$$

$$x_j \geq 0 \quad j = 1, n \quad [1.3]$$

are satisfied, where  $a_{ij}$ ,  $c_j$ , and  $b_i$  are constants. The function [1.1] is the objective in terms of the variables  $x_j$ . The constraints [1.2] reflect the restrictions placed on the variables in the objective function. The non-negativity constraints [1.3] are consistent with most linear programming problems. A simple diet problem will be used as an example to illustrate the formulation of a linear programming problem. The problem is to find the minimum cost diet which meets given calorie and vitamin standards. The calorie and vitamin composition and the costs of each food item are given in Table 1.1. Let  $x_1$ ,  $x_2$ , and  $x_3$  denote the amounts in kg of bread, meat and vegetables in the diet. Assume that the diet must contain at least 200 calories and

100 mg of vitamins. The objective is minimize the cost of diet. The formulation as a linear program is

$$\begin{aligned} \min z &= 2x_1 + 3x_2 + 2x_3 \\ \text{s.t. } 24x_1 + 32x_2 + 16x_3 &\geq 200 \\ 8x_1 + 20x_2 + 12x_3 &\geq 100 \\ x_1, x_2, x_3 &\geq 0. \end{aligned}$$

Table 1.1. Food composition and costs for the diet problem.

Food	calories per kg	vitamin mg per kg	cost per kg
Bread	24	8	2
Meat	32	20	3
Vegetable	16	12	2

## LinPro Software

LinPro is an application for solving linear programming problems on an Apple Macintosh computer. LinPro was developed to solve large linear programming problems. The size of linear programming problem that LinPro can solve is limited by the amount of RAM available. The algorithm employed by LinPro has been optimized to maximize the speed of the application. Two versions of LinPro are supplied;

Enhanced LinPro

Standard LinPro.

Enhanced LinPro is optimized for a math coprocessor such as the Motorola 68881, 68882 or the 68040 chip and requires a 62080, 63080 or 68040 central processing unit. Standard LinPro will run on any Macintosh. If you have an SE/30, Mac IIx, IICx, IICI, IIfx, Powerbook 170, Quadra 700 or 900 use the Enhanced LinPro. If have a IISI, LC, Classic II or Powerbook 140 with a math coprocessor installed use the Enhanced LinPro. Otherwise use the Standard LinPro application. If your are unsure of you system configuration run Standard LinPro and choose **System** under the About in the Apple menu. **LinPro Requirements:** System 6.07 or greater.

## Features

LinPro uses the standard Macintosh interface. Enhanced LinPro has been optimized for use with a math coprocessor. The application uses the native floating point format of the math coprocessor to maximize speed. LinPro has three options to further increase computational efficiency:

1. **Optimal Pivoting:** Optimal Pivoting chooses the column to pivot on, which will result in the largest gain in the objective function for a given iteration. Optimal pivoting will increase computational effort for each iteration. However, optimal pivoting usually substantially reduces the total number of iterations required to solve the problem subsequently increasing the overall computational efficiency.

2. **Removal of redundant columns:** A optimal solution to a linear programming problem has can have variables that have values equal to zero (non-basic). If variables that will be non-basic in the optimal solution can be identified and eliminated during the simplex procedure, then computational efficiency of the simplex may be improved.

3. **Lower bounds:** Linear programming problems with lower bound constraints can be transformed to reduce the size of the linear program problem.

LinPro will accept linear programming data with negative right hand sides. The application automatically changes the right hand side to a positive value and updates the constraint type.

## Using LinPro

### Help Menu

The help menu is located in the Apple menu. This contains most of the information in the following sections.

### File Menu

The file menu allows you to run a linear program by selecting Run LP or typing ⌘-R. This will activate the standard file dialog box, where you can select the file containing the linear programming data. LinPro allocates enough memory to solve the problem and reads the data. A second standard file dialog box appears for you to specify a file for the linear programming output. LinPro solves the linear programming problem writes the results to the output file and may display the results on screen depending on which display options have been selected in the display menu. Close (⌘-W) will close the current window. Print (⌘-P) will print contents of the current window. This will activate the standard print dialog box.

Page Setup will activate the standard page setup dialog box. Quit (⌘-Q) terminates the LinPro application.

## Options Menu

Selecting Optimal Pivoting (⌘-O) will turn this feature on or off. Optimal pivoting will have a tick when the feature is on. Optimal pivoting is used in phase one. This option can decrease the amount of time required to solve a problem, especially for larger problems (i.e., greater than 100 constraints). There is no advantage in using this option for most small problems. Selecting Remove Redundant Columns (⌘-E) will turn this feature on or off. Removing redundant columns will decrease the amount of time required to solve larger problems that have considerably more variables than constraints. Transform Lower Bounds (⌘-T) will transform lower bound constraints reducing the size of the linear programming problem. LinPro automatically re-transforms the solutions and give the values for the lower bounds. This feature should be used with any linear programming problem with lower bounds. Maximize (⌘-M) and Minimize (⌘-N) specify whether the linear programming problem is to be maximized or minimized. The defaults for LinPro are to have optimal pivoting, remove redundant columns and transform lower bounds off and maximize on. Run Mode—Foreground (⌘-F) runs LinPro in the foreground not releasing the mac to do other tasks until the problem is solved. Background (⌘-B) allows the mac to do other tasks (multitask) while LinPro solves a problem. This option is slower. Iterations Displayed—when LinPro is solving a problem LinPro will report how many iterations it has completed. The values in this menu are number of iterations between reports of LinPro's progress.

## Text Menu

Font and Size are hierarchical menus for selecting which font and font size you wish the text in the active window to be displayed in. The default for windows when they are first opened is monaco 9pt. You should use a fixed pitch font such as courier or monaco.

## Output Menu

Display Summary Info will display the LinPro kernel information for each linear program which is run. When Show Results Window is on, a window displaying the optimal solution to the linear programming problem will be created. LinPro allows several results' windows to be open simultaneously. If Show Results Window is off then the output is written to file only, and the optimal objective

value is given in the LinPro kernel window if it is active. **Print LinPro Kernel** (⌘-L) will print the text in the LinPro kernel window invoking the standard print dialog box. **Show Dual Activities** (⌘-S) will give the solution to the corresponding dual linear programming problem in the LinPro output file and display these solution in the output window. **Stagger Output Windows** (⌘-W) the first six output windows are staggered for user convenience.

## Edit Menu

**Copy** (⌘-C) copies the selected text from the active window to the clipboard. This allows pieces of the results or the LinPro log to be pasted in other applications. **Clear LinPro Kernel** (⌘-K) clears all the text in the LinPro kernel window. **Select All** selects all the text in the active window.

## Input data

The linear programming data is stored in a text (ASCII) file. This file can be created using *TeachText*, a spreadsheet or a word processor. The file must have the correct format. The following example to illustrate the format of the input file. The problem is

$$\begin{array}{ll}
 \text{min} & 0.025x_1 + 0.028x_2 + 0.022x_3 + 0.011x_4 \\
 \text{s.t.} & x_2 \leq 1000 \\
 & 0.06x_1 + 0.025x_2 + 0.07x_3 \leq 90 \\
 & 0.0115x_1 + 0.086x_2 + 0.13x_3 \geq 200 \\
 & 0.02x_1 + 0.038x_2 + 0.03x_3 \geq 54 \\
 & x_1 + x_2 + x_3 + x_4 = 2000 \\
 & x_4 = 5 \\
 & x_2 \geq 400
 \end{array}$$

This example has 4 variables, 2 less than constraints, 2 greater than constraints, 2 equality constraints and 1 lower bound. This problem would be entered in a file as (*italics indicate the meaning of numbers and are not entered in the file*):

2	<i>number of less than constraints</i>
2	<i>number of greater than constraints</i>
2	<i>number of equality constraints</i>
4	<i>number of variables</i>
1	<i>number of lower bounds</i>

0.025 0.028 0.022 0.011	<i>objective function</i>
0 1 0 0 1000	<i>1st less than constraint</i>
0.06 0.025 0.07 0 90	<i>2nd less than constraint</i>
0.115 0.086 0.13 0 200	<i>1st greater than constraint</i>
0.02 0.038 0.03 0 54	<i>2nd greater than constraint</i>
1 1 1 1 2000	<i>1st equality constraint</i>
0 0 0 1 5	<i>2nd equality constraint</i>
2 400	<i>lower bounds.</i>

The input file called small example contains this data in the LinPro folder. All LinPro input files have this format—the number of  $\geq$ ,  $\leq$ , and  $=$  constraints, number of variables, number of lower bounds,  $\leq$  constraints (if any),  $\geq$  constraints (if any),  $=$  constraints (if any) and the lower bounds (if any). If there are no lower bounds, say, then zero is placed in the position for number of lower bounds and no lower bounds are entered. Similarly for  $\leq$  constraints,  $\geq$  constraints, and  $=$  constraints. To enter a lower bound you enter the number of the variable (i.e., 2 for  $x_2$ ) and the lower bound value. The lower bound value must be greater than zero. The lower bounds may be entered on separate line or on the same line.

## Running a linear program

The small example file will be used to show how a linear programming problem is solved.

- Step 1. Select Minimize from the Options menu.
- Step 2. Select Transform Lower Bounds from the Options menu because the problem has a lower bound.
- Step 3. Select Run LP from the File menu.
- Step 4. Select the Small Example file from the file dialog and choose Open.
- Step 5. LinPro will allocate memory for the problem and read the file then ask for a file to save the results in. Type Small Example Output and select Save.
- Step 6. LinPro will solve the linear programming problem and produce the output in two windows. The kernel window gives the details about the problem. LinPro will display the solution in the Small Example Output window. The contents of this window are also saved in a text file called "Small Example Output."

Not all linear programming problems will have optimal solutions. Two situations can occur, either the problem will be infeasible or it will be unbounded. An infeasible problem occurs when there are no values of the  $x$ 's which satisfy the constraints. For example:

$$x_1 \geq 10$$

$$x_1 \leq 8$$

$x_1$  can not be greater than 10 and less than 8 at the same time. LinPro will notify you of an infeasible problem with the following dialog.

An unbounded problem occurs when a variable can increase or decrease without bound. This will cause the objective function to increase to infinity or decrease to negative infinity. If you have an unbounded problem LinPro will determine which variable is unbounded and display the following dialog.

## Interpreting LinPro output

The small example output will be used to explain LinPro output. The information displayed in the Linear Programming Kernel window gives the time taken to run the linear programming problem, the options used, and the size of the problem. The number of variables displayed includes the original number of variables in the problem plus the number of slack, surplus and artificial variables required to solve the problem. In addition, the optimum objective value is displayed. The Small Example Output window displays the linear programming results stored in the Small Example Output file. The results are presented in four sections. The first section gives the general information about the linear programming problem,

```
LinPro Output
12:52:09 PM Tue, Feb 18, 1992
A Simplex based Approach
```

---

```
Number of variables      : 12
Number of greater than constraints: 2
Number of less than constraints : 2
Number of equality constraints : 2
```

---

The second section gives the optimal values for the variables in the original linear programming problem,

```
Final Solution
```

Basis	Value
-------	-------



x 1	465.0000
x 2	1000.0000
x 3	530.0000
x 4	5.0000

for example the optimal value for  $x_2$  is 1000. If a variable in the original problem is not in the final solution then it has optimal value of zero. The third section gives the optimal non-zero values of the slack and surplus variables required to solve the linear programming problem and the constraint that they are associated with and the optimal solution.

Basis	Constraint	Value
Surplus	3	8.3750
Surplus	4	9.2000
Objective Optimum		51.3400

The final section gives the non-zero dual activity values which are the solution to the dual linear programming problem associated with the original (primal) problem if Show Dual Activities in the Output menu is selected. For more information on duality see Sposito (1989).

Constraint	Dual Activity
1	0.0030
5	0.0250
6	-0.0140

## Performance of LinPro

A second example input file called "Large Example" is included in the LinPro folder. This problem was developed to test the performance of the software. This is a taxing problem to solve. The following table gives the time in seconds Enhanced LinPro needed to solve the problem on different Macintoshs in foreground. Also given is a measure of the mathematical performance of each Macintosh, called Whetstones, where a larger amount equates to better mathematical performance. If you wish to solve this problem on a Macintosh

10

without a math coprocessor be prepared for a long wait.

Macintosh Model

K Whetstones

Time required without optimal pivoting (sec)

Time required with optimal pivoting (sec)

Ilcx

790

222.0

109.3

Ilisi with 68882 chip

795

178.6

90.5

Ilci

1011

162.4

81.5

Quadra 900

3530

29.3

17.2

Ilcx with 68040 radius accelerator

3750

28.9

16.2

## Information

LinPro was developed using Think C 5.01. The algorithm for solving the linear programming problems conforms to ANSI C standards. Please send comments, reports of bugs, or inquiries about the availability and cost of the source code to:

**Prior to 1 August 1992**

B. L. Harris,  
 245 Kildee Hall,  
 Iowa State University,  
 Ames, IA 50011.  
 E Mail – kiwi@vincent.iastate.edu

**After to 1 August 1992**

B. L. Harris,  
 Research and Development,  
 Livestock Improvement Corp.,  
 Private Bag,  
 Hamilton,

New Zealand. **Disclaimer**

THIS PROGRAM MAY BE FREELY DISTRIBUTED, BUT IS **NOT** PUBLIC DOMAIN.

IN **NO** EVENT WILL THE DEVELOPER OF THE PROGRAM BE LIABLE FOR ANY DAMAGES RESULTING FROM THE USE OF THIS PROGRAM. THIS PROGRAM IS USED ENTIRELY AT THE RISK OF THE USER.

THE DEVELOPER RETAINS ALL RIGHTS TO THE CODE AND VISUAL PRESENTATION THEREOF. THIS PROGRAM MAY **NOT** BE DISTRIBUTED IN ANY COMMERCIAL FORMAT WITHOUT EXPRESS WRITTEN PERMISSION FROM THE DEVELOPER. THIS INCLUDES DISK'S OR CD-ROM'S OF SHAREWARE AND FREeware WHERE A FEE IS CHARGED, WITH THE EXCEPTION OF MACINTOSH USER GROUP LIBRARIES, AND INCLUDES ALL HARDWARE PRODUCTS.

ALTHOUGH THE PROGRAM AND REFERENCE MANUAL INCLUDE COMPANY AND UNIVERSITY NAMES, THIS IN **NO WAY** IMPLIES THAT THOSE COMPANIES OR UNIVERSITIES ENDORSE OR SUPPORT LINPRO.

## References

Dorfman, R., Samuelson, P. A. & Solow, R. W. (1958). *Linear programming and economic*

*analysis*. McGraw Hill, New York.

Hadley, G. (1962). *Linear Programming*. Addison-Wesley, Reading, Mass.

Heady, E. O. & Chandler, E. O. (1958). *Linear programming Methods*. Iowa State Press, Ames, IA.

Murtagh, B. A. (1981). *Advanced linear programming*. McGraw Hill, New York.

Pfaffenberger, R. C. & Walker, D. A. (1976). *Mathematical Programming for Economics and Business*. Iowa State Press, Ames, IA.

Saaty, T. L. (1988). *Mathematical Methods of Operations Research*. Dover Publ. Inc., New York.

Sposito, V. A. (1989). *Linear Programming with Statisitcal Applications*. Iowa State Press, Ames, IA.