Calhoun

Institutional Archive of the Naval Postgraduate School

Calhoun: The NPS Institutional Archive

Reports and Technical Reports | All Technical Reports Collection

2003

Calling the LP Solve linear program software from Excel, S-Plus and R

Buttrey, Samuel E.

Monterey, California. Naval Postgraduate School

http://hdl.handle.net/10945/736

NPS-OR-03-006

NAVAL POSTGRADUATE SCHOOL
Monterey, California

Calling the LP_Solve Linear Program Software From Excel, S-Plus and R
by
Samuel E. Buttrey
August 2003

Approved for public release; distribution is unlimited.

Prepared for: Naval Postgraduate School
Monterey, CA 93943-5000

REPORT DOCUMENTATION PAGE		*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC 20503.		

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE August 2003	3. REPORT TYPE AND DATES COVERED Technical Report
4. TITLE AND SUBTITLE: Calling the LP_Solve Linear Program Software From Excel, S-Plus and R		**5. FUNDING NUMBERS**
6. AUTHOR(S) Associate Professor Samuel E. Buttrey		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		**8. PERFORMING ORGANIZATION REPORT NUMBER** NPS-OR-03-006
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		**10. SPONSORING / MONITORING AGENCY REPORT NUMBER** N/A
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.		
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.		**12b. DISTRIBUTION CODE**

13. ABSTRACT *(maximum 200 words)*

We present software that allows Excel, S-Plus and R to call the functions in the lp_solve system. Lp_solve is free software (licensed under the Gnu Lesser GPL) that solves linear and mixed-integer linear programs of moderate size (on the order of 10,000 variables and 50,000 constraints). Since these problems are substantially larger than those that can be handled by Excel's built-in solver, this link will allow Excel users to handle large problems at no extra cost.

14. SUBJECT TERMS Software, lp_solve, integer programming, linear programming, Excel, freeware			**15. NUMBER OF PAGES** 15
			16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

CALLING THE LP_SOLVE LINEAR PROGRAM SOFTWARE FROM EXCEL, S-PLUS AND R

Samuel E. Buttrey

Associate Professor
Department of Operations Research
Naval Postgraduate School
Monterey, CA 93943 USA

ABSTRACT

We present software that allows Excel, S-Plus and R to call the functions in the `lp_solve` system. `Lp_solve` is free software (licensed under the Gnu Lesser GPL) that solves linear and mixed-integer linear programs of moderate size (on the order of 10,000 variables and 50,000 constraints). Since these problems are substantially larger than those that can be handled by Excel's built-in solver, this link will allow Excel users to handle large problems at no extra cost.

DISCLAIMER

The reader is cautioned that computer programs developed in this research may not have been exercised for all cases of interest. While every effort has been made, within the time available, to ensure that the programs are free of computational and logic errors, they cannot be considered validated. Any application of these programs without additional verification is at the risk of the user.

1. INTRODUCTION

A. The Excel Solver and Its Limitations

The spreadsheet program Microsoft Excel 2000 for Windows (Walkenbach, 1999) comes supplied with a "Solver" add-in that performs numerical optimization including linear and integer programming. However, the Solver can handle only comparatively small problems (200 "adjustable cells," according to the on-line help for Excel 2000). Additional "add-ins" that allow the solution of larger problems are available for purchase. This paper addresses a need for a free solver to handle linear or mixed integer programs of substantial size (on the order of 10,000 variables and 50,000 constraints). We use the free software `lp_solve` (see the ftp site at `ftp://ftp.ics.ele.tue.nl/pub/lp_solve`), re-compiled into a dynamic linked library (DLL) using a free compiler and development environment. This DLL works together with an interface written in Visual Basic for Applications on the Excel end and in C on the DLL end to allow calls to the `lp_solve` application programming interface (API).

The same DLL and interface allow calls to `lp_solve` from the statistical environments S-Plus (Insightful, 1999) and R (R Project Home Page, `http://www.r-project.org`). The latter is another piece of freeware. `Lp_solve` is licensed under the Gnu Lesser General Public License; see the documentation for specific terms of the licensing agreement.

B. DLLs and the Development Environment

For this project we used the Cygwin development environment (Cygwin project, `http://www.cygwin.com`). This environment supplies a set of Unix-like tools to the Windows application developer. In particular, it allows the use of the free Gnu compiler gcc (Gnu project, `http://gcc.gnu.org`). Through this environment the developer is spared the necessity of having to purchase a commercial development environment.

The gcc compiler and its associated tools allow the production of DLLs. The DLL contains the essential instructions associated with the program. The DLL acts as a "library" that can be attached to the main program (Excel, for example) and whose member functions can be called as needed.

2. THE PIECES OF THE LINK

A. Description

The link that we have constructed consists of two parts. The first is Visual Basic for Applications (VBA) code that is part of the Excel workbook. This code contains a form that allows the user to specify the objective function, constraints, integer variables, and the place for the results to appear. It then declares and calls the second part, a C function that acts as the interface between the VBA call and the functions exposed by the `lp_solve` API. Figure 1 shows a schematic diagram of the link.
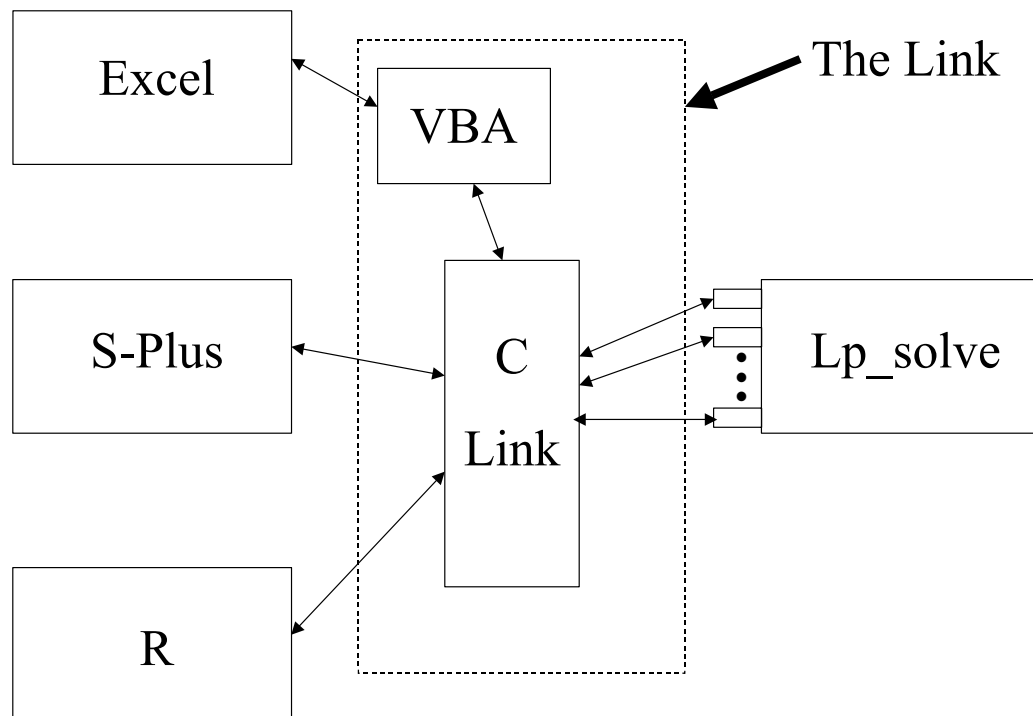
Figure 1: Schematic of Link Between Excel and S-Plus

B. The C Link

The C link is necessary in order to call `lp_solve` from S-Plus (or R; for brevity "S-Plus" will include R in this discussion). This is because in some instances lp_solve passes function arguments by value, but when S-Plus calls a DLL all arguments must be passed by reference. Furthermore, C functions called from S-Plus cannot return values. The C link allows function calls originating in S-Plus to be properly "packaged." The link is not necessary when calling `lp_solve` from VBA, since VBA can accept return values and pass arguments in either style. Still, the C link provides a simple, one-call interface to the `lp_solve` system. Currently, the caller provides the coefficients of the objective function, the matrix of constraints, arrays containing the directions and right-hand sides of each constraint, and a vector indicating which variables are required to be integers. Details of handling these arguments (for example, the objective function array needs to have an additional leading 0) are managed by the caller.

Inside the C link, the code dispatches calls to some of the `lp_solve` functions. There are about 150 of these, but at the moment only nine are used. These calls create the program, set the objective function and the optimization direction, add constraints, set integer variables, solve the program, extract the results, and delete the program.

C. The VBA Link

VBA is a dialect of Visual Basic that serves as a scripting language for all Microsoft applications (for an in-depth introduction, see Walkenbach (1999)). In the present case, its usefulness descends from its ability to handle forms and to call DLLs. The spreadsheet used for calling `lp_solve` contains a button that produces a form when pressed. This form allows the user to select whether the problem is a general mixed linear program, or specifically an assignment or transportation problem. This selection produces a second form. For a general mixed integer program, the user fills in the form to describe whether the problem is a maximization or a minimization, and to give the ranges of the objective function, constraints, integer variables, and the location where results should be placed. A

4

press of a button then produces the solution. The current interface is admittedly primitive. Figure 2 shows a screen shot of a spreadsheet; the "Start Solver" button, the "Choose a Problem" window, and the linear program form are visible.

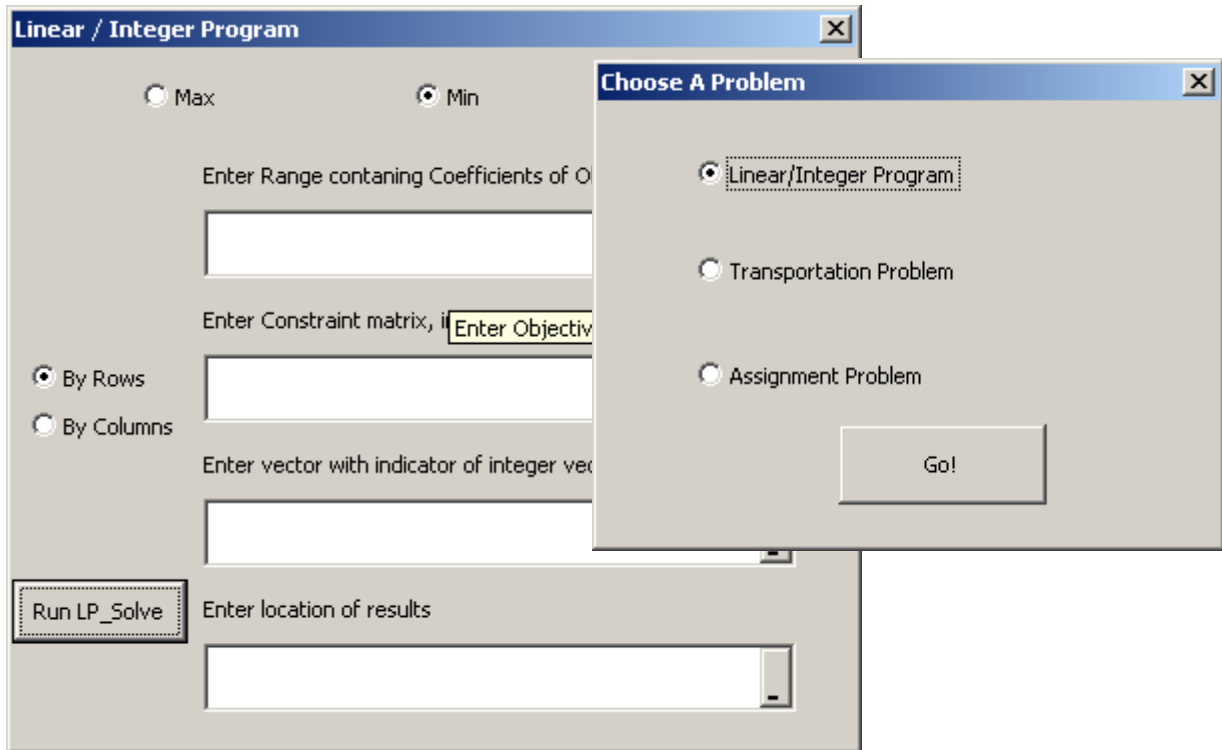**Figure 2: Problem Selection Window and Linear Program Solver Form**

3. DETAILS AND SOME TECHNICAL NOTES

A. Types of Problems

As Figure 2 suggests, three types of problems are currently supported in the link. The first is a general linear program. Here the objective function and constraints are laid out in rows, with each variable occupying a column. Any linear program in standard form can be represented in this way.

However, two specific types of linear programs that arise in practice are also made available. In the transportation problem, the decision variables are arranged in a rectangular matrix, say I by J, so that the decision variables can be denoted by x_{ij}, $i = 1$, …, I; $j = 1$, …, J. There is a cost associated with each of the x_{ij}, and there are constraints on row and column sums. For this problem the user needs to enter a rectangular matrix containing the decision variables, the signs of the constraints, and the constraint values, and a second matrix containing the costs. The program converts the constraints to the needed form and requires that all variables be non-negative integers.

The assignment problem is a special case of the transportation problem in which all the rows and columns are constrained to add up to one. In our implementation, the user needs to provide one matrix with decision variables and a second with cost; the program then prepares the constraints and requires that all variables be non-negative integers.

B. Compilation

We created DLLs containing both the C link and the `lp_solve` code using the gcc compiler in the Cygwin environment. This allows other users to modify the code without having to purchase a commercial development environment. Code and Makefiles are available from the author. Note that by default the Cygwin compiler produces DLLs that themselves depend on another DLL, `cygwin1.dll`, that is installed with that environment. Since most users will not have this DLL, compilation should include the `-mnocygwin` flag and references to the MinGW libraries so that the resulting DLL can stand alone (see the Makefile for details).

When producing a DLL for Excel, calls to the compiler should include the `-mrtd` flag so that the resulting objects use the Pascal (stdcall) calling convention. This flag is included in the `Makefile` file. The DLL for R needs to use the C calling convention, and so calls to the compiler need to omit the `-mrtd` flag. Although S-Plus can seemingly use either convention, we use the Excel DLL for S-Plus use. Users creating DLLs for R

should use the `Makefile.R` file, which also defines a compile-time definition BUILDING_FOR_R that accounts for the fact that integer variables are passed as `int *` in R but as `long *` in S-Plus.

C. Calling from Excel

Called a DLL from Excel requires two steps. First the DLL should be declared, using a `Declare` statement in the VBA code. Second, it is called using a `Call` statement. The documentation suggests that arrays should be declared as arrays in the `Declare` statement, and passed as arrays in the `Call` statement, like this:

```
Private Declare Sub lpslink Lib "lpsolve.dll" _
    (ByRef objOut() As Double, ....)
Call lpslink (objOut(), ...)
```

In fact, though, this does not work. Instead, array arguments should be declared as scalars, and the call should refer to the first element of the array (element 0 unless specified otherwise), like this:

```
Private Declare Sub lpslink Lib "lpsolve.dll" _
    (ByRef objOut As Double, ....)
Call lpslink (objOut(0), ...)
```

A workbook containing all the examples in this document is available from `http://web.nps.navy.mil/~buttrey/Software/lpsolve`.

D. Calling from S-Plus or R

In S-Plus 6.1 (Insightful Corp., 1999), DLLs are loaded with the `dyn.open()` function; `dyn.load()` accomplishes that task in R (we are currently using version 1.7.1). Recall that while the code used to produce the DLLs for S-Plus and R is the same

7

the compilation schemes are different. Three functions, useable in either S-Plus or R, serve as the interface to the DLLs. The function `lp()` accepts, as arguments, the vector of objective function coefficients, a matrix of constraints, vectors containing the signs of the constraints and their right-hand side values, and a vector indicating which variables should be required to be integers. The return value is a list that includes (among other things) the optimal values of the decision variables and the objective function value. The related functions `lp.assign()` and `lp.transport()` handle the assignment and tranportation problems in a way analogous to the way they are handled in Excel (see Examples).

4. EXAMPLES

A. Example 1: Where Solver is Wrong

An example of where Excel's solver gives the wrong answer appears in Nemhauser and Wolsey (1988, p. 443). (This example was brought to our attention by the `lp_solve` documentation.) The problem is:

Maximize $592 x_1 + 381 x_2 + 273 x_3 + 55 x_4 + 48 x_5 + 37 x_6 + 23 x_7$
subject to $3534 x_1 + 2356 x_2 + 1767 x_3 + 589 x_4 + 528 x_5 + 451 x_6 + 304 x_7 > 119567$
with $x_1, x_2, x_3, x_4, x_5, x_6, x_7$ all integer.

Nemhauser and Wolsey (1988) say (in our notation) "[i]t is not hard to show an optimal solution is $x_1 = 33$, $x_2 = 1$, $x_3 = 0$, $x_4 = 1$, $x_5 = 0$, $x_6 = 0$, $x_7 = 0$, and that the optimal [objective function] value is 19972." This is the solution produced by Excel's solver using the default settings. In fact, though, the solution (32, 2, 1, 0, 0, 0, 0) meets the constraint and produces an objective function value of 19979. This is the solution produced by the `lp_solve` link. (If "tolerance" is set to 0 in Excel, the solver produces the correct response, but it gives a message saying "solver could not find a feasible solution.") Figure 3 shows a screen shot of the `lp_solve` link solver producing the

correct answer, shown in grey cells (see the "Nemhauser Example" worksheet in the example workbook).
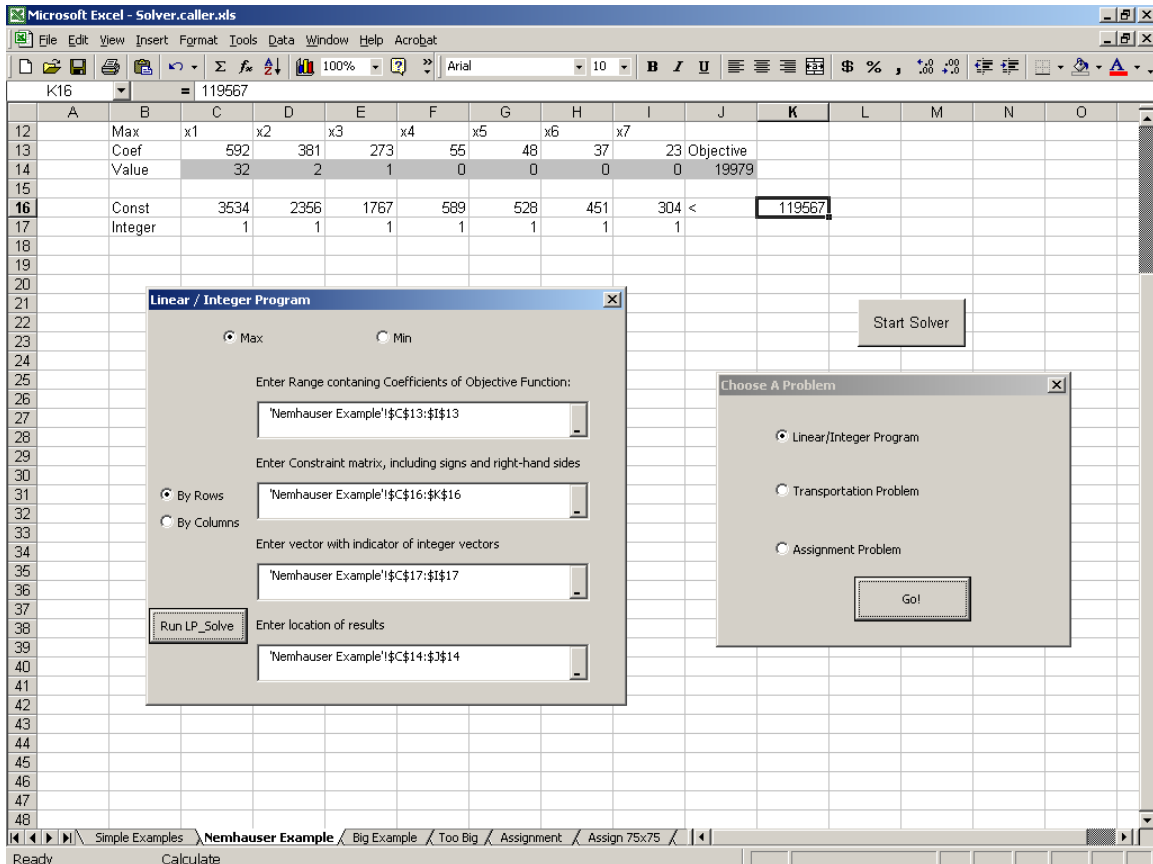
Figure 3: Lp_Solve Link Finding the Correct Answer to Example 1

B. Example 2: Problems Too Big for Excel's Solver

It is easy to construct examples too big for Excel's solver. A simple one on the "Assignment" worksheet demonstrates the assignment problem. Here there are 15 sources (say, operators) to be assigned to 15 destinations (say, jobs). Each decision variable represents the assignment of a source to a destination, so in this example there are 225 variables and, with one constraint per row and column, 30 constraints. Figure 4 shows a screen shot of the assignment problem in the example worksheet. (The lower matrix shows the assignments, with the total cost of 24 just visible to the bottom right; the upper matrix shows the costs, with highlighted cells showing the actual assignments.)

9

Since there are more than 200 adjustable cells in this problem, the Excel solver will not run. On a laptop computer equipped with Windows 2000, a 2.2GHz processor and 1GB RAM, Lp_solve handles this problem in very much less than a second. A 75×75 problem takes about three seconds; a 150×150 problem takes about 30 seconds; a 200×200 problem fails, seemingly for lack of memory.
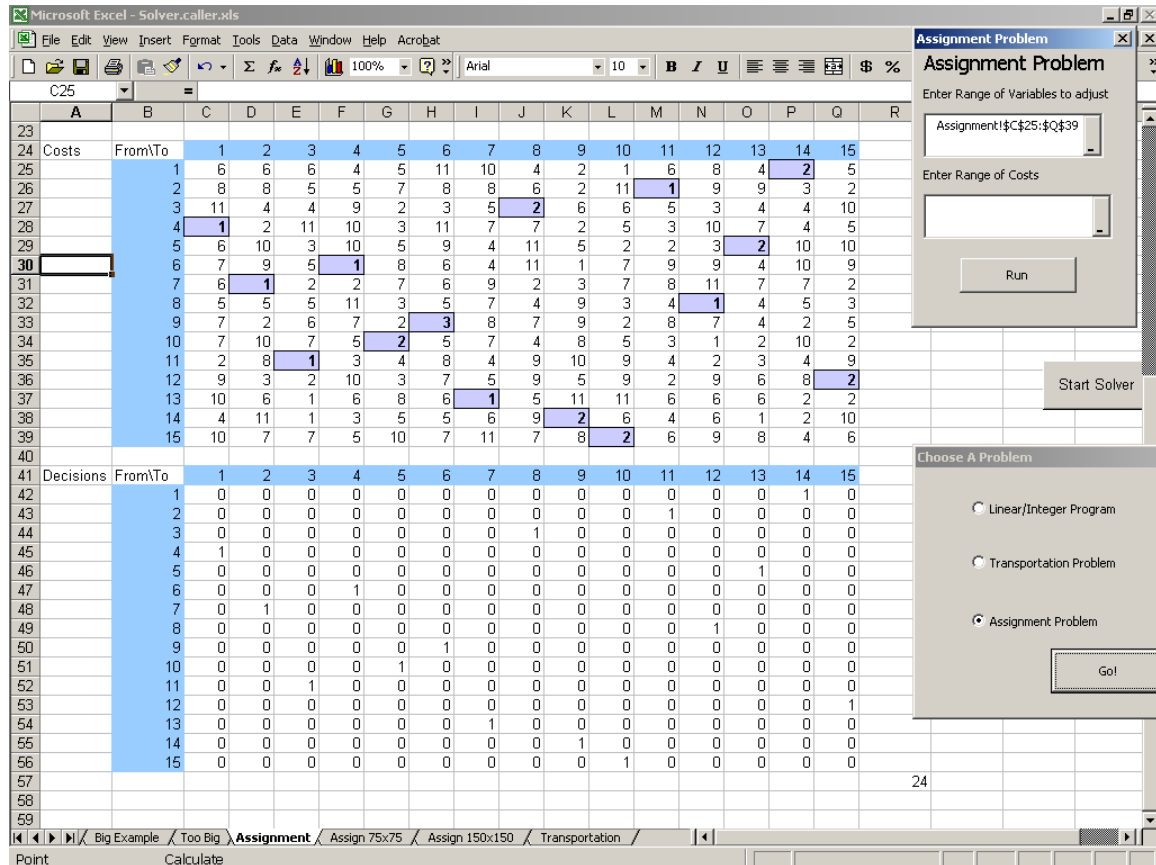
Figure 4: Assignment Problem Example. The upper matrix shows the costs (highlighted cells show the optimal assignment); the lower shows the decision variables. The optimal cost of 24 is visible in the bottom right of the lower matrix.

C. Example 3: Transportation Problem

An example from Bronson (1982) demonstrates the transportation problem and its implementation. In the transportation problem, the row and column sums are constrained, but not necessarily to be equal to one. The user enters the sign and value associated with each constraint, as in the screen shot in Figure 5. As with the assignment problem, the $I \times J$

transportation problem is assumed to be a minimization problem, with *IJ* integer variables and exactly *I+J* constraints. The optimal value for the example problem, 7790, is visible in the bottom-right corner of the upper matrix.

5. CONCLUSION AND FUTURE DEVELOPMENT EFFORTS

A. Conclusion

This report describes progress in solving linear programs in Excel. The existing Solver built in to Excel handles only small problems and occasionally, as in Example 1 above, produces the wrong answer. Through the C and VBA link described here we can call the `lp_solve` free software from Excel and from S-Plus and R to find solutions for problems orders of magnitude larger. This allows users to solve moderate to large linear and mixed integer linear programs at no additional cost. In the case of R, this adds a freeware linear program solution capability to a high-quality freeware statistical software environment. A workbook available from the author's website demonstrates the link in general linear programs and some functionality specific to transportation and assignment problems.

Although `lp_solve` appears to find the correct solution in the above examples, it cannot be considered validated software.

From\To	Spring	Summer	Fall	Winter	dummy	Sign	Supply
RegSpr	0	100	0	100	0	<	200
RegSum	0	0	300	0	0	<	300
RegFall	0	0	0	350	0	<	350
Init	200	0	0	0	0	<	200
OTSpr	50	0	0	0	50	<	100
OTSum	0	0	0	0	50	<	50
OTFall	0	0	100	0	0	<	100
OTWinter	0	0	0	50	100	<	150
Sign	>	>	>	>	>		
Demand	250	100	400	500	200		7790

Costs

From\To	Spring	Summer	Fall	Winter	dummy
RegSpr	10000	7.00	7.70	8.40	0
RegSum	10000	10000	7.00	7.70	0
RegFall	10000	10000	10000	7.00	0
Init	0	0.70	1.40	2.10	10000
OTSpr	8.00	10000	10000	10000	0
OTSum	10000	9.00	10000	10000	0
OTFall	10000	10000	8.00	10000	0
OTWinter	10000	10000	10000	10.00	0

Transportation Problem

Enter Range of Variables to adjust, with row and column directions and RHSs

Transportation!$C$19:$I$28

Enter Range of Costs

Transportation!$C$32:$G$39

Run

Start Solver

Choose A Problem

○ Linear/Integer Program
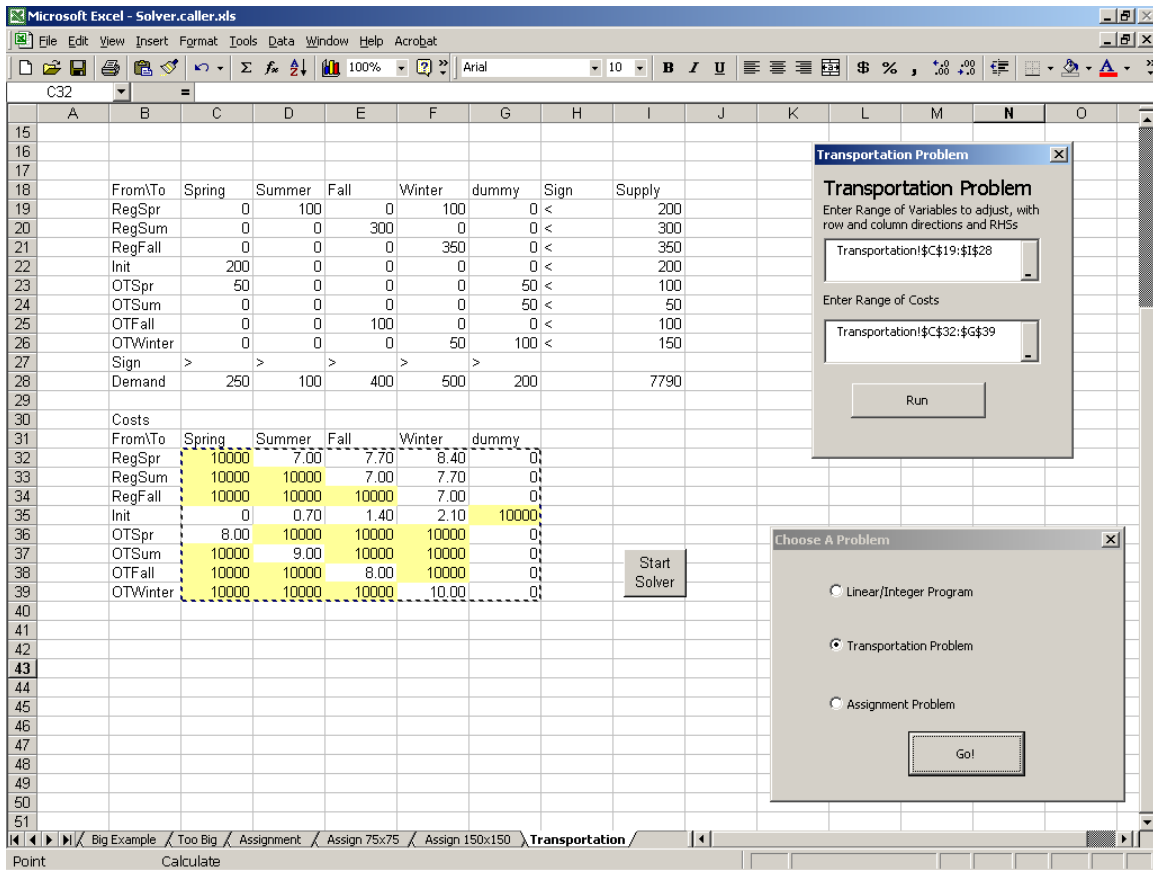
● Transportation Problem

○ Assignment Problem

Go!

Figure 5: Transportation Problem. The user supplies signs and values for each row and column constraint (upper matrix). The costs (lower matrix) include shaded cells set to a large value as a penalty. The optimal value of 7790 is visible in the bottom right-hand corner of the constraint matrix.

B. Further Development

Right now the link is fairly primitive. For example, no constraints other than those on the row and column sums can be added to transportation or assignment problems, and the location of the optimal value cannot be selected by the user. The error-checking is also primitive: if a user enters no text at all into the objective function box, for example, that error will be caught and a reasonable message produced. However, if she enters some text that is not a range at all, the error is not currently caught by the link. Instead, Excel returns a run-time error whose text is largely indecipherable. A number of improvements to the form interface can be made; for example, right now constraints must appear in

12

rows, not in columns. The link might be upgraded to allow calls to other `lp_solve` functions so as to, for example, establish feasibility, find constraint values, print dual values, and so on.

For the moment the user needs either to install the DLL into one of the system directories, or to edit the code so that the location of the DLL is made explicit. A more elegant solution would be to convert the workbook to an add-in; the add-in could be managed by Excel's built-in add-in manager. We anticipate distributing the link under the same license as the original `lp_solve` software.

REFERENCES

Bronson, R. (1982), *Operations Research*, Schaum's Outline Series, McGraw-Hill, New York, NY.

Insightful Corp. (1999), *S-Plus 6 for Window's Programmer's Guide*, Insightful Corporation, Seattle, WA.

Nemhauser, G.L. and Wolsey, L.A. (1988), *Integer and Combinatorial Optimization*, John Wiley and Sons, New York, NY.

Walkenbach, J. (1999), *Microsoft Excel 2000 Power Programming With VBA*, IDG Books, Foster City, CA.

INITIAL DISTRIBUTION LIST

1. Research Office (Code 09)..1
   Naval Postgraduate School
   Monterey, CA 93943-5000

2. Dudley Knox Library (Code 013)...2
   Naval Postgraduate School
   Monterey, CA 93943-5002

3. Defense Technical Information Center...2
   8725 John J. Kingman Rd., STE 0944
   Ft. Belvoir, VA 22060-6218

4. Associate Professor Samuel E. Buttrey 2
   Department of Operations Research
   Naval Postgraduate School
   Monterey, CA 93943-5000

5. Richard Mastowski (Editorial Assistant)..2
   Department of Operations Research
   Naval Postgraduate School
   Monterey, CA 93943-5000