OpenSolver - An Open Source Add-in to Solve Linear and Integer Progammes in Excel

Andrew J Mason

Abstract OpenSolver is an open source Excel add-in that allows spreadsheet users to solve their LP/IP models using the COIN-OR CBC solver. OpenSolver is largely compatible with the built-in Excel Solver, allowing most existing LP and IP models to be solved without change. However, OpenSolver has none of the size limitations found in Solver, and thus can solve larger models. Further, the CBC solver is often faster than the built-in Solver, and OpenSolver provides novel model construction and on-sheet visualisation capabilities. This paper describes OpenSolver's development and features. OpenSolver can be downloaded free of charge at http://www.opensolver.org.

1 Introduction

Microsoft Excel for Windows [5], and its inbuilt Solver optimiser [3], form an ideal tool for delivering optimisation systems to end users. However, Solver imposes limitations on the maximum size of problem that can be solved. OpenSolver has been developed as a freely available open-source Excel add-in for Microsoft Windows that uses the COIN-OR (Computational Infrastructure for OR) [2] CBC optimiser [1] to solve large instances of linear and integer programs.

We start by briefly describing how optimisation models are built using Solver. Readers familiar with Solver may wish to skip to the next section. A typical linear programming model and its Solver data entry are shown in Figure 1. The decision variables are given in cells C2:E2; of these, C2 must be binary and C3 must be integer. Cell G4 defines the objective function using a 'sumproduct' formula defined in terms of the decision variables and the objective coefficients in cells C4:E4. Each of the constraints are similarly defined in terms of a constraint left hand side (LHS) in G6:G9 and right hand side (RHS) in I6:I9. Note that the 'min', 'binary', 'integer'

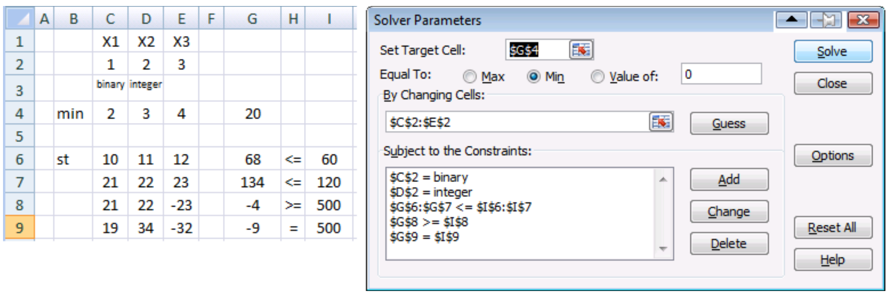
Andrew J Mason
Department of Engineering Science, University of Auckland, Private Bag 92019, Auckland, e-mail: a.mason@auckland.ac.nz

Subsequently published as:

Mason, A.J., "OpenSolver - An Open Source Add-in to Solve Linear and Integer Progammes in Excel", Operations Research Proceedings 2011, eds. Klatte, Diethard, Lüthi, Hans-Jakob, Schmedders, Karl, Springer Berlin Heidelberg
pp 401-406, 2012, http://dx.doi.org/10.1007/978-3-642-29210-1_64, http://opensolver.org

Please cite as the above. Also, see http://opensolver.org

and constraint relations shown on the spreadsheet are for display purposes only, and are not used by Solver.



	A	B	C	D	E	F	G	H	I
1			X1	X2	X3				
2			1	2	3				
3			binary	integer					
4		min	2	3	4		20		
5									
6		st	10	11	12		68	<=	60
7			21	22	23		134	<=	120
8			21	22	-23		-4	>=	500
9			19	34	-32		-9	=	500

Solver Parameters

Set Target Cell: $G$4

Equal To: Max [✓ Min] Value of: 0

By Changing Cells: $C$2:$E$2

Subject to the Constraints:

```
$C$2 = binary
$D$2 = integer
$G$6:$G$7 <= $I$6:$I$7
$G$8 >= $I$8
$G$9 = $I$9
```

Fig. 1 A typical spreadsheet optimization model (left) and the Solver entry for this. Many models do not follow this layout, but instead 'hide' the model inside the spreadsheet formulae.

2 Constructing a Mathematical Model from a Solver Model

The means by which Solver stores a model does not appear to be documented. However, using the Excel Name Manager add-in [7], developed by Jan Karel Pieterse of Decision Models UK, shows that Solver uses hidden 'names' to contain all the model's details. OpenSolver reads these values to determine the cells that define the model and Solver options.

Excel's representation of the optimisation model is given in terms of cells that contain constants and formulae. Because OpenSolver restricts itself to linear models, we wish to analyse the spreadsheet data to build an optimisation model with equations of the form:

$$\begin{aligned} &\text{Min/max} \quad c_1x_1 + c_2x_2 + \ldots + c_nx_n \\ &\text{Subject to } a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n \geq/=/\leq \quad b_i,\ i = 1,2,\ldots,m \end{aligned}$$

where $\geq/=/\leq$ denotes either $\geq$, $=$ or $\leq$. Assuming the model is linear, then the Excel data can be thought of as defining an objective function given by

$$\text{Obj}(\mathbf{x}) = c_0 + c_1x_1 + c_2x_2 + \ldots c_nx_n$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ is the vector of n decision variables, $\text{Obj}(\mathbf{x})$ is the objective function cell value, and c_0 is a constant. Similarly, each constraint equation i is defined in Excel by

$$\text{LHS}_i(\mathbf{x}) \geq/=/\leq \text{RHS}_i(\mathbf{x}) \Rightarrow \text{LHS}_i(\mathbf{x}) - \text{RHS}_i(\mathbf{x}) \geq/=/\leq 0, \quad i = 1,2,\ldots,m$$

where $\mathrm{LHS}_i(\mathbf{x})$ and $\mathrm{RHS}_i(\mathbf{x})$ are the cell values for the left hand side and right hand side of constraint i respectively given decision variable values $\mathbf{x}$. Because Solver allows both $\mathrm{LHS}_i(\mathbf{x})$ and $\mathrm{RHS}_i(\mathbf{x})$ to be expressions, we assume that both of these are linear functions of the decision variables. Thus, assuming the model is linear, we have

$$a_{i1}x_1 + a_{i2}x_2 + ...a_{in}x_n - b_i = \mathrm{LHS}_i(\mathbf{x}) - \mathrm{RHS}_i(\mathbf{x}), \quad i = 1,2,...,m.$$

OpenSolver determines the coefficients for the objective function and constraints through numerical differentiation. First, all the decision variables are set to zero, $\mathbf{x} = \mathbf{x}_0 = (0,0,...,0)$ giving:

$$c_0 = \mathrm{Obj}(\mathbf{x}_0)$$
$$b_i = \mathrm{RHS}_i(\mathbf{x}_0) - \mathrm{LHS}_i(\mathbf{x}_0), i = 1,2,...,m$$

Then, each variable x_j is set to 1 in turn on the spreadsheet, giving a sequence of decision variable values $\mathbf{x} = \mathbf{x}_j$, $j = 1,2,...,n$ where $\mathbf{x}_j = (x_{1j}, x_{2j}, ..., x_{nj})$ is a unit vector with the single non-zero element $x_{jj} = 1$. The spreadsheet is recalculated for each of these $\mathbf{x}_j$ values and the value of the objective and the left and right hand sides of each constraint recorded. This allows us to calculate the following coefficients:

$$c_j = Obj(\mathbf{x}_j) - c_0$$
$$a_{ij} = \mathrm{LHS}(\mathbf{x}_j) - \mathrm{RHS}(\mathbf{x}_j) + b_i, i = 1,2,...,m$$

The speed of this process depends on both the speed at which OpenSolver can read and write data to the spreadsheet, and the time Excel takes to re-calculate the spreadsheet. Newer versions of OpenSolver have been optimised to access larger groups of cells in each operation, reducing run times for large models. As an example, a staff scheduling model with 532 variables and 1909 constraints takes 36s to build (and just 1 second to solve) on an Intel Core-2 Duo 2.66GHz laptop.

3 Excel Integration and User Interface

OpenSolver is coded in Visual Basic for Applications and runs as an Excel add-in. The add-in presents the user with new OpenSolver controls within the standard ribbon interface that provide buttons for common operations and a menu for more advanced operations. (In earlier versions of Excel without a ribbon, OpenSolver appears as a menu item.) These OpenSolver controls are shown in Figure 2.

OpenSolver is downloaded as a single .zip file which when expanded gives a folder containing the CBC files and the OpenSolver.xlam add-in. Double clicking OpenSolver.xlam loads OpenSolver and adds the new OpenSolver buttons and menu to Excel. OpenSolver then remains available until Excel is quit. If required, the

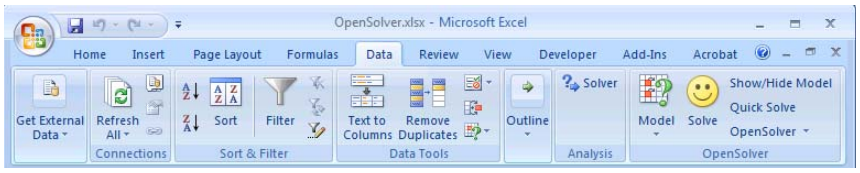


Fig. 2 OpenSolver's buttons and menu appear in Excel's Data ribbon.

OpenSolver and CBC files can be copied to the appropriate Microsoft Office folder to ensure OpenSolver is available every time Excel is launched.

Users can construct their models either using the standard Solver interface or using a new OpenSolver dialog. The new dialog provides a number of advantages, including highlighting of selected constraints on the sheet, and easier editing of constraints.

We have found OpenSolver's performance to be similar or better than Solver's. CBC appears to be a more modern optimizer than Solver's, and so gives much improved performance on some difficult problems. For example, large knapsack problems which take hours with the Excel 2007 Solver are solved instantly using OpenSolver, thanks to the newer techniques such as problem strengthening and pre-processing used by CBC [1].

To review an optimisation model developed using the built-in Solver, the user needs to check both the equations on the spreadsheet and the model formulation as entered into Solver. This separation between the equations and the model form makes checking and debugging difficult. OpenSolver provides a novel solution to this in the form of direct model visualisation on the spreadsheet. As Figure 3 shows, OpenSolver can annotate a spreadsheet to display a model in which the objective cell is highlighted and labeled min or max, the adjustable cells are shaded with any binary and integer decision variable cells being labeled 'b' and 'i' respectively, and each constraint is highlighted and its sense shown. We have found this model visualisation to be very useful for checking large models.

4 Automatic Model Construction

We have developed additional functionality that allows OpenSolver to build Solver-compatible models itself without requiring the usual step-by-step construction process. Our approach builds on the philosophy that the model should be fully documented on the spreadsheet. Thus, we require that the spreadsheet identifies the objective sense (using the keyword 'min' or 'max' or variants of these), and gives the sense ($\leq$, $=$, or $\geq$) of each constraint. Our example spreadsheet shown in Figure 1 satisfies these layout requirements.

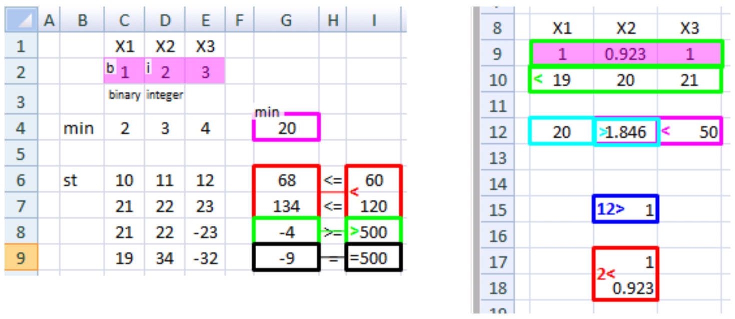
To identify the model, OpenSolver starts by searching for a cell containing the text 'min' or 'max' (or variants of these terms). It then searches the cells in the vicinity of this min/max cell to find a cell containing a formula (giving preference to any cell containing a ‘sumproduct’ formula); if one is found, this is assumed to define the objective function. The left and/or right hand side formulae for the constraints are then located in a similar fashion by searching for occurrences of $<=$ (or ‘$<$’), $=$ and, $>=$ (or ‘$>$’). The predecessor cells of all these formulae are then found, and the decision variables are then taken as those predecessors cells that have as successors either (1) at least two constraints or (2) the objective and at least one constraint.

The final step is to identify any binary or integer restrictions on the decision variables. These are assumed to be indicated in the spreadsheet by the text ‘binary’ or ‘integer’ (and variants of these) entered in the cells beneath any restricted decision variables of these types.

5 Advanced Features

OpenSolver offers a number of features for advanced users, including:

- The ability to easily solve an LP relaxation with a single menu click,
- Interaction with the COIN-OR CBC solver via the command line,
- Faster running using ‘Quick Solve’ when repeatedly solving the same problem with a succession of different right hand sides,
- Viewing of the CBC .lp input file showing the model’s equations, and,
- Detection and display of non-linearities in the model.



	A	B	C	D	E	F	G	H	I
1			X1	X2	X3				
2			b 1	i 2	3				
3			binary	integer			min		
4		min	2	3	4		20		
5									
6		st	10	11	12		68	<=	< 60
7			21	22	23		134	<=	120
8			21	22	-23		-4	>=	>500
9			19	34	-32		-9	=	=500

8	X1	X2	X3
9	1	0.923	1
10	< 19	20	21
11			
12	20	>1.846	< 50
13			
14			
15		12> 1	
16			
17		1	
18		2< 0.923	

Fig. 3 OpenSolver can display an optimisation model directly on the spreadsheet. The screenshot on the left shows OpenSolver’s highlighting for the model given earlier, while the screenshot on the right illustrates OpenSolver’s highlighting for several other common model representations.

6 User Feedback

It is difficult to determine how OpenSolver is being used, but a comment by Joel Sokol from Georgia Tech on the OpenSolver web site [6] describes one use of OpenSolver as follows:

> Thanks, Andrew! I'm using OpenSolver on a project I'm doing for a Major League Baseball team, and it's exactly what I needed. It lets the team use their preferred platform, creates and solves their LPs very quickly, and doesn't constrain them with any variable limits. Thanks again! - Joel Sokol, August 11, 2010

OpenSolver has been used to successfully solve problems which have as many as 70,000 variables and 76,000 constraints [6]. Further, users report that they appreciate being able to view the algebraic form of the Excel model given in the .lp file [4].

7 Conclusions

We have shown that it is possible to leverage the COIN-OR software to provide users with a new open source option for delivering spreadsheet-based operations research solutions. While our software is compatible with Solver, it also provides new innovative tools for visualising and building spreadsheet models that we hope will benefit both students and practitioners of spreadsheet optimisation.

Acknowledgements The author wishes to thank Iain Dunning (Department of Engineering Science, University of Auckland) who has made a huge contribution in developing the AutoModel and Model dialog code. The author also wishes to thank Kathleen Gilbert (Dept of Engineering Science) for the many improvements she has made to OpenSolver, including the non-linearity checks. The author gratefully acknowledges the contribution of Paul Becker of Eclipse Engineering for developing the Excel 2003 interface code in OpenSolver. Finally, OpenSolver would not have been possible without the huge software contributions made by the COIN-OR team.

References

1. CBC: A COIN-OR Integer Programming Solver, https://projects.coin-or.org/Cbc. Cited November 2010
2. COIN-OR: Computational Infrastructure for Operations Research, http://www.coin-or.org. Cited July 2011
3. Frontline Systems: http://www.solver.org. Cited July 2011
4. Martin, K.: 'COIN-OR: Software for the OR Community,' Interfaces 40(6), pp. 465-476, INFORMS (2011)
5. Microsoft Excel. In: Wikipedia, http://en.wikipedia.org/wiki/Microsoft_Excel. Cited July 2011
6. OpenSolver web site, http://opensolver.org. Cited July 2011
7. Pieterse, J.K.: Name Manager for Excel, http://www.jkp-ads.com/officemarketplacenm-en.asp. Cited November 2010