Operations Management Education Review 19: 161-178.
© 2025 NeilsonJournals Publishing. *Promo Version*

Solving Linear Programs in a Loop by Automating Excel Solver

Kingsley Gnanendran and Ozgur Isil
Arthur J. Kania School of Management, The University of Scranton, USA

Abstract. This paper presents an efficient method that instructors can use to demonstrate how to solve a series of related linear programs in a loop by automating the native Excel Solver using VBA (Visual Basic for Applications), the built-in programming environment in Microsoft Office. The approach is illustrated on two elementary linear programming models, the product mix problem (maximization objective) and the transportation problem (minimization objective), but this automation approach can easily be applied to any linear programming model to examine the impact of arbitrary changes in any of the model's input parameters (e.g., a right-hand side value or objective function coefficient). VBA offers students several advantages over invoking Solver via the standard Excel interface. It provides a customizable sensitivity analysis that goes beyond the fixed parameter limits in the Sensitivity Report. It requires very little coding, so no prior programming background would be expected; on the contrary, it can act as a gentle introduction to increasingly indispensable programming languages.

Keywords: Linear Programming, Sensitivity Analysis, Excel, VBA, Solver Automation.

1. Introduction

Linear Programming is a very widely taught topic in undergraduate business core courses in the areas of management science or analytics. Typically, students are taught how to solve single instances of various common classes of linear programs using the Excel interface, see e.g. Hillier and Hillier (2023), Render *et al.* (2024), Ragsdale (2022), and Taylor (2022). Students frequently will have questions about the impact on the optimal solution of a change in an input parameter, say, in a right-hand side value. While such questions can be answered quite easily if the change is within the limited range (optimality or validity) provided in the standard sensitivity analysis output, e.g. Gurobi (2024), an alternative and robust confirmation could be obtained by solving a series of linear programs by automating the native Excel Solver using VBA (Visual Basic for Applications), the built-in programming environment in Microsoft Office. It requires minimal coding, so that no prior programming background is needed. Moreover, the examined change in parameter is not limited to the range specified in the standard sensitivity analysis results. Therefore, one can examine arbitrary changes in a model parameter or even simultaneous changes over multiple parameters. Moreover, this automation approach can readily be applied to any linear programming model. The rest of the paper is structured as follows: We begin with a literature review, followed by a description of the method of invoking Excel Solver in VBA code. Then we apply this method to two different linear programming examples, the product mix problem (maximization objective) and the transportation problem (minimization objective). Finally, we summarize our main insights in the conclusion.

2. Literature Review

Achieving critical thinking skills is a common program goal in business education today. However, achieving this goal is not easy. van Gelder (2005) offers several suggestions from the field of cognitive science, including “deliberate” practice. Kydd (2012) presented a Web-based Java script applet as an active learning technique to investigate fundamental linear programing concepts. Nargundkar, Samaddar, and Mukhopadhyay (2014) describe a problem-based learning approach that is guided by the use of a “reversed” textbook, and which they show improves student performance by a statistically significant amount.

In the fields of optimization and analytics, a particular challenge has been to teach model formulation. Translating word problems to mathematical constructs (equations, inequalities) is not easy (Verschaffel, Greer, and de Corte (2000)). In this regard, the “translation” effort of Stevens and Palocsay (2004) is noteworthy. They change model formulation from an art to a science via a streamlined process: first, the measurable quantities in the problem are written down, and then a “coefficient rule” is used to determine the numerical coefficients in those algebraic expressions.

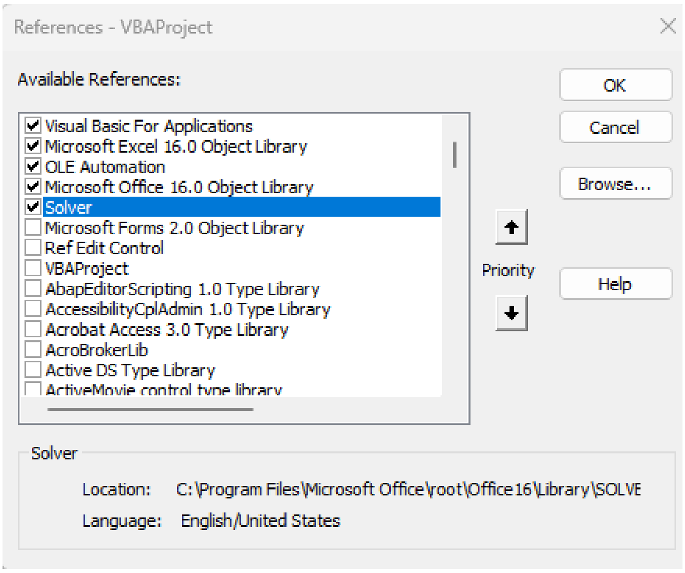
Paul and MacDonald (2015) offer a teaching supplement on sensitivity analysis of linear programs. However, that approach is limited to a change in a single parameter with all other parameters being held at the same level. Thus, the question of the impact on the optimal solution of multiple, simultaneous parameter changes is not addressed. In this paper, we propose the use of VBA coding to automate Excel Solver thereby making it quite easy to examine multiple, arbitrary changes in the input parameters.

3. Invoking Excel Solver in VBA Code

The VBA editor, see e.g. Albright (2015), can be invoked from the Excel spreadsheet by pressing Alt-F11. (This is a toggle, allowing the user to switch quickly between spreadsheet and VBA editor window.) Next, to enable the use of the built-in Excel Solver functionality, a reference needs to be set to the

Solver Library. This can be done by clicking on “Tools” on the menu bar in the VBA editor, then “References”, and finally selecting “Solver” in the dialog. (See Figure 1.)

Figure 1: Setting a Reference to the Solver Library


References - VBAProject

Available References:

- [✓] Visual Basic For Applications
- [✓] Microsoft Excel 16.0 Object Library
- [✓] OLE Automation
- [✓] Microsoft Office 16.0 Object Library
- [✓] Solver
- [ ] Microsoft Forms 2.0 Object Library
- [ ] Ref Edit Control
- [ ] VBAProject
- [ ] AbapEditorScripting 1.0 Type Library
- [ ] AccessibilityCplAdmin 1.0 Type Library
- [ ] Acrobat Access 3.0 Type Library
- [ ] AcroBrokerLib
- [ ] Active DS Type Library
- [ ] ActiveMovie control type library

Solver

Location: C:\Program Files\Microsoft Office\root\Office16\Library\SOLVE

Language: English/United States

Once in the VBA editor, the following code can be used to run Solver to solve any linear program:

```
Sub LPSolve()
    SolverReset
    SolverAdd CellRef := LHS Range, Relation := Constraint Type, FormulaText := RHS
    Range
    SolverOk SetCell := Objective Cell Range, MaxMinVal := Objective Direction, By Change
    := Decision Variables Range
    SolverSolve
End Sub
```

In the above code, the four ranges (*LHS Range, RHS Range, Objective Cell Range,* and *Decision Variables Range*) are placeholders that refer to the specific spreadsheet cells that contain the indicated model component. *Constraint Type* indicates whether the constraint is “less than or equal to” (1), “equal to” (2), or “greater than or equal to” (3). Note that a separate SolverAdd statement is required for each group of constraints. *Objective Direction* indicates whether the problem seeks to maximize the objective function (1) or minimize it (2). It is useful to create “Named Ranges” on the spreadsheet as they make the model more descriptive and the VBA code clearer to the reader.

In the following sections, we illustrate Solver automation using this VBA code on two elementary LP examples (the product mix and transportation models).

4. The Product Mix Problem

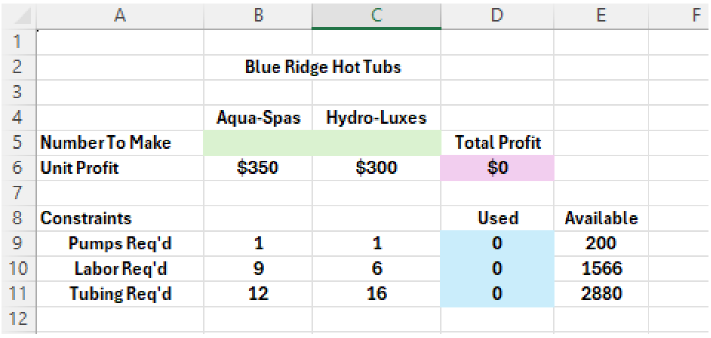
For an example of the product mix problem, we turn to Ragsdale (2022, pp. 47-64). Here is a description of the problem. *Blue Ridge Hot Tubs sells two models of hot tubs: the Aqua-Spa and the Hydro-Luxe. Each Aqua-Spa requires 1 water pump, 9 hours of labor, and 12 feet of tubing; each Hydro-Luxe requires 1 water pump, 6 hours of labor, and 16 feet of tubing. The company expects to have 200 pumps, 1,566 hours of labor, and 2,880 feet of tubing available in the next production period. If each Aqua-Spa generates a profit of $350, and each Hydro-Luxe $300, what is the optimal number of the two products that Blue Ridge must make to maximize total profit?*

With X_1 and X_2 defined as the number of Aqua-Spas and Hydro-Luxes to make, respectively, in the next production period, the linear programming model may be written as:

Maximize	$350X_1 + 300X_2$
Subject to:	
Pumps Required:	$X_1 + X_2 \leq 200$
Labor Required:	$9X_1 + 6X_2 \leq 1566$
Tubing Required:	$12X_1 + 16X_2 \leq 2880$
Non-Negativity Constraint:	$X_1, \quad X_2 \leq 0$

This problem can be set up in the Excel interface as follows. First, we enter the model data on the spreadsheet (see Figure 2).

Figure 2: Setting up the Product Mix Example



	A	B	C	D	E	F
1						
2		Blue Ridge Hot Tubs				
3						
4		Aqua-Spas	Hydro-Luxes			
5	Number To Make			Total Profit		
6	Unit Profit	$350	$300	$0		
7						
8	Constraints			Used	Available	
9	Pumps Req'd	1	1	0	200	
10	Labor Req'd	9	6	0	1566	
11	Tubing Req'd	12	16	0	2880	
12						

Next, we type the cell formulas as indicated in Table 1.