

Implementing Large-Scale Optimization Models in Excel Using VBA

Larry J. LeBlanc

Owen Graduate School of Management, Vanderbilt University,
Nashville, Tennessee 37203, larry.leblanc@owen.vanderbilt.edu

Michael R. Galbreth

Department of Management Science, Moore School of Business, University of South Carolina,
Columbia, South Carolina 29208, galbreth@moore.sc.edu

We discuss the importance of spreadsheets for optimization modeling, including a description of their limitations for large-scale problems. We then describe efficient ways to overcome these limits. Our approach makes use of Excel's standard functionality but augments Excel with its programming language, Visual Basic for Applications (VBA), where necessary. We show how using VBA within Excel to generate and solve large linear programs (LPs) overcomes many of the problems inherent in purely spreadsheet-based models and greatly increases model usability. The techniques described were instrumental in our successful development of a large-scale procurement/distribution LP that resulted in savings of approximately \$1,000,000 in the first year, with even greater annual savings expected in the future.

Key words: computers/computer science: system design, operations; planning: corporate.

History: This paper was refereed.

Spreadsheets have significantly increased managers' awareness of optimization modeling, primarily because most nontechnical managers can more easily visualize and comprehend spreadsheet models than classic algebraic models. For example, they understand such concepts as constraints more quickly when explained in spreadsheets terms (Figure 1). Conway and Ragsdale (1997), Powell (1997), Savage (1997, 2003), Willemain et al. (1997), Grossman (1999), and LeBlanc (2000) have discussed the importance of spreadsheets for explaining MS/OR concepts. MS/OR analysts have used spreadsheets successfully in developing models for many industrial applications (Jacobs and Peck 2000; Appa and Sridharan 2000; LeBlanc et al. 2000, 2004; Brown et al. 2001; Fader and Hardie 2001; Gupta et al. 2002; Srinivasan et al. 2003; Gordon and Erkut 2004; Tyagi et al. 2004).

Thus, MS/OR analysts should design and present optimization models within spreadsheets for projects that bridge theory and practice. However, spreadsheets have inherent limitations when dealing with large models, and the intuitive procedures recommended in textbooks are often unsuccessful. For the

special case of supply chain problems, we have described efficient implementation of spreadsheet linear programs (LeBlanc and Galbreth 2007). In this paper, we generalize that work, discussing the difficulties that can be encountered when developing optimization models in spreadsheets.

A Spreadsheet Linear Program (LP) for Supply Chain Optimization

We introduce the importance of efficient spreadsheet design for optimization modeling with an overview of a project in which the initial design was so inefficient that the project almost failed. Nu-kote International asked us to develop Excel-based LP models for managing its supply chain. Nu-kote is the world's largest independent remanufacturer of cartridges for printers, copiers, and fax machines, with a global network of suppliers, production facilities, and customers. Its facilities are located in various cities in the US, China, Thailand, and Mexico. In addition, empty cartridges can be sourced from brokers in Asia and Europe. Because of the lead times for shipping to and from

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
1																									
2																									
3																									
4	Demand	Units from China in Week 1 for Denver & Chicago Demands in Weeks:																							
5	Site	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20				
6	Denver	X	X	X	X	X	X	85	60	30	10	5	5	5	5										
7	Chicago	X	X	X	X	X	X	75	70	60	20	10	5	5	5										
8																									
9		X	X	X																					
10																									
11																									
12																									
13																									
14																									

$$\sum_{j \in J} \sum_{t \in T_{21j}} X_{21jt} \leq C_{21}$$

Figure 1: We represent in spreadsheet terms and algebraically the constraint that “Total goods shipped from a China manufacturing site in week 1 to all demand sites to meet their demands in all weeks cannot exceed China’s capacity in week 1.” Column A of the spreadsheet prominently shows that the demand sites are Denver and Chicago, instead of the algebraic representation by indices $j \in J$. The spreadsheet shows the source, China, and the time period for each demand site, whereas the algebraic representation denotes these by site 2 and period t , respectively. Spreadsheet cells with Xs represent shipments that are not included as changing cells (decision variables) because they cannot arrive on time. Nontechnical managers understand the spreadsheet much more easily than the algebraic representation $t \in T_{21j}$ of the demand weeks that site 2 can supply with shipments originating in week 1 to demand site j . The spreadsheet also shows the numerical values of the total shipped out of China in week 1 and the corresponding capacity side by side (V7 and X7) for easy comparison.

overseas locations, Nu-kote’s supply chain problem requires a multiperiod model.

The Nu-kote model has changing cells that specify shipments in each period of empty cartridges for remanufacturing and of remanufactured cartridges to customer sites. The model chooses shipments to meet demands in each period while minimizing total procurement, shipping, holding, and remanufacturing costs. Considerations include lead times for shipping and remanufacturing and, for each period, the limited availabilities of empty cartridges at each source, remanufacturing capacities at each site, and customer demands for remanufactured cartridges.

In our first attempt at an Excel-based model of Nu-Kote’s problem, we used a rectangular range for the changing cells and their target cell coefficients. This model worked well for the initial problem. However, as the project with Nu-Kote evolved, the model size expanded. The frequent changes in model size were time consuming, requiring constant modifications to the basic spreadsheet design. We had to insert new rows to accommodate additional sites in Mexico and later in Europe. On several occasions, we needed new columns to incorporate additional time

periods into the model. Although cell formulas adjust when rows and columns are inserted within a range, they do not adjust when ranges are expanded to include additional columns that previously bordered the range. Furthermore, we had to manually update documentation within each worksheet, such as cell comments containing formulas, when we added new rows or columns.

Eventually, the problem reached the point at which we could no longer use rectangular ranges—they wouldn’t fit in the 256 columns available in each worksheet. Thus, we modified it to use individual columns containing the changing cells and used Ragsdale’s (2007, pp. 180–182) and Winston and Albright’s (2001, pp. 210–211) approach with Excel’s SUMIF worksheet function for representing constraints. This function adds cells in a range whose corresponding cell in a comparison range has a specified value. The SUMIF naturally represents demand, capacity, and conservation-of-flow constraints by summing all changing cells that correspond to shipments into or out of the given site. In Nu-kote’s LP, we used extensive SUMIFs for constraints in each time period that limited shipments of empties to be no more than the

available supplies, required that the flow in of empty cartridges equal the flow out of remanufactured cartridges at remanufacturing sites, and required that shipments of remanufactured goods meet customer demands.

However, as the problem expanded further, the SUMIF approach proved unworkable. When solving the LP, the values of the changing cells are updated at every iteration of the solution algorithm, requiring recalculation of every SUMIF. This recalculation can take several seconds, greatly slowing the process of solving large-scale LPs that require a huge number of iterations. Nu-kote's LP had thousands of cells containing the SUMIF, each of which required comparing and adding values in tens of thousands of other cells. This overload caused frequent Excel crashes, even on a very powerful PC. Projected solution time for the LP model (if Excel did not crash) was longer than 27 hours.

Formulas also became quite complex. For example, determining whether lead times allowed shipments from one site in a specific time period to arrive at another site in another time period was complex and could be confusing to those who would inherit the model.

Nu-kote's LP grew from 12 to 78 one-week planning periods (1.5 years) and from a single cartridge type to 86 different types. At one point, the Excel file containing the LP model had nearly 39,000 rows and was more than 50 megabytes in size. Because the Excel file was enormous and unstable, we first tried to reduce the size of the model by using traditional approaches. We reduced the number of periods from 78 to 38 by aggregating some periods (Table 1). This is a natural approach given that forecasts of demands, available empty cartridges, and plant capacities are less accurate for weeks well into the future than for near-term weeks. After discussion with Nu-kote's managers, we also eliminated changing cells referring to shipping empty or remanufactured cartridges for use more than 26 weeks into the future. Thus,

the remanufacture week must be within 26 weeks of receipt of the empty cartridges and also within 26 weeks of the demand that the remanufacturing serves. These two approaches significantly reduced the number of variables in the LP.

However, even after we reduced the model size using these standard approaches, it was still so large that it caused Excel crashes. Fortunately, we were successful in implementing the LP using Visual Basic for Applications (VBA)-based techniques such as the ones described in this paper. This LP model is now being used to support strategic decisions in a global supply chain. Savings in the first year of use were approximately \$1,000,000, with much higher savings anticipated in subsequent years.

Efficient Implementation of Optimization Models in Excel

The example in the previous section demonstrates some of the potential issues with using spreadsheets for optimization modeling. Of course, spreadsheets, even when enhanced using VBA, are not appropriate for all optimization models because both model size and structure affect the complexity of the formulation and solution process. Generally, models fall into three categories:

- (1) For small models (a few hundred variables and constraints) with simple structures that don't change often, Excel without VBA is usually sufficient.
- (2) For enormous models with more than approximately 100,000 variables or constraints, spreadsheets may not be appropriate. However, spreadsheets with VBA might work well if the problems are well structured or very sparse.
- (3) A large middle ground exists between categories 1 and 2. For these problems, the inherent appeal of spreadsheets to managers is such a significant advantage that spreadsheet models augmented with VBA should be used.

In this section, we present approaches for the efficient implementation of optimization models falling

Period	1	2	3	...	22	23	24	25	26	27	28	29	...	37	38
Weeks	1	2	3	...	22	23	24	25–28	29–32	33–36	37–40	41–44	...	73–76	77–78

Table 1: We reduced the number of time periods by aggregating some intervals into a single time period.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	
1	Foreign exchange problem															
2			Exchange Rate To													
3			USA	Europe	Britain	Canada	Mexico	Japan	Yemen	Kuwait	UAE					
4			US\$	Euro	Pound	Can. \$	Peso	Yen	Rial	Dinar	Dirham					
5	Exchange Rate From	US\$	1	0.815	0.549	1.245	10.872	106.840	181.120	0.292	3.673					
6		Euro	1.227	1	0.674	1.527	13.347	131.107	222.358	0.358	4.507					
7		Pound	1.821	1.484	1	2.266	19.809	194.435	330.371	0.532	6.686					
8		Can. \$	0.803	0.655	0.441	1.000	8.746	85.833	145.491	0.235	2.951					
9		Peso	0.092	0.075	0.050	0.114	1.000	9.803	16.664	0.027	0.050					
10		Yen	0.009	0.008	0.005	0.012	0.102	1	1.696	0.003	0.034					
11		Riyal	0.006	0.004	0.003	0.007	0.060	0.590	1	0.078	0.979					
12		Dinar	3.424	2.790	1.881	4.261	37.286	365.473	12.841	1	12.575					
13		Dirham	0.272	0.222	0.150	0.339	19.822	29.062	1.021	0.080	1					
14																
15			Amount of Currency (000) Exchanged To													
16			US\$	Euro	Pound	Can. \$	Peso	Yen	Rial	Dinar	Dirham					
17	Currency Exchanged From	US\$	-	-	-	-	-	-	-	110.000	-	110	<=<	\$ 110		
18		Euro	-	-	-	-	-	-	-	-	-	-	<=<	\$ -		
19		Pound	-	175.22	-	-	-	-	75.67	7.002	8.23	266	<=<	\$ 615		
20		Can. \$	-	-	-	-	-	-	-	100.00	-	100	<=<	\$ 100		
21		Peso	-	-	-	-	-	-	-	-	-	-	<=<	\$ -		
22		Yen	-	-	-	-	8,822.43	-	-	2,077.57	-	10,900	<=<	\$ 10,900		
23		Rial	-	-	-	-	-	-	-	-	-	-	<=<	\$ -		
24		Dinar	-	-	-	-	-	-	-	-	-	-	<=<	\$ -		
25		Dirham	-	-	-	-	-	-	-	-	-	-	<=<	\$ -		
26	Total received	-	260	-	-	900	-	25,000	65	55						
27		=	=	=	=	=	=	=	=	=						
28	Required (000)	-	260	-	-	900	-	25,000	65	55						

Figure 2: In this LP for a firm needing to exchange currencies, the rectangular range C5:K13 contains fixed currency exchange rates, and changing cells in C17:K25 specify the quantities of currencies exchanged. The LP determines quantities to convert between pairs of currencies to meet the required currency amounts (rows 16 and 28) without exceeding the available currency amounts (columns B and N). Row sums in L17:L25 give the total amounts of each currency exchanged for other currencies. Column sums often represent totals received, but in this problem a SUMPRODUCT in each column gives the total receipts of the corresponding currencies. For example, J26 contains the SUMPRODUCT of the two columns above—110 US\$ @ 0.292 + 0 Euros @ 0.358 + 7 Pounds @ 0.532 + The target cell, N8, is the US dollar equivalent of the total currencies spent in L17:L25—SUMPRODUCT(L17:L25, C5:C13).

into category 3 above, a common situation in real-world applications.

Representing Changing Cells and Coefficients

It is natural to represent many optimization problems in spreadsheets using rectangular ranges for changing cells and their target cell (objective function) coefficients. Such problems include assigning managers to accounts or machines to jobs, exchanging foreign currencies, locating facilities, routing vehicles, and scheduling transportation in one or more time periods. The visual appeal of rectangular ranges is that

the available managers, machines, or currencies (supplies) are listed along the left side, and the accounts, jobs, or required currencies (demands) are listed along the top. An additional benefit of using rectangular ranges for changing cells is that row and column sums represent constraint totals in an intuitive manner (Figure 2).

Unfortunately, the structural limitations of Excel make rectangular ranges unsuitable for many large-scale applications (Figure 3). Because of Excel’s worksheet limit of 256 columns, using rectangular ranges is often awkward for real-world problems because

		Customer Sites																																							
		A								B								C								Z															
Available Vehicle		Time Periods								Time Periods								Time Periods								Time Periods															
		1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8								
1	123ZFFZ	X	X	X	X	X	X	X	X									X	X	X	X	X	X	X	X																
2	169ZOFN									X	X	X	X	X	X	X	X									X	X	X	X	X	X	X	X								
3	21LVGR																	X	X	X	X	X	X	X	X									X	X	X	X	X	X	X	X
4	235YIBN																									X	X	X	X	X	X	X	X								
5	268NIFV	X	X	X	X	X	X	X	X																	X	X	X	X	X	X	X	X								
6	381NKIN									X	X	X	X	X	X	X	X																	X	X	X	X	X	X	X	X
7	389JAKP																	X	X	X	X	X	X	X	X																
8	462PIGC																									X	X	X	X	X	X	X	X								
9	484UMXR	X	X	X	X	X	X	X	X																									X	X	X	X	X	X	X	X
10	491BCTH																	X	X	X	X	X	X	X	X																
11	496VYTZ																									X	X	X	X	X	X	X	X								
12	819GXYI	X	X	X	X	X	X	X	X																									X	X	X	X	X	X	X	X

Figure 3: This problem is to assign vehicles to customers in each time period. Column A shows the 1,000 vehicles that can be assigned, and rows 2 through 4 show the 26 customers labeled A through Z in each of eight periods. The LP with rectangular ranges for changing cells uses 209 columns, A through HA. If we use 15 periods for each customer instead of eight, the model would require at least $26 * 15 = 390$ columns, which would not fit as a single range because each sheet in the current version of Excel has only 256 columns. Cells marked with Xs represent assignments that management does not permit, which result in an inherent waste of space with rectangular ranges.

one must divide large ranges into smaller subranges. Regardless of Excel’s column limit (which will increase substantially with the release of Excel 2007), there is an inherent waste of space in using a rectangular range when customer requirements do not allow many assignments. This wasted space is repeated in the range containing the target cell coefficients of the allowable changing cells. The feasible changing cells can consist of hundreds of separate subranges. These could be passed individually to a solution engine (for example, one of Frontline System’s Solver products), but such an approach is time consuming and prone to error. Even using VBA can be infeasible because Excel limits the number of characters in dialog box entries.

Because Excel has far more rows than columns, using two or more sets of columns is a good approach to avoid the inefficiencies of rectangular ranges—one set of columns for supplies (such as available currencies, vehicles, managers, or machines) and another set for demands (such as required currencies, customers, or jobs). Each set of columns contains the labels, coefficients, changing cells, and formulas, so we can pass all changing cells to a solver as two or three column ranges of cells. Because it is difficult to identify all feasible combinations of vehicles, customers, time periods, or manufacturing sites manually, we can use

VBA to generate these columns. We give an example of this in a subsequent section.

Representing Constraints

For problems in which inputs, such as funds, machine hours, or materials, are limited, SUMIFs can naturally represent constraints. Doing so is ideal for small problems but, as noted above, can make solving large-scale LPs difficult. We illustrate this with an example of a pension fund manager planning investments from a portfolio of 2,000 possible instruments for clients in 10 age groups. The manager must consider clients’ different risk tolerances and requirements for growth and income, so she identifies different subsets of possible investment instruments for the various groups. The manager uses an LP with 8,000 combinations of investments (10 age groups multiplied by an average of 800 instruments per age group) and 2,000 constraints for upper limits on investments in each instrument. The LP model can use Excel’s SUMIF to implement constraints on the total invested in each of the 2,000 instruments (Figure 4).

In many other cases in which funds, machine hours, or materials are limited, SUMIFs also naturally represent constraints. Doing so is ideal for small problems but impractical for very large LPs. When too many cells contain such SUMIF worksheet functions,

	A	B	C
1	Age Group	Instrument	\$ Invested
2	25 to 30	1	50
3	25 to 30	7	20
4	25 to 30	9	30
850	25 to 30	1997	0
851	25 to 30	1998	50
852	31 to 35	4	30
853	31 to 35	7	0
854	31 to 35	10	40
1641	31 to 35	1999	40
1642	31 to 35	2000	20
1643	36 to 40	2	20
1644	36 to 40	3	20
1645	36 to 40	8	30
7250	66 to 70	1996	20
7251	66 to 70	2000	40
7252	71 to 75	3	0
7253	71 to 75	6	50
7254	71 to 75	15	10
8000	71 to 75	1828	30
8001	71 to 75	1831	20

	S	T	U	V
1	Instrument	Total \$ Invested		Upper Limit
2	1	0	<=	\$ 500
3	2	0	<=	\$ 400
4	3	0	<=	\$ 1,100
2001	2000	0	<=	\$ 600
2002				
2003				
2004	Total invested in instrument 2000			
2005	=+C1642+C3246+C4051			
2006	+C5657+C6451+C7251			

	S	T	U	V
1	Instrument	Total \$ Invested		Upper Limit
2	1	190	<=	\$ 500
3	2	90	<=	\$ 400
4	3	140	<=	\$ 1,100
2001	2000	170	<=	\$ 600
2002				
2003				
2004	Total invested in instrument 2000			
2005	=SUMIF(\$B\$2:\$B\$8001,\$2001,			
2006	\$C\$2:\$C\$8001)			

Figure 4: In this pension fund investment LP, changing cells in column C specify dollars invested in each combination of age group and instrument. Constraints in columns T:V limit the total investments in each of the 2,000 instruments. At the lower right, the inefficient approach has SUMIFs in column T to add the cells in C2:C8001 whose corresponding cell in the comparison range B2:B8001 equals the investment instrument in column S. At the upper right, the much more efficient approach directly sums the appropriate cells. Because the labels in columns A and B never change for any problem instance, the direct-sum formula is entirely equivalent to the SUMIF. This direct-sum formula involves at most 10 cells (one for each age group), instead of the SUMIF's reference to 8,000 cells in columns B and 8,000 additional cells in column C. There is a total of 2,000 such formula cells in T2:T2001, so the direct-sum formula approach vastly reduces the computational effort.

each referring to many comparison cells and changing cells, the recalculation is so time consuming that even a state-of-the art solver will bog down and can cause Excel itself to crash. To overcome this problem, we can eliminate the SUMIFs and instead use a formula summing only the specifically applicable cells (Figure 4).

However, because these direct-sum formulas are unique, we cannot copy and paste the first direct-sum formula throughout the rest of the range. For small problems, manual entry of unique formulas in each cell is possible. For problems with thousands of such formulas, however, the manual approach is tedious and prone to error, and the SUMIF approach is too slow. In such cases, we can use a VBA procedure to generate the formulas. Code for producing the desired formulas is shown in Figure A.1 in the appendix. This VBA consists of only a few lines of code.

Developing the Initial Model

When initially developing the labels, coefficients, changing cell ranges, and formulas in a spreadsheet optimization model, it is tempting to use Excel's copy/paste and find/replace features. However, managers often ask analysts to expand an LP model to examine the problem in greater detail, such as multi-period scenarios. With manual copy/paste and find/replace operations, such modifications can be difficult and prone to error.

Savage (2003, p. 14) refers to the problem of changing a spreadsheet's dimensionality as *hyperscaling* and notes that Geoffrion called it *dimensional arthritis*. Savage discusses how this problem arises when a small prototype LP is expanded. For example, to expand a static model to include 12 periods, one must add 11 copies of the original model, one for each new each period. To add a new plant, one must add a copy of the 12-period model. To add a new product, one must then revise all 24 ranges, which can be difficult. However, we can use VBA code to generate all aspects of an LP model easily. Figure A.2 in the appendix shows the VBA code to generate the previous pension fund investment LP with new investments and age groups (or with new machines or suppliers for other applications) to keep up with changes that occur in the system being modeled. Given this flexibility, management is more likely to use the model on an ongoing basis.

Anticipating Formula Complexity Due to Model Evolution

As a model evolves and increases in complexity, Excel formulas can become so complicated that developers may balk at making additional changes. However, they can often use VBA to include new complexities much more easily than they could using only Excel formulas. For example, in a truck assignment problem, a scheduler must assign some or all of the available vehicles to a set of customer demands. The analyst initially uses a static model for the case of a single type of truck (Figure 5). If the scheduler wants to expand the model to include a dynamic aspect with specified times for truck availabilities and requirements, the analyst can calculate the cost of assigning each available vehicle to a demand site with worksheet functions using a travel cost table,

1													
	2	3	4	Trucks Required (Location)									
				1	1	1	2	2	3	3	3		
4	1	4	\$ 400	\$ 400	\$ 400	\$ 320	\$ 320	\$ 80	\$ 80	\$ 80			
5	2	4	\$ 400	\$ 400	\$ 400	\$ 320	\$ 320	\$ 80	\$ 80	\$ 80			
6	3	3	\$ 360	\$ 360	\$ 360	\$ 200	\$ 200	\$ -	\$ -	\$ -			
7	4	3	\$ 360	\$ 360	\$ 360	\$ 200	\$ 200	\$ -	\$ -	\$ -			
8	5	3	\$ 360	\$ 360	\$ 360	\$ 200	\$ 200	\$ -	\$ -	\$ -			
9	6	2	\$ 160	\$ 160	\$ 160	\$ -	\$ -	\$ 200	\$ 200	\$ 200			
10	7	2	\$ 160	\$ 160	\$ 160	\$ -	\$ -	\$ 200	\$ 200	\$ 200			
11	8	2	\$ 160	\$ 160	\$ 160	\$ -	\$ -	\$ 200	\$ 200	\$ 200			
12	9	1	\$ -	\$ -	\$ -	\$ 160	\$ 160	\$ 360	\$ 360	\$ 360			
13	10	1	\$ -	\$ -	\$ -	\$ 160	\$ 160	\$ 360	\$ 360	\$ 360			
14	11	1	\$ -	\$ -	\$ -	\$ 160	\$ 160	\$ 360	\$ 360	\$ 360			
15		Etc.	Etc.										

Figure 5: In this truck assignment spreadsheet, the assignment costs in D4:K15 are given data and are input manually into this range. Available vehicles are shown in B2:C15. We use a rectangular range of assignment costs, D4:K15, for this small example problem.

assuming that the vehicle can arrive in time to meet the demand. If the vehicle cannot arrive in time, the cost is set to \$9,999 to preclude the combination (Figure 6). Enhancing the model in this way is within the ability of most skilled developers.

Next, suppose that the model is so successful that the scheduler wants to expand it to include multiple types of vehicles. The analyst can revise the model by replacing “=IF” formulas with “=IF(AND(...” formulas to test whether the available vehicle can get to

				Trucks Required (Location and Date/Time)								
1	2	3	4	1	1	1	2	2	3	3	3	
				1/5 00:00	1/5 00:00	1/5 00:00	1/6 02:17	1/6 08:17	1/9 01:02	1/9 01:02	1/9 01:02	
4	1	4	1/2 16:00	\$ 400	\$ 400	\$ 400	\$ 320	\$ 320	\$ 80	\$ 80	\$ 80	
5	2	4	1/2 16:00	\$ 400	\$ 400	\$ 400	\$ 320	\$ 320	\$ 80	\$ 80	\$ 80	
6	3	3	1/3 19:30	\$ 360	\$ 360	\$ 360	\$ 200	\$ 200	\$ -	\$ -	\$ -	
7	4	3	1/3 19:30	\$ 360	\$ 360	\$ 360	\$ 200	\$ 200	\$ -	\$ -	\$ -	
8	5	3	1/3 19:30	\$ 360	\$ 360	\$ 360	\$ 200	\$ 200	\$ -	\$ -	\$ -	
9	6	2	1/4 15:50	\$ 9,999	\$ 9,999	\$ 9,999	\$ -	\$ -	\$ 200	\$ 200	\$ 200	
10	7	2	1/4 15:50	\$ 9,999	\$ 9,999	\$ 9,999	\$ -	\$ -	\$ 200	\$ 200	\$ 200	
11	8	2	1/4 15:50	\$ 9,999	\$ 9,999	\$ 9,999	\$ -	\$ -	\$ 200	\$ 200	\$ 200	
12	9	1	1/4 23:45	\$ -	\$ -	\$ -	\$ 160	\$ 160	\$ 360	\$ 360	\$ 360	
13	10	1	1/4 23:45	\$ -	\$ -	\$ -	\$ 160	\$ 160	\$ 360	\$ 360	\$ 360	
14	11	1	1/4 23:45	\$ -	\$ -	\$ -	\$ 160	\$ 160	\$ 360	\$ 360	\$ 360	
15		Etc.	Etc.									

Table of Travel Days To City					Table of Travel Costs To City						
		1	2	3	4			1	2	3	4
From City	1		0.40	0.90	1.00			\$ -	\$ 160	\$ 360	\$ 400
	2	0.40		0.50	0.80			\$ 160	\$ -	\$ 200	\$ 320
	3	0.90	0.50		0.33			\$ 360	\$ 200	\$ -	\$ 132
	4	1.00	0.80	0.20				\$ 400	\$ 320	\$ 80	\$ -
		Etc.						Etc.			

Figure 6: Truck availability locations and times are in C4:D15, and required truck locations and pick-up times are in E2:L3. Each assignment cost in E4:L15 is calculated with an IF worksheet function that tests whether the vehicle can arrive at the required location in time.

	B	C	D	E	F	G	H	I	J	K	L	
1		Substitution Costs for Truck Types										
2		Platform	Flatbed	Van	Box	Etc.		Truck Types				
3	Platform	\$ -	\$ 50	\$ 200	\$ 100			Platform				
4	Flatbed	\$ 100	\$ -	\$ 300	\$ 200			Flatbed				
5	Van	\$ 75	\$ 100	\$ -	\$ 35			Van				
6	Box	\$ -	\$ 99,999	\$ 99,999	\$ -			Box				
7	Etc.			Etc.				Etc.				
8												
9												
10	Type	Type	Cost To									
11	Requested	Assigned	Substitute									
12	Platform	Flatbed	\$ 50									
13	Van	Box	\$ 35									
14	Van	Flatbed	\$ 100									
15		Etc.	Etc.									

Figure 7: Additional truck substitution costs (losses in profits) are calculated by matching the names of the requested and assigned truck types in the named range “Truck_Types” (\$I\$3:\$I\$7). The positions in this range are used to offset from \$B\$2.

the required location in time and whether the vehicle type is the same type as required.

These approaches work well to a point but can become troublesome as models become very complex. For example, suppose that the scheduler wants to allow substitution of some types of vehicles for others at a small loss in profit, such as a platform truck at a discount for a flatbed truck. The cost of the discount could be accommodated (Figure 7), but the analyst

may balk at incorporating the challenging formulas needed to do so. Given the frequency with which analysts hand off MS/OR models, such a complex model is not a good long-term solution for a company. However, VBA makes such complexities easier to manage. This is discussed in the appendix (Figure A.3).

Because model complexity often increases over time, analysts should consider using VBA even when the initial model does not seem to require it. Subsequent

	A	B	C	D	E	F	G	H	I	J	K	
1		Overall Cost Coefficients (US\$ Per Unit Shipped)										
2		MEX	CAN	VEN	FRA	GAR	SUN					
3	MEX	\$ 98.88	\$100.10	\$149.07	\$109.45	\$104.50	\$106.20				\$5,297.93	
4	CAN	\$124.23	\$ 72.97	\$115.57	\$ 87.72	\$ 80.33	\$ 86.19					
5	VEN	\$ 87.84	\$ 54.91	\$ 54.89	\$ 60.13	\$ 57.38	\$ 58.21					
6	FRA	\$ 87.96	\$ 57.85	\$ 84.95	\$ 51.61	\$ 59.96	\$ 61.91					
7	GAR	\$112.13	\$ 69.83	\$106.75	\$ 79.89	\$ 63.83	\$ 80.16					
8	SUN	\$173.50	\$108.48	\$162.65	\$118.78	\$113.35	\$108.36					
9												
10		Amounts Shipped					Total Sent	Plant				
11		MEX	CAN	VEN	FRA	GAR	SUN	From	<=	Capacity		
12	MEX	3.0	0.0	0.0	0.0	0.0	3.2	6.2	<=	22.0		
13	CAN	0.0	2.6	0.0	0.0	1.1	0.0	3.7	<=	3.7		
14	VEN	0.0	0.0	4.5	0.0	0.0	0.0	4.5	<=	4.5		
15	FRA	0.0	0.0	11.5	20.0	6.8	8.7	47.0	<=	47.0		
16	GAR	0.0	0.0	0.0	0.0	18.5	0.0	18.5	<=	18.5		
17	SUN	0.0	0.0	0.0	0.0	0.0	0.0	0.0	<=	0.0		
18	Total	3.0	2.6	16.0	20.0	26.4	11.9					
19	Sent TO	=	=	=	=	=	=					
20	Demand	3.0	2.6	16.0	20.0	26.4	11.9					
21												
22												

Figure 8: In the original Applichem LP, overall cost coefficients in B3:G8 are calculated using current foreign-currency exchange rates. Annual plant capacities and customer demands in millions of pounds are in J12:J17 and B20:G20, respectively. The optimal solution, shown in B12:G17, calls for closing the SunChem plant because there are no shipments from it.

	A	B	C	D	E	F	G	H	I	J	K	
25		Overall Cost Coefficients (Foreign Currency Units)								Exchange Rates/US\$		
26		MEX	CAN	VEN	FRA	GAR	SUN			Original	Simulated	
27	MEX	925.5	936.9	1,395.3	1,024.4	978.1	994.0		MEX	9.36	4.91	
28	CAN	182.6	107.3	169.9	128.9	118.1	126.7		CAN	1.47	0.79	
29	VEN	60,346.2	37,722.0	37,712.0	41,307.6	39,419.4	39,989.0		VEN	687.00	821.23	
30	FRA	81.8	53.8	79.0	48.0	55.8	57.6		FRA	0.93	0.87	
31	GAR	112.1	69.8	106.7	79.9	63.8	80.2		GAR	1.00	1.00	
32	SUN	18,930.8	11,836.0	17,747.0	12,960.4	12,368.0	11,823.0		SUN	109.11	134.27	
33												
34		Formulas for Simulated Overall Cost Coefficients (US\$ Per Unit)								Simulated Japanese yen/US\$:		
35		MEX	CAN	VEN	FRA	GAR	SUN			=0.5*J32+RAND()*J32 (Between half of original value and 1.5 times original value)		
36	MEX	\$ 188.3	\$ 190.6	\$ 283.9	\$ 208.5	\$ 199.0	\$ 202.3					
37	CAN	\$ 231.3	\$ 135.8	\$ 215.1	\$ 163.3	\$ 149.5	\$ 160.4					
38	VEN	\$ 73.5	\$ 45.9	\$ 45.9	\$ 50.3	\$ 48.0	\$ 48.7					
39	FRA	\$ 94.0	\$ 61.8	\$ 90.8	\$ 55.2	\$ 64.1	\$ 66.2					
40	GAR	\$ 112.1	\$ 69.8	\$ 106.7	\$ 79.9	\$ 63.8	\$ 80.2					
41	SUN	\$ 141.0	\$ 88.1	\$ 132.2	\$ 96.5	\$ 92.1	\$ 88.1					

Figure 9: The revised Applichem LPs use simulated currency exchange rates. B27:G32 contains the overall cost coefficients in foreign currencies, and I25:K32 gives the original and simulated exchange rates. Excel formulas simulate exchange rates using “[Smallest Possible Value] + [Random 0 to 1] * [Range of Possible Values].” Formulas in B36:G41 convert the costs from foreign currencies to US dollars using the simulated exchange rates. VBA copies the values from the formulas in B36:G41 to a separate range B3:G8 (Figure 8)—not the formulas themselves because formulas recalculate during the Solver solution process. If B3:G8 contained formulas, then the overall cost coefficients would change while Solver was finding the optimal solution.

analysts who inherit the model will appreciate the ease with which they can manage added complexities.

Using VBA to Control Multiple Runs of an LP

Many real-world problems require that an analyst set up and solve an LP numerous times, with each run varying slightly from the previous run. VBA greatly simplifies the process of passing the model to a solver and presenting the solution in a user-friendly format. For example, consider the suggestion by Huchzermeier (2005) of updating the Applichem (1986) LP to reflect future exchange rates. Applichem produces a chemical at different plants that serve the same markets—Mexico, Canada, Venezuela, Frankfurt (Germany), Gary (USA), and the SunChem site in Japan. The overall cost coefficients (Figure 8) for shipping one unit from a source country to a market country include costs for manufacturing, transportation, and (when applicable) import duties. However, these cost coefficients depend on the exchange rates for foreign currencies, which change and thus could change the optimal shipments. The solution to the LP with current exchange rates has no shipments from SunChem, suggesting that this plant be closed (Figure 8).

To analyze the decision of whether or not to close SunChem, we assume that future exchange rate values will be uniformly distributed in the following interval:

$$[0.5 * \text{Original Exchange Rate}, \\ 1.5 * \text{Original Exchange Rate}].$$

We then generate a wide range of potential scenarios using a simulation/optimization model with 1,000 replications:

- (1) Simulate each exchange rate.
- (2) Solve two LPs whose overall cost coefficients result from the simulated exchange rates: one with the SunChem plant open and another with it closed.
- (3) Record the overall costs for these two scenarios.

We compare the average costs with the SunChem plant open and with it closed (Figure 9). At each iteration, we use VBA to

- Copy the new overall cost coefficients from one range to another,
- Solve the two resulting LPs with the SunChem plant open and with it closed, and
- Accumulate the total costs for each LP.

Figure A.4 (appendix) shows the VBA code to perform these steps. Running this VBA to solve the simulation/optimization model showed that the cost

increase from closing the SunChem plant would be only 1.5 percent. This is important information that Applichem managers can use to make an informed decision regarding the plant.

Conclusions and Lessons Learned

Spreadsheets make optimization models accessible to managers. Thus, for researchers wanting to bridge the gap to practice, the ability to model complex real-world problems in spreadsheets is valuable. We have successfully implemented large-scale LPs in Excel, and in developing these LPs, we learned valuable lessons regarding the scalability and flexibility of established spreadsheet modeling techniques:

—Manually developed models lack the high degree of flexibility of those developed with VBA. With VBA, one can easily create new versions of the model based on updated user preferences.

—Except for small problems, rectangular ranges are often wasteful and inefficient. Using a column-based approach, generated by VBA, is preferable.

—The SUMIF function, although ideal for small models, is inadequate for many practical-sized problems and should be replaced with a VBA-generated formula specifying the equivalent direct sum of applicable cells.

—Using VBA for model parameter calculations is appropriate for complex situations, particularly when one anticipates model evolution and hand-off among developers.

—It is easier to control multiple runs of an LP, each solving a different variation of the problem, with VBA than manually.

The approaches presented in this paper will help researchers overcome the challenges of spreadsheet modeling and create robust, user-friendly tools for real-world applications.

Appendix

Direct Sum Formulas for the Pension Fund Investment LP

This VBA code (Figure A.1) produces the direct-sum formulas to replace the inefficient SUMIF worksheet functions for the pension fund investment LP (Figure 4). The code assumes that several variables (TopRowInstr, MyCompareCol, RowTopAgeInstr, ...) have been defined from named ranges based on the layout of the worksheet. (TopRowInstr and RowTopAgeInstr are the top row numbers of named ranges referencing S2:V2001 and A2:C8001 in Figure 4. The use of named ranges ensures that the variables adjust if new rows or columns are inserted.)

Model Regeneration for the Pension Fund Investment LP

VBA code can quickly regenerate the pension fund investment LP model (Figure 4) with new investments and age groups. This allows the model to keep up with changes that occur in the system (Figure A.2).

```
For I = TopRowInstr to BotRowInstr
    Criterion = Range(MyCompareCol & I)
    ,
    Range(MyTotCol & I) = SumCells(Criterion)
Next I
.
.
.
Function SumCells(Criterion)
SumCells = "="
For I = RowTopAgeInstr To RowBotAgeInstr
    ' If cell in column B equals Criterion, augment the formula's cell reference by
    ' concatenating the column letter and row number:
    If Range(MyInstrCol & I) = Criterion Then SumCells = SumCells & "+" & MyColLetter & I
    ,
    MyInstrCol is "B"
    MyColLetter is "C"
Next I
End Function
```

Figure A.1: This VBA code produces the direct-sum formulas to replace the SUMIF worksheet functions that are very inefficient for large problems.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	Age Group	Instrument	\$ Invested		A1 and A2 are named Top1 and Top2, respectively			Number of instruments in age group 1									
2	25 to 30	1	50					1	2	3	4	5	6	7	8	9	10
3	25 to 30	7	20					850	791	809	795	805	800	806	794	800	750
4	25 to 30	9	30					1	4	2	1	1	3	1	1	1	3
5	25 to 30	10			Number of Age Groups			7	7	3	2	11	5	2	16	3	6
6	25 to 30	13	10		10			9	10	8	3	14	8	3	19	5	15
7	25 to 30	15						10	13	11	15	17	28	10	22	36	16
8	25 to 30	16						13	15	14	19	19	32	13	25	38	17
9	25 to 30	19			Number of Investments			15	18	16	21	22	35	15	26	41	18
10	25 to 30	22	20		2,000			16	21	18	22	24	38	18	29	42	20
11	25 to 30	23						19	24	21	25	25	41	21	32	45	21
12	25 to 30	24	10					22	27	23	27	28	44	24	34	47	22
13	25 to 30	25	10					23	30	26	31	31	47	27	36	49	25
14	25 to 30	28						24	33	27	34	33	49	29	37	52	28
15	25 to 30	31						25	35	30	35	34	51	32	40	53	31
16	25 to 30	34						28	38	33	38	35	53	35	43	55	34
17	25 to 30	37	10					31	41	36	40	38	54	38	46	58	35
18	25 to 30	40			Lower Age	Upper Age		34	44	39	43	39	57	40	48	61	38
19	25 to 30	42	20		25	30		37	45	42	45	42	58	41	49	62	41
20	25 to 30	43	10		31	35		40	48	43	46	43	62	43	52	64	44
21	25 to 30	46	20		36	40		42	51	46	49	44	65	46	55	67	47
22	25 to 30	47			41	45		43	52	49	50	46	67	49	57	70	50
23	25 to 30	48	10		46	50		46	55	50	51	49	70	52	58	73	53
24	25 to 30	51	10		51	55		47	56	51	53	50	72	56	60	76	56
25	25 to 30	54			56	60		48	59	53	57	54	75	60	63	77	59
26	25 to 30	56			61	65		51	62	56	61	57	78	63	64	80	62
27	25 to 30	58	20		66	70		54	64	57	63	59	81	66	65	84	63
28	25 to 30	61	20		71	75		56	67	58	65	61	84	67	68	87	65

```

NumAgeGrps = Range("Age_Groups") 'Define the number of age groups (E7)

For i = 1 To NumAgeGrps
  Lower(i) = Range("Low_Age_Anchor").Offset(i, 0) 'Define starting and ending ages identifying each age group (E18:F18)
  Upper(i) = Range("Up_Age_Anchor").Offset(i, 0)
Next i

For i = 1 To NumAgeGrps
  NumAllowed(i) = Range("Anchor").Offset(0, i) 'Define number of allowable instruments for each age group (G3)
  For j = 1 To NumAllowed(i)
    Allowed(i, j) = Range("Anchor").Offset(j, i) 'Instruments allowed for age group i
  Next j
Next i

' Clear earlier labels and changing cells
Range("Output_1").Clear 'Output_1 is A2:C8001

For i = 1 To NumAgeGrps
  For j = 1 To NumAllowed(i)
    NextRow = NextRow + 1
    Range("Top1").Offset(NextRow, 0) = Lower(i) & " to " & Upper(i)
    Range("Top2").Offset(NextRow, 0) = Allowed(i, j)
  Next j
Next i

'Etc.-produce column T (see Figures 4 and A1)

```

Figure A.2: This VBA code creates the labels in columns A and B for the pension fund investment LP.

Substitution Costs for Vehicle Assignment LP

The VBA code for the case of multiple truck types with a substitution cost whenever the assigned truck type is different from the one requested (Figure A.3)

is part of a larger macro that sets up the entire worksheet. The code assumes that travel costs and times and vehicle substitution costs have been initialized. The only change in the VBA code to accommodate

```
' Find the cost of assigning a type-I vehicle at available site i to a customer at
' site j wanting a type-J vehicle:
If AvailableTime + TravelTime > RequiredTime Then
    Cost = 9999          ' If the vehicle can't arrive on time, the cost is infinite
Else
    Cost = TravelCost(i,j) + SubstituteCost(I,J)
End If
```

Figure A.3: This VBA code easily extends the assignment cost to include substitution costs.

```
Sub Solve_Applichem_LP()

For LP_Run = 1 To Num_Replications          'Solve LP models--Sun open then closed

'Copy values from simulated costs to the LP model:
ActiveSheet.Range("B36:G41").Copy
ActiveSheet.Range("B3:G8").PasteSpecial (xlPasteValues)

Range("J17") = Sun_Cap                      'Sun is open with original capacity of 5
Stop_Value = SolverSolve(True)              'Find LP solution
Cost_Sun_Open = Cost_Sun_Open + Range("I3")

Range("J17") = 0                            'Sun is closed with capacity of 0
Stop_Value = SolverSolve(True)              'Find LP solution
Cost_Sun_Closed = Cost_Sun_Closed + Range("I3")

Next LP_Run

Range("M3") = Cost_Sun_Open / Num_Replications 'Record the average
Range("N3") = Cost_Sun_Closed / Num_Replications 'costs in M3 and N3

End Sub
```

Figure A.4: This VBA code solves Applichem LPs with costs from simulated exchange rates.

vehicle substitutability is the underlined “+ SubstituteCost(I,J)” term. This single code change is not only easy to make, but its function will also be clear to other developers who have responsibility for the model in the future.

VBA Code for Applichem LP

This code solves the simulation/optimization model for the Applichem case (Figure A.4). The variables Sun_Cap and Num_Replications were previously initialized at 5 and 1,000. The process of copying and pasting causes the Excel exchange rate formulas to recalculate so that new exchange rates, and thus new overall cost coefficients, result for the next LP (Figures 8 and 9).

References

Appa, G., R. Sridharan. 2000. OR helps the poor in a controversial irrigation project. *Interfaces* 30(2) 13–28.

Applichem (A). 1986. Harvard Business School Case 9-685-051. Harvard Business School, Boston, MA.

Brown, G., J. Keegan, B. Vigus, K. Wood. 2001. The Kellogg Company optimizes production, inventory, and distribution. *Interfaces* 31(6) 1–15.

Conway, D., C. Ragsdale. 1997. Modeling optimization problems in the unstructured world of spreadsheets. *Omega Internat. J. Management Sci.* 25(3) 313–332.

Fader, P., B. Hardie. 2001. Forecasting repeat sales at CDNOW: A case study. *Interfaces* 31(3) S94–S107.

Gordon, L., E. Erkut. 2004. Improving volunteer scheduling for the Edmonton Folk Festival. *Interfaces* 34(5) 357–376.

Grossman, T. 1999. Teachers’ forum: Spreadsheet modeling and simulation improves understanding of queues. *Interfaces* 29(3) 88–103.

Gupta, V., E. Peters, T. Miller, K. Blyden. 2002. Implementing a distribution-network decision-support system at Pfizer/Warner-Lambert. *Interfaces* 32(4) 28–45.

Huchzermeier, A. 2005. The real option value of operational and managerial flexibility in global supply chain networks. M. Frenkel, U. Hommel, M. Rudolf, eds. *Risk Management: Challenge and Opportunity*, 2nd ed. Springer-Verlag, New York, 609–629.

- Jacobs, D. P., J. C. Peck. 2000. A simple heuristic for maximizing service of carousel storage. *Comput. Oper. Res.* **27**(13) 1351–1356.
- LeBlanc, L., M. Galbreth. 2007. Designing large-scale supply chain linear programs in spreadsheets. *Comm. ACM* **50**(8).
- LeBlanc, L., J. Hill, G. Greenwell, A. Czesnat. 2004. Nu-kote's spreadsheet linear programming models for optimizing transportation. *Interfaces* **34**(2) 139–146.
- LeBlanc, L., D. Randels Jr. T. Swain, E. Redden. 2000. Heery International's spreadsheet optimization model for assigning managers to construction projects. *Interfaces* **30**(6) 95–106.
- LeBlanc, L. J. 2000. Teaching management science/operations research using spreadsheets. *Ricerca Operativa* **30**(94/95) 75–100.
- Powell, S. 1997. The teacher's forum: From intelligent consumer to active modeler, two MBA success stories. *Interfaces* **27**(3) 88–98.
- Ragsdale, C. 2007. *Spreadsheet Modeling and Decision Analysis*, 5th ed. Thomson/South-Western, Mason, OH.
- Savage, S. 1997. Weighing the pros and cons of decision technology in spreadsheets. *OR/MS Today* **24**(1) 42–45.
- Savage, S. 2003. *Decision Making with Insight*. Thomson Brooks/Cole, Belmont, CA.
- Srinivasan, M., S. Ebbing, A. Swearingen. 2003. Woodward Aircraft Engine Systems sets work-in-process levels for high-variety, low-volume products. *Interfaces* **33**(4) 61–69.
- Tyagi, R., P. Kalish, K. Akbay, G. Munshaw. 2004. GE Plastics optimizes the two-echelon global fulfillment network at its high performance polymers division. *Interfaces* **34**(5) 359–366.
- Willemain, T., E. Jordan, L. Lasdon, M. Lenard, J. Moore, S. Powell. 1997. OR/MS and MBAs: Mediating the mismatches. Report of the operating subcommittee of the INFORMS business school education task force. *OR/MS Today* **24**(1) 36–41.
- Winston, W., S. Albright. 2001. *Practical Management Science*. Thomson/South-Western, Mason, OH.

Greg Greenwell, director, business development, Nukote, wrote on October 5, 2005: "We estimate that savings from the use of the linear programming model developed by LeBlanc and Galbreth have been approximately \$1,000,000 in the past 12 months. Anticipated future annual savings are estimated to be in the millions of dollars.

"We have used these models to support strategic decisions regarding when and where to send products for remanufacturing and shipping to customers. Specifically, we have opened a warehouse in Macau, and our operations in China have increased significantly. All key strategic supply chain decisions have been supported by the results obtained from the linear program.

"Sufficient confidence has been developed in the model that it is anticipated that continued use of this linear program in the future will be critical as our product line changes to reflect the dynamic nature of our market."