Spreadsheets in Education (eJSiE)

Volume 9 | Issue 2 Article 6

11-17-2016

Cryptarithms: A Non-Programming Approach Using Excel

Keith Luoma
Augusta University, kluoma@augusta.edu

Follow this and additional works at: http://epublications.bond.edu.au/ejsie

This work is licensed under a Creative Commons Attribution-Noncommercial-No Derivative Works 4.0 License.

Recommended Citation

Luoma, Keith (2016) Cryptarithms: A Non-Programming Approach Using Excel, *Spreadsheets in Education (eJSiE)*: Vol. 9: Iss. 2, Article 6.
Available at: http://epublications.bond.edu.au/ejsie/vol9/iss2/6

This Regular Article is brought to you by the Bond Business School at ePublications@bond. It has been accepted for inclusion in Spreadsheets in Education (eJSiE) by an authorized administrator of ePublications@bond. For more information, please contact Bond University's Repository Coordinator.

Cryptarithms: A Non-Programming Approach Using Excel

Abstract
This paper presents a method for solving cryptarithms (variously known as alphametics, crypto-arithmetics, arithmetical restorations, and verbal arithmetic) using Microsoft Excel along with the Solver add-in. In order to make it suitable for classroom presentation, elements of the method are introduced incrementally, first as a linear program, then as an integer linear program, and finally as a integer linear program with "all different" variables. Unlike other approaches discussed in the literature (the "brute force" method, logic programming, and parallel genetic algorithms), this method does not require a knowledge of computer programming. It also introduces users of Excel to the often neglected integer and alldifferent constraints, as well as the increase in computational time associated with these constraints.

Keywords
cryptarithms, alphametics, solver linear programming, alldifferent constraint

Distribution License
This work is licensed under a Creative Commons Attribution-Noncommercial-No Derivative Works 4.0 License.

1. Introduction

A cryptarithm is a puzzle where a mathematical operation involving numbers is encrypted by using letters of the alphabet to replace the numerical digits. The following is a well-known example [1].

SEND + MORE = MONEY

Generally speaking, the following rules are assumed:

1. Each letter represents a decimal digit (that is, one of {0,1,2,3,4,5,6,7,8,9}).
2. If a letter occurs more than once, in each occurrence it represents the same digit.
3. Different letters represent different digits.
4. The first letter in a word does not represent the number 0.

Solving a cryptarithm involves finding the values of the letters that satisfy the equation. For the above example, the reader can verify that the solution is given by 9567 + 1085 = 10652

Thus, S = 9, E = 5, N = 6, D = 7, M = 1, O = 0, R = 8, and Y = 2.

Other names for cryptarithms include alphametics, crypto-arithmetics, arithmetical restorations, or verbal arithmetic. Cryptarithms are popular in recreational mathematics [2], but they have also been the objects of serious study [3], [4], [5].

Cryptarithms can be solved using a combination of logic and trial and error, although doing so may be quite challenging. An example of the required logic may be found in [6].

With the aid of a computer, cryptarithms can also be solved by brute force: There can be at most 10 different letters, each corresponding to the number 0-9, so an upper bound on the numbers of trials is 10! A series of nested loops—easily represented in any procedural computer language—can be used to find the solution [6],[7]. Although cryptarithms are often used to illustrate the brute force in computer programming, other, more sophisticated methods for solving them include logic programming [8] and evolutionary methods [5].

The goal of this paper is to show that cryptarithms can be solved using spreadsheets, using nothing other than Excel and the freely available Solver add-in. The ideal time to introduce cryptarithms to students would be just after basic linear programming problems have been covered. Two things will be accomplished. First, students will be exposed to the formulation of a non-trivial and atypical problem, and second, students will be introduced to the seldom covered Integer and AllDifferent constraints in Excel.

Mathematical programming refers to optimization problems consisting of an objective function and a set of constraints. The goal is to find values of the variables that maximize (or minimize) the objective function while satisfying all of the constraints. If the objective function and constraints are all linear, then the problem is known as a linear program.

2. Basic Setup

In order to solve the cryptarithm SEND + MORE = MONEY, we need to define the variables, formulate an objective function, and determine the constraints.

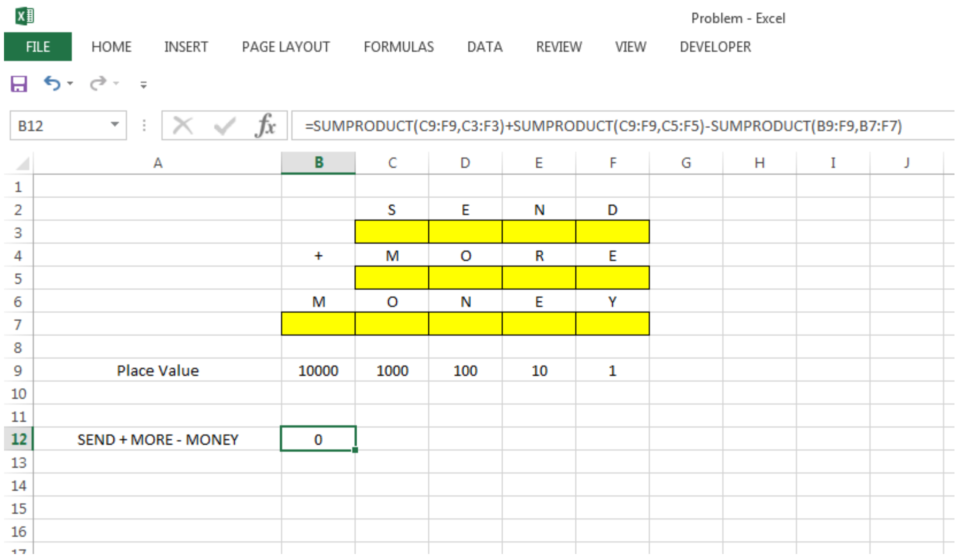
The variables consist of the set of letters contained in the cryptarithm. Thus, the variables are members of the set {D,E,M,N,O,R,S,Y}.

Next, we need to construct an objective function. To do so, we subtract "MONEY" from each side of the cryptarithm, resulting in SEND + MORE – MONEY = 0. Thus, our objective function is SEND + MORE – MONEY, with its desired value being 0.

At the heart of this problem are the constraints. Perhaps the most obvious constraints are those specifying the range of values on each variable. Specifically:

- Each letter must be at least zero and no more than 9.
- The initial letter must be at least 1.
- Identical letters must represent the same number (for example, all three E's must represent the same numerical value).

As a first approximation to the answer, we will formulate the above using Excel. In Figure 1, the cell below each letter will correspond to its numerical value. Thus, C3:F3, C5:F5, and B7:F7, highlighted in yellow, represent the variables.



B12 =SUMPRODUCT(C9:F9,C3:F3)+SUMPRODUCT(C9:F9,C5:F5)-SUMPRODUCT(B9:F9,B7:F7)

	A	B	C	D	E	F	G	H	I	J
1										
2			S	E	N	D				
3			🟨	🟨	🟨	🟨				
4		+	M	O	R	E				
5			🟨	🟨	🟨	🟨				
6		M	O	N	E	Y				
7		🟨	🟨	🟨	🟨	🟨				
8										
9	Place Value	10000	1000	100	10	1				
10										
11										
12	SEND + MORE - MONEY	0								
13										
14										
15										
16										

Figure 1

Row 9 shows the place value corresponding to the variables found in the corresponding columns. For example, "MONEY" is equal to the number 10000*M + 1000*O + 100*N + 10*E + 1*Y. An efficient way to perform this product is by using the SUMPRODUCT function, SUMPRODUCT(B9:F9,B7:F7).

We are now ready to construct the objective function, whose value resides in B12: SUMPRODUCT(C9:F9,C3:F3)+SUMPRODUCT(C9:F9,C5:F5)-SUMPRODUCT(B9:F9,B7:F7).

We also need the following groups of constraints, which are entered in the Solver Parameters box.

Each digit must be positive.

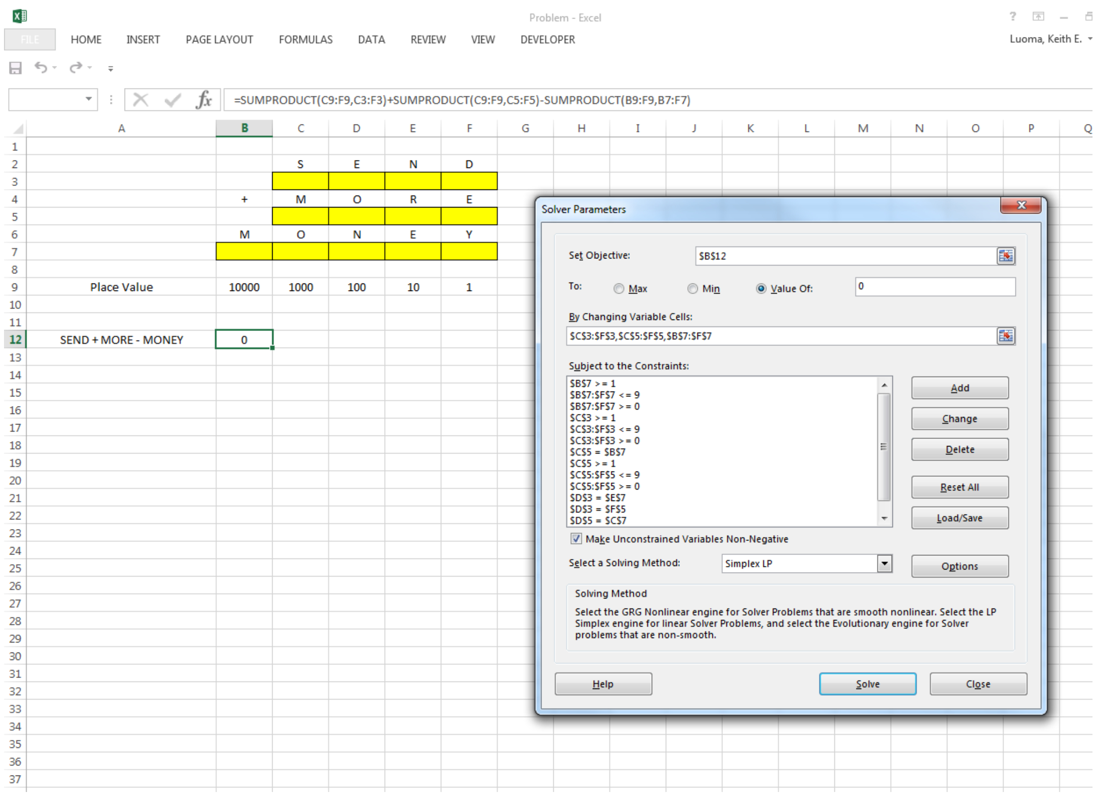
$C$3:$F$3>=0
$C$5:$F$5>=0
$B$7:$F$7>=0

Each digit must be less than or equal to 9.
$C$3:$F$3<=9
$C$5:$F$5<=9
$B$7:$F$7<=9

The first digit of each number must be greater than or equal to one.
$C$3>=1
$C$5>=1
$B$7>=1

Identical letters represent the same number.
$D$3=$F$5
$D$3=$E$7
$E$3=$D$7
$C$5=$B$7
$D$5 = $C$7

These constraints are shown as entered in the Solver dialog box in Figure 2. The objective value is set to the desired value of 0, and since the objective function and constraints are linear, the Simplex method for Linear Programming is selected.



```
=SUMPRODUCT(C9:F9,C3:F3)+SUMPRODUCT(C9:F9,C5:F5)-SUMPRODUCT(B9:F9,B7:F7)
```

	A	B	C	D	E	F
1						
2			S	E	N	D
3			🟨	🟨	🟨	🟨
4		+	M	O	R	E
5			🟨	🟨	🟨	🟨
6		M	O	N	E	Y
7		🟨	🟨	🟨	🟨	🟨
8						
9	Place Value	10000	1000	100	10	1
10						
11						
12	SEND + MORE - MONEY	0				

Solver Parameters

Set Objective: $B$12

To: Max Min [✓ Value Of:] 0

By Changing Variable Cells:
$C$3:$F$3,$C$5:$F$5,$B$7:$F$7

Subject to the Constraints:
```
$B$7 >= 1
$B$7:$F$7 <= 9
$B$7:$F$7 >= 0
$C$3 >= 1
$C$3:$F$3 <= 9
$C$3:$F$3 >= 0
$C$5 = $B$7
$C$5 >= 1
$C$5:$F$5 <= 9
$C$5:$F$5 >= 0
$D$3 = $E$7
$D$3 = $F$5
$D$5 = $C$7
```

☑ Make Unconstrained Variables Non-Negative

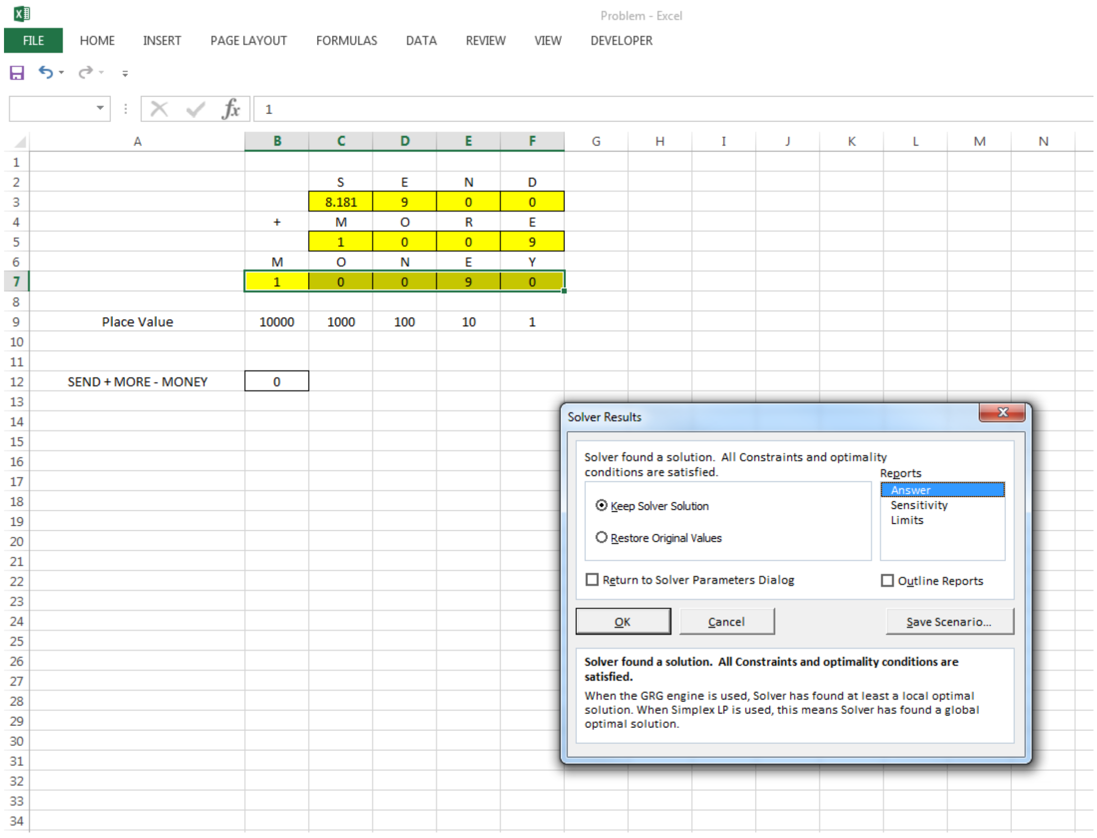
Select a Solving Method: Simplex LP

Solving Method

Select the GRG Nonlinear engine for Solver Problems that are smooth nonlinear. Select the LP Simplex engine for linear Solver Problems, and select the Evolutionary engine for Solver problems that are non-smooth.

Figure 2

Clicking on the "Solve" button leads to the message box shown in Figure 3. Although the solution values can be seen on the original spreadsheet (in the highlighted cells), selecting the Answer Report will provide additional information about the algorithmic process.



	A	B	C	D	E	F
1						
2			S	E	N	D
3			8.181	9	0	0
4		+	M	O	R	E
5			1	0	0	9
6		M	O	N	E	Y
7		1	0	0	9	0
8						
9	Place Value	10000	1000	100	10	1
10						
11						
12	SEND + MORE - MONEY	0				

Solver Results

Solver found a solution. All Constraints and optimality conditions are satisfied.

(●) Keep Solver Solution
( ) Restore Original Values

Reports: [✓ Answer] Sensitivity Limits

☐ Return to Solver Parameters Dialog
☐ Outline Reports

OK | Cancel | Save Scenario...

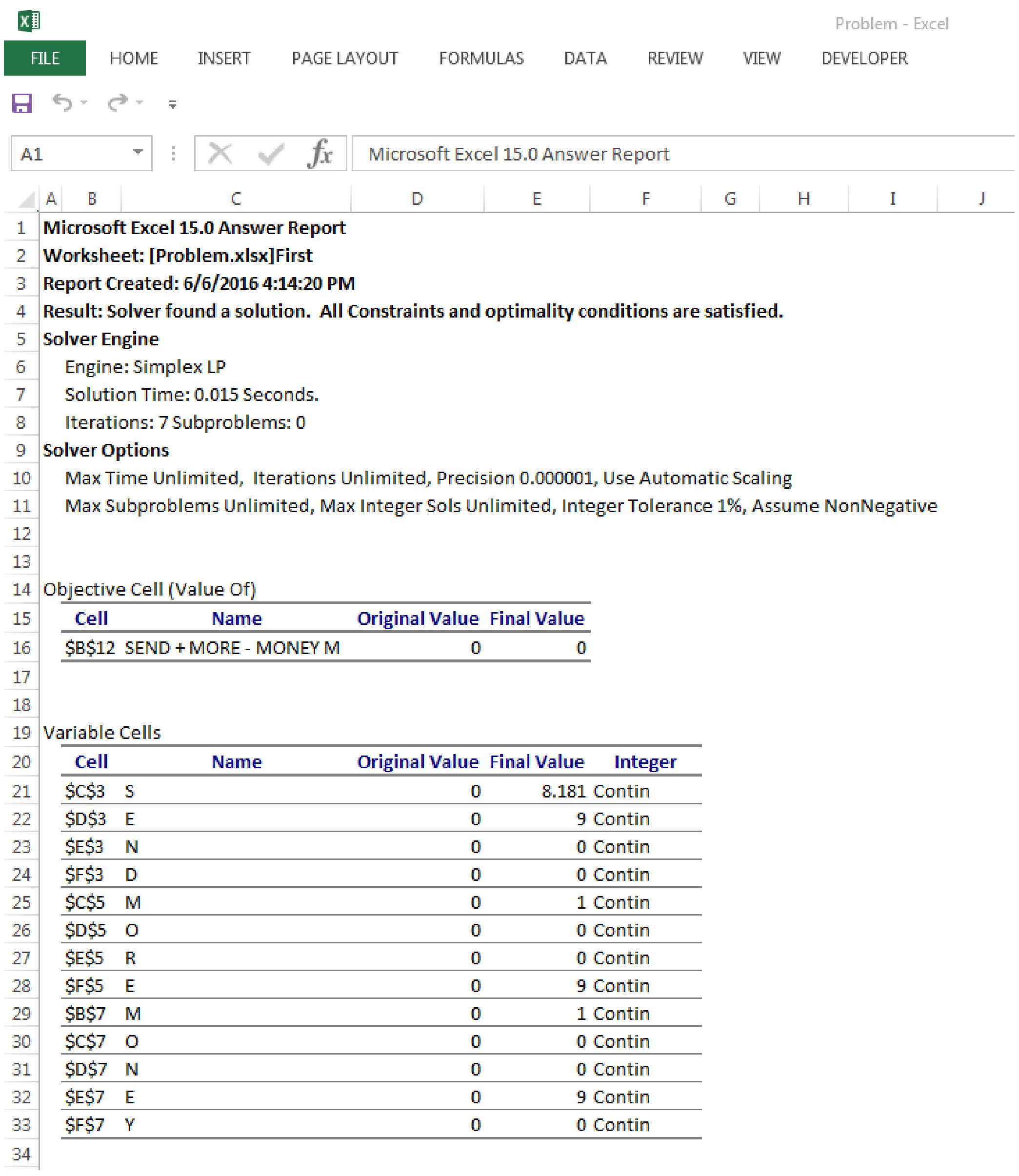
Solver found a solution. All Constraints and optimality conditions are satisfied.

When the GRG engine is used, Solver has found at least a local optimal solution. When Simplex LP is used, this means Solver has found a global optimal solution.

Figure 3

A glance at figure 3 reveals "solutions" that are clearly inadequate. Cell C3 gives a non-integer value for S, and the variables N,D,O,R, and Y all have the same value—zero.

The "Answer Report", which opens up on a separate page, is shown in Figure 4. In addition to the above "solutions", reports that the solution time is 0.015 seconds.



Microsoft Excel 15.0 Answer Report
Worksheet: [Problem.xlsx]First
Report Created: 6/6/2016 4:14:20 PM
Result: Solver found a solution. All Constraints and optimality conditions are satisfied.
Solver Engine
Engine: Simplex LP
Solution Time: 0.015 Seconds.
Iterations: 7 Subproblems: 0
Solver Options
Max Time Unlimited, Iterations Unlimited, Precision 0.000001, Use Automatic Scaling
Max Subproblems Unlimited, Max Integer Sols Unlimited, Integer Tolerance 1%, Assume NonNegative

Objective Cell (Value Of)

Cell	Name	Original Value	Final Value
$B$12	SEND + MORE - MONEY M	0	0

Variable Cells

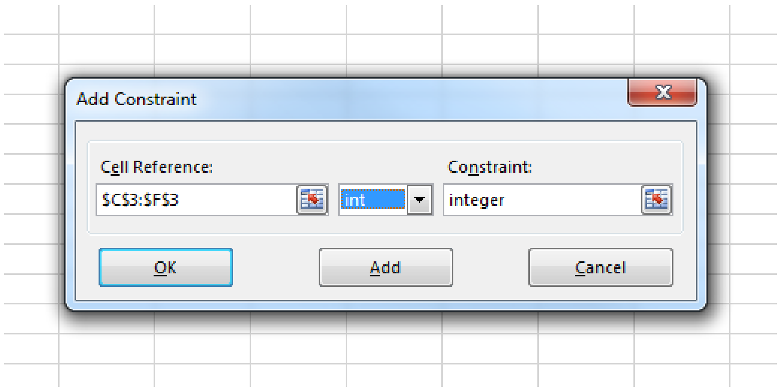
Cell	Name	Original Value	Final Value	Integer
$C$3	S	0	8.181	Contin
$D$3	E	0	9	Contin
$E$3	N	0	0	Contin
$F$3	D	0	0	Contin
$C$5	M	0	1	Contin
$D$5	O	0	0	Contin
$E$5	R	0	0	Contin
$F$5	E	0	9	Contin
$B$7	M	0	1	Contin
$C$7	O	0	0	Contin
$D$7	N	0	0	Contin
$E$7	E	0	9	Contin
$F$7	Y	0	0	Contin

Figure 4

3. The Integer Constraint

Obviously, we need a way to specify that the variables must be integers. Excel makes it easy by offering by offering the integer constraint in the dropdown box. This leads to the following three constraints.
$C$3:$F$3 integer
$C$5:$F$5 integer
$B$7:$F$7 integer

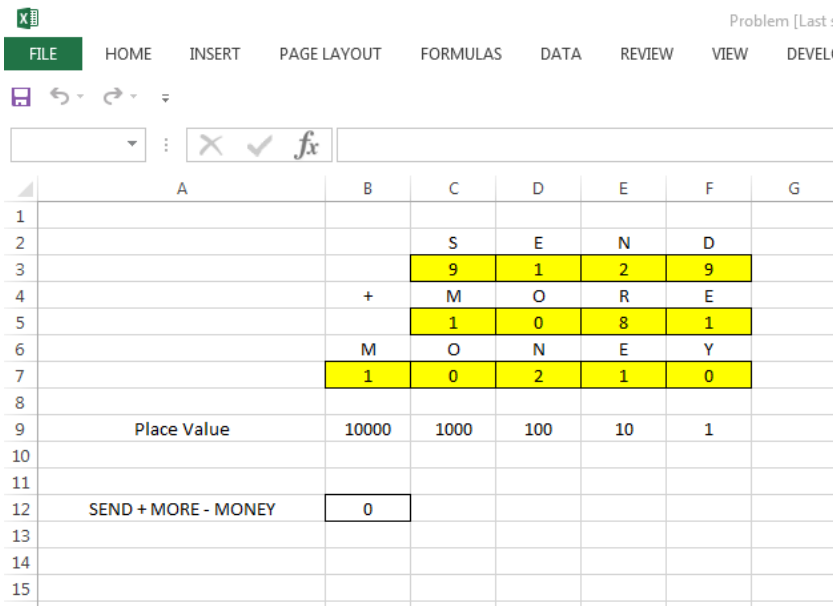
Figure 5 shows the first of these being added in Solver.


Add Constraint

Cell Reference: $C$3:$F$3 | int | Constraint: integer

Figure 5

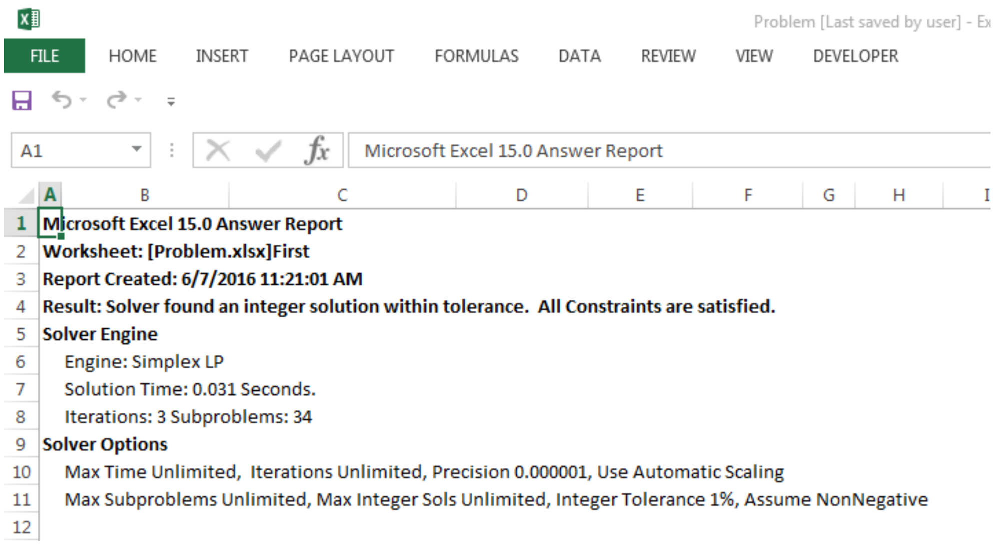
The revised output, shown in Figure 6, is still incorrect. For example, in row 3 we find that S and D are both assigned a value of 9. Clearly, we have not accounted for the rule that says that different letters represent different numerical values.



	A	B	C	D	E	F	G
1							
2			S	E	N	D	
3			9	1	2	9	
4		+	M	O	R	E	
5			1	0	8	1	
6		M	O	N	E	Y	
7		1	0	2	1	0	
8							
9	Place Value	10000	1000	100	10	1	
10							
11							
12	SEND + MORE - MONEY	0					
13							
14							
15							

Figure 6

For comparison's sake, it should be noted that the addition of the integer constraint more than doubled the computing time, as shown in Figure 7. When an integer constraint is introduced to an otherwise linear program, the resulting problem is known as an integer linear program.


Problem [Last saved by user] - Ex
FILE
HOME
INSERT
PAGE LAYOUT
FORMULAS
DATA
REVIEW
VIEW
DEVELOPER
A1
Microsoft Excel 15.0 Answer Report

```
Microsoft Excel 15.0 Answer Report
Worksheet: [Problem.xlsx]First
Report Created: 6/7/2016 11:21:01 AM
Result: Solver found an integer solution within tolerance.  All Constraints are satisfied.
Solver Engine
   Engine: Simplex LP
   Solution Time: 0.031 Seconds.
   Iterations: 3 Subproblems: 34
Solver Options
   Max Time Unlimited,  Iterations Unlimited, Precision 0.000001, Use Automatic Scaling
   Max Subproblems Unlimited, Max Integer Sols Unlimited, Integer Tolerance 1%, Assume NonNegative
```

Figure 7

4. The Alldifferent Constraint

If we go back to the "Add Constraint" dialogue box, we find that Solver offers the AllDifferent constraint, which at first glance seems made to order, since we want each letter to represent a different number. However, the AllDifferent constraint, as interpreted by Solver, does not behave quite as we would like it to. According to documentation provided by FrontlineSolvers [9]:

A constraint such as A1:A5 = alldifferent, where A1:A5 are decision variable cells, requires that these cells must be integers in the range 1 to N (N = 5 in this example), with each variable *different* from all the others at the solution. Hence, A1:A5 will contain a *permutation* of integers, such as 1,2,3,4,5 or 1,3,5,2,4.

Note that the range must begin with 1 and end with N. Of course, what we desire is a permutation of the numbers 0,1,2,3,4,5,6,7,8,9. However, since we have eight different letters, we will end up with a permutation on 1,2,3,4,5,6,7,8. Hence, 0 and 9 will be excluded from consideration. And so we face two problems. First, we need our permutation to begin with 0 instead of 1, and second, we need a permutation of the ten digits {0,1,2,3,4,5,6,7,8,9}.

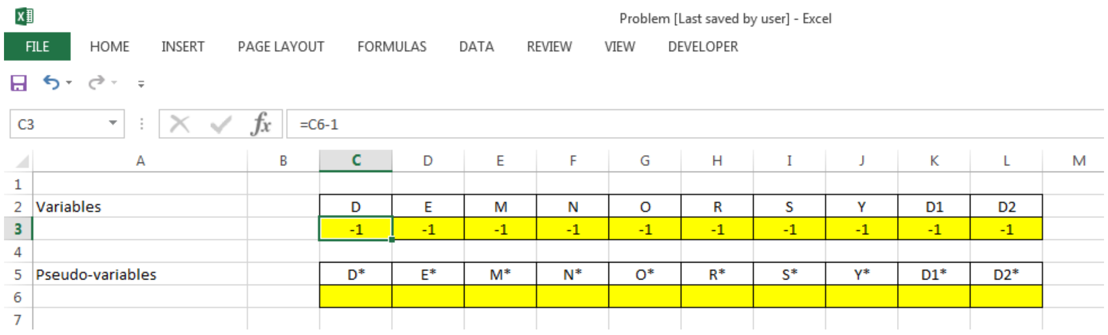
Let's address the second point first by introducing two new variables D1 and D2. Since they will not occur in the objective function, but only exist to provide us with the proper permutation, they are examples of dummy-variables.

Our set of variables now consists of {D,E,M,N,O,R,S,Y, D1, D2}, so AllDifferent will now give us a permutation from 1 to 10. Although it would be nice if we could "tell" the alldifferent function to begin with 0, this is not an option. Instead, we will have to find a logical workaround.

The easiest way to solve this is to create a set of pseudo-variables {D*,E*,M*,N*,O*,R*,S*,Y*,D1*, D2*}. Using AllDifferent, the values these receive will be a permutation from 1 to 10. Our original variables {D,E,M,N,O,R,S,Y, D1, D2} will correspond to these pseudo-variables in the following way:

D=D*-1
E=E*-1
...
D2 = D2* - 1

Thus, the values of {D,E,M,N,O,R,S,Y, D1, D2} will from a permutation from 0 to 9, as desired. This correspondence is shown in Figure 8. For example, in the formula box we see that C3=C6-1, which corresponds to D = D*-1.

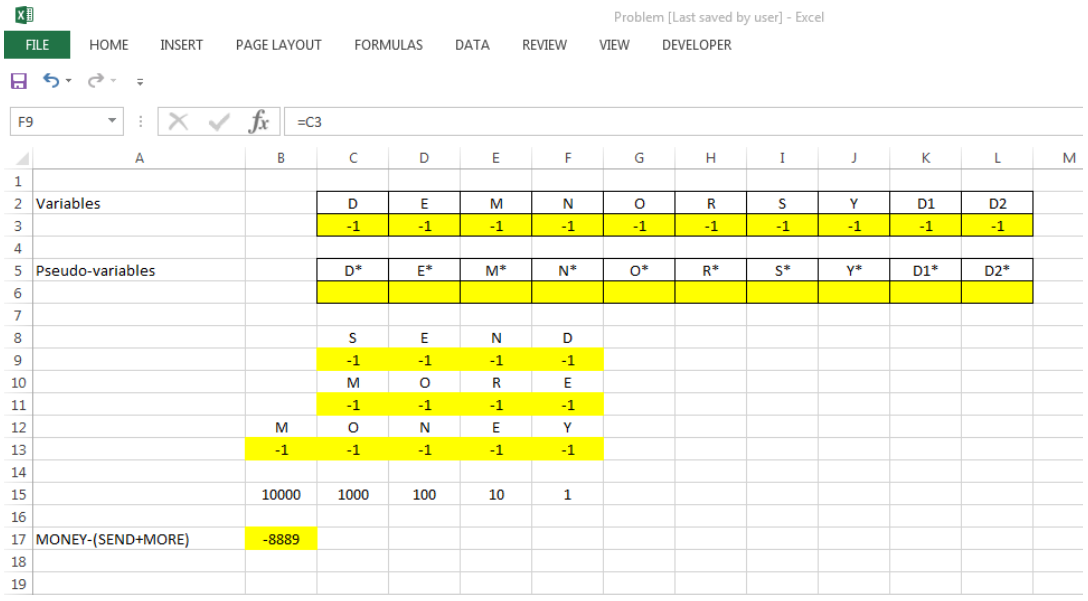

Problem [Last saved by user] - Excel
FILE HOME INSERT PAGE LAYOUT FORMULAS DATA REVIEW VIEW DEVELOPER

C3 =C6-1

	A	B	C	D	E	F	G	H	I	J	K	L	M
1													
2	Variables		D	E	M	N	O	R	S	Y	D1	D2	
3			-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	
4													
5	Pseudo-variables		D*	E*	M*	N*	O*	R*	S*	Y*	D1*	D2*	
6													
7													

Figure 8

It is now an easy matter to map the letters in SEND + MORE = MONEY to the cells in row 3. In Figure 9, C9 = I3. Then D9 = D3, E9 = F3, F9 = C3, etc.

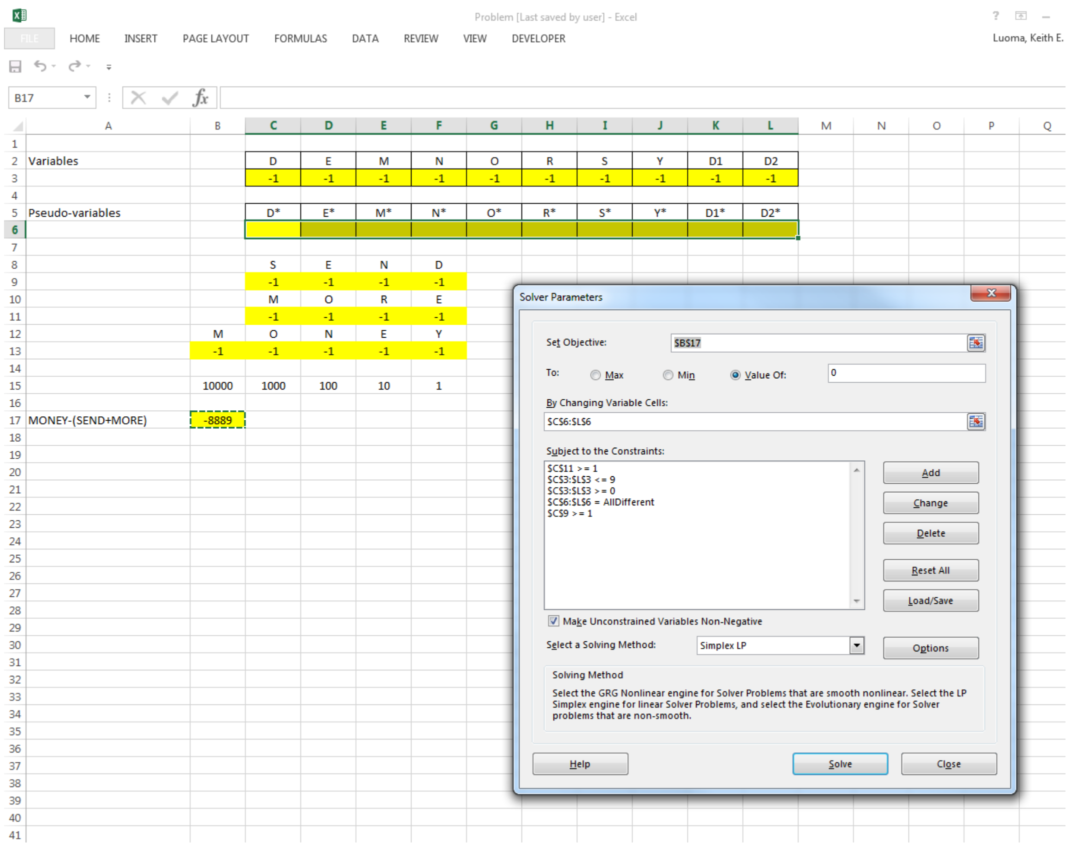

Problem [Last saved by user] - Excel
FILE HOME INSERT PAGE LAYOUT FORMULAS DATA REVIEW VIEW DEVELOPER

F9 =C3

	A	B	C	D	E	F	G	H	I	J	K	L	M
1													
2	Variables		D	E	M	N	O	R	S	Y	D1	D2	
3			-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	
4													
5	Pseudo-variables		D*	E*	M*	N*	O*	R*	S*	Y*	D1*	D2*	
6													
7													
8			S	E	N	D							
9			-1	-1	-1	-1							
10			M	O	R	E							
11			-1	-1	-1	-1							
12		M	O	N	E	Y							
13		-1	-1	-1	-1	-1							
14													
15		10000	1000	100	10	1							
16													
17	MONEY-(SEND+MORE)	-8889											
18													
19													

Figure 9

With the above mapping of variables completed, the list of constraints is considerably shortened. The pseudo-variables in row 6 must be integer and all different, and the variables in row 3 must be greater than or equal to 1 and less than or equal to 9. Finally, S and M must be at least 1. One further simplification is possible—the alldifferent constraint already assigns integer values to the variables, therefore the separate integer constraint is unnecessary and can be dropped. The set of the constraints is shown in Figure 10.



B17

	A	B	C	D	E	F	G	H	I	J	K	L
1												
2	Variables		D	E	M	N	O	R	S	Y	D1	D2
3			-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
4												
5	Pseudo-variables		D*	E*	M*	N*	O*	R*	S*	Y*	D1*	D2*
6												
7												
8			S	E	N	D						
9			-1	-1	-1	-1						
10			M	O	R	E						
11			-1	-1	-1	-1						
12		M	O	N	E	Y						
13		-1	-1	-1	-1	-1						
14												
15		10000	1000	100	10	1						
16												
17	MONEY-(SEND+MORE)	-8889										

Solver Parameters

Set Objective: $B$17

To: ○ Max ○ Min ● Value Of: 0

By Changing Variable Cells: $C$6:$L$6

Subject to the Constraints:

```
$C$11 >= 1
$C$3:$L$3 <= 9
$C$3:$L$3 >= 0
$C$6:$L$6 = AllDifferent
$C$9 >= 1
```

☑ Make Unconstrained Variables Non-Negative

Select a Solving Method: Simplex LP

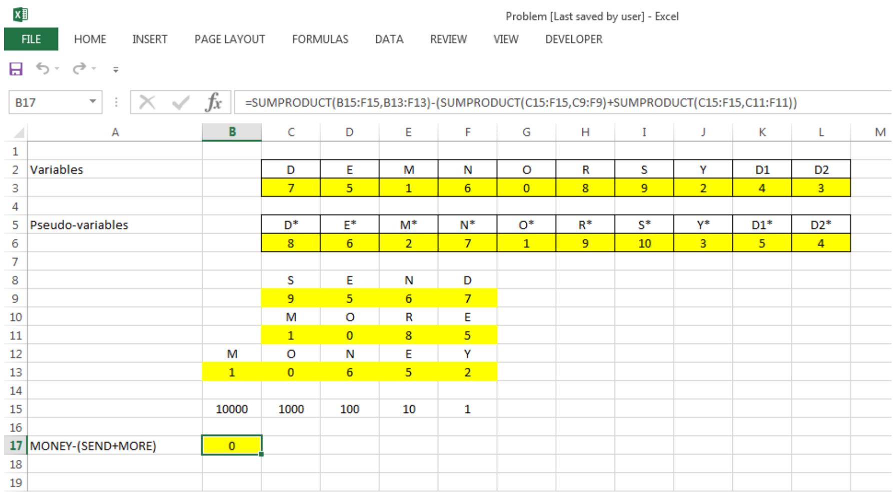
Solving Method

Select the GRG Nonlinear engine for Solver Problems that are smooth nonlinear. Select the LP Simplex engine for linear Solver Problems, and select the Evolutionary engine for Solver problems that are non-smooth.

Problem [Last saved by user] - Excel
FILE HOME INSERT PAGE LAYOUT FORMULAS DATA REVIEW VIEW DEVELOPER
Luoma, Keith E.
Add
Change
Delete
Reset All
Load/Save
Options
Help
Solve
Close

Figure 10

The output from this formulation is shown in Figure 11. The solution agrees with that found in the introduction to this article.



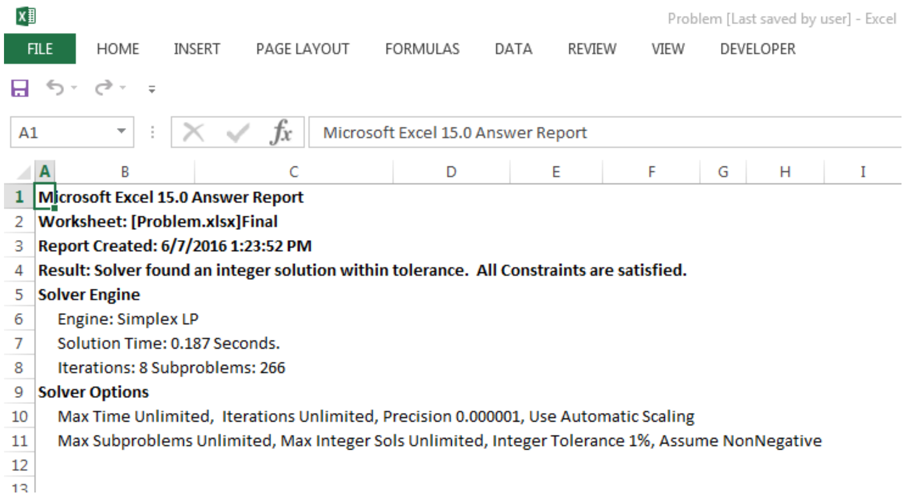
B17 =SUMPRODUCT(B15:F15,B13:F13)-(SUMPRODUCT(C15:F15,C9:F9)+SUMPRODUCT(C15:F15,C11:F11))

	A	B	C	D	E	F	G	H	I	J	K	L
1												
2	Variables		D	E	M	N	O	R	S	Y	D1	D2
3			7	5	1	6	0	8	9	2	4	3
4												
5	Pseudo-variables		D*	E*	M*	N*	O*	R*	S*	Y*	D1*	D2*
6			8	6	2	7	1	9	10	3	5	4
7												
8			S	E	N	D						
9			9	5	6	7						
10			M	O	R	E						
11			1	0	8	5						
12		M	O	N	E	Y						
13		1	0	6	5	2						
14												
15		10000	1000	100	10	1						
16												
17	MONEY-(SEND+MORE)	0										

Problem [Last saved by user] - Excel
FILE HOME INSERT PAGE LAYOUT FORMULAS DATA REVIEW VIEW DEVELOPER

Figure 11

Even though we were able to reduce the number of constraints, Figure 12 reveals a sixfold increase in computing time compared to the previous formulation. Clearly, the use of the alldifferent constraint leads to a significant drain on computer resources.



```
Microsoft Excel 15.0 Answer Report
Worksheet: [Problem.xlsx]Final
Report Created: 6/7/2016 1:23:52 PM
Result: Solver found an integer solution within tolerance.  All Constraints are satisfied.
Solver Engine
   Engine: Simplex LP
   Solution Time: 0.187 Seconds.
   Iterations: 8 Subproblems: 266
Solver Options
   Max Time Unlimited,  Iterations Unlimited, Precision 0.000001, Use Automatic Scaling
   Max Subproblems Unlimited, Max Integer Sols Unlimited, Integer Tolerance 1%, Assume NonNegative
```

Figure 12

5. Conclusion

Cryptarithms are interesting mathematical puzzles that are readily accessible to non-mathematicians, and they can also be used to provide a simple example of cryptography. Although the algorithm for solving cryptarithms shown in this article may not be as sophisticated as some other methods [5], its ease of presentation would make it suitable for courses including Management Science, Cryptography, Linear Algebra, or even College Algebra. In summary, it provides a useful teaching tool for the following reasons:

1. It does not require computer programming skills.
2. It introduces Excel users to Solver, and especially to the seldom taught integer and alldifferent constraints.
3. It shows the significant effect that the integer and alldifferent constraints have on computation time.
4. It emphasizes the value of adaptability, which was especially necessary in the implementation of the alldifferent constraint.

References

[1] Dudeney, H.E. *Strand Magazine*. Vol. 68, July 1924, PP. 97, 214.

[2] *Journal of Recreational Mathematics*, Vol. 1-38, 1961-2014.

[3] Eppstein, David. On the NP-completeness of cryptarithms. *SIGACT News*. Vol. 18, No. 3, 1987, pp 38-40.

[4] Matsui, Tomomi. NP-completeness of arithmetical restorations. *Journal of Information Processing*. Vol. 21, No. 2, 2013, pp 402-404.

[5] Shedge, Kishor N. and Kumar G. Sravan, Solving verbal crypto-arithmetic problem by parallel genetic algorithm (PGA). *International Journal of Computer Technology and Electronics Engineering*. Vol. 2, No. 4, 2012, pp 51-56.

[6] Brooks, Sarah. Computer solution of alphametics. *The Two-Year College Mathematics Journal*. Vol. 11, No. 2, 1980, 110-114.

[7] Ashbacker, Charles. Using alphametics in introductory programming classes: nifty tools and assignments. Journal of Computing Sciences in Colleges. Volume 24, No. 1, 1988, p 102.

[8] Bratko, Ivan. *Prolog*: *Programming for Artificial Intelligence*. 1986. Pp 158-162. Addison-Wesley.

[9] FrontlineSolvers, *Solver Platform SDK--Alldifferent Constraint,* http://www.solver.com/solver-platform-sdk-alldifferent-constraint