Module 14: Multiple Objectives via Goal

Reading Material: 14.1 – Introduction to Goal Programming

There are many formal mathematical programming approaches that deal with multiple objectives (as the situation dictates). Volumes of books have been written. The world is definitely a multi-objective place, and these multiple objectives are often in conflict. To complete this book, we ought to look at an approach that can accommodate more than a single objective.

For a variety of reasons, the book presents preemptive priority GP as a method to cover situations with multiple objectives. First, GP is a natural extension of LP. Second, because of this, it is quickly grasped due to this similarity. Finally, it is a multi-objective approach that past graduates have used in practice. A recent graduate reported a strategic staffing problem remarkably similar to one he dealt with in class. GP is representative of the many ways one could deal multiple objectives in quantitative scenarios and serves well as an introduction to the entire field.

14.1.1 Comparison of Linear Programming and Goal Programming

If LP is algebra on steroids, then GP is LP on legal controlled substances (steroids of course are legal in many circumstances). We will introduce GP by discussing its fundamental differences with LP. See Table 14.1 for an additional summary.

	Linear Programming	**Goal Programming**
Tagline	Algebra on steroids	LP on steroids
Changing cells	Decision variables	Decision variables Deviational variables
Objective function	Single, quantifiable	Multiple, “at least” or “no more than” or “equal”
	MAX, MIN	MIN from “goals” created in constraints “Barney” order
Constraints	SYSTEM - ABSOLUTE (<=, =, >=)	SYSTEM - ABSOLUTE (<=, =, >=) Goal: see objective function (ALWAYS =)

Constraints	X + Y <= 30	X + Y + UNDER(q) - OVER(q) = 30 At most, one of UNDER/OVER can be >= 0
Solution	One time	N times with N objectives Add constraint for previous goal, resolve.
Philosophy	MUST SATISFY ALL CONSTRAINTS OR INFEASIBLE!!!!	Try to satisfy goals, if cannot, set at "best value" Then move on to next goal without "undoing"

Objective: To this point, the LP models illustrated have had a single quantifiable objective that was either minimized or maximized. In GP, by definition, we will have a prioritized set of goals/objectives. These objectives, by convention, will be specified through a target value that one wants to attain "at least" the target, "not exceed" the target, or achieve "exactly" the target. For instance, a goal in a production situation might try to achieve a profit of at least $5000. Another goal in that same problem scenario might try to not exceed production capacity by attempting to producing no more than 250 products.

The convention in GP is that the decision maker can specify these goals in strict priority order (or, using fancy terms, "preemptive priority" of goals). I sometimes refer to this as the "Barney" order, in honor of the purple dinosaur that invaded our world back in the 1990s. In GP, the first goal is "super-dee-duper" more important than our second goal, which is "super-dee-duper" more important than our third goal, etc. You can replace "super-dee-duper" with "infinitely" if you'd prefer a less silly definition. To summarize, GP has multiple objectives specified as targets that will be treated in priority order.

Also note that the specified objectives in GP are likely in conflict with each other – that is, some will be achievable and some will not (like in reality!). This is why specified priority order is important and pertinent to the model.

Variables: LP has decision variables – those items for which our LP model determined optimal values. In GP, we still have decision variables (used for the same reasons) and deviational variables, as well, which Excel will treat just like decision variables (as changing cells).

In GP, deviational variables are used to express goals. They work in conjunction with our targets and goal constraints (discussed below) to specify the prioritized model objectives.

Constraints: In LP, we treated the constraints as system constraints (even though we didn't realize we were doing this!). Put another way, the LP model structure considered these constraints unbreakable. If an LP model could not find a solution that satisfied all (system) constraints, the Solver reported that the model was infeasible.

In GP, there will still be some unbreakable system constraints. Mainly, though, models will have goal constraints – constraints that allow us to specify our objectives (targets) as right-hand side (RHS) values and allow the goals/targets to NOT be met through the specification of the aforementioned deviation variables. In each goal constraint, there are two deviational variables – one that allows the model solution to UNDER-achieve the target, and one that allows the model solution to exceed the target (or go OVER the target). These deviational variables, unique for each goal constraint, are also termed UNDER and OVER variables.

Solution process: In LP, a model was set up in Excel and the Solver, one clicked solve, and the Solver found the optimal solution. In GP, a model is set up in Excel and the Solver, but multiple iterations of solutions will be required to find the optimal GP solution. The Solver will be executed once for each specific goal.

The philosophy of the GP solution methodology is to give preference to our most important goals. We do this by implementing the following systematic process. We find a solution that best achieves the present goal (highest priority goal remaining). Once we identify the best we can do, we FIX that level of achievement and move to the next most important goal. "Fixing" achievement means that the process will not allow our achievement of more important goals to be 'undone' when we consider new model solutions driven by the less important goals. We continually repeat this process until we've considered our least important goal.

Solutions may change at each step throughout the solution process – or the solution values may eventually converge to a solution that no longer varies. Depending on the circumstance, the higher priority goals may dictate a final solution before the lesser goals can be considered.

To implement this process in Excel, a "scoreboard" in the spreadsheet that allows a visual tracking of the progress will be used. This is just a Wilson habit, but one that allows a clear illustration of the progression in the solution process.

Reading Material: 14.2 – Example: The Three LEGO Furniture Product Example

Consider the setting of the modified LEGO furniture production problem examined in Module 7, which included a third product, stools. We further modify the problem scenario as follows.

14.2.1. The Goal Programming Problem Statement

PROBLEM: Determine the optimal production plan for three LEGO furniture products:
tables (made of two large LEGO blocks and two small LEGO blocks, and a total of three space units), chairs (mode of one large LEGO block and two small LEGO blocks, and a total of two space units) and stools (made of one large LEGO block and one small LEGO block, and a total of 1.5 space units) based on the following prioritized goals. Also, because of the configuration of the furniture (space units), there is one capacity constraint that CANNOT be violated – the total

amount of furniture produced cannot exceed 475 units of space. The production plan should address the following goals, stated in strict priority order. Also, all sub-goals present should be weighted equally.

Goal 1: Obtain at least $1100 of profit.

Goal 2: Use no more than 280 large LEGO blocks.

Goal 3: Produce at least 70 units of each furniture product.

Goal 4: Use no more than 300 small LEGO blocks.

Goal 5: Produce no more than 80 units of each furniture product.

Note that tables contribute $7 per unit to profit, chairs $5 per unit, and stools $3 per unit.

14.2.2 Detour: Linear Programming with All Hard (System) Constraints – WRONG MODEL ALERT!!!

Previously when we've examined production problems like this, we created an algebraic LP model that might look something like the following (with an assumption of a single goal of maximizing profit).

MAX 7 TAB + 5 CHR + 3 STL

ST

3 TAB + 2 CHR +1.5 STL <=475

2 TAB + 1 CHR + 1 STL <=280

TAB >=70

CHR >=70

STL >=70

2 TAB + 2 CHR + 1 STL <=300

TAB <=80

CHR <=80

STL <=80

If one tried to solve this LP model, the Solver would report that it is infeasible. Not all requirements can be satisfied.

This shouldn't be surprising because this wasn't a correct model anyway! (Again, I've added an incorrect model in a book – sacrilege!). We took the goals specified in the problem definition and made them all system constraints; again, these goals are likely in conflict with each other and need to be treated differently – thus, the GP formal approach.

This also brings us to another point: expect that there will be goals in GP that are not achievable. Just like goals conflicting in reality. If they weren't in conflict, we could treat them all as hard system constraints. GP allows us flexibility in not achieving each goal so that we can find the best overall solution based on goal prioritization.

The next section will illustrate the algebraic representation of the correct GP model. Then we have the two algebraic models to compare and contrast – the (incorrect) LP model stated earlier and the (correct) GP model. Historically, even though we try to avoid too much math, having the different models side by side has shown to be helpful to illustrate that GP is truly just LP and "a little bit more."

14.2.3 Constructing the GP Model: Preliminaries

Creating a GP model requires a slightly different approach than that used in LP modeling. The mantra of course was to separate out the three main model components (decision variables, objective and constraints). In GP, the (goal) constraints and objective are linked together, so they must be discussed simultaneously.

The process still begins with an identification of the decision variables – those aspects of the situation that we control and want the model to determine optimal numbers. In this case, obviously, the decision variables represent the number of the three furniture types to be produced.

Next, any system constraints should be identified – those aspects of the problem environment that absolutely must be met. In this specific case, the capacity requirement is to be treated as an absolute system requirement. The constraint looks like it has in previous examples:

3 TAB + 2 CHR +1.5 STL <=475

14.2.4 Constructing the Goal Programming Model: Goal Constraints – The First Goal

Goal constraints consist of three parts: targets, or RHS values; the left-hand side (LHS) summation, and the UNDER and OVER deviational variables.

Goals are expressed in terms of achieving 'targets'. The targets are the RHS values of the constraints. Goal constraints also have the traditional (for lack of a better term) left-hand side

calculations (LHS), whereby the decision variables contribution to the criteria of interest is tallied. The relationship between the LHS and the RHS is modified by the inclusion of deviational variables – specifically, the UNDER variable and the OVER variable. They allow the solution to achieve less than the target, or more than the target. This modeling convention permits a goal to not be attained – which is the key distinction in GP vs. LP. Symbolically, then, a goal constraint looks like this:

LHS + UNDER – OVER = TARGET (RHS)

Consider the actual first goal – obtaining a profit of at least $1100. Using the symbolic framework above, we can determine the following:

LHS = profit of the three products = 7 TAB + 5 CHR + 3 STL

TARGET = $1100

So the goal constraint would be: 7 TAB + 5 CHR + 3 STL + $UNDER_1$ – $OVER_1$ = 1100

(Note the use of subscripts – important!).

In a hypothetical solution, should the profit of the three products be $1200 (the LHS = $1200), this would mean that our solution provided $100 OVER the target profit. It would then seem reasonable that the OVER variable would be equal to 100. Mathematically, the goal constraint formulation does exactly that, because by adding the UNDER variable and subtracting the OVER variable, everything works out duckily (word in honor of the NCIS coroner!).

1200 + (UNDER = 0) – (OVER = 100) = 1100, or 1200 + 0 – 100 = 1100.

Reverse the scenario – say the profit of the three products was only $1020, $80 under the target. Mathematically,

1020 + (UNDER = 80) – (OVER = 0) = 1100, or 1020 + 80 – 0 = 1100.

So, UNDER = 80 tells us we missed by $80 making the target profit of $1100.

Why not have just one deviational variable instead of the matched pair? Two reasons. First, there is that concept of non-negativity. If we make the deviational variables behave just like decision variables, they can only be zero or positive. So we would have mathematical anarchy with only using one deviational variable – it would have to be negative. Second, and most important, we want the deviational variables to mean something to us in practice – and in this formulation they do. UNDER tells us how much a solution is under the goal target, and OVER tells us how much a solution is over the goal target. This is both helpful and straightforward.

A summary of goal constraint rules so far:

Every goal constraint is an equality. This is because every goal constraint has two matched unique deviational variables that allow variation from the goal target.

UNDER represents the amount less than the target a solution provides, OVER the amount more than a specified target. Mathematically, UNDER is always added to the LHS of the constraint and OVER is always subtracted.

Now, let's turn our attention to the use of the goal constraint to specify the explicit objective. Goal 1 is to try to achieve at least $1100 of profit. That means that it would be achieved if the production of furniture provided profit of $1100 or more. Thus, from a deviational variable standpoint, it would be okay if OVER was positive, but we would NOT achieve this goal if UNDER was non-zero. We could say that the goal would be achieved if the value of UNDER was equal to zero.

Thus, we will tell the Solver (when we get there) to minimize the value of the UNDER variable – that will be our objective statement. If the value of UNDER can be made 0, then we will say that we achieved this goal. If in minimizing the value, it cannot be made to be equal to zero, the goal is not achieved, but THE WORLD DOES NOT END! More on that when the GP solution process is discussed subsequently.

For the algebraic notation then, we will use: Minimize $UNDER_1$ to represent the implementation of the first goal.

Intermediate summary: Goals will always be expressed by minimizing the value of one or more deviational variables. They may or may not (in our definition) be achieved.

14.2.5 Goal Constraints: Goal 2 and Beyond

The second goal is to try to use no more than 280 large LEGO blocks. Following the same logic used in the previous section, noting the large LEGO block use by the three products, the second goal constraint is:

$2\ TAB + 1\ CHR + 1\ STL + UNDER_2 - OVER_2 = 280$

In this case, because we wish to try to use less than the target of 280, it is okay if UNDER is >0, but we want to try to have a solution in which OVER = 0. Thus, the objective/goal can be stated as:

MINIMIZE $OVER_2$

The third goal is an example of having multiple constraints needed to express one goal. The goal is to produce at least 70 units of each product – thus, we will need a goal constraint for each product. The three goal constraints:

TAB + $UNDER_3$ – $OVER_3$ = 70

CHR + $UNDER_4$ – $OVER_4$ = 70

STL + $UNDER_5$ – $OVER_5$ = 70

To achieve this goal, we need to actually consider all three sub-goals at the same time. Because we want to make "at least" 70 units, having OVER >0 is fine – so we want to minimize the UNDER deviational variables to try to achieve the goal. Thus, the goal can be stated as:

MINIMIZE $UNDER_3 + UNDER_4 + UNDER_5$

The convention we use is to always treat sub-goals equally (sub-goals being the three deviational variables that we wish to minimize). Not enough time and not enough value-added to worry about different weighting mechanisms.

Goal 4 is similar to Goal 2, except for it being small LEGO blocks and a target of 300.

2 TAB + 2 CHR + 1 STL + $UNDER_6$ – $OVER_6$ = 300, MINIMIZE $OVER_6$.

Finally, Goal 5 is similar to Goal 3, consisting of goal constraints for each product, but the goal is "opposite", to produce no more than 80 units of each. Therefore,

TAB + $UNDER_7$ – $OVER_7$ = 80

CHR + $UNDER_8$ – $OVER_8$ = 80

STL + $UNDER_9$ – $OVER_9$ = 80

And the statement of the goal:

MINIMIZE $OVER_7 + OVER_8 + OVER_9$

All goal constraints (and the one system constraint) have been defined, and the proper deviational variables for goal specification have been noted. Next up – how do we solve this goal programming model in Excel?

Reading Material: 14.3 – Solving Goal Programming Models in Excel

14.3.1 Restatement of Goal Programming Model

Below is the GP model developed in the previous section, all in one place!

MINIMIZE:

GOAL1 $UNDER_1$

GOAL2 $OVER_2$

GOAL3 $UNDER_3 + UNDER_4 + UNDER_5$

GOAL4 $OVER_6$

GOAL5 $OVER_7 + OVER_8 + OVER_9$

3 TAB + 2 CHR +1.5 STL <=475

7 TAB + 5 CHR + 3 STL $+ UNDER_1 - OVER_1 = 1100$

2 TAB + 1 CHR + 1 STL $+ UNDER_2 - OVER_2 = 280$

TAB $+ UNDER_3 - OVER_3 = 70$

CHR $+ UNDER_4 - OVER_4 = 70$

STL $+ UNDER_5 - OVER_5 = 70$

2 TAB + 2 CHR + 1 STL $+ UNDER_6 - OVER_6 = 300$

TAB $+ UNDER_7 - OVER_7 = 80$

CHR $+ UNDER_8 - OVER_8 = 80$

STL $+ UNDER_9 - OVER_9 = 80$

The strategy for implementation: Row/Column format. Each column will correspond to one of the three decision variables, representing the production level of the furniture. Also, we will have columns for UNDER and OVER variables, because they are also changing cells. Finally, the multi-faceted objective represents a challenge to implement – a scoreboard mechanism will be used on the spreadsheet to facilitate the iterative goal programming solution approach.

14.3.2 Creating the Framework in Excel

Figure 14.1 illustrates the basics of the model – note that this is shown PRIOR to any solving by Excel.

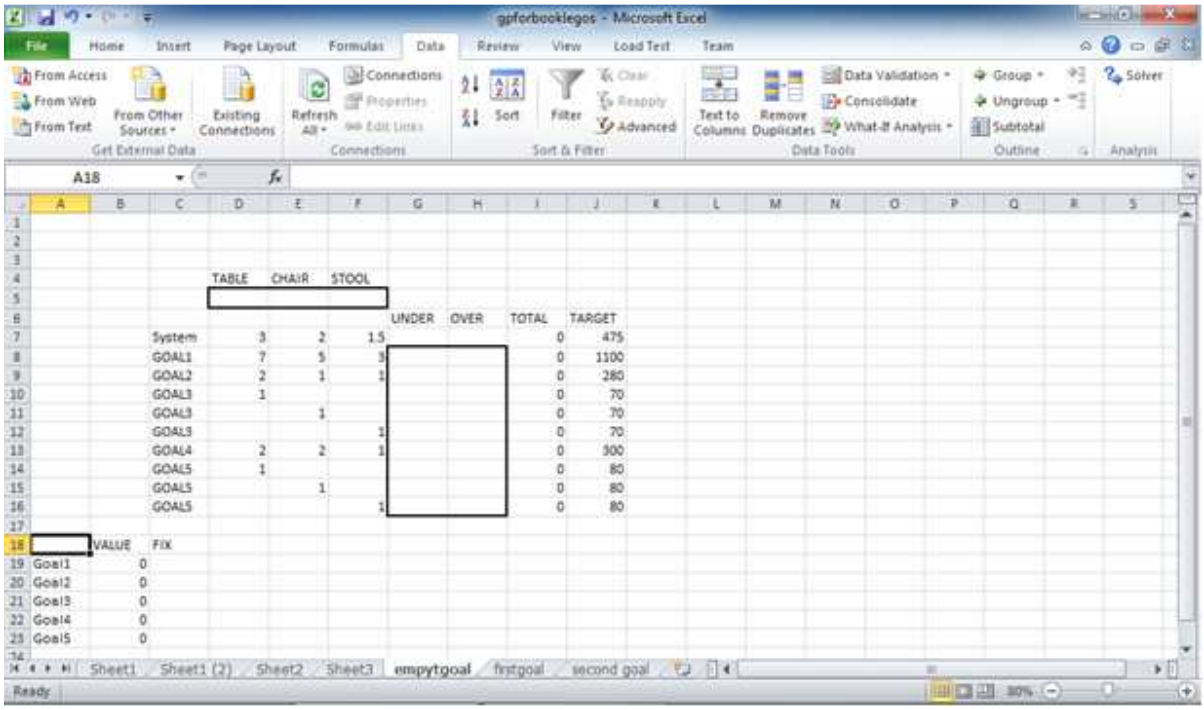

Figure 14.1

The figure should look very familiar at this stage of the course, because it is just a reflection of the algebraic model. Row 5 is the location of the decision variables for Tables, Chairs, and Stools. Columns D through F are the product's corresponding columns. Column C indicates the corresponding goal or system constraint reference.

Row 7 is the model's one system constraint. Rows 8 to 16 are the goal constraints. Obviously, the coefficients of the goal constraints are input in the corresponding columns D through F.

The UNDER and OVER deviational variable placeholders are in Columns G and H. They too will need to be defined in the Solver as changing cells. Column J is the RHS value of the constraints (the target for the goal constraints).

Column I contains the action cells in this model. Cell I7 is a usual SUMPRODUCT because row 7 is the system constraint (no deviational variables). The formula: SUMPRODUCT(D$5:F$5, D7:F7). Cell I7 is also set <= J7 in the Solver.

For the goal constraint rows (Rows 8 to 16), Column I formula includes the usual SUMPRODUCT, but also adds the value from Column G (the UNDER deviational variable) and subtracts the value from Column H (the OVER deviational variable). For instance, the formula in Cell I8 is: SUMPRODUCT(D$5:F$5,D8:F8) + G8 – H8. This formula can be copied down through row 16 in Column I. Also, note that the goal constraints will need to be set in the Solver: I8:I16 = J8:J16 – yes, that simple.

Finally, at the bottom left of the spreadsheet is what we've called the scoreboard. Column A is just a label for the goals. Column B references the pertinent deviational variables that specify the goals of interest. This is where the Solver will be "pointed" – the target cell. Column C ("FIX") will be used as one moves from goal to goal. Here, we will put the best value that can be obtained for that goal and set a constraint that fixes that goal – so its achievement level is not undone as we look at lesser super-dee-duper goals.

For this example, the following cell references (i.e., formulas) are found in Column B.

Cell B19: = G8

Cell B20: = H9

Cell B21: = G10 + G11 + G12

Cell B22: = H13

Cell B23: = H14 + H15 + H16

The cell references correspond to the deviational variables minimized in each of the five goals. Refer back to the original algebraic statement of the GP model if it is unclear. For instance, B19, the statement of Goal 1, is a reference to $UNDER_1$, the underachievement of the profit goal of $1100.

And now the big moment – we're actually going to solve something!

14.3.3 The Goal Programming Iterative Solution Process

Figure 14.2 is the Solver input for the model as we look to solve for Goal 1. Figure 14.3 is the resultant spreadsheet after it is solved. (Note that for simplicity, we are not worrying about integers, pieces of furniture for this example, though we could if we wanted to.)

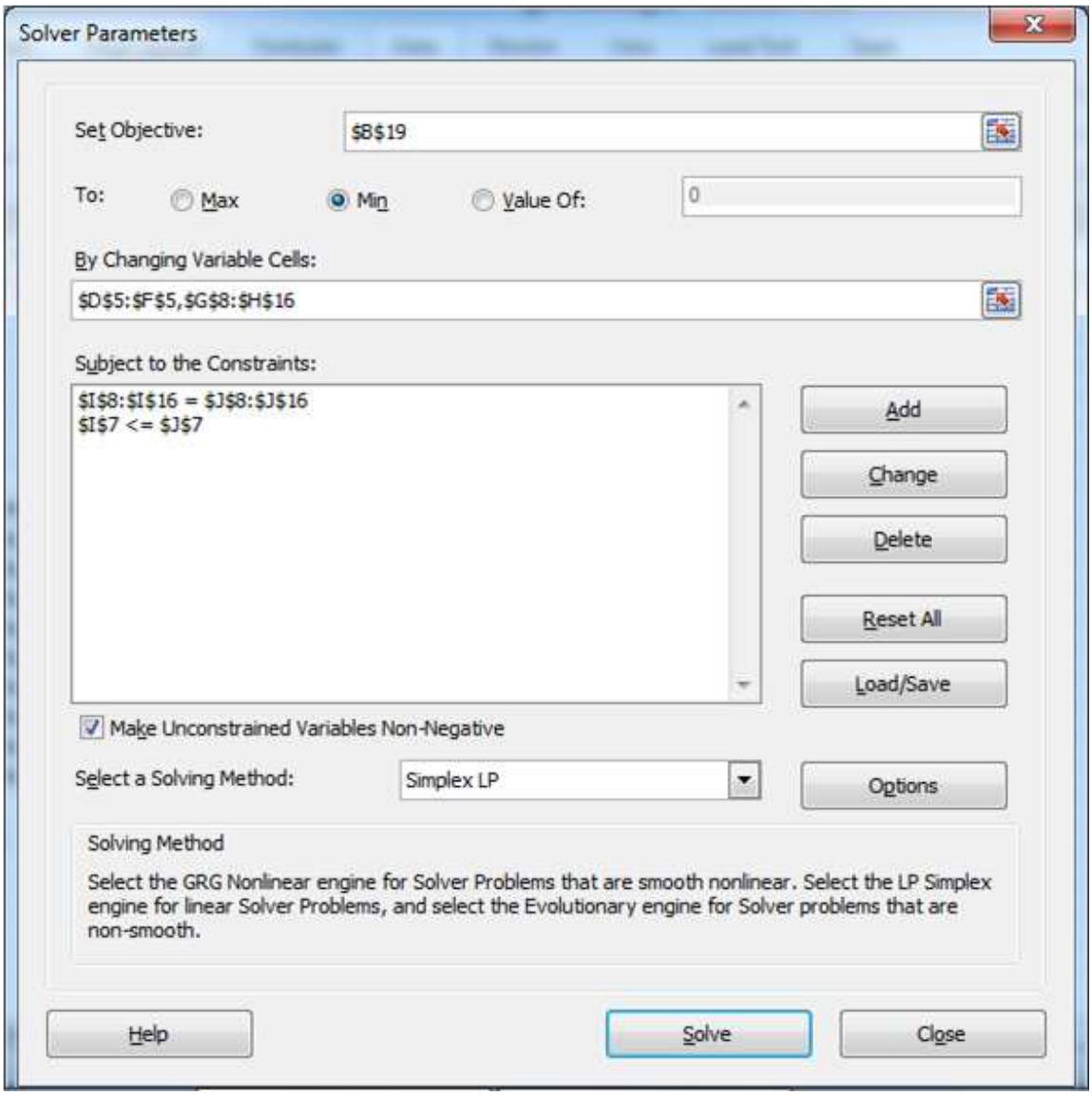
Solver Parameters
Set Objective:
$B$19
To:
Max
Min
Value Of:
0
By Changing Variable Cells:
$D$5:$F$5,$G$8:$H$16
Subject to the Constraints:
$I$8:$I$16 = $J$8:$J$16
$I$7 <= $J$7
Add
Change
Delete
Reset All
Load/Save
Make Unconstrained Variables Non-Negative
Select a Solving Method:
Simplex LP
Options
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.
Help
Solve
Close

Figure 14.2

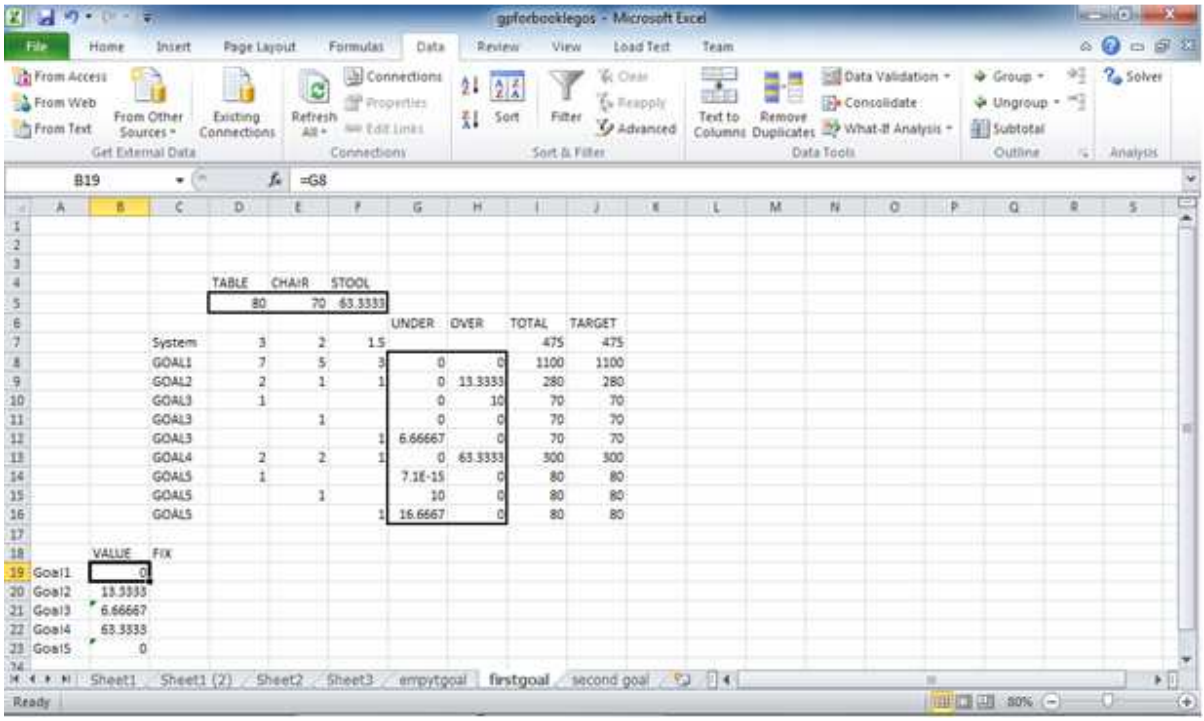

Figure 14.3

By definition, we were able to achieve Goal 1, because the target cell B19 (the reference to the algebraic variable $UNDER_1$) was minimized to 0. Also, note that this initial solution of 80 tables, 70 chairs and 63.3333 stools achieves Goal 5, but not Goals 2, 3, or 4. This is determined by the scoreboard – Column B values. When those values are 0, a Goal is said to be achieved.

The intermediate solutions that occur as we iterate through the goals are not necessarily meaningful in terms of the final solution. For discussion sake, they can help us see how the solutions are gradually narrowed with each successive goal. Ultimately, the goals will converge on a solution – sometimes all the way at the end of investigating all goals, sometimes they converge earlier. Because we don't know when this will happen, we MUST investigate each goal, even if the solution appears not to be further modified.

Next comes a very important step in the GP solution process. We have determined that Goal 1 can be achieved. We need to now consider Goal 2, but also fix Goal 1 so it is not undone. To do this, the target cell is changed (now to point at B20, the placeholder for Goal 2), and a constraint is added to the model: B19=C19, while adding the value of 0 to C19. The constraint forces Goal 1 to remain achieved and allows us to consider possible alternative solutions that may achieve Goal 2 (or improve Goal 2's achievement). This is the two-step process we use each time we go from goal to goal: change the target cell, and add a constraint to fix the previous goal.

Figure 14.4 shows the new Solver model after the change, and Figure 14.5 shows the results of considering Goal 2 (with the Solver model of Figure 14.4).

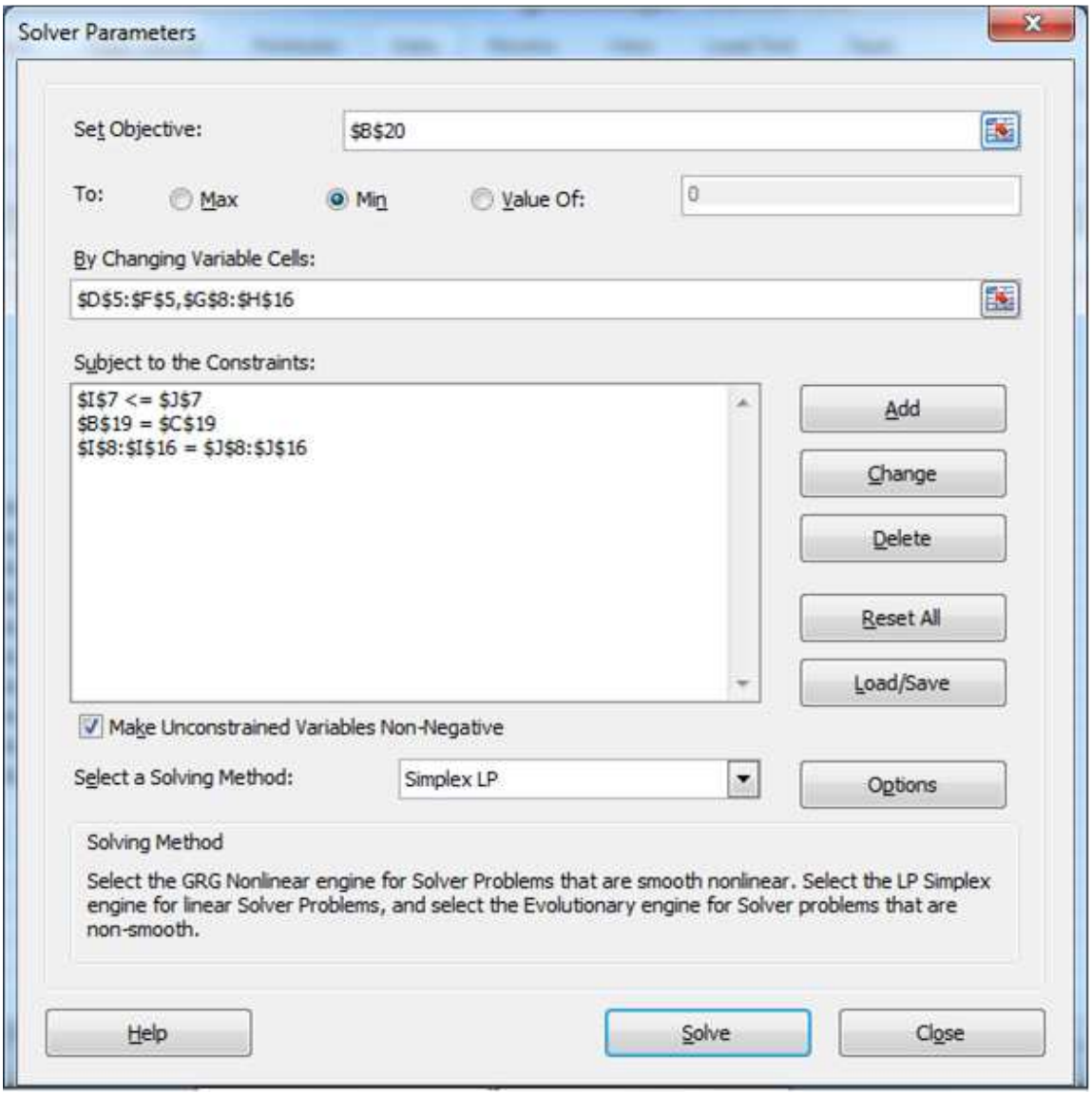

Figure 14.4

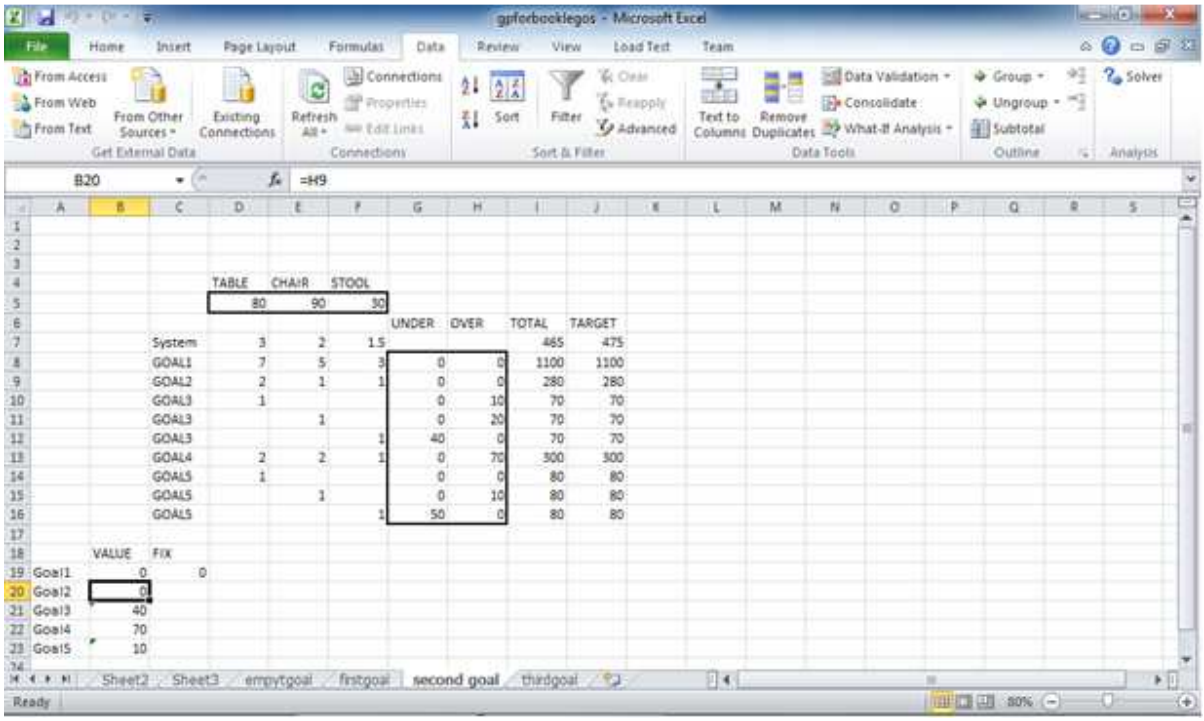

Figure 14.5

The solution has changed (80 tables, 90 chairs and 30 stools), and Goal 2 was achieved (B20=0) without undoing Goal 1 (B19=0). With the new solution, though, Goals 3, 4, and 5 are now not achieved. Of course, this is okay because they are lesser priority goals and will be considered in due time.

Now it is time to analyze Goal 3. The target cell is changed to B21, and a constraint B20=0 is added – I choose to do that explicitly in the spreadsheet, but of course you could also do it directly in the Solver. Figure 14.6 shows the new Solver model, Figure 14.7 the results of considering Goal 3.

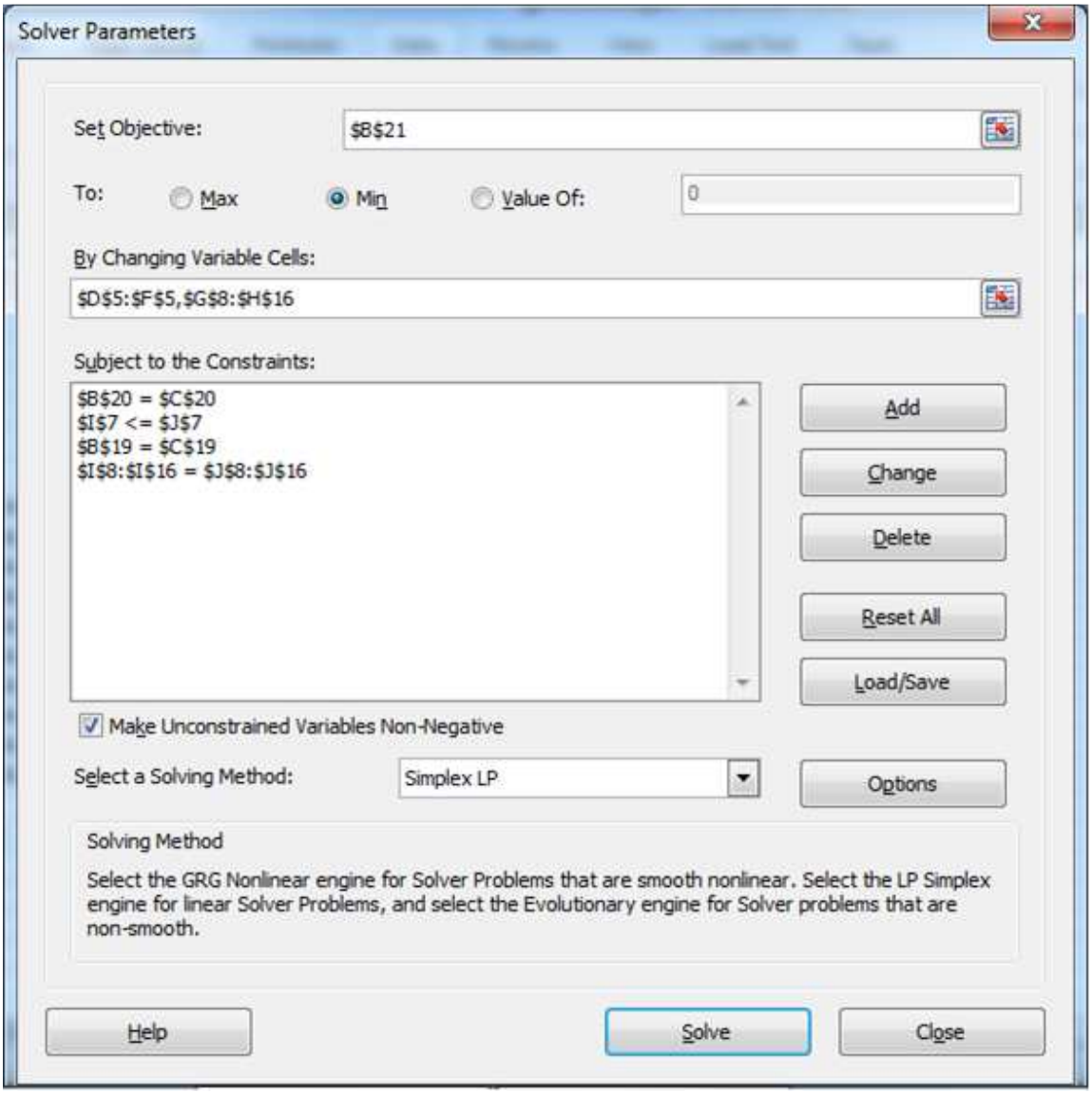
Solver Parameters
Set Objective:
$B$21
To:
Max
Min
Value Of:
0
By Changing Variable Cells:
$D$5:$F$5,$G$8:$H$16
Subject to the Constraints:
$B$20 = $C$20
$I$7 <= $J$7
$B$19 = $C$19
$I$8:$I$16 = $J$8:$J$16
Add
Change
Delete
Reset All
Load/Save
Make Unconstrained Variables Non-Negative
Select a Solving Method:
Simplex LP
Options
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.
Help
Solve
Close

Figure 14.6

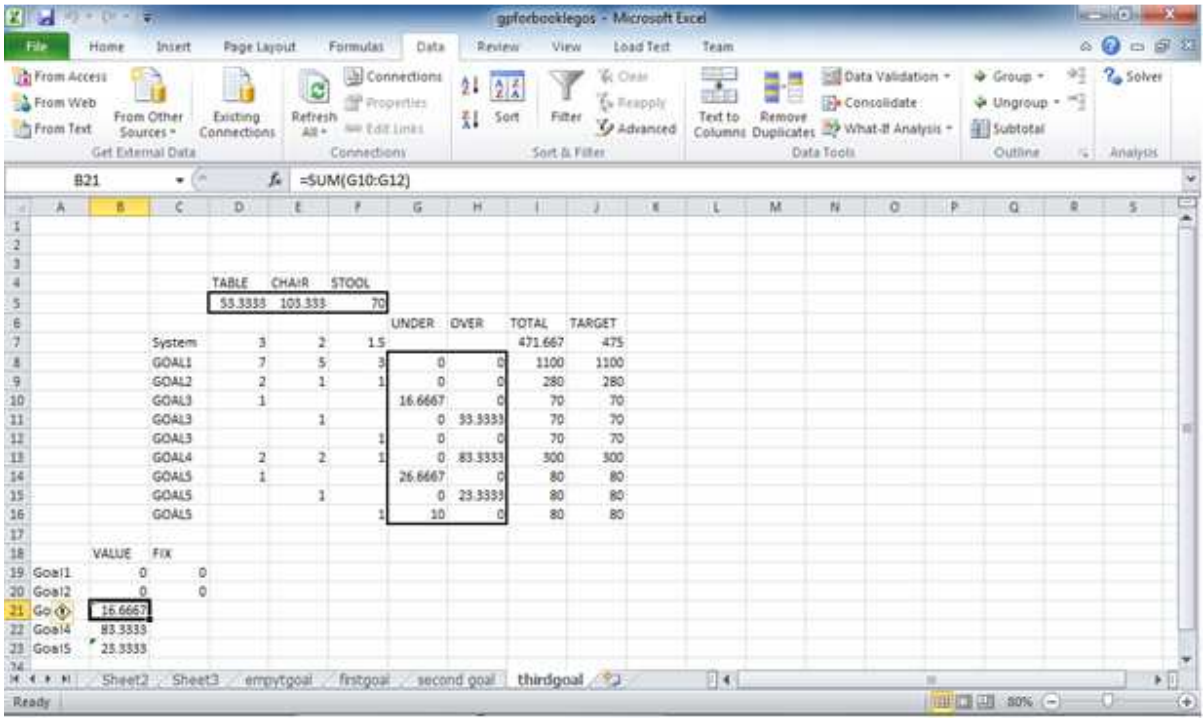

Figure 14.7

The results show that we could NOT achieve Goal 3 – the minimized value of the three deviational variables added together was 16.6667. We can see that in our attempt to achieve Goal 3 (but not at the expense of Goals 1 and 2) we cannot produce enough tables to meet the minimum target of 70. In essence, the combination of Goals 1 and 2 are likely in conflict with Goal 3 – probably because of the requirement for increased production of stools (if you compare Figure 14.5 to Figure 14.7). Goals 4 and 5 are also not achieved.

So, is this bad? No – it just means that Goal 3 cannot be achieved if we achieve Goal 1 and Goal 2. And, Goal 1 and Goal 2 are more important than Goal 3 (by definition). So, to quote the late Walter Cronkite "And that's the way it is."

And – we are not finished either. Whether we can achieve a goal or not, we keep considering every goal until all goals have been considered. We do not know if further improvements to the lesser goals can be made until we explicitly investigate it.

So, on to Goal 4 – we change target cells (B22 now) and must carefully add a constraint that limits B21 <= 16 2/3. Why carefully? If we are sloppy in rounding-off the non-achievement value, we could create an infeasible solution. If you set a constraint that says B21 = 16.666, it will be infeasible – the smallest that B21 can be is 16 2/3, which is larger than the decimal number 16.666 (keep in mind the infinitely repeating decimals – e-mail me if you don't see this).

I have actually typed the number 50/3 (16 and 2/3) in cell C23 to avoid any round-off issues.

Figure 14.8 is the final result after considering Goals 4 and 5. I do not show the Solver input again – the iterative process should be clear after three examples. Nothing changed after Goal 3's solution, but we still need to continue through all the goals in case subsequent goals can be marginally improved. This is very important to be thorough!

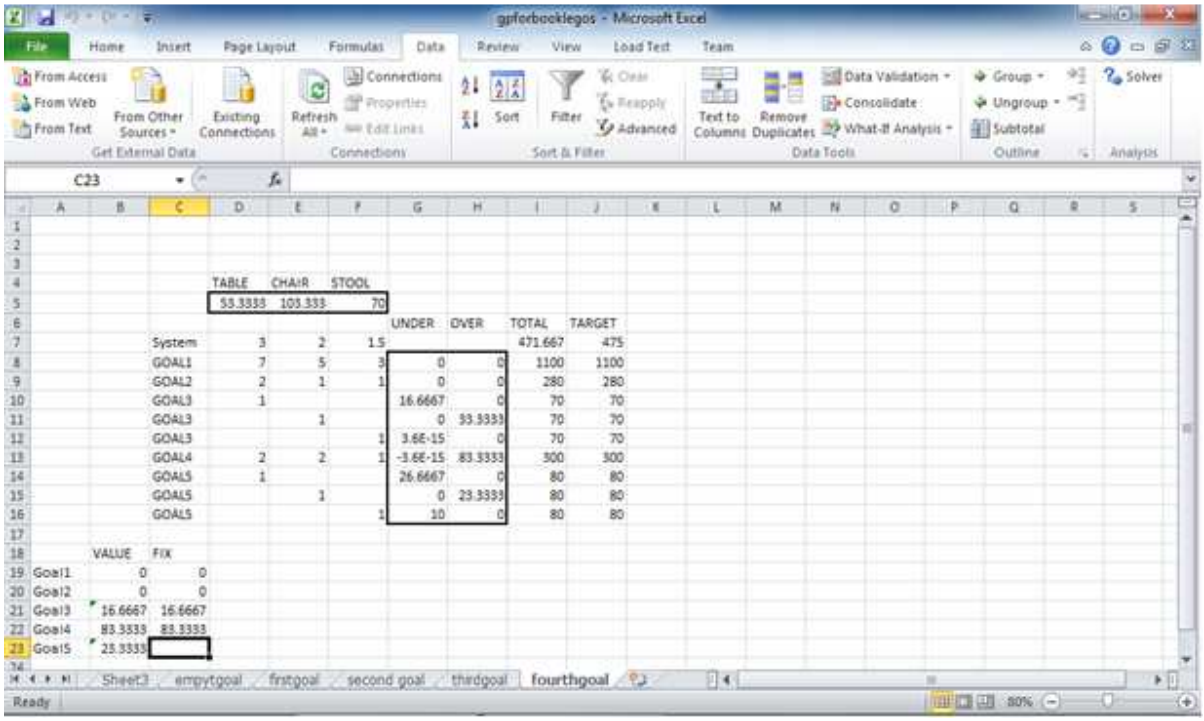

Figure 14.8

So, to summarize the solution to our goal programming model, the optimal production level of furniture (we are ignoring integrality for now) is 53 1/3 tables, 103 1/3 chairs, and 70 stools. This solution achieves Goals 1 and 2, but does not achieve Goals 3, 4, and 5.

Reading Material: 14.4 – Goal Programming Sensitivity Analysis

After this example, it might be apparent that the goal order could dramatically affect the optimal solution. Thus, sensitivity analysis in terms of GP models consider how changes in the order of the goals affect the determined optimal solutions.

Obviously, one could create all kinds of order combinations with five goals. The point of this section isn't to be overwhelming, but to show what one could do in practice to gain additional insight into the relative trade-offs of the goals.

For the sake of example, let us reverse the order of the goals – Goal 5 above became Goal 1, Goal 4 the second most important goal and so forth up to the original Goal 1, the profit goal, being the least important.

Figure 14.9 shows the very last spreadsheet in the sequence of five for this goal order. The solution, as you would expect, is quite different. The profit Goal and the "produce at least 70" goal are not achieved, but the other three goals are (large LEGO blocks, small LEGO blocks, and produce no more than 80 individual products). The optimal furniture production is 70 each of tables and stools and 45 chairs. Obviously, the solution here sacrifices meeting the "at least" type goals by satisfying the "no more than" type goals – lower production doesn't exceed upper bounds in LEGO blocks and production, but doesn't meet minimum production goals or the desired amount of profit. Thus, from this analysis, you gain an understanding of the tradeoffs involved among the goals.

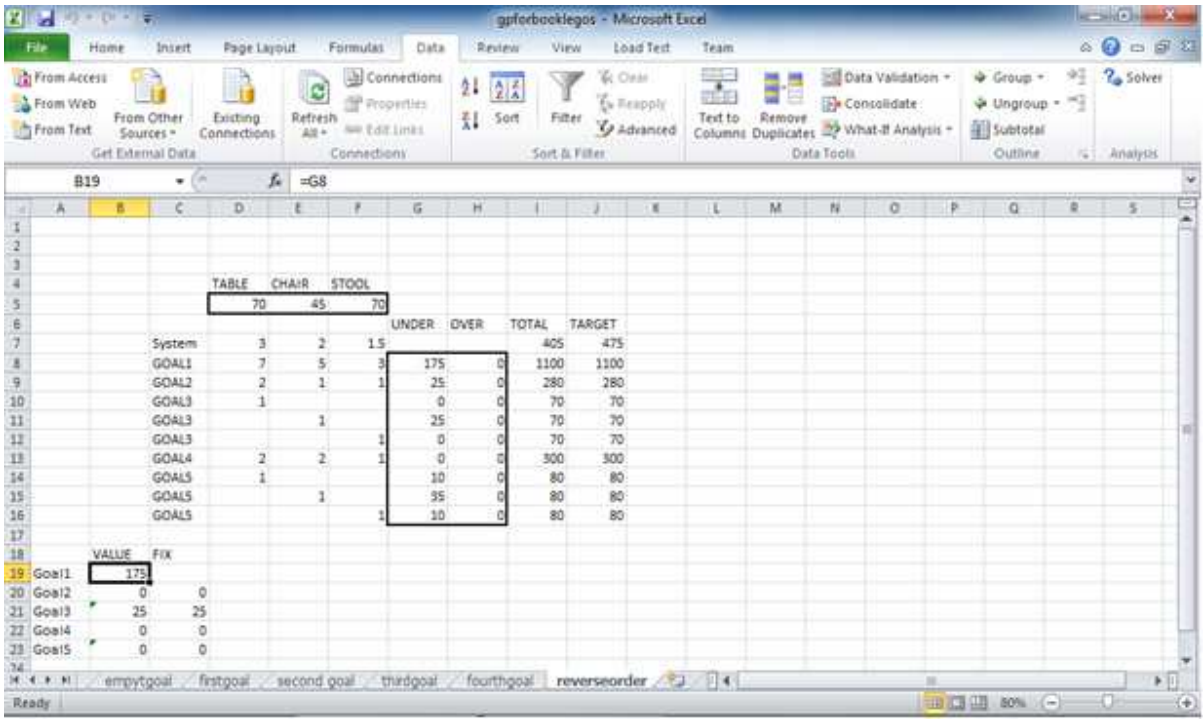

Figure 14.9

An interesting point: to get the same optimal solution in Figure 14.9, one MUST examine all goals, even after it appears we might be able to stop earlier. This is a great example of why one must be thorough. Make sure you match the solution found in Figure 14.9.

Which of the two solutions are best? One would assume that the decision maker puts the goals in order of their true importance, and thus the best order leads to the best solution. But if the

decision maker is simply exploring the impacts of different goals and their order, this is where the partnership between model and modeler is clearly important. All kinds of iterations can be done very easily using this basic goal programming framework. GP may epitomize the concept of decision model/decision maker partnerships.

Reading Material: 14.5 – Final Comments on Goal Programming

There are all kinds of ways in the realm of mathematical programming to treat multiple objectives. GP is just one way. But it is a good way.

It may not be apparent, and it certainly isn't necessarily important, but do you realize that every LP model also could be represented as a GP model? I mention this because others have stumbled on to this fact, and it shows that LP and GP really are of the same family of quantitative techniques.

GP is also useful in those circumstances in which we recognize we have multiple solutions, and we have a secondary objective that we might want to use as a tie-breaker for determining which extreme point (of the multiple optima) might be better on that lesser dimension.

So we've come to the end of the algebra on steroids on steroids chapter. Practice problems await.