Nonlinear Programming and Evolutionary Optimization
Ragsdale, Cliff (2010-12-06). Spreadsheet Modeling & Decision Analysis: A Practical Introduction to Management Science (Page 351).

8.0 Introduction

Up to this point in our study of optimization, we have considered only mathematical programming models in which the objective function and constraints are linear functions of the decision variables. In many decision problems, the use of such linear functions is appropriate. Other types of optimization problems involve objective functions and constraints that cannot be modeled adequately using linear functions of the decision variables. These types of problems are called nonlinear programming (NLP) problems.

The process of formulating an NLP problem is virtually the same as formulating an LP problem. In each case, you must identify the appropriate decision variables and formulate an appropriate objective function and constraints using these variables. As you will see, the process of implementing and solving NLP problems in a spreadsheet is also similar to that for LP problems. However, the mechanics (that is, mathematical procedures) involved in solving NLP problems are very different. Although optimization software such as Solver makes this difference somewhat transparent to the user of such systems, it is important to understand these differences so you can understand the difficulties you might encounter when solving an NLP problem. This chapter discusses some of the unique features and challenges involved in solving NLP problems and presents several examples of managerial decision-making problems that can be modeled as NLP problems.

8.1 The Nature of NLP Problems

The main difference between an LP and NLP problem is that NLPs can have a nonlinear objective function and/or one or more nonlinear constraints. To understand the differences and difficulties nonlinearities introduce to an optimization problem, consider the various hypothetical NLP problems shown in Figure 8.1.

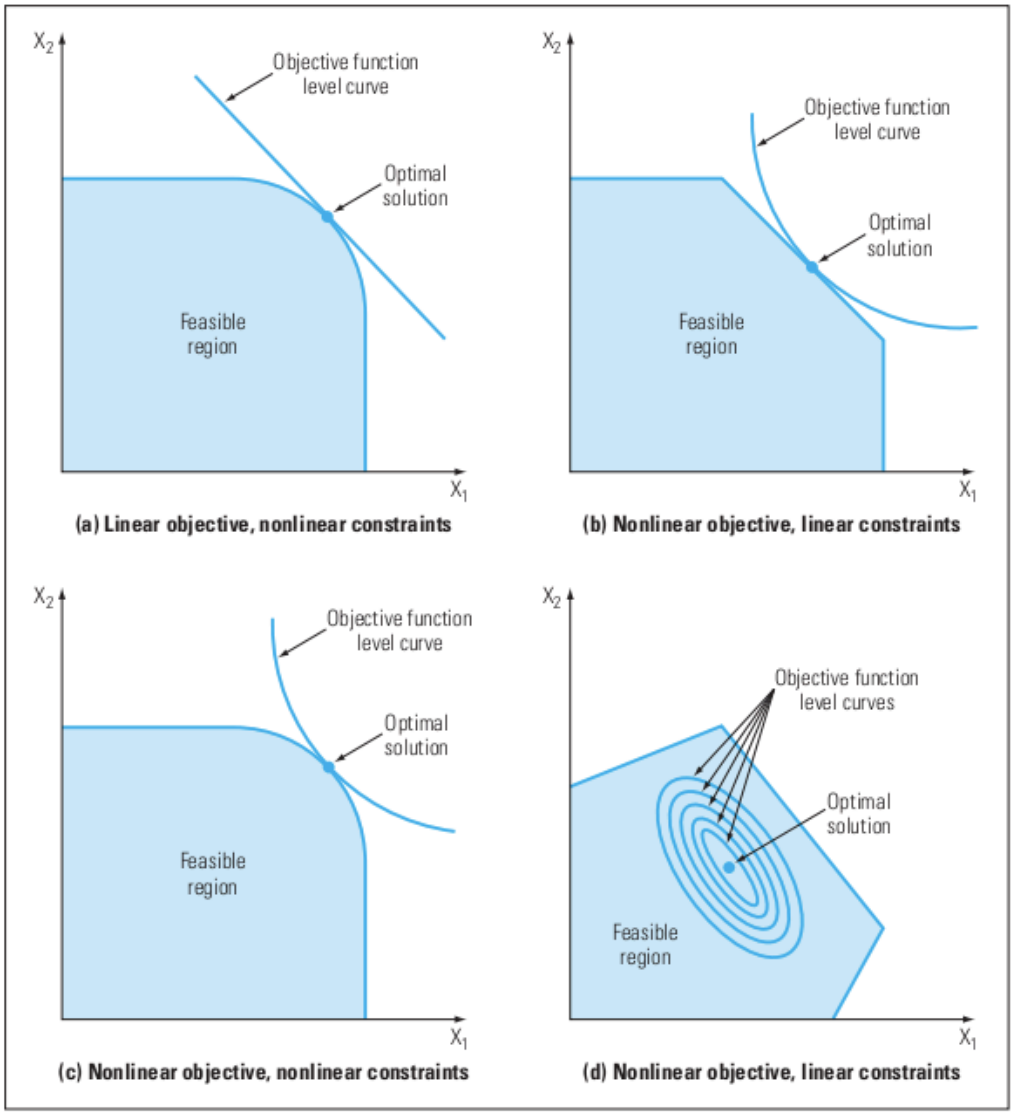
The first graph in Figure 8.1, labeled (a), illustrates a problem with a linear objective function and a nonlinear feasible region. Note that the boundary lines of the feasible region for this problem are not all straight lines. At least one of the constraints in this problem must be nonlinear to cause the curve in the boundary line of the feasible region. This curve causes the unique optimal solution to this problem to occur at a solution that is not a corner point of the feasible region.

The second graph in Figure 8.1, labeled (b), shows a problem with a nonlinear objective function and a linear constraint set. As indicated in this graph, if an NLP problem has a nonlinear objective function, the level curves associated with the objective are also nonlinear. So from this graph, we observe that a nonlinear objective can cause the optimal solution to the NLP problem to occur at a solution that is not a corner point of the feasible region—even if all the constraints are linear.

The third graph in Figure 8.1, labeled (c), shows a problem with a nonlinear objective and a nonlinear constraint set. Here again, we see that the optimal solution to this NLP problem occurs at a solution that is not a corner point of the feasible region.

FIGURE 8.1

Examples of NLP problems with optimal solutions not at a corner point of the feasible region



Finally, the fourth graph in Figure 8.1, labeled (d), shows another problem with a nonlinear objective and a linear constraint set. The optimal solution to this problem occurs at a point in the interior of the feasible region.

These graphs illustrate the major difference between LP and NLP problems—an optimal solution to an LP problem always occurs at a corner point of its feasible region, but this is not true of NLP problems. The optimal solution to some NLP problems might not

occur on the boundary of the feasible region at all but at some point in the interior of the feasible region. Therefore, the strategy of searching the corner points of the feasible region employed by the simplex method to solve LP problems will not work with NLP problems. We need another strategy to solve NLP problems.

8.2 Solution Strategies for NLP Problems

The solution procedure Solver uses to solve NLP problems is called the generalized reduced gradient (GRG) algorithm. The mathematics involved in this procedure are rather complex and go beyond the scope and purpose of this text. However, the following discussion should give you a very basic (if somewhat imprecise) understanding of the ideas behind the GRG and other NLP solution algorithms.

NLP algorithms begin at any feasible solution to the NLP problem. This initial feasible solution is called the starting point. The algorithm then attempts to move from the starting point in a direction through the feasible region that causes the objective function value to improve. Some amount of movement (or a step size) in the selected feasible direction is then taken resulting in a new, and better, feasible solution to the problem. The algorithm next attempts to identify another feasible direction in which to move to obtain further improvements in the objective function value. If such a direction exists, the algorithm determines a new step size and moves in that direction to a new and better feasible solution. This process continues until the algorithm reaches a point at which there is no feasible direction in which to move that results in an improvement in the objective function. When no further possibility for improvement exists (or the potential for further improvement becomes arbitrarily small), the algorithm terminates.

Figure 8.2 shows a graphical example of how a crude NLP algorithm might work. In this graph, an initial feasible solution occurs at the origin (point A). The fastest rate of improvement in the objective function value occurs by moving from point A in the direction that is perpendicular to (or forms a 90-degree angle with) the level curves of the objective function. Feasible movement in this direction is possible from point A to point B where a boundary of the feasible region is encountered. From point B, moving along the edge of the feasible region to point C further improves the objective function value. At point C, the boundary of the feasible region begins to curve; therefore, continued movement in the direction from point B to point C is no longer feasible. From point C, a new direction through the interior of the feasible region allows movement to point D. This process continues from point D until the solution becomes arbitrarily close (or converges) to point E—the optimal solution.

In moving from point A in Figure 8.2, we selected the direction that resulted in the fastest rate of improvement in the objective function. In retrospect, we can see that it would have been better to move from point A in the direction of point E. This direction does not result in the fastest rate of improvement in the objective as we move from point A, but it would have taken us to the optimal solution in a more direct fashion. Thus, it is not always best to move in the direction producing the fastest rate of improvement in the objective, nor is it always best to move as far as possible in that

direction. The GRG algorithm used by Solver takes these issues into consideration as it determines the direction and step size of the movements to make. Thus, although the GRG algorithm usually cannot move directly from a starting point to an optimal solution, it does choose the path it takes in a more refined manner than outlined in Figure 8.2.

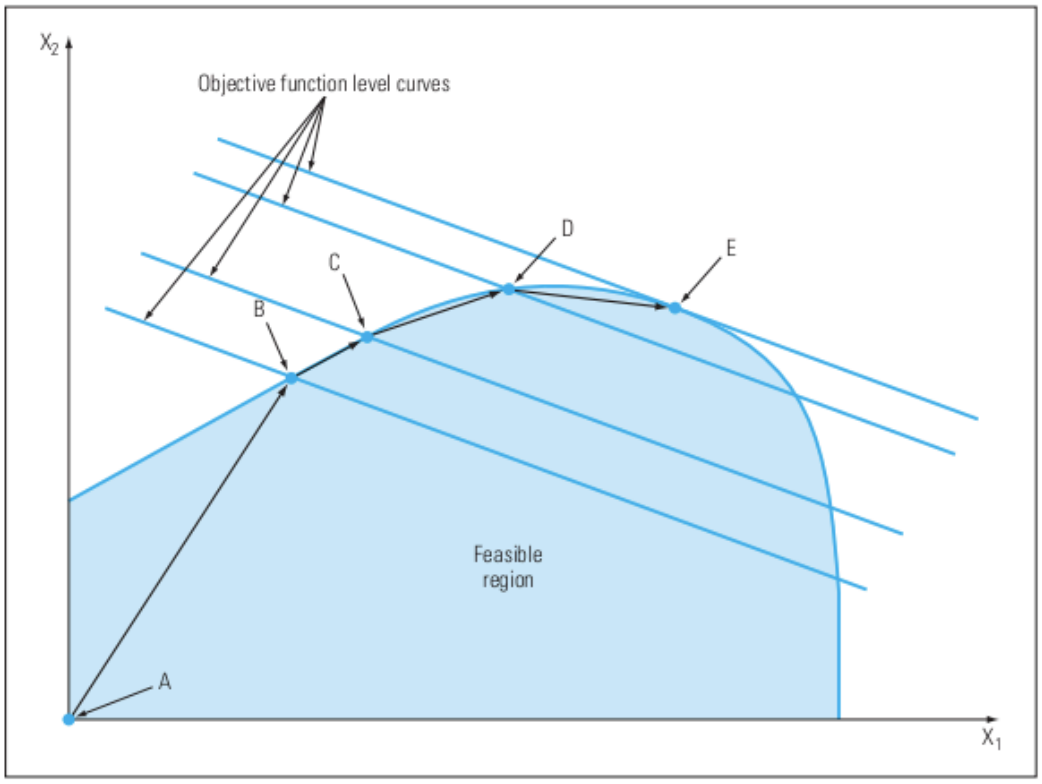


FIGURE 8.2

Example of an NLP solution strategy

8.3 Local vs. Global Optimal Solutions

NLP solution algorithms terminate whenever they detect that no feasible direction exists in which it can move to produce a better objective function value (or when the amount of potential improvement becomes arbitrarily small). In such a situation, the current solution is a local optimal solution—a solution that is better than any other feasible solution in its immediate, or local, vicinity. However, a given local optimal solution might not be the best possible, or global optimal, solution to a problem. Another local optimal solution in some other area of the feasible region could be the best possible solution to the problem. This type of anomaly is illustrated in Figure 8.3.

If an NLP algorithm starts at point A in Figure 8.3, it could move immediately to point B and then along the feasible direction from B to C. Because no feasible point in the vicinity of C produces a better objective function value, point C is a local optimal solution, and the algorithm terminates at this point. However, this is clearly not the best possible solution to this problem. If an NLP algorithm starts at point D in Figure 8.3, it could move immediately to point E, and then follow the feasible direction from E to F

and from F to G. Note that point G is both a local and global optimal solution to this problem.

It is important to note that the feasible region of the problem in Figure 8.3 is nonconvex, while those in Figures 8.1 and 8.2 are convex. A set points X is called a convex set if for any two points in the set a straight line drawn connecting the two points falls entirely within X. (A line connecting points B and E in Figure 8.3 would not fall within the feasible region. Therefore the feasible region in Figure 8.3 is nonconvex.) A function is convex (or concave) if the line connecting any two points on the function falls entirely above (or below) the function. Optimization problems with convex feasible regions and convex (or concave) objective functions are considerably easier to solve to global optimality than those that do not exhibit these properties.

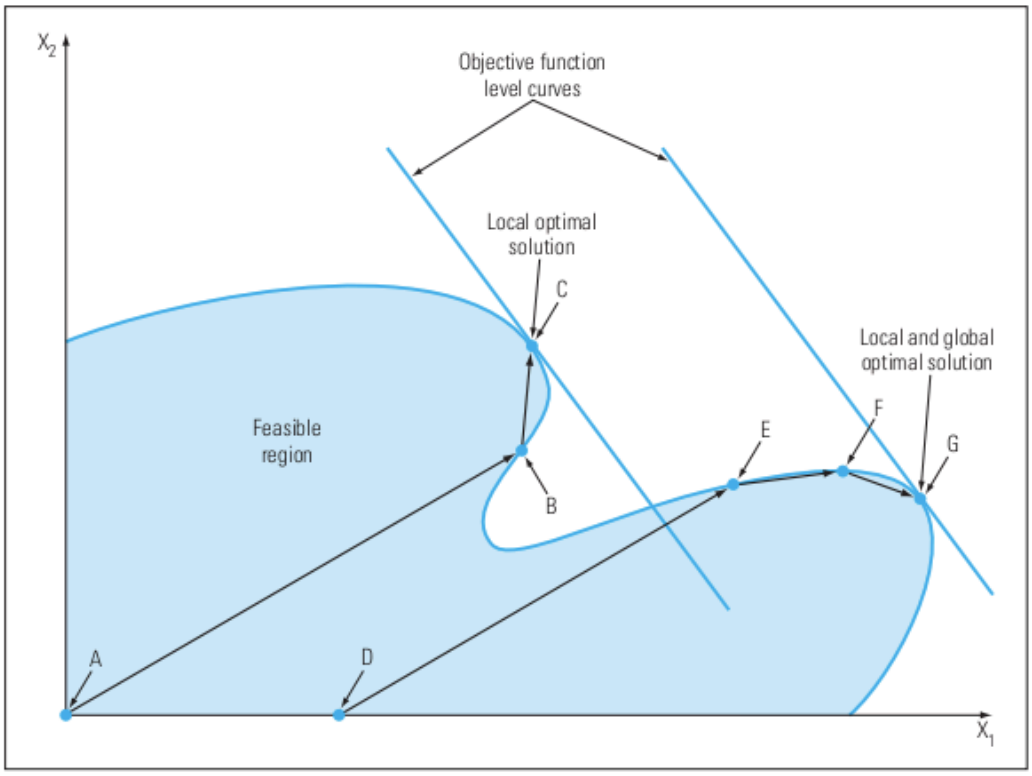


FIGURE 8.3
Local vs. global optimal solutions

Figure 8.3 highlights two important points about the GRG and all other NLP algorithms:

- NLP algorithms can terminate at a local optimal solution that might not be the global optimal solution to the problem.
- The local optimal solution at which an NLP algorithm terminates depends on the initial starting point.

The possibility of terminating at a local optimal solution is undesirable—but we have encountered this type of difficulty before. In our study of integer programming, we

noted that suboptimal solutions to ILPs might be acceptable if they are within some allowable tolerance of the global optimal solution. Unfortunately, with NLP problems, it is difficult to determine how much worse a given local optimal solution is than the global optimal solution because most optimization packages do not provide a way of obtaining bounds on the optimal objective function values for these problems. However, many NLP problems have a single local optimal solution that, by definition, must also be the global optimal solution. So in many problems, NLP algorithms will locate the global optimal solution, but as a general rule, we will not know whether the solution obtained is a global optimal solution. However, in Risk Solver Platform, some information about this issue can be obtained by running the convexity tester (by choosing Optimize, Analyze Without Solving, or by clicking the X-Checkbox icon on the Model tab in the task pane). The result of convexity testing may be Proven Convex, Proven Nonconvex, or Nothing Proven. If you see Model Type - NLP Convex in the Model Diagnosis area of the task pane Model tab, then you know that a local optimal solution is also a global optimal solution. If you see NLP NonCvx or just NLP, then you have to assume that you have only a local optimal solution. In the nonconvex case, it is usually a good idea to try starting NLP algorithms from different points to determine if the problem has different local optimal solutions. This procedure often reveals the global optimal solution. (Two questions at the end of this chapter illustrate this process.)

"Optimal" Solutions

When solving an NLP problem, Solver normally stops when the first of three numerical tests is satisfied, causing one of the following three completion messages to appear:

1. **"Solver found a solution. All constraints and optimality conditions are satisfied."** This means Solver found a local optimal solution, but does not guarantee that the solution is the global optimal solution. Unless you know that a problem has only one local optimal solution (which must also be the global optimal solution), you should run Solver from several different starting points to increase the chances that you find the global optimal solution to your problem. The easiest way to do this is to set the Engine tab Global Optimization group MultiStart option to True before you solve. This will automatically run the Solver from several randomly (but efficiently) chosen starting points.
2. **"Solver has converged to the current solution. All constraints are satisfied."** This means the objective function value changed very slowly for the past few iterations. If you suspect the solution is not a local optimal solution, your problem may be poorly scaled. The convergence option in the Solver Options dialog box can be reduced to avoid convergence at suboptimal solutions.
3. **"Solver cannot improve the current solution. All constraints are satisfied."** This rare message means that your model is degenerate, and the Solver is cycling. Degeneracy can often be eliminated by removing redundant constraints in a model.