Paper 158-2013

Parallel Multistart Nonlinear Optimization with PROC OPTMODEL

Ed Hughes, Tao Huang, and Yan Xu, SAS Institute Inc.

ABSTRACT

Nonlinear optimization has many compelling applications, including finance, manufacturing, pricing, health care, telecommunications, engineering, and statistics. Often a nonlinear optimization problem has many locally optimal solutions, making it much more difficult to identify a globally optimal solution. That's why the multistart feature in PROC OPTMODEL selects a number of initial points and starts optimization from each one, significantly improving your chances of finding a global optimum.

In SAS/OR® 12.1, the multistart feature adds parallel execution. This paper explores the multistart feature and its parallel optimization feature, illustrating with examples drawn from research and industry.

INTRODUCTION

This paper begins with a brief exploration of nonlinear optimization and the challenges it presents, especially if the nature of the functions involved in the constraints or the objective might result in multiple local optima in addition to the global optimum. The multistart approach addresses this difficulty by determining multiple starting points for optimization and reporting back the best locally optimal solution that it finds. Using multiple computational cores in parallel improves both the identification of good starting points and the execution of the optimization process from each of the starting points.

In discussing the multistart feature, this paper describes the algorithms that are used to create a pool of candidate initial points and to select a set of candidates for optimization. It explores the various options that you can use to control and configure starting-point selection and multistart optimization, and it describes the information that the OPTMODEL procedure reports back on the progress and results of the multiple optimization processes.

This paper does not describe the details of the active set and interior point nonlinear optimization methods provided by SAS/OR; instead it focuses on their use in the multistart approach. For more information about these algorithms, see the *SAS/OR 12.1 User's Guide: Mathematical Programming*. This paper assumes that you have a basic understanding of optimization.

INTRODUCTION TO NONLINEAR OPTIMIZATION

Nonlinear optimization models have proven to be useful in many areas of organizational and business planning. Examples of nonlinear relationships and effects that have been captured in optimization modeling include sales revenue in response to price changes, economies of scale in ordering and production, performance of blended chemicals, and returns on financial investments (Wagner 1975). Specialized nonlinear optimization methods are used when nonlinear functions appear in the objective or the constraints of an optimization model. Symbolically, a nonlinear optimization problem can be described as

$$\begin{array}{ll} minimize & f(x) \\ subject\ to & h_i(x) = 0, i \in \{1,2,\dots,p\} \\ & g_i(x) \le 0, i \in \{1,2,\dots,q\} \\ & l \le x \le u \end{array}$$

where $x \in \mathbb{R}^n$ is the vector of decision variables, $f: \mathbb{R}^n \to \mathbb{R}$ is the objective function, $h : \mathbb{R}^n \to \mathbb{R}^p$ is the vector of equality constraints ($h = \{h_1, \dots, h_p\}$), $g : \mathbb{R}^n \to \mathbb{R}^q$ is the vector of inequality constraints ($g = \{g_1, \dots, g_q\}$), and $l, u \in \mathbb{R}^n$ are the vectors of the upper and lower bounds, respectively, on the decision variables. Any or all of the functions $f, g,$ and h can be nonlinear. A feasible solution is a set of values for the decision variables $x_1, \dots, x_n$ that satisfies all the constraints. An optimal solution to this problem is a feasible solution that produces the smallest possible value for the objective function f.

There are no limits on the form that nonlinear functions can take, and consequently the structure of a nonlinear optimization model can vary greatly. In turn, this means that nonlinear optimization methods cannot rely as heavily as other optimization methods (such as linear and mixed integer methods) on the existence of a particular type of mathematical structure in the model to be solved. Nonlinear optimization must, in a sense, be largely noncommittal regarding problem structure. As a result, when a nonlinear optimization method identifies a local minimum (a solution that produces an objective value at least as small as that of any *nearby* feasible solution), you generally cannot draw any conclusion about whether it is also a global minimum (which produces an objective at least as small as *any* feasible solution).

For some types of nonlinear programs, you can make a connection between local optimality and global optimality. A "convex" nonlinear optimization problem is one in which the objective function f is convex, the equality constraint functions h are linear, and the inequality constraint functions g are concave. For this type of problem, any local minimum is also a global minimum. All other types of nonlinear optimization problems are termed "nonconvex" and can possess multiple local minima that are not also global minima. Because nonlinear optimization methods identify only locally optimal solutions, if a problem has many local (and nonglobal) minima, then optimization can report back a globally suboptimal solution. This occurs because a given starting point (initial solution) for optimization can be sufficiently close to a local minimum that the optimization algorithm inevitably proceeds toward and terminates at that local minimum.

THE MULTISTART FEATURE

The multistart feature of the nonlinear optimization solver in SAS/OR helps address the difficulties of solving nonconvex problems that might have multiple local minima. The basic concept is simple: start optimization from several starting points, in hopes of locating local minima of better quality (which have smaller objective function values by definition), and then report back the local minimum that has the smallest objective function value. The main challenges in multistart optimization are selecting good starting points for optimization and conducting the subsequent multiple optimization processes efficiently.

The multistart approach delivers a clear benefit by increasing the quality of the reported optimal solution. If a problem has many local minima, then starting optimization from multiple well-selected initial points makes it more likely that all or most of the local minima are identified. As a result, the identified local minimum that produces the smallest objective value is more likely to be a global minimum. This is helpful if solving a nonlinear optimization problem is your only goal. It is even more beneficial if the optimization is part of a larger solution process that uses the reported optimal solution as input to the next step; a more accurate optimal solution for the optimization problem tends to produce better solutions in later stages. In contrast, a poor optimization solution can substantially invalidate the analyses that are done in succeeding steps of the overall solution process.

Implementing the multistart feature in a parallel computing environment provides even greater benefits. Most obviously, you can execute optimization from several different starting points much more quickly in parallel than serially. More subtly, parallel execution also assists in the preliminary steps of identifying candidate starting points via sampling and using clustering to determine which starting points to select.

THE MULTISTART ALGORITHM

The algorithm that is used in the multistart approach is based on the concept of *regions of attraction* to local minima. The region of attraction to a local minimum for a nonlinear optimization problem is a set of starting points from which optimization converges to that specific local minimum. For the multistart algorithm, the goal is to start optimization exactly once from within the region of attraction of each local minimum, thus ensuring that all local minima are identified and the global minimum is selected. Practically, the algorithm strives to come as close to that ideal as possible within a reasonable amount of time, iterating through four major steps.

UNIFORM SAMPLING

In order for the multistart optimization algorithm to proceed effectively, it must operate within a finite space. You specify this finite space by a combination of explicit bounds that you can include in the formulation of the problem and by using the MSBNDRANGE= parameter for the nonlinear solvers. Within this space, the algorithm selects a set of uniformly distributed points as candidate starting points for optimization.

STARTING-POINT SELECTION

After sampling, the algorithm determines which of the sampled points to select as starting points. First, the objective function value is evaluated at each sampled point. Next, the algorithm uses a result (Rinnooy Kan and Timmer 1987) stating that among the sampled points are sequences of points that have decreasing objective function values, each sequence connecting to a local minimum. Moreover, any two points within a calculated critical distance of each other are members of the same sequence.

Relying on this result, the algorithm performs an implicit form of clustering according to the critical distance; a sample point is selected only if no other points that have a smaller objective function value fall within the critical distance. Because the objective function value decreases as you approach a local minimum along a sequence of points, this method selects from each sequence the sampled point that is as close to a local minimum as possible.

The last step in the selection process is to choose the 10% to 20% of the previously selected points that have the lowest objective function values. The percentage is heuristically determined and helps to ensure that excess starting points are not selected.

LOCAL OPTIMIZATION AND SAMPLE-POINT UPDATE

Local optimization is initiated and completed from each selected starting point. Each identified local minimum is used to update the clustering criteria, because some sampled points might be within the critical distance of the newly found local minimum and thus in its region of attraction. For constrained optimization, Lagrangian multipliers (used to incorporate constraints into the objective function) are updated.

RECURSION

The uniform sampling, starting-point selection, local optimization, and sample-point update steps are repeated in sequence until either no sampled points are selected or the maximum number of local optimizations (specified via the MSMAXSTARTS= parameter) has been completed. At termination the algorithm reports the best local minimum that it has found.

PARALLEL IMPLEMENTATION OF THE MULTISTART ALGORITHM

You can implement the multistart algorithm in parallel for both single-machine (multiprocessor) and distributed computing environments. For sampling, the bounded space is divided into as many subspaces as there are computational threads (or nodes) available. Sampling is performed in parallel in the subspaces. Starting-point selection is also performed in parallel in the subspaces, for points that are sufficiently far from the boundaries of the subspaces; this is termed *local selection*. After local selection concludes, the same selection process, here termed *global selection*, is applied to the comparatively few sample points that are located near subspace boundaries that were excluded from local selection.

Local optimization from the selected starting points is assigned among the available threads (or nodes). Sample-point updates are similarly assigned, and the updates are done in a manner that ensures deterministic results overall—repeated applications of the multistart algorithm to the same problem identify identical local optima. This is especially important if you need to optimize repeatedly to test alternative scenarios. In such a case, it is important to ensure that the only source of variation that influences the results is the makeup of the scenario itself.

The flow diagram in Figure 1 illustrates the steps in the multistart algorithm and indicates how parallel computing is incorporated.

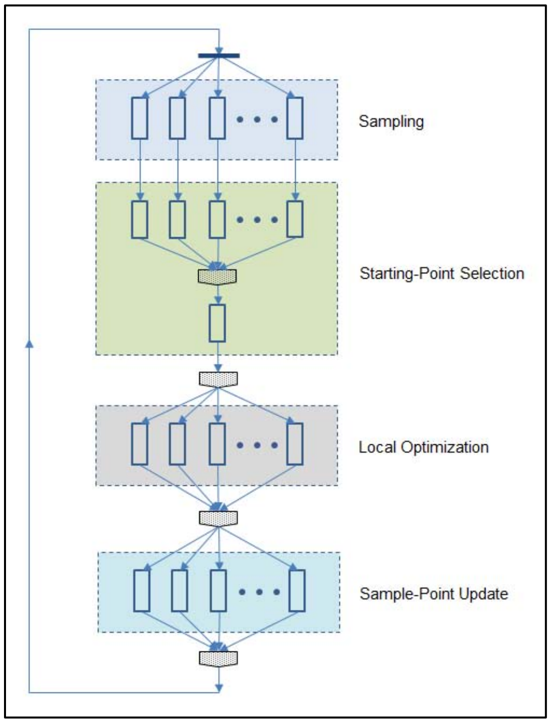


Figure 1. The Multistart Algorithm with Parallel Implementation

Each step is implemented in parallel, as indicated by the multiple parallel nodes in the flow diagram. For the starting-point selection step, the single node that follows the parallel nodes represents global selection.

PROC OPTMODEL SYNTAX FOR THE MULTISTART FEATURE

To invoke the multistart algorithm, you use the WITH clause in the SOLVE statement to specify a nonlinear optimization (NLP) solver and include the MS keyword as an option:

```
   solve with nlp / tech=ip ms;
```

PROC OPTMODEL offers the following options to configure starting-point selection and optimization from the chosen points.

MSBDNRANGE=*M*

ensures that the space in which uniform sampling of starting points occurs is bounded; it supplements any bounds that you specify in the formulation of the nonlinear optimization problem. It’s important to point out that the value of this option (either specified or default) does not affect the feasible region of the nonlinear optimization problem itself; it is used only to contain the sampled space.

If you specify both upper and lower bounds for all decision variables, then any value specified for the MSBNDRANGE= option is not used. If for some decision variable you specify only a lower (upper) bound, then for sampling purposes the value of that variable cannot differ from the lower (upper) bound by more than *M*. If both bounds are missing for a decision variable, then during sampling its value cannot vary by more than *M*/2 from a default or specified starting point.

By default, MSBNDRANGE=200 for a single-machine environment, and MSBNDRANGE=1000 for distributed computing.

MSDISTTOL=ϵ

establishes a tolerance level that determines whether two local minima are considered distinct; the Euclidean distance between them must be at least ϵ in order for them to be deemed distinct. This is important in reducing the work done in the sample-point update step of the multistart algorithm.

By default, MSDISTTOL=1.0E–6.

MSMAXTIME=*T*

defines the maximum time to be used for local optimization by the multistart algorithm; it complements the MAXTIME= option for nonlinear optimization (which specifies the maximum time to be used by a single invocation of the nonlinear optimization solver). Because by its nature the multistart algorithm invokes the nonlinear optimization solver multiple times, the value of the MSMAXTIME= option should exceed the value of the MAXTIME= option. The MSMAXTIME= option is the only option available to control the time that is used by the multistart algorithm because local optimization is responsible for the vast majority of the multistart time. For parallel use of the multistart algorithm, the MSMAXTIME= option controls the local optimization time that is consumed among all threads or nodes.

If you do not specify the MSMAXTIME= option, optimization does not stop based on the amount of time that local optimization consumes. Thus, the default value of the MSMAXTIME= option is effectively infinity.

The value of the TIMETYPE= option determines whether the MSMAXTIME= option refers to real time or CPU time.

MSMAXSTARTS=*n*

limits the total number of starting points that the multistart algorithm uses for local optimization. By default, MSMAXSTARTS=100 for a single-machine environment (serial or parallel); in a distributed computing environment, the default value is the lesser of 100 multiplied by the number of nodes available and 1,000.

MSLOGLEVEL=NUMBER

controls the amount of information that the SAS log displays for the multistart algorithm. A value of 0 blocks all SAS log messages that are related to the multistart algorithm, a value of 1 displays summary information at termination of the algorithm, and a value of 2 (the default) displays the multistart iteration log along with summary information when the algorithm terminates.

PARALLEL IMPLEMENTATION AND THE PERFORMANCE STATEMENT

You can use the PERFORMANCE statement to control the use of threads by the multistart algorithm in a parallel computing environment. The NTHREADS= option specifies the maximum number of threads to be used, defaulting to the value of the CPUCOUNT= SAS system option (the total number of available threads). This setting applies both to computational threads on a single machine and to nodes in a distributed computing grid.

APPLICATIONS OF MULTISTART NONLINEAR OPTIMIZATION

The multistart approach to nonlinear optimization can be useful wherever nonconvex nonlinear optimization is used. Practical instances of nonconvex optimization occur in many industries (Floudas 2010). Examples in chemical engineering include alkylation process design; reactor, reactor sequence, and reactor network design; and heat exchanger design. The optimal design of multiproduct batch plants, in which products are produced in batches and multiple production steps are required for each product, can demand a global optimization approach.

Maximum likelihood estimation of parameters in a nonlinear model is a nonconvex problem that has applications in science and engineering. Practical examples include pharmacokinetics and chemical engineering. Applications of nonconvex optimization in communications include Internet congestion control, wireless network power control, and DSL spectrum management (Chiang 2006).

In medicine, nonconvex optimization can be used to devise optimal radiation treatments for cancerous tumors, manipulating the number, intensity, and angles of radiation beams in order to deliver sufficient radiation to the tumor while minimizing the impact on noncancerous tissue (Bertsimas, Nohadani, and Teo 2009). Equilibrium problems in physics and economics can be modeled as complementarity problems, a type of nonconvex optimization model (Isac, Bulavsky, and Kalashnikov 2010).

In general, because nonconvex optimization denotes the absence of a simplifying mathematical structure in optimization problems and because real-world problems are progressively less likely to possess such structure as you examine them more closely, it's reasonable to assert that techniques such as multistart for dealing with nonconvexity are likely to find increasing application and take on greater relevance as optimization is used more broadly and in greater detail around the world.

EXAMPLE 1: SOLVING A SET OF NONLINEAR EQUATIONS

Suppose you want to find a root of the following pair of nonlinear equations:

$$\begin{aligned} x - \sin(2x + 3y - 52) - \cos(3x - 5y + 36) &= 8 \\ y - \sin(x - 2y + 16) + \cos(x + 3y - 44) &= 12 \end{aligned}$$

For any chosen values of x and y, the approximation errors are defined as the difference between the right-hand and left-hand sides of the equations:

$$\begin{aligned} e_1(x,y) &= x - \sin(2x + 3y - 52) - \cos(3x - 5y + 36) - 8 \\ e_2(x,y) &= y - \sin(x - 2y + 16) + \cos(x + 3y - 44) - 12 \end{aligned}$$

One valid approach is to minimize the sum of squares of these error terms:

$$minimize\ \mathcal{E}(x,y) = \frac{1}{2}\left(e_1(x,y)^2 + e_2(x,y)^2\right)$$

To pursue this approach, the resulting unconstrained nonlinear optimization problem is formally stated as

$$minimize\ \frac{1}{2}\left(e_1(x,y)^2 + e_2(x,y)^2\right)$$

where

$$e_1(x,y) = x - \sin(2x + 3y - 52) - \cos(3x - 5y + 36) - 8$$

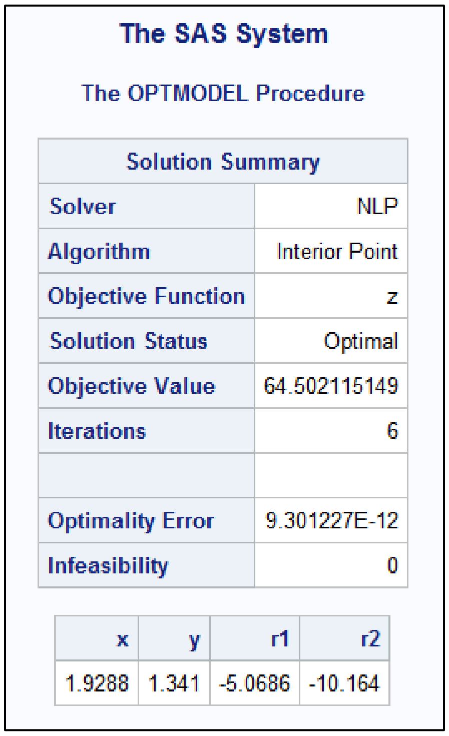
$$e_2(x,y) = y - \sin(x - 2y + 16) + \cos(x + 3y - 44) - 12$$

The following PROC OPTMODEL statements model and solve this optimization problem:

```
proc optmodel;
   var x init 1, y init 1;
   impvar e1 = x - sin(2*x + 3*y - 52) - cos(3*x - 5*y + 36) - 8;
   impvar e2 = y - sin(x - 2*y + 16) + cos(x + 3*y - 44) - 1
   min z = 0.5*(e1^2 + e2^2);
   solve with nlp / tech=ip;
```

```
      print x y r1 r2;
   quit;
```

The IMPVAR (implicit variable) statements establish the two approximation error functions e1 and e2. This SAS program solves the problem once (using the interior point solver) from the initial point (1, 1), which prior experience identifies as a good starting point. The results appear in Figure 2.



The SAS System

The OPTMODEL Procedure

Solution Summary	
Solver	NLP
Algorithm	Interior Point
Objective Function	z
Solution Status	Optimal
Objective Value	64.502115149
Iterations	6
Optimality Error	9.301227E-12
Infeasibility	0

x	y	r1	r2
1.9288	1.341	-5.0686	-10.164

Figure 2. Results of a Single Optimization for Example 1

Brief inspection of the objective function for this problem reveals that it is nonconvex. Based solely on the optimization results in Figure 2, you cannot know whether you have located a global minimum or a local minimum and, if the latter, whether its objective function value is close to that of the global minimum. You do know that when solving for a root of a system of equations, you would like the least squares objective to be as close to 0 as possible. For this problem, however, you have specific information about the appearance of the objective function; it is graphed in Figure 3.

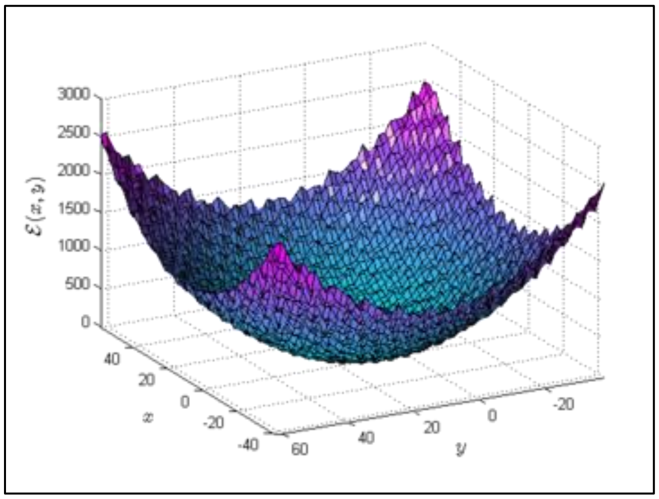


Figure 3. Graph of the Objective Function for Example 1

Even at this relatively low level of detail, you can see that the surface of the objective function is not smooth but instead has many small peaks and troughs, meaning that it has numerous local maxima and local minima. A more detailed view, shown in Figure 4, makes this point even more clearly.

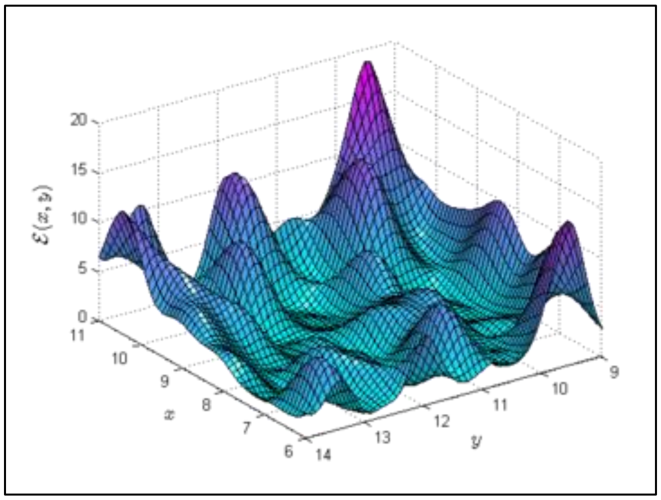


Figure 4. Detailed Graph of the Objective Function for Example 1

Here it's clear that a single optimization might easily converge to a local minimum that has an objective function value that is far from that of the global minimum. Use of the multistart algorithm is strongly recommended.

To invoke the multistart algorithm, only a slight modification of the preceding PROC OPTMODEL code is needed:

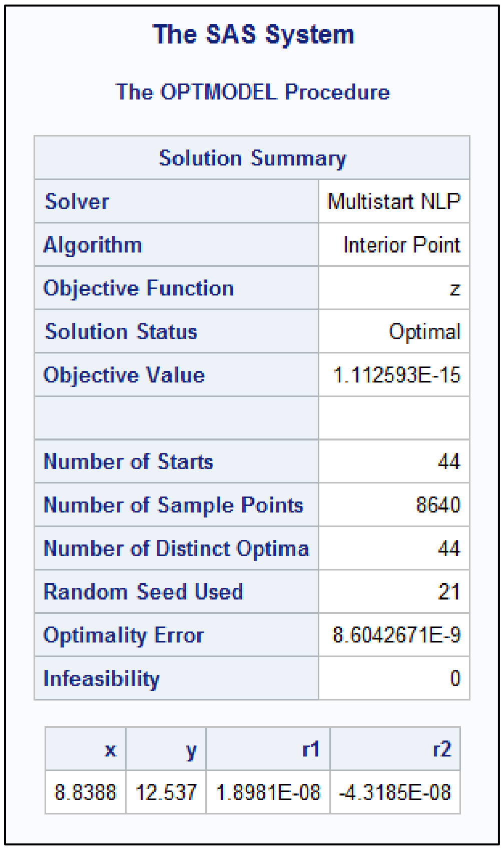
```
proc optmodel;
   var x init 1, y init 1;
   impvar e1 = x - sin(2*x + 3*y - 52) - cos(3*x - 5*y + 36) - 8;
   impvar e2 = y - sin(x - 2*y + 16) + cos(x + 3*y - 44) - 1
   min z = 0.5*(e1^2 + e2^2);
   performance nthreads=4;
   solve with nlp / tech=ip ms seed=21 msmaxstarts=100 msbndrange=50;
   print x y r1 r2;
quit;
```

Here the MS keyword invokes the multistart feature, MSMAXSTARTS=100 limits the number of starting points for optimization, MSBNDRANGE=50 focuses the sampling on a smaller region, and SEED=21 ensures repeatable results. The PERFORMANCE statement specifies that four computational threads can be used. An excerpt from the SAS log for this program appears in Figure 5.

```
NOTE: The Multistart algorithm is executing on the client.
NOTE: The Multistart algorithm is using up to 4 threads.
NOTE: Random number seed 21 is used.
                    Best       Local  Optimality    Infeasi-  Local  Local
      Start    Objective   Objective       Error      bility  Iters  Status
          1   19.8450796  19.8450796  1.68166E-8           0      4  Optimal
          2 * 19.8450796  64.5021151  9.3012E-12           0      6  Optimal
          3   1.1126E-15  1.1126E-15  8.60427E-9           0      6  Optimal
                                         .
                                         .
                                         .
         44   1.1126E-15  158.515678  3.12421E-7           0      3  Optimal
NOTE: The Multistart algorithm generated 8640 sample points.
NOTE: 44 distinct local optima were found.
NOTE: The best objective value found by local solver = 1.112593E-15.
```

Figure 5. SAS Log Excerpt for Multistart Optimization of Example 1

The second optimization (as indicated by the asterisk after the Start column value in the multistart iteration log) begins at the user-supplied starting point, (1, 1), and thus identifies the same local minimum found earlier. Later optimizations that start from other initial points find local minima that have lower objective function values. The values in the Best Objective column of the multistart iteration log decrease steadily. The best local minimum found has an objective function value of 1.112593E–15, just a fraction of the value that is found by optimization from a single starting point and quite close to the desired value of 0 when you are finding a root of a system of equations. A total of 8,640 candidate starting points were created by sampling; of these, only 44 were selected for optimization. The best local minimum is displayed in Figure 6.



The SAS System

The OPTMODEL Procedure

Solution Summary	
Solver	Multistart NLP
Algorithm	Interior Point
Objective Function	z
Solution Status	Optimal
Objective Value	1.112593E-15
Number of Starts	44
Number of Sample Points	8640
Number of Distinct Optima	44
Random Seed Used	21
Optimality Error	8.6042671E-9
Infeasibility	0

x	y	r1	r2
8.8388	12.537	1.8981E-08	-4.3185E-08

Figure 6. Results of Multistart Optimization for Example 1

Parallel implementation of the algorithm enabled all phases of the algorithm to proceed more quickly by enabling work to be done simultaneously on multiple computational threads. In practice, this benefit should be seen most clearly in the local optimization step, because optimization usually accounts for the vast majority of the time that the multistart algorithm consumes.

EXAMPLE 2: HEAT EXCHANGER NETWORK DESIGN

Heat exchangers are used in industry and in industrial design to minimize the cost and maximize the efficiency of required heating and cooling operations. This example is based on Visweswaran and Floudas (1996) and involves two heat exchangers that must handle two incoming streams of hot fluids and one incoming stream of cold fluids. There are three categories of variables: f variables represent flow volume along various branches of the network, ΔT variables represent the temperature changes produced by the two heat exchangers for each of the input streams, and T variables represent variable input and output temperatures for the two heat exchangers. The objective is to minimize the cost of operating the heat exchangers, and the constraints include flow balance, temperature consistency, and targeted output stream temperature requirements. The formulation is as follows:

$$\textit{minimize}\ 1300\left(\frac{1000}{\frac{1}{30}(\Delta T_{1,1}\Delta T_{1,2})+\frac{1}{6}(\Delta T_{1,1}\Delta T_{1,2})}\right)^{0.6}+1300\left(\frac{600}{\frac{1}{30}(\Delta T_{2,1}\Delta T_{2,2})+\frac{1}{6}(\Delta T_{2,1}\Delta T_{2,2})}\right)^{0.6}$$

$$\textit{subject to}\qquad f_{1,1}+f_{2,1}=10$$
$$f_{1,1}+f_{2,3}-f_{1,2}=0$$

$$\begin{aligned}
f_{2,1} + f_{1,3} - f_{2,2} &= 0\\
f_{1,4} + f_{1,3} - f_{1,2} &= 0\\
f_{2,4} + f_{2,3} - f_{2,2} &= 0\\
150 f_{1,1} + T_{2,o} f_{2,3} - T_{1,i} f_{1,2} &= 0\\
150 f_{2,1} + T_{1,o} f_{1,3} - T_{2,i} f_{2,2} &= 0\\
f_{1,2}\left(T_{1,o} - T_{1,i}\right) &= 1000\\
f_{2,2}\left(T_{2,o} - T_{2,i}\right) &= 600\\
\Delta T_{1,1} + T_{1,o} &= 500\\
\Delta T_{1,2} + T_{1,i} &= 250\\
\Delta T_{2,1} + T_{2,o} &= 350\\
\Delta T_{2,2} + T_{2,i} &= 200
\end{aligned}$$

$$\begin{aligned}
\textit{with bounds} \qquad 10 \le\ & \Delta T_{1,1} \le 350\\
10 \le\ & \Delta T_{1,2} \le 350\\
10 \le\ & \Delta T_{2,1} \le 350\\
10 \le\ & \Delta T_{2,2} \le 350\\
0 \le\ & f_{1,j} \le 10, \quad j = 1, \dots, 4\\
0 \le\ & f_{2,j} \le 10, \quad j = 1, \dots, 4\\
150 \le\ & T_{j,i} \le 310, \quad j = 1, \dots, 2\\
150 \le\ & T_{j,o} \le 310, \quad j = 1, \dots, 2
\end{aligned}$$

The following PROC OPTMODEL statements model and solve this problem:

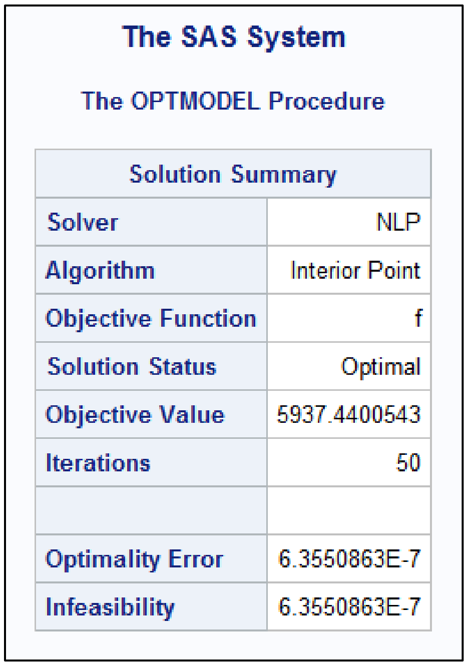
```
proc optmodel;
   var dT11, dT12, dT21, dT22,
       f11, f12, f13, f14, f21, f22, f23, f24,
       T1i, T1o, T2i, T2o;

   min f = 1300*(1000/(1/30*dT11*dT12+1/6*(dT11+dT12)))**0.6
          +1300*(600/(1/30*dT21*dT22+1/6*(dT21+dT22)))**0.6;
   con g1  : f11+f21 = 10;
   con g2  : f11+f23-f12 = 0;
   con g3  : f21+f13-f22 = 0;
   con g4  : f14+f13-f12 = 0;
   con g5  : f24+f23-f22 = 0;
   con g6  : 150*f11+T2o*f23-T1i*f12 = 0;
   con g7  : 150*f21+T2i*f13-T1o*f22 = 0;
   con g8  : f12*T2i-f12*T1i = 1000;
   con g9  : f22*T2o-f22*T1o = 600;
   con g10  : dT11+T2i = 500;
   con g11  : dT12+T1i = 250;
   con g12  : dT21+T2o = 350;
   con g13  : dT22+T1o = 200;


   dT11.lb = 10; dT11.ub = 350; dT12.lb = 10; dT12.ub = 350;
   dT21.lb = 10; dT21.ub = 200; dT22.lb = 10; dT22.ub = 200;
   f11.lb = 0;   f11.ub = 10;   f12.lb = 0;   f12.ub = 10;
   f13.lb = 0;   f13.ub = 10;   f14.lb = 0;   f14.ub = 10;
   f21.lb = 0;   f21.ub = 10;   f22.lb = 0;   f22.ub = 10;
   f23.lb = 0;   f23.ub = 10;   f24.lb = 0;   f24.ub = 10;
   T1i.lb = 150; T1i.ub = 310;  T1o.lb = 150; T1o.ub = 310;
   T2i.lb = 150; T2i.ub = 310;  T2o.lb = 150; T2o.ub = 310;

   solve with nlp /tech=ip;
quit;
```

The SOLVE statement invokes the interior point solver once, and optimization produces a local minimum that has an objective value of 5,937.44005. Figure 7 shows the results of this optimization process.



The SAS System

The OPTMODEL Procedure

Solution Summary	
Solver	NLP
Algorithm	Interior Point
Objective Function	f
Solution Status	Optimal
Objective Value	5937.4400543
Iterations	50
Optimality Error	6.3550863E-7
Infeasibility	6.3550863E-7

Figure 7. Results of a Single Optimization for Example 2

This objective function value is significantly higher than the global minimum (approximately 4,845) for this problem. To find a better solution, you can use the multistart algorithm. Add a PERFORMANCE statement and alter the SOLVE statement as follows:

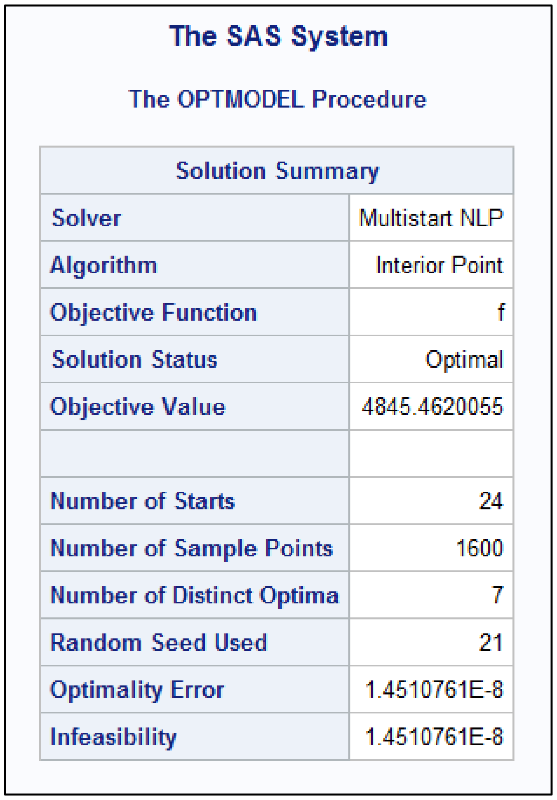
```
performance nthreads=4;
solve with nlp / tech=ip ms seed=21 msmaxstarts=25;
```

As in the previous example, the MS keyword invokes the multistart feature, SEED=21 ensures repeatable results, and MSMAXSTARTS=25 limits the number of local optimizations. Four threads can be used, as specified in the PERFORMANCE statement. An excerpt from the SAS log appears in Figure 8.

```
NOTE: The Multistart algorithm is executing on the client.
NOTE: The Multistart algorithm is using up to 4 threads.
NOTE: Random number seed 21 is used.
                     Best        Local   Optimality    Infeasi-  Local   Local
      Start     Objective    Objective        Error      bility  Iters   Status
          1    5937.43735   5937.43735  3.48738E-8   3.48738E-8     10   Optimal
          2    5937.43735   5937.43735  2.17313E-7   2.17313E-7      7   Optimal
                                       .
                                       .
                                       .
          6 r  5937.43735   5937.43743  7.48096E-7   7.48096E-7     11   Optimal
          7    4845.46201   4845.46201  1.45108E-8   1.45108E-8      7   Optimal
                                       .
                                       .
                                       .
         23    4845.46201   6439.57781  5.39503E-7   5.39503E-7    134   Optimal
         24 r  4845.46201   5937.43972  1.42151E-7   1.42151E-7    257   Optimal
NOTE: The Multistart algorithm generated 1600 sample points.
NOTE: 7 distinct local optima were found.
NOTE: The best objective value found by local solver = 4845.4620055.
```

Figure 8. SAS Log Excerpt for Multistart Optimization for Example 2

By using the multistart algorithm, you find a local minimum that is very close to the global minimum. Among the 1,600 sample points that were created, 24 were selected for local optimization. The best local minimum was located by optimizing from the seventh starting point. Figure 9 shows the solution summary for multistart optimization.



The SAS System

The OPTMODEL Procedure

Solution Summary	
Solver	Multistart NLP
Algorithm	Interior Point
Objective Function	f
Solution Status	Optimal
Objective Value	4845.4620055
Number of Starts	24
Number of Sample Points	1600
Number of Distinct Optima	7
Random Seed Used	21
Optimality Error	1.4510761E-8
Infeasibility	1.4510761E-8

Figure 9. Results of Multistart Optimization for Example 2

CONCLUSION

Nonlinear optimization has many diverse applications, making it a particularly relevant and useful type of optimization. However, nonlinear optimization owes much of its breadth of application to the wide range of mathematical forms that optimization models of this type can assume. Many of these models are nonconvex and thus are especially challenging to solve on a global basis. Multistart nonlinear optimization is an excellent means of improving the quality of solutions to nonconvex optimization problems.

PROC OPTMODEL in SAS/OR makes the multistart nonlinear optimization approach even more valuable by implementing it in parallel. This capability enables you to exploit your available computing resources fully and effectively, whether you work with multiple threads on a single computer or you communicate with a grid of computational nodes. Sampling and evaluation of candidate starting points and local optimization from the selected starting points are as thorough as possible, ultimately producing better solutions.

REFERENCES

Bertsimas, D., Nohadani, O., and Teo, K. M. (2009). “Nonconvex Robust Optimization for Problems with Constraints.” *INFORMS Journal on Computing, Articles in Advance*, 1–15.

Chiang, M. (2006). “Nonconvex Optimization in Communication Systems.” In *Advances in Mechanics and Mathematics*. Vol. 3. Edited by D. Y. Gao and H. D. Sherali. New York: Springer.

Floudas, C. A. (2010). *Handbook of Test Problems in Local and Global Optimization.* Dordrecht, Netherlands: Kluwer Academic.

Isac, G., Bulavsky, V. A., and Kalashnikov, V. V. (2010). *Complementarity, Equilibrium, Efficiency, and Economics: Nonconvex Optimization and Its Applications.* Vol. 63. Dordrecht, Netherlands: Kluwer Academic.

Rinnooy Kan, A. H. G. and Timmer, G. T. (1987). “Stochastic Global Optimization, Part II: Multi Level Methods.” *Mathematical Programming* 39:57-78.

SAS Institute Inc. (2012). *SAS/OR 12.1 User’s Guide: Mathematical Programming.* Cary, NC: SAS Institute Inc.

Visweswaran, V. and Floudas, C. A. (1996). “Computational Results for an Efficient Implementation of the GOP Algorithm and Its Variants.” In *Global Optimization in Engineering Design*, edited by I. E. Grossman. Dordrecht, Netherlands: Kluwer Academic.

Wagner, H. M. (1975). *Principles of Operations Research*. Englewood Cliffs, NJ: Prentice-Hall.

RECOMMENDED READING

- *SAS/OR 12.1 User's Guide: Mathematical Programming*

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Ed Hughes
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
919-531-6916
Ed.Hughes@sas.com

Tao Huang
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
919-531-6128
Tao.Huang@sas.com

Yan Xu
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
919-531-0138
Yan.Xu@sas.com