Paper 194-2011

New Features in SAS/OR® 9.3

Ed Hughes and Manoj Chari, SAS Institute Inc., Cary, NC, USA

ABSTRACT

SAS/OR 9.3 adds a variety of new and enhanced operations research capabilities. This paper surveys these added capabilities, highlighting their advantages and benefits.

In optimization, new nonlinear and quadratic solvers provide improved performance, and a multistart algorithm for nonlinear solvers enables you to search more effectively for globally optimal solutions to difficult problems. SAS® Simulation Studio adds support for 64-bit Windows environments and enhances graphical plots and icons, model editing and viewing, and distribution fitting with JMP® software. The CLP procedure enhances its activity-scheduling capabilities and adds support for a linear objective function, providing an alternate means of optimization.

INTRODUCTION

This paper details the enhancements added for SAS/OR 9.3 and SAS Simulation Studio 1.6 (which is available both as a component of SAS/OR 9.3 and as SAS Simulation Studio for JMP, an add-on product for JMP). The anticipated practical impact of each enhancement on your work with SAS and SAS/OR software is described, and a small number of examples illustrate these enhanced features.

This paper assumes that you have at least a basic level of experience with operations research and accordingly spends relatively little time describing the basic elements of operations research methods and technologies. Instead this paper focuses on how the additional features of SAS/OR 9.3 and SAS Simulation Studio 1.6 make your operations research work easier and more productive.

SAS/OR 9.3 NEW FEATURES AND ENHANCEMENTS

Highlights of the changes in SAS/OR 9.3 include the following:

- The linear programming solver adds a network simplex algorithm for problems with a dominant or embedded network structure.
- The linear programming interior point solver adds an experimental crossover option that converts the optimal solution found to an optimal basic feasible solution.
- The nonlinear programming solver adds an active-set solution algorithm and also adds a multistart capability.
- All optimization solvers (linear, mixed-integer linear, quadratic, and nonlinear) add performance improvements.
- The ability of the CLP procedure to solve scheduling constraint satisfaction problems (CSPs) is now production. The CLP procedure now supports the specification of an objective function.
- The CPM procedure adds a control on the use of progress-update information with resource-constrained schedules.
- The Microsoft Project conversion macro %SASTOMSP is production.

MATHEMATICAL PROGRAMMING

Like every SAS/OR release since the debut of the OPTMODEL family of optimization procedures and solvers, SAS/OR 9.3 includes performance enhancements for a full range of solvers—linear, mixed integer, quadratic, and nonlinear. These improvements enable you to solve optimization problems more quickly and in turn permit you to build and solve optimization models that are greater in scope and detail than was practicable before. Looking further, acceleration of the SAS/OR optimization solvers makes it easier for you to embed optimization into larger solutions that might be subject to stringent “time windows” for delivering results even though they also encompass data integration, other analytic methods, and reporting capabilities.

Linear Optimization: PROC OPTLP and PROC OPTMODEL

The OPTLP procedure solves linear optimization problems specified via an input SAS data set. The OPTMODEL procedure solves multiple classes of optimization problems (including linear optimization) and also includes an algebraic modeling language for creating optimization models and solution algorithms.

For SAS/OR 9.3, two major improvements have been made in linear optimization. First, a new network simplex solver focuses on solving linear optimization problems in which a significant portion of the problem can be modeled as a network, with arcs (or links) connecting nodes that represent physical locations, time periods, specific products or materials, or a combination of these factors. Network structures can be prominent features of optimization models that are created to address business problems in supply-chain design and management, oil and gas production, electrical power generation and distribution, financial investment portfolio planning, and many other areas.

For such problems, the network simplex solver operates by first identifying and extracting the largest possible network structure within the problem. It finds an optimal solution to the network problem and then uses this solution as the core of an advanced initial solution for the entire problem, which is solved with the primal or dual simplex solver. As with other linear programming solvers, the network simplex solver is specified using the SOLVER= option (value NS). The new SOLVER2= option, used only when SOLVER=NS, specifies the solver to be used for solving the overall problem once the network problem has been solved. If SOLVER2= is not specified, the solver determines the best algorithm to use (primal simplex or dual simplex).

A second new feature for linear optimization can improve the usability of solutions generated by the interior point linear programming solver. Typically an interior point solver finds an optimal solution on a face of the feasible region, as opposed to simplex-based methods which locate an optimal solution on a vertex. An interior point solver often locates on optimal solution more quickly than a simplex-based method, but vertex solutions (also known as optimal basic solutions) found by simplex-based methods are usually preferable. While in an interior point optimal solution many decision variables might have nonzero (and often fractional) values, in an optimal basic solution there tend to be fewer nonzero decision variables and more integer-valued decision variables. In a sense, the optimal basic solution represents a more "focused," more practical set of decisions than the interior point optimal solution. In many cases the optimal basic solution is easier to translate into business terms and is also easier to use if the linear optimization just completed is only one step in a larger algorithm that uses the current optimal solution to begin the next step.

The experimental CROSSOVER= option specifies that a solution found by the interior point solver should be converted to an optimal basic solution. CROSSOVER=ON activates the crossover algorithm, which by default (CROSSOVER=OFF) is not used. The crossover algorithm converts the optimal solution found by the interior point solver to an initial basic solution and then calls a simplex solver to produce an optimal basic solution.

Crossover Example for Interior Point Solver

The following simple example illustrates the use of the CROSSOVER= option and the resulting improvement in the optimal solution. Consider this optimization problem:

$$
\begin{aligned}
&\textit{minimize} && x_1 + x_2 + x_3 \\
&\textit{subject to} && x_1 + x_2 \geq 10 \\
& && x_1 + x_3 \leq 4 \\
& && x_2 + x_3 \geq 4 \\
& && x_1, x_2 \geq 0 \\
& && 0 \leq x_3 \leq 3
\end{aligned}
$$

The PROC OPTMODEL statements to build and solve this optimization model with the default (dual simplex) linear optimization solver are straightforward:

```
proc optmodel presolver=none;
   /* declare variables */
   var x{1..3} >=0;

   /* upper bound on variable x[3] */
   x[3].ub = 3;

   /* objective function */
   min obj = x[1] + x[2] + x[3];

   /* constraints */
   con c1: x[1] + x[2] >= 10;
   con c2: x[1] + x[3] <= 4;
   con c3: x[2] + x[3] >= 4;

   solve with lp;

   /* print solution */
   print x;
```

```
quit;
```

The solution summary and the optimal decision variable values are shown in Output 1.

The SAS System

The OPTMODEL Procedure

Solution Summary	
Solver	Dual Simplex
Objective Function	obj
Solution Status	Optimal
Objective Value	10
Iterations	1
Primal Infeasibility	0
Dual Infeasibility	0
Bound Infeasibility	0

[1]	x
1	0
2	10
3	0

Output 1. Solution Summary and Optimal Decision Variable Values: Dual Simplex Solver

In order to solve this problem with the interior point method, only a minor adjustment is needed in the SOLVE statement:

```
solve with lp/solver=ii;
```

The solution summary and optimal decision variable values for the interior point solver are shown in Output 2.

The SAS System

The OPTMODEL Procedure

Solution Summary	
Solver	Iterative Interior
Objective Function	obj
Solution Status	Optimal
Objective Value	10
Iterations	5
Primal Infeasibility	1.244301E-17
Dual Infeasibility	0
Bound Infeasibility	0
Duality Gap	0

[1]	x
1	3.1446
2	6.8554
3	0.0000

Output 2. Solution Summary and Optimal Decision Variable Values: Interior Point Solver

Comparing the two optimal solutions, you can see that the dual simplex solution (0, 10, 0) is simpler than the interior point solution (3.1446, 6.8554, 0). However, by using the CROSSOVER=ON specification, you can convert the interior point optimal solution to a basic optimal solution. Simply add the specification to the SOLVE statement:

```
solve with lp/solver=ii crossover=on;
```

The solution summary and final decision variable values produced are shown in Output 3. Note that the decision variable values at optimality now match those found by the dual simplex method.

The SAS System

The OPTMODEL Procedure

Solution Summary	
Solver	Iterative Interior
Objective Function	obj
Solution Status	Optimal
Objective Value	10
Iterations	4
Primal Infeasibility	0
Dual Infeasibility	0
Bound Infeasibility	0
Duality Gap	0

[1]	x
1	0
2	10
3	0

Output 3. Solution Summary and Optimal Decision Variable Values: Interior Point Solver with Crossover

Although for this simple example the dual simplex solver produces a better solution more quickly than the interior point solver, for some more complex models (especially those with sparse constraint matrices) the combination of the interior point solver and the crossover operation can find a basic optimal solution much more quickly than purely simplex-based methods.

PROC OPTMODEL and Nonlinear Optimization

The OPTMODEL procedure enables you to build and solve multiple classes of optimization models, including linear, mixed integer, quadratic, and general nonlinear programming. For SAS/OR 9.3, PROC OPTMODEL adds features that increase your ability to control how optimization models built with PROC OPTMODEL are stored and how solutions found with PROC OPTMODEL are formatted. The SAVE MPS and SAVE QPS statements add the ability to specify an objective function, which enables you to determine which among several alternative objectives is saved as the problem objective in the MPS or QPS data set. The CREATE DATA statement adds options that enable you control the formatting, length, and labeling of output data set variables created from an optimization model or its solution.

For nonlinear optimization with PROC OPTMODEL, two significant upgrades are delivered with SAS/OR 9.3. An experimental active-set solver performs well for both small and large scale problems. The active-set solver is often the preferred solver if the problem being addressed contains only bound constraints; it can also deliver superior performance for other classes of problems.

In SAS/OR 9.3, the nonlinear optimization solver also introduces a multistart method, which investigates multiple starting points for the optimization process. This approach is useful for nonlinear optimization problems that might have many locally optimal solutions; this is common if either or both of the objective and constraint functions are non-convex. The situation is akin to exploring a mountain range, trying to find the highest of all the peaks. Each peak is higher than the immediately surrounding terrain (locally optimal), but just finding a single peak does not make it possible to determine whether it is also the highest of all the peaks (globally optimal). It is better to start exploring at several points and to compare the corresponding peaks that are thus located.

With the multistart algorithm, the nonlinear optimization solver starts at several different initial points and then reports back the best among the multiple locally optimal solutions that it finds. This approach makes it far more likely that the optimization algorithm will locate a globally optimal solution, and at least should identify a better solution than would be found with a single starting point.

In the first phase of the multistart algorithm, the feasible region of the optimization problem is explored and candidate starting points are identified as the most likely to produce good locally optimal solutions. In the second phase, a subset of the candidates is selected (according to criteria designed to produce better and distinct locally optimal solutions) and used as starting points for the nonlinear optimization solver. Two options can be used to control these phases: the MSBNDRANGE= option limits the initial exploration of the feasible region and is especially useful with unbounded problems, and the MSNUMSTARTS= options specifies the number of starting points to be used. The MSPRINTLEVEL= option controls the amount of detail shown in the SAS log regarding the details of the solver's progress from each of the starting points.

Once the multistart algorithm has concluded, the solution with the best objective function is reported. The .msinit suffix can be used to produce the values of the decision variables at the starting point that ultimately lead to the discovery of this solution.

Multistart Nonlinear Programming Example

The following nonlinear optimization problem is a minimization for five decision variables over a space that contains several local minima. The challenge is to avoid stopping at a local minimum rather than locating a global minimum, which is the goal of the optimization. The difficulty is that you cannot "see" this five-dimensional space in any real sense and thus you cannot identify which starting points are near local minima versus global minima. The best approach in cases such as this is to use a multistart approach and compare the claimed optimal solutions that are found from each selected starting point.

The problem, taken from Hock and Schittkowski (1981), is as follows:

$$
\begin{aligned}
&\textit{minimize} && f(x) = (x_1 - 1)^2 + (x_1 - x_2)^2 + (x_2 - x_3)^3 + (x_3 - x_4)^4 + (x_4 - x_5)^5 \\
&\textit{subject to} && x_1 + x_2^2 + x_3^3 = 2 + 3\sqrt{2} \\
& && x_2 + x_4 - x_3^2 = -2 + 2\sqrt{2} \\
& && x_1 x_5 = 2 \\
& && -5 \le x_i \le 5, i = 1, \ldots, 5
\end{aligned}
$$

The following PROC OPTMODEL statements call the nonlinear optimization solver in multistart mode to solve this problem:

```
proc optmodel;
   var x{i in 1..5} >= -5 <= 5 init -2;

   min f=(x[1] - 1)^2 + (x[1] - x[2])^2 + (x[2] - x[3])^3 +
         (x[3] - x[4])^4 + (x[4] - x[5])^4;

   con g1: x[1] + x[2]^2 + x[3]^3 =  2 + 3*sqrt(2);
   con g2: x[2] + x[4]   - x[3]^2 = -2 + 2*sqrt(2);
   con g3: x[1]*x[5] = 2;

   solve with nlp/ms msnumstarts=3 msprintlevel=3 printfreq=50;
   print x.init x.msinit x;
quit;
```

The PROC OPTMODEL statements create the optimization model that corresponds to the formulation, defining the decision variables, objective, and constraints. The SOLVE statement specifies that the multistart method be used with three starting points (MSNUMSTARTS=3), and further specifies (MSPRINTLEVEL=3) that the progress of the optimization be tracked from all three starting points. The PRINTFREQ=50 option limits the volume of progress tracking information produced. At optimality the procedure prints, for each decision variable, the initial value specified when the variable was declared (x.init), the starting value that produces the best locally optimal solution (x.misinit),

and the value in the best locally optimal solution found (x).

Output 4 shows the progress tracking for one of these starting points.

```
NOTE: The solver is called with starting point 2.
                        Objective                           Optimality
            Iter            Value     Infeasibility            Error
               0     2191.15299165       15.28205927       68.72857614
              50        2.29737977        0.00006383        0.35732798
             100        0.10416978        0.00002183        0.26033668
             120        0.02931083   0.0000000929926   0.0000000929926
NOTE: Optimal.
NOTE: Objective = 0.0293108334.
NOTE: Best objective = 0.0293108334.
NOTE: Best objective found at starting point 2.
```

Output 4. Progress Tracking (SAS Log) for a Selected Starting Point

Output 5 shows the solution summary produced by PROC OPTMODEL along with the initial, best starting point, and best locally optimal values for the decision variables.

The SAS System

The OPTMODEL Procedure

Solution Summary	
Solver	NLP/INTERIORPOINT
Objective Function	f
Solution Status	Optimal
Objective Value	0.0293108334
Iterations	3
Optimality Error	9.2992635E-8
Infeasibility	9.2992635E-8

[1]	x.INIT	x.MSINIT	x
1	-2	-3.1504	1.1166
2	-2	4.7009	1.2204
3	-2	-1.0018	1.5378
4	-2	-2.4060	1.9728
5	-2	4.2160	1.7911

Output 5. Solution Summary and Decision Variable Values

A more extensive exploration of this problem can be found in Example 7.5 in the *SAS/OR 9.3 User's Guide: Mathematical Programing.*

THE CLP PROCEDURE

The CLP procedure is a finite-domain constraint programming solver for solving constraint satisfaction problems (CSPs) with linear, logical, global, and scheduling constraints. Often constraint programming provides a more direct method of modeling and solving highly constrained problems than traditional optimization approaches such as integer programming. In some cases, constraint programming can model and solve problems that cannot practically be addressed with traditional optimization methods. The CLP procedure, including its use of scheduling constraints, is production in SAS/OR 9.3.

New in SAS/OR 9.3, the _TYPE_ variable values MAX and MIN enable you to specify an objective in the Constraint input data set. The experimental OBJ statement enables you to set upper (UB= option) and lower (LB= option) bounds on the value of an objective function that is specified in the Constraint data set. You can also use the OBJ statement to specify the tolerance (TOL= option) used for finding a locally optimal objective value. If upper and lower bounds for the objective value are not specified, the CLP procedure tries to derive bounds from the domains of the variables that appear in the objective function.

The variable and activity selection strategy options have been expanded in SAS/OR 9.3. The ACTSELECT= option in the SCHEDULE statement adds the PRIORITY value, specifying that activities of highest priority be selected to break ties between activities with identical start times. The experimental EVALACTSEL option in the SCHEDULE statement evaluates all of the possible activity selection strategies by attempting to find a solution with each, and the experimental EVALVARSEL option in the PROC CLP statement does the same for variable selection. The new macro variables _ORCLPEAS_ and _ORCLPEVS_ record the results of these evaluations.

Constraint Programming Example: Workforce Scheduling Optimization

In this problem six workers (Alan, Bob, John, Mike, Scott, and Ted) are to be assigned to work in one or more of three working shifts. Alan cannot work the first shift, and Bob can work only the third shift. Along with these constraints, the first shift needs between one and four workers, the second shift needs between two and three workers, and exactly two people must work on the third shift. The goal is to minimize the costs associated with the work assignments. With the variable Ci representing the cost of assigning worker i and Wi representing the shift to which worker i is assigned (i=1, 2,…, 6 corresponds to Alan, Bob,…, Ted), the Constraint input data set for PROC CLP is as follows, specifying the objective of minimizing total cost:

```
data indata;
    input C1 C2 C3 C4 C5 C6 _TYPE_ $ _RHS_;
    datalines;
1 1 1 1 1 1 MIN .
;
```

The PROC CLP call begins by using the OBJ statement to set upper and lower bounds on the objective function so that finite domain constraint programming can be used to solve this problem. The decision variables W1–W6 and C1–C6 are declared. The work-shift requirements and the workers' work-shift restrictions are specified using GCC and LINCON statements, respectively:

```
proc clp data=indata out=clpout;
obj lb=1 ub=100;

/* Six workers (Alan, Bob, John, Mike, Scott and Ted)
are to be assigned to 3 working shifts. */
var (W1-W6) = [1,3];
var (C1-C6) = [1,100];

/* The first shift needs at least 1 and at most 4 people;
the second shift needs at least 2 and at most 3 people;
and the third shift needs exactly 2 people. */
gcc (W1-W6) = ( ( 1, 1, 4) ( 2, 2, 3) ( 3, 2, 2) );

/* Alan doesn't work on the first shift. */
lincon W1 <> 1;

/* Bob works only on the third shift. */
lincon W2 = 3;
```

Finally the relationships between costs and assignments of workers to shifts are specified using ELEMENT statements, and PROC CLP is run.

```
/* Specify the costs of assigning the workers to the shifts.
Use 100 (a large number) to indicate an assignment
that is not possible.*/
element (W1, (100, 12, 10), C1);
element (W2, (100, 100, 6), C2);
element (W3, ( 16, 8, 12), C3);
element (W4, ( 10, 6, 8), C4);
element (W5, ( 6, 6, 8), C5);
element (W6, ( 12, 4, 4), C6);

run;
```

The output data set, specified by the OUT=CLPOUT option in the PROC CLP statement, is printed and shows the optimal values of the work shift assignments and associated costs.

```
proc print;
run;
```

The output shown in Output 6 indicates that Alan and Bob are assigned to the third shift; John, Mike, and Ted to the second shift; and Scott to the first shift. All the constraints are satisfied by this assignment at a total cost of 40. This analysis was completed, thanks to the OBJ statement, with a single run of PROC CLP. Without the ability to specify an objective and bounds on its value, multiple runs of PROC CLP (each specifying a target value for the objective) would have been required.

The SAS System

Obs	W1	W2	W3	W4	W5	W6	C1	C2	C3	C4	C5	C6
1	3	3	2	2	1	2	10	6	8	6	6	4

Output 6. Optimal Work-Shift Assignments and Costs

THE CPM PROCEDURE

The CPM procedure performs project and resource scheduling and can be used for planning, controlling, and monitoring a project. For SAS/OR 9.3, PROC CPM adds the experimental SETFINISH= option, which enables progress update information to override resource considerations when determining the resource-constrained finish times of activities. Specifying SETFINISH=EARLY gives priority to progress-update information, enabling you to account for alterations to work done by resources that might result in altered finish times. The default value SETFINISH=MAX sets the resource-constrained finish times to ensure that all resources assigned to an activity complete their work as originally scheduled.

MICROSOFT PROJECT CONVERSION MACROS

The SAS macro %MSPTOSAS converts Microsoft Project 98 (and later) data into SAS data sets that can be used as input for project scheduling with SAS/OR software. This macro generates the necessary SAS data sets, determines the values of the relevant options, and invokes the PM procedure with the converted project data. The %MSPTOSAS macro enables you to use Microsoft Project for the input of project data and still take advantage of the excellent SAS/OR project and resource scheduling capabilities. The %MSPTOSAS macro converts Microsoft Project 98, 2000, 2002, 2003, and 2007 data, including the Microsoft Access Database (MDB) and XML formats.

The %SASTOMSP macro converts data sets that are used by the CPM and PM procedures into an MDB file that is readable by Microsoft Project 2000, 2002, and 2003. The macro converts information that is common to PROC CPM,

PROC PM, and Microsoft Project; this information includes hierarchical relationships, precedence relationships, time constraints, resource availabilities, resource requirements, project calendars, resource calendars, task calendars, holiday information, and work-shift information. In addition, the early and late schedules, the actual start and finish times, the resource-constrained schedule, and the baseline schedule are also extracted and stored as start-finish variables.

Together, the %MSPTOSAS and %SASTOMSP macros enable organizations to maintain a closed-loop relationship between Microsoft Project as an interface and the superior SAS/OR scheduling capabilities. Execution of the %MSPTOSAS and %SASTOMSP macros requires SAS/ACCESS® software.

SAS SIMULATION STUDIO 1.6

SAS Simulation Studio is a graphical application that enables you to build, run, and analyze discrete event simulation models. Application areas include retail, customer service, health care, transportation, and many other industries. The graphical user interface of SAS Simulation Studio provides extensive modeling tools suitable for both novice and advanced simulation users. SAS Simulation Studio 1.6 provides the following enhancements:

- The 64-bit Windows platform is supported along with the 32-bit Windows platform.
- The following improved features and usability enhancements are available:
  - A new set of icons for all blocks improves the appearance of the application.
  - New graphics technology is used for graphical display blocks (bar charts, scatter plots, histograms, and so on).
  - Copy-and-paste capabilities aid in replicating sections of models.
  - A new Snapshot feature provides a scaled-down view of the entire model, which can be used to navigate to sections of interest in larger models that extend beyond the boundaries of one monitor screen.
- The ability to work with data and generate samples from probability distributions is enhanced. You can sample from nonhomogeneous Poisson processes and empirical distributions (discrete and continuous).
- Integration with JMP distribution-fitting capabilities is tighter than in previous releases. Now you can select a candidate fitted distribution from JMP software and with one click transmit the distribution and parameter settings back to the appropriate Numeric Source block in SAS Simulation Studio.
- The following new blocks are available:
  - The Observation Source block enables you to sample an entire observation from a source data set in a single step; this is a useful, compact method for accessing many variables from the same data set that are to be used in a simulation model.
  - The Dataset Writer block, when signaled to do so, writes data collected during a simulation model run to a specified location.
  - The Dataset Holder block also receives data collected during a simulation model run but makes the data available for queries during the same run.
  - The Stats Collector block enables you to collect time-persistent statistics and values.
  - The Table block provides a real-time display of data collected during the simulation run, in tabular form. The Bucket, Number Holder, and the various Stats Collector blocks are among the blocks that can supply data to the Table block.
  - The Stopper block enables you to create a signal that immediately stops a simulation model run and can also trigger the saving of key simulation data near or at the end of the simulation model run.
- Access to SAS software (to run SAS programs during or after a simulation model run) is available not only on the local PC but also via a remote SAS server.

Improvements in Usability

SAS Simulation Studio debuts a new set of icons for all blocks and new graphics technology for graphical display blocks. It also provides several other enhancements aimed at making SAS Simulation Studio easier to use. Copy-and-paste capabilities enable you to replicate whole models or sections of models for reuse. You can copy an individual or compound block from one model to another, whether the two models in question are in the same or different projects. This is essential if a particular multiple-block construct needs to be used (perhaps with minor variations) at several points in a model, or if you want to use one model (or part of it) as an advanced starting point for another model.

Viewing large models is also easier thanks to the new Snapshot feature, which is accessible by right-clicking in the background of a model. The Snapshot (shown in Figure 1) produces a scaled-down view of the entire model with a blue highlighted area that indicates the currently visible portion of the model. By dragging this highlighted area, you can navigate the model and change the portion that is visible in the Model window. A related Track Animation feature, also accessible by right-clicking in the background of a model, causes the visible portion of the model to shift so that the current animation in the model is visible; in effect the visible portion of the model "tracks" or follows model animation as it occurs during the simulation run.

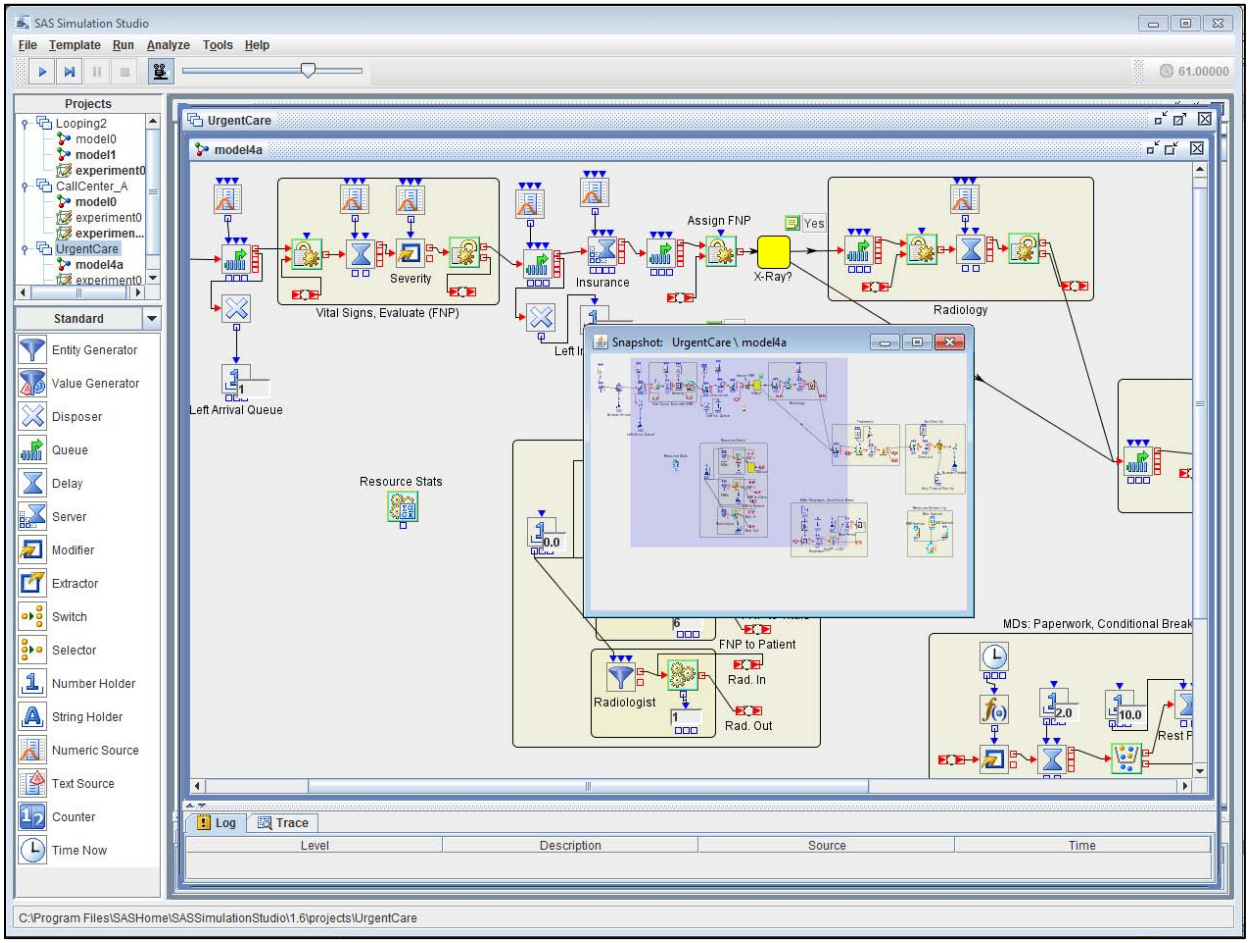

Figure 1. The Snapshot Feature in SAS Simulation Studio 1.6

Another new feature expands the basis on which SAS Simulation Studio can interact with other SAS software and SAS data sets. Although previous releases of SAS Simulation Studio required that any SAS software used be installed on the same PC, SAS Simulation Studio 1.6 also can connect to SAS software that is installed on a remote server. This greatly expands the possible uses of both SAS analytical capabilities and data by SAS Simulation Studio.

Advances in Data Use and Data Creation

SAS Simulation Studio 1.6 enhances its ability to work with empirical data. The new Observation Source block enables you to read an entire observation from a SAS data set or JMP table in a single step; this is an expansion of the ability of the "SAS Data Column" choice for the Numeric Source block, which reads one variable at a time. This enhanced capability is especially useful with models in which a great deal of data must be read from the same data source at one time. In the past, several Numeric Source and Text Source blocks had to be connected to, for example, a Modifier block in order to assign several data-driven attribute values to entities. Now a single Observation Source block replaces them. In general, the Observation Source block enables you to create much more compact models.

An example of a model built prior to the debut of the Observation Source block appears in Figure 2. The Modifier block assigns five attributes (two string attributes and three numeric attributes) to entities. A total of three Numeric Source blocks and two Text Source blocks, each specifying a single variable in the same external SAS data set, must be connected to the input ports of the Modifier block.

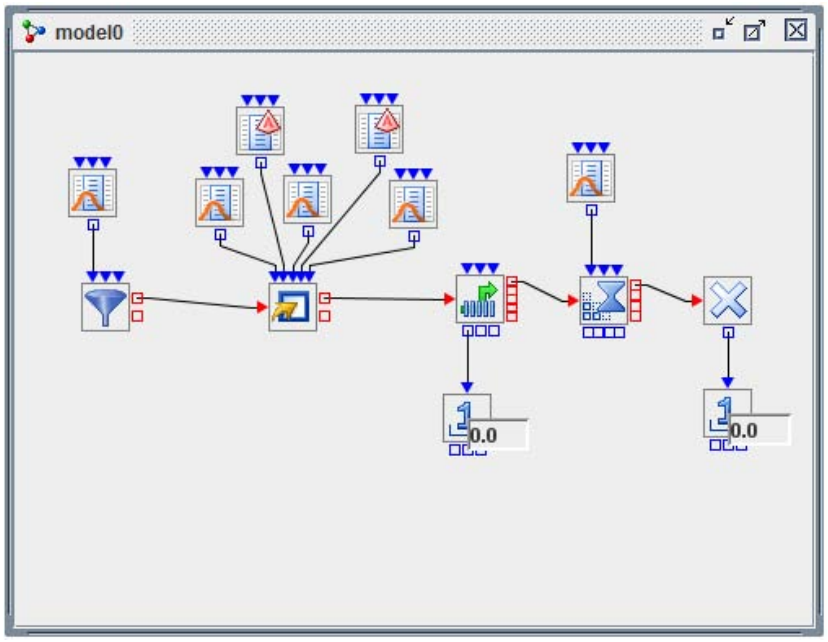

Figure 2. SAS Simulation Studio Model without the Observation Source Block

The model in Figure 3 uses the Observation Source block instead and configures the Modifier block to receive an observation rather than five individual variables. This model is considerably simpler as a result.

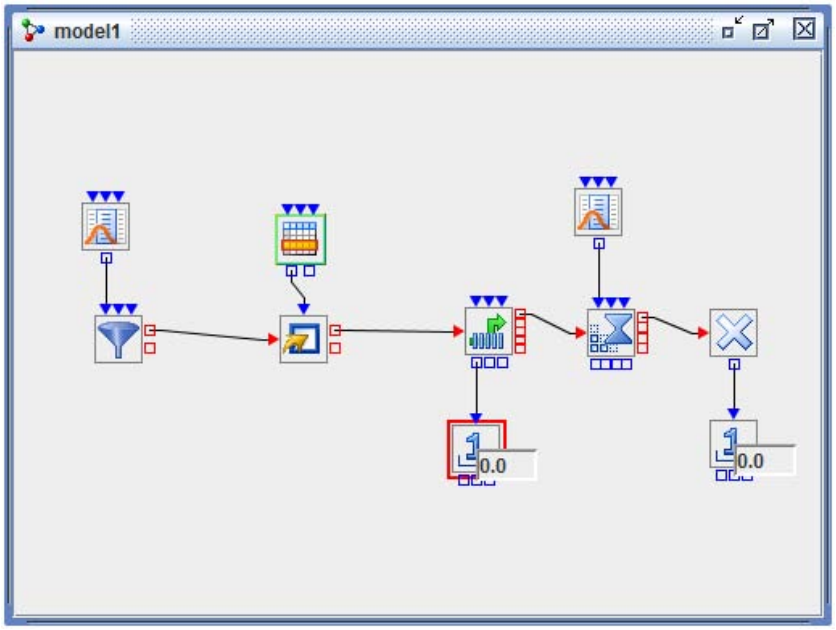

Figure 3. SAS Simulation Studio Model with the Observation Source Block

In another update, SAS Simulation Studio 1.6 is tightly integrated with JMP distribution-fitting capabilities, which are now directly incorporated into the Numeric Source block. This integration enables you to use the JMP "fit all" capability with your data to view numerous candidate distributions and graphs of their respective fits. Selecting your preferred fit among these candidate distributions automatically populates the appropriate Numeric Source block with the chosen distribution and its parameter values. Figure 4 shows the new Numeric Source block dialog box with the **Fitted** option selected. The distribution selected in JMP has been sent back to SAS Simulation Studio and is shown in the bottom half of the dialog box along with its chosen parameters.

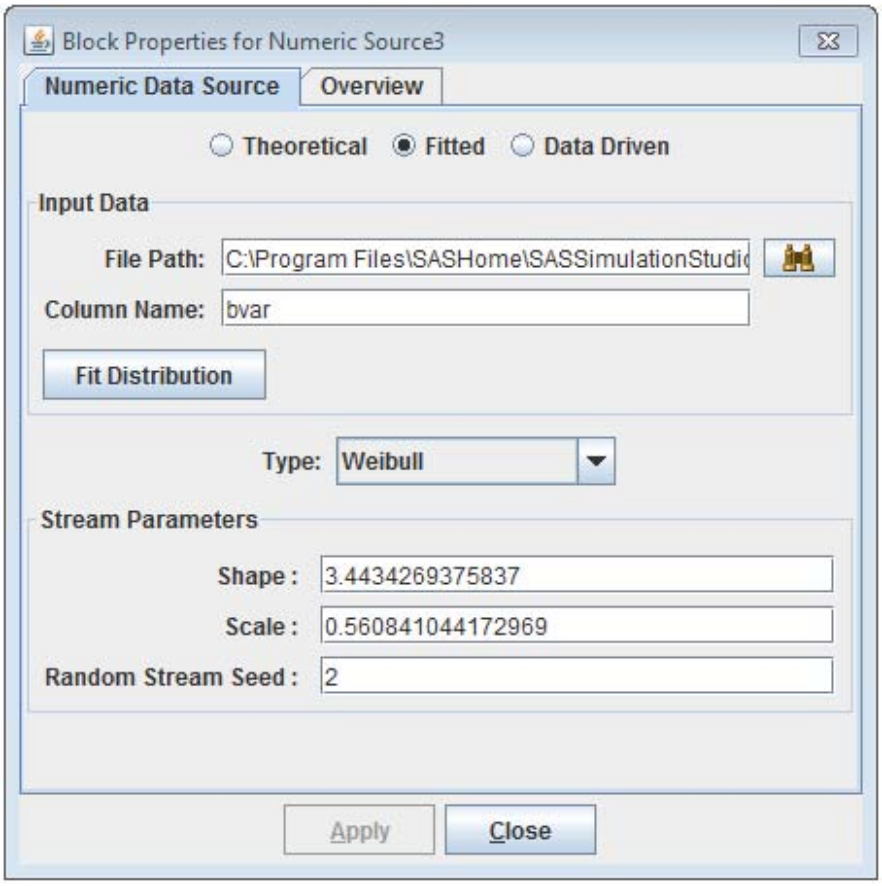

Figure 4. Numeric Source Block Dialog Box with Distribution as Fit by JMP

SAS Simulation Studio 1.6 also adds new capabilities for building and sampling from data-driven probability distributions. You can use data to specify a discrete empirical distribution (by providing data that list a set of possible individual values and their corresponding probabilities of occurrence) or a continuous empirical distribution (by providing data that describe possible intervals of values and their corresponding cumulative probabilities of occurrence). An empirical distribution enables you to use historical data to characterize a source of variation in your model without first needing to fit a specific probability distribution to your data.

Additionally, you can build nonhomogeneous Poisson processes, featuring an arrival rate that varies over time. These include count-based processes (for which input data specify time intervals and their associated arrival counts) and rate-based processes (for which input data specify time intervals and their associated arrival rates). The judicious use of nonhomogeneous Poisson processes greatly increases the realism of models that must include variations in arrival patterns. This is common in any system that features uncontrolled arrivals, such as those in retail service, medicine, transportation, cargo handling, and elsewhere. Figure 5 shows the Numeric Source block dialog box that specifies a count-based nonhomogeneous Poisson process.

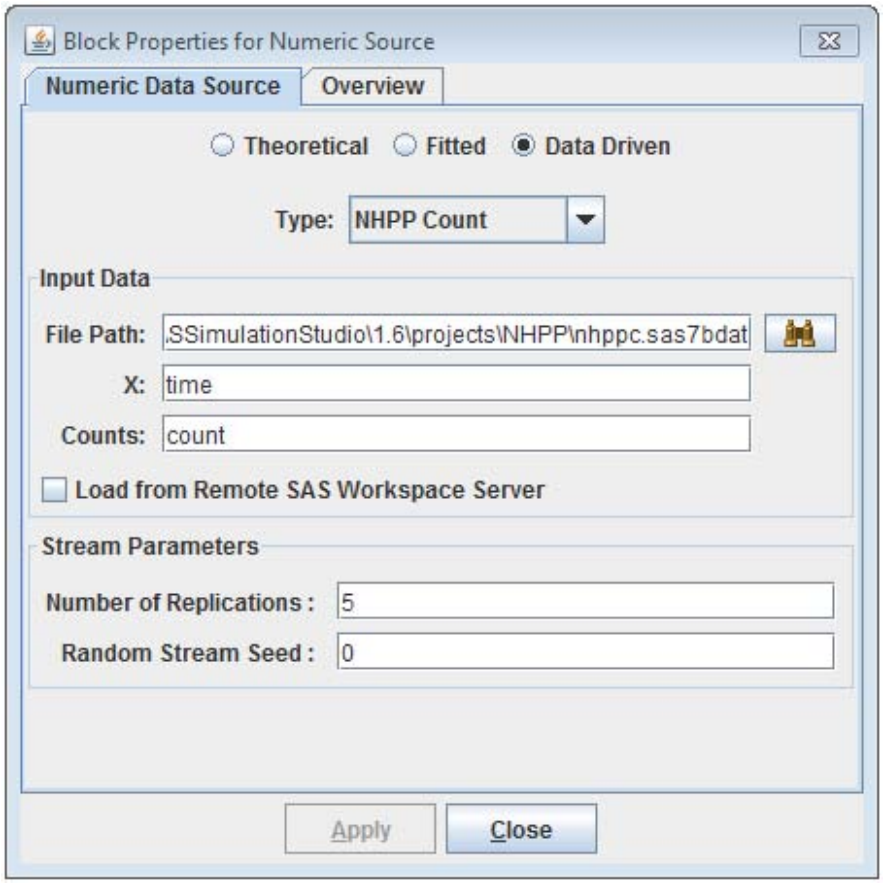

Figure 5. Numeric Source Block Dialog Box with Count-Based Nonhomogeneous Poisson Process Specified

Two new blocks, the Dataset Holder and Dataset Writer, work together to provide more flexible and more extensive access to data, including empirical data used to drive the simulation and the data produced (and possibly consumed subsequently) by the simulation. The Dataset Holder block provides a repository for data and enables customized queries and extractions from the data; it enables you to view and access the entire data set and does not limit you to working with a single variable or a single observation. You can use the Dataset Writer block to create output data at multiple points during the simulation run. Collectively, the Dataset Holder and Dataset Writer block enable event-driven data interactions (read and write) throughout the simulation run. Each is compatible with both SAS data sets and JMP tables.

The Stats Collector block expands your ability to calculate statistics (including time-persistent statistics) on simulation-generated data, generalizing capabilities already found in the Queue Stats Collector and Server Stats Collector blocks to work with any specified sources of data. Finally, the new Stopper block enables you to create an event that immediately stops the simulation run when specified criteria are met and can also trigger the saving of key simulation data near or at the end of the simulation run.

Figure 6 depicts a model that uses the Stopper block to halt the simulation when specified conditions have been met. In this model, entities of Types 1 through 4 are generated and served. The Stopper halts the simulation run when at least 400 units of time have elapsed and at least 20 entities of Type 4 have been served. This is accomplished by the portion of the model in the compound block at the bottom; essentially an auxiliary entity is created at time 400 and recirculates in this compound block every one unit of time, checking whether 20 Type 4 entities have been served. If so, the auxiliary entity flows through a Gate block that sends a true Boolean value to the Stopper block, halting the simulation.

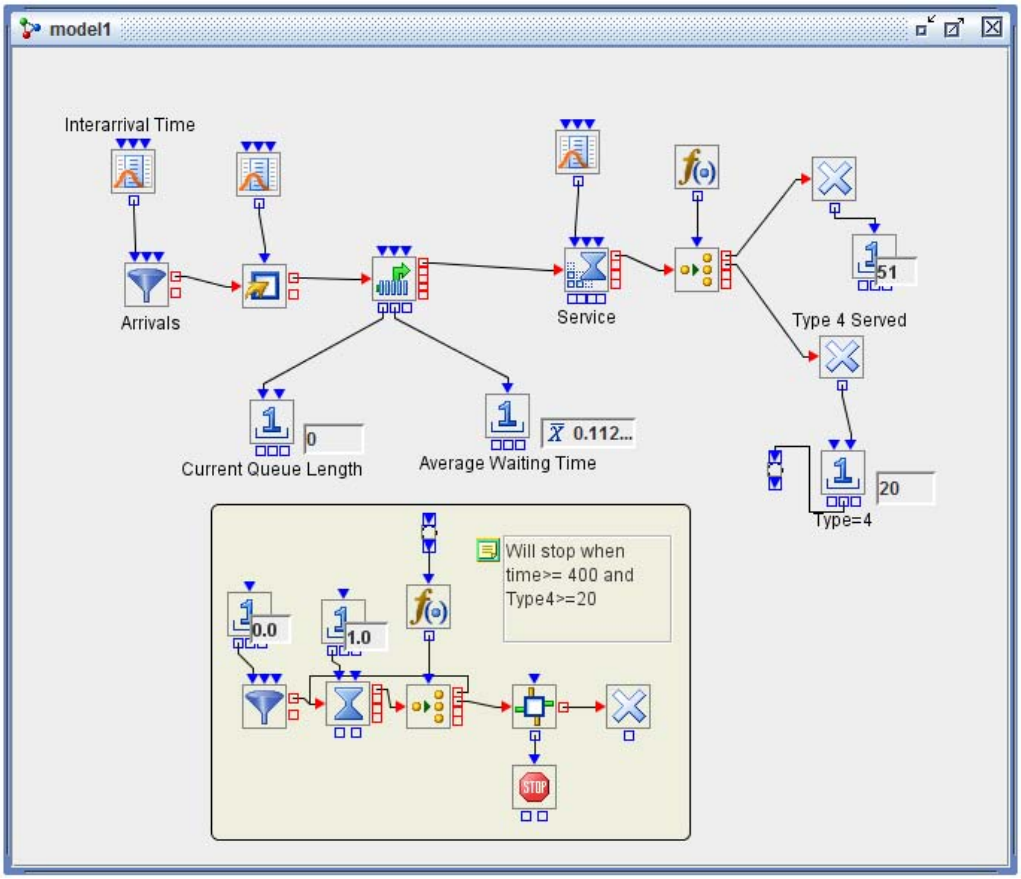

Figure 6. Use of the Stopper Block to Halt a Simulation Run

CONCLUSION

SAS/OR 9.3, including SAS Simulation Studio 1.6, introduces several new features in optimization, constraint programming, scheduling, and discrete event simulation. Collectively these enhancements make SAS/OR software easier to use and also improve its performance significantly. The scope of features and details that you can include in models expands notably, and you can produce results much more efficiently than before for both new and pre-existing models. Ultimately these improvements enable you to build and use operations research models that are more realistic, more detailed, and of greater scale and scope than was possible before.

REFERENCES

Hock,W. and Schittkowski, K. 1981. *Lecture Notes in Economics and Mathematical Systems: Test Examples for Nonlinear Programming Codes*. Berlin: Springer-Verlag.

SAS Institute Inc. 2011. *SAS/OR 9.3 User's Guide: Mathematical Programing*. Cary, NC: SAS Institute Inc.

ACKNOWLEDGMENTS

The authors would like to thank Phillip Meanor, Emily Lada, and Lindsey Puryear for their contributions to this paper and would also like to thank Melanie Gratton, Tao Huang, and Liping Cai for their assistance with the examples shown. Finally, the authors express their gratitude to Anne Baxter for her invaluable contributions as editor.

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
Work Phone: 919-531-6916
Fax: 919-677-4444
Ed.Hughes@sas.com

Manoj Chari
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
Work Phone: 919-531-9274
Fax: 919-677-4444
Manoj.Chari@sas.com