1. Introduction

This paper presents an overview of the enhancements to the SAS/OR software. We assume the reader is familiar with the 82.4 release of the SAS/OR product. In the beginning sections, we discuss the changes to the mathematical programming capabilities of the product. This includes enhancements to PROC LP. We continue the discussion of mathematical programming by introducing PROC TNETFLOW, a new procedure for solving network flow problems with side constraints. In the final sections we present the new capabilities in the area of project management with a discussion of the enhancements to PROC CPM. We also introduce PROC GANTT, a new procedure for producing Gantt charts on printers and high resolution graphic devices.

2. PROC LP Enhancements

PROC LP has been improved in several ways. The most significant of these is the addition of integer and mixed integer programming capabilities. In addition, the procedure has been made interactive, so that the iterative solution process can be both monitored and controlled.

Parametric programming and ranging are two areas for analyzing the sensitivity of the solution to perturbations which have also been included in the Version 5 release. And finally, a new sparse data input format has been integrated into the LP procedure in order to improve performance and allow larger problems to be solved.

Integer and Mixed Integer Programming

The emphasis in the design of PROC LP's integer and mixed integer programming capability has been on flexibility in the control of the solution process. The branch and bound form of implicit enumeration is the algorithm used. The implementation is such that a dual feasible solution is always maintained. As a result, each node in the tree of active problems need only have a minimum amount of data identified with it. Furthermore, because the current solution is dual feasible the objective is decreasing (with each dual pivot) for a maximization and can often enable the node to be fathomed before a primal feasible solution (if there is one) is found.

In conjunction with the flexibility gained by maintaining dual feasibility, the procedure gives the user considerable control over rules for selecting the branching variable and the next node to explore in the branch and bound tree. Moreover, the interactive capabilities allow the user the option of modifying the node and variable branch selection rules during the course of the solution process.

Finally, PROC LP provides the user with a mechanism for stopping the iterative process midstream, saving the current best integer solution (if any), and saving a copy of the current tree of active problems. Then, the user can restart the process continuing iterations where they were left off. As a result, not only can the heuristics used for searching the tree be modified mid-solution, but the tree can be hand pruned.

An example illustrates the use of the integer programming capabilities. Integer variables are identified in the SAS data set that contains the model. The type variable should contain the keyword INTEGER in an observation. Those variables that have non-zero and non-missing values in the observation are interpreted as integer variables. Consider the data set read in Figure 1. The model is an elaboration of a simple product mix example. Briefly put, a company has three products, french fries, hash browns, and flakes. The single period demand for these are 1, 2, and 4, respectively. The company has access to two sources of potatoes, source 1 and source 2. Since the quality of potatoes differs for the two sources the proportion of the three products that a unit from each source can make differs and so does the profit. The data show that one unit of source 1 potatoes generates a profit of 5 whereas one unit of source 2 potatoes generates a profit of 6. The details are apparent from an examination of the data in Figure 1.

So far we have described the standard product mix model. However, suppose that there is a fixed setup cost associated with ordering any source 1 or source 2 potatoes. In particular, suppose that if any source 1 potatoes are ordered there is a fixed charge of 15 and if any source 2 potatoes are ordered there is a fixed charge of 20. Then, the problem becomes a product mix example with setup costs.
The PRODUCT MIX EXAMPLE WITH FIXED SETUP COSTS is a section of output that is new in Version 5. This section reports on the growth of the branch and bound tree. Figure 3 shows the INTEGER ITERATION LOG for the product mix example with fixed setup costs. The ITER column gives the number of the iteration; PROBLEM gives the iteration number from which the current problem is defined, namely the parent node; CONDITION tells about the status of the problem in the tree, OBJECTIVE gives the objective value of the problem; BRANCHED names the variable that is being branched; VALUE gives the value of the branched variable; SINFEAS contains the sum of integer infeasibilities; and ACTIVE tells the number of problems that remain in the tree of active nodes.

By invoking PROC LP as shown in Figure 1, the model is solved. PROC LP first produces the familiar PROBLEM SUMMARY. Now, however, the procedure reports on the number of integer variables as well, as shown in Figure 2.

Figure 2. The PROBLEM SUMMARY.

The SOLUTION, VARIABLE, and CONSTRAINT SUMMARIES are as before.

Figure 4a. The SOLUTION SUMMARY.
PRODUCT MIX EXAMPLE WITH FIXED SETUP COSTS

LINEAR PROGRAMMING PROCEDURE

VARIABLE SUMMARY

<table>
<thead>
<tr>
<th>COL NAME</th>
<th>STATUS</th>
<th>TYPE</th>
<th>PRICE</th>
<th>ACTIVITY</th>
<th>COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOURCE1</td>
<td>DEGEN</td>
<td>NON-NEG</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SOURCE2</td>
<td>BASIC</td>
<td>NON-NEG</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>SETUP1</td>
<td>INTEGER</td>
<td>-15</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>F_FIXES</td>
<td>BLACK</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S_COST1</td>
<td>SLACK</td>
<td>0</td>
<td>0</td>
<td>-20.0000</td>
<td>0</td>
</tr>
<tr>
<td>S_COST2</td>
<td>SLACK</td>
<td>4.0000</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 4b. The VARIABLE SUMMARY.

Figure 4c. The CONSTRAINT SUMMARY.

Interactive Capabilities

The interactive capabilities of PROC LP give the user considerable flexibility and control over the solution process. This is particularly useful when solving integer and mixed integer problems. For example, the user can interactively modify variable and candidate selection rules, and consequently adjust the algorithm to suit the needs of the model.

In addition, the interactive features provide the capability for performing sensitivity analysis while sitting at a terminal. The right-hand-side or price change vector can be specified interactively, the results of the analysis examined, then additional analysis performed, all in a single session.

Furthermore, the LP procedure also has the ability to print submatrices of the coefficient matrix. This allows the user to debug the model within PROC LP.

How is the interactive system used? The user sets break points in the initial call to PROC LP. Break points are times at which the procedure stops execution and waits for further instructions from the user. For example, the option ENDPause on the PROC LP statement tells the procedure to pause after the linear program has been solved but before the solution has been printed.

Break points can also be set after a specified number of pivots or integer iterations, or after a primal feasible or integer feasible solution has been found.

Once the procedure has paused at a break point and is waiting for additional instructions from the user, it can accept instructions: to print the current solution, perform sensitivity analysis on the current solution, print the tableau or coefficient matrix, reset PROC LP options (including the naming of output data sets), and continue its usual execution. For example, the following program causes PROC LP to solve a linear program but pause before printing the solution. The procedure is then instructed to print the section of the VARIABLE SUMMARY that contains the structural variables named in the input data set between X1 and Z inclusive. The user then requests that the submatrix of the coefficient matrix that contains all the rows and the columns X1 to Z be printed. The RESET statement tells PROC LP to save its final primal solution in the output data set named SOLUN. Then, the procedure is instructed to terminate processing and save the current solution as previously instructed in options on the PROC LP and RESET statements.

PROC LP ENDPause;
RUN;
PRINT COL(X1 -- Z);
PRINT MATRIX(_ALL, X1 -- Z);
RESET PRIMALOUT=SOLUN;
QUIT/SAVE;

Parametric Programming

Version 5 of PROC LP has been upgraded to perform parametric programming on the right-hand-side constants and the objective coefficients. The model for parametric programming for the right-hand-side constants finds all solutions for \( \alpha \) for \( 0 \) > \( \alpha \) > \( 0 \) for \( \alpha \) > \( 0 \) to the family of linear programs

\[
\text{max} \ c'x \\
\text{st.} \ Ax = b + \alpha r \\
x \geq 0 
\]

Earlier version of PROC LP provided the range on \( \alpha \) for which the optimal solution (when \( \alpha = 0 \)) remained optimal. Now, you specify the \( \alpha \) on the PROC LP or RESET statements using the
RHS PHI= option and PROC LP will give all the requested solutions.

Analogously, Version 5 of PROC LP finds all solutions of \( ax \geq 0 \) \( (a \geq 0) \) for \( \lambda > 0 \) to the family of linear programs

\[
\begin{align*}
\text{max } & (c + \delta \lambda)'x \\
\text{st. } & Ax = b \\
x & \geq 0.
\end{align*}
\]

The user specifies the value of \( \lambda \) on the PROC LP or RESET statement with the PRICEPHI= option and PROC LP gives all the requested solutions.

Sparse Data Input Capability

The handling of sparse models was a limitation of earlier releases of PROC LP. With Version 5 the user can specify a model in a sparse format so that large models with few non-zero entries can be easily specified. Although not identical to the MPS format, the sparse format is consistent with industry standards.

The SAS data set that specifies the model contains four variables: a TYPE variable, a COLNAME variable, a ROWNAME variable, and a COEF variable. The TYPE variable contains keywords such as LE, GE, EQ, MAX, and MIN, and is used to identify rows in the linear programming model. The COLNAME, ROWNAME, and COEF variables contain the names of a column, a row, and the coefficient for that column and row. There can be numerous pairs of ROWNAME COEF variables in the data set. Figure 3 illustrates the discussion.

Variables in the SAS Data Set

<table>
<thead>
<tr>
<th>TYPE</th>
<th>COLNAME</th>
<th>ROWNAME</th>
<th>COEF</th>
</tr>
</thead>
<tbody>
<tr>
<td>keyword</td>
<td>name</td>
<td>name</td>
<td>coefficient</td>
</tr>
<tr>
<td>of column</td>
<td>of row</td>
<td>for the column and row</td>
<td></td>
</tr>
</tbody>
</table>

Figure 3. The Sparse Input Format.

There are several minor differences between the PROC LP sparse data format and the MPS format. One difference is that in the PROC LP format the COLNAME and ROWNAME variables only contain the names of columns and rows in the model. The MPS format mixes row and column names in the field which is equivalent to the ROWNAME variable. Another difference regards observation order. The PROC LP format does not expect the data to appear in any particular order. As long as all of the information is in the data set the model will be interpreted correctly. Because of the close similarity between the two formats the Version 5 SASMPSX macro, which converts MPS format to PROC LP format, is simpler than previous versions and consists of a single data step. As a result it is significantly faster than older versions.

PROC LP Performance Improvements

In addition to the performance improvements gained from the new sparse data input format and the new SASMPSX macro for data conversion, Version 5 PROC LP has undergone several algorithmic improvements designed to shorten solution time. Among these are new internal data representation schemes designed to minimize the number of multiplications.

Also added to the procedure are default dynamic pricing strategies. These strategies keep the procedure from pricing the entire matrix at each pivot by adjusting the number of columns that are priced dynamically, as the algorithm progresses, in a way that limits the number of times the complete matrix is priced.

The procedure can also internally handle ranges on rows. For example, if you want to specify the two constraints

\[
\begin{align*}
ax & \leq b \\
ax & \geq b - r \quad \text{for } r > 0
\end{align*}
\]

In previous versions you were required to specify two constraints. In Version 5 you can specify these two constraints by placing implicit bounds on the logical variable. This is done automatically by PROC LP when you include a variable in the input data set that contains the range data, \( r \) in this example, and specify that variable with the new RANGE statement. By limiting the number of rows in the model in this way, significant decreases in solution time can be realized.

3. PROC TNETFLOW: A New Procedure

PROC TNETFLOW is a new procedure in SAS/OR. In a test level for Version 5, PROC TNETFLOW will eventually replace PROC NETFLOW. It has many improvements over PROC NETFLOW including the ability to solve network flow problems with side constraints. Many linear programs which are currently solved with PROC LP could be reformulated and solved more easily and efficiently as constrained networks with PROC TNETFLOW. This is true even for problems for which the network structure is not apparent, like multiperiod inventory problems.

PROC TNETFLOW features an optimizer that uses a primal simplex algorithm which minimizes arithmetic by exploiting the network structure. When a network has side constraints, the procedure uses the pure network representation of the basis where possible and an LU decomposition of the basis otherwise. These
features of the optimizer account for the procedure's speed in solving constrained network flow problems.

The procedure also provides the ability to save intermediate solutions in a SAS data set, then restart from these intermediate solutions. This is particularly useful for 'what if' analyses.

An Oil Industry Example

To demonstrate the kind of model that can be solved with PROC TNETFLOW and how one would use the procedure consider the following scenario. An oil company has two suppliers of crude oil, one in the middle east and one in the United States. In addition it has two refineries each of which produces gas and diesel fuel in the ratio of .75 units of gas and .25 units of diesel for each unit of crude. The company also operates two distribution stations, each station demanding different quantities of gas and diesel. Figure 4 shows a network which illustrates the flow of crude oil from the supply points to the refineries, and then as the products gas and diesel, to the station demand points.

An Oil Industry Example

To demonstrate the kind of model that can be solved with PROC TNETFLOW and how one would use the procedure consider the following scenario. An oil company has two suppliers of crude oil, one in the middle east and one in the United States. In addition it has two refineries each of which produces gas and diesel fuel in the ratio of .75 units of gas and .25 units of diesel for each unit of crude. The company also operates two distribution stations, each station demanding different quantities of gas and diesel. Figure 4 shows a network which illustrates the flow of crude oil from the supply points to the refineries, and then as the products gas and diesel, to the station demand points.

The company wants to find the minimum cost routing of crude, gas, and diesel through the network which satisfies the demand with the supply at the supply points without violating any side constraints or the capacities of the shipping routes. Notice that the figure shows the available supplies and the required demands at each of the supply and demand points. However, it does not show the unit shipping costs or the arc capacities.

The requirement that a unit of crude produces .75 units of gas and .25 units of diesel is not represented in Figure 4. These additional constraints must be specified in the side constraints. In particular, let

\[\text{THRUNPUT}_1 = \text{total output from Refinery 1}\]
\[\text{R1.GAS} = \text{gas output from Refinery 1}\]
then the side constraints are written as

\[4 \times \text{R1.GAS} = 3 \times \text{THRUNPUT}_1\]
\[4 \times \text{R2.GAS} = 3 \times \text{THRUNPUT}_2\]

We use PROC TNETFLOW to find the minimum cost flow through the network which satisfies the demand at each of the demand points with the supply at the supply points without violating the side constraints. First, we specify the model to the procedure. This is done in three data sets: a node data set, an arc data set, and a constraint data set. The node data set, named in the PROC TNETFLOW statement with the NODEDATA= option, tells the procedure the names of the supply and demand nodes in the model and the supply and demand data at those nodes. Figure 5 shows the data step for saving the supply and demand node information for this model. Notice that the supply data given in the variable _SD_ are positive quantities while the demand data, also given in the variable _SD_, are negative quantities.

\[
\begin{align*}
\text{DATAS_D_NODE;} \\
\text{INPUT NODE_ $15. _SD_;} \\
\text{CARDS;} \\
\text{MIDDLE EAST} & 100 \\
\text{U.S.A.} & 80 \\
\text{SERVSTN1 GAS} & -95 \\
\text{SERVSTN1 DIESEL} & -30 \\
\text{SERVSTN2 GAS} & -40 \\
\text{SERVSTN2 DIESEL} & -15
\end{align*}
\]

Figure 5. The Node Data Set.

The pure network part of the model is specified in the arc data set named in the PROC TNETFLOW statement with the ARCDATA= option. This data set specifies the network part of the model as is currently done in PROC NETFLOW. The nodes at the head and tail of each arc in the network are given. Also specified are the costs for a unit of flow over the arc, the arc capacity, minimum allowable flow on the arc and an arc name. Figure 6 shows the arc data set for the example network. Notice that several of the arcs have nonzero minimum allowable flows as specified in the variable named _LO. Also notice that although all the arcs could have been named, only those involved in side constraints have been given names.

\[
\begin{align*}
\text{DATAS_A_DATA;} \\
\text{INPUT A_DATA_ $15. $20. $150. _LO. _SD.;} \\
\text{CARDS;} \\
\text{R1_GAS} & \text{Station}_1 \text{ gas (95)} \\
\text{R1_DIESEL} & \text{Station}_1 \text{ diesel (30)} \\
\text{R2_GAS} & \text{Station}_2 \text{ gas (40)} \\
\text{R2_DIESEL} & \text{Station}_2 \text{ diesel (15)}
\end{align*}
\]

Figure 4. An Oil Industry Model
DATA NETWORK;
  INPUT _FROM_ $11._TO_ $15._COST_ _CAPAC_ _LO_ _ARCNAME_ $ ;
CARDS;
  MIDDLE EAST REFINERY 1 63 95 20 .
  MIDDLE EAST REFINERY 2 81 80 20 .
  U.S.A. REFINERY 1 55 .
  U.S.A. REFINERY 2 49 .
  REFINERY 1 R1 200 150 50 THRUPUT1
  REFINERY 2 R2 220 100 30 THRUPUT2
  R1 REFl GAS 140 RLGAS
  REF1 DIESEL 75
  R2 REF2 GAS 100 R2_GAS
  REF2 DIESEL 75
  SERVSTN1 GAS 15 90 .
  SERVSTN2 GAS 22 60 .
  REF1 DIESEL 18
  REF1 GAS 37
  SERVSTN1 GAS 17 35 .
  SERVSTN2 GAS 41 .
  REF2 DIESEL 36
  REF2 GAS 23 .

Figure 6. The Network Data Set.

Finally, the side constraints are specified in the constraint data set named in the PROC TNETFLOW statement with the CONDATA= option. This data set specifies the side constraints similarly to the way the constraints are specified for PROC LP. As with the LP procedure the constraint data set contains a type variable which tells PROC TNETFLOW how to interpret each observation in the data set. Figure 7 shows the constraints needed for the example model. Although in this example the structural variables named in the constraint data set are arcs in the network, PROC TNETFLOW does not require this. The constraint data set can name variables which are not arcs in the model.

DATA CONSTRT;
  INPUT THRUPUT1 R1_GAS THRUPUT2 R2_GAS _TYPE_ $ _RHS_ ;
CARDS;
  -3  4 =  0
  .  -3  4 =  0 .

Figure 7. The Constraint Data Set.

Note that PROC TNETFLOW also supports a sparse input format that is similar to the format supported by PROC LP.

The minimum cost flow is found by invoking PROC TNETFLOW as

PROC TNETFLOW
  NODEDATA=S_D_NODE
  ARCDATA=NETWORK
  CONDATA=CONSTRT
  CONOUT=SOLUTN;
PROC PRINT DATA=SOLUTN; SUM _FCOST_;

As with PROC NETFLOW, PROC TNETFLOW does not produce any printed output. The solution, namely the minimum cost flow, is saved in an output data set named with the CONOUT= option. In this example the data set named SOLUTN is printed with PROC PRINT and shown in Figure 8.

In addition to the input data, for each arc in the network the solution data set contains: the optimal flow (the _FLOW_ variable), the cost of flow (the _FCOST_ variable), the reduced cost (the _RCOST_ variable), and restart information (the _STATUS_ variable).

<table>
<thead>
<tr>
<th>ARCNAME</th>
<th><em>FLOW</em></th>
<th><em>FCOST</em></th>
<th><em>RCOST</em></th>
<th><em>STATUS</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>KEY_ARC</td>
<td>BASIC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KEY_ARC</td>
<td>BASIC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KEY_ARC</td>
<td>BASIC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KEY_ARC</td>
<td>BASIC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KEY_ARC</td>
<td>BASIC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KEY_ARC</td>
<td>BASIC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KEY_ARC</td>
<td>BASIC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KEY_ARC</td>
<td>BASIC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KEY_ARC</td>
<td>BASIC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KEY_ARC</td>
<td>BASIC</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 8. The Solution Data Set.

PROC TNETFLOW is an important addition to the SAS/OR software.

4. Project Management with SAS/OR

There have been several improvements made to the project management capability of the SAS/OR software. In Version 5 the ability to account for detailed modeling information has been improved in PROC CPM. The procedure can schedule a project around precedence, time, and resource constraints. Previous versions could only schedule around precedence constraints and some elementary time constraints. In addition, Version 5 PROC CPM has a powerful resource utilization summary capability.
With Version 5, SAS/OR also introduces PROC GANTT, a new procedure for producing Gantt charts on line printers and on high resolution devices. This procedure is very flexible and can be used to tailor charts to your graphic needs for project management.

PROC CPM Enhancements

To demonstrate some of the enhancements to PROC CPM we present a simple project for planning the introduction of a new product. The project encompasses seven activities. The activities and the precedence relationships among them are shown in the activity network in Figure 9.

![Activity Network](image)

Figure 9. Product Introduction Activity Network.

The precedence constraints on the activities are specified in a SAS data set along with the duration of each of the activities. This is not new in Version 5. Figure 10 shows a printout of the data set.

<table>
<thead>
<tr>
<th>NODE</th>
<th>SUC_NODE</th>
<th>DURATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRODUCT CONCEPT</td>
<td>PRODUCT DESIGN</td>
<td>25</td>
</tr>
<tr>
<td>PRODUCT CONCEPT</td>
<td>MARKET RESEARCH PLAN</td>
<td>25</td>
</tr>
<tr>
<td>PRODUCT DESIGN</td>
<td>PROTOTYPE DEVELOPMENT</td>
<td>70</td>
</tr>
<tr>
<td>PRODUCT DESIGN</td>
<td>MARKET SURVEY</td>
<td>70</td>
</tr>
<tr>
<td>MARKET RESEARCH PLAN</td>
<td>MARKET SURVEY</td>
<td>40</td>
</tr>
<tr>
<td>MARKET SURVEY</td>
<td>SALES/PRODUCTION PLAN</td>
<td>45</td>
</tr>
<tr>
<td>PROTOTYPE DEVELOPMENT</td>
<td>COST ESTIMATES</td>
<td>35</td>
</tr>
<tr>
<td>COST ESTIMATES</td>
<td>SALES/PRODUCTION PLAN</td>
<td>20</td>
</tr>
<tr>
<td>SALES/PRODUCTION PLAN</td>
<td></td>
<td>50</td>
</tr>
</tbody>
</table>

Figure 10. The Precedence Constraints.

The project is scheduled without violating the precedence constraints by invoking PROC CPM. For example, suppose that you want to schedule the project to start on March 15, 1985. Then you call PROC CPM as follows:

```sas
PROC CPM DATE='15MAR85'D OUT=SCH;
ACTIVITY NODE;
SUCCESSOR SUC_NODE;
DURATION DURATION;
```

The procedure finds the early and late start schedules along with the total and free float, and saves this information in the SAS data set named SCH. These data can be printed or displayed in any number of ways.

Time Constrained Scheduling

Often, in addition to the precedence relationships among activities in a project and the time constraint on the project as a whole, there are time constraints on the individual activities in the project. PROC CPM can be used to schedule subject to these time constraints. Each activity can be required to start or finish 'on', 'on or before', or 'on or after' a given date. The data are given to PROC CPM in the input data set.

For example, suppose that PRODUCT DESIGN is to finish on or before April 15, 1985, MARKET RESEARCH PLAN is to start on or after May 1, 1985, and PROTOTYPE DEVELOPMENT is to start on April 20, 1985.

Figure 11 shows how these time constraints would be specified. The two new variables CON_TYPE and DATE would be added to the input data set. These contain the time constraint information.

<table>
<thead>
<tr>
<th>NODE</th>
<th>CON_TYPE</th>
<th>DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRODUCT CONCEPT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRODUCT DESIGN</td>
<td>FLE</td>
<td>15APR85</td>
</tr>
<tr>
<td>MARKET RESEARCH PLAN</td>
<td>SGE</td>
<td>01MAY85</td>
</tr>
<tr>
<td>MARKET SURVEY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROTOTYPE DEVELOPMENT</td>
<td>SEQ</td>
<td>20APR85</td>
</tr>
<tr>
<td>COST ESTIMATES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SALES/PRODUCTION PLAN</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 11. Time Constraint Data.

To schedule the project invoke PROC CPM as:

```sas
PROC CPM DATE='15MAR85'D;
ACTIVITY NODE;
SUCCESSOR SUC_NODE;
DURATION DURATION;
ALIGNTYPE CON_TYPE;
ALIGNDATE DATE;
```

The ALIGNTYPE and ALIGNDATE statements identify the names of these new variables and tell the procedure to account for the time constraints when calculating the schedule. As in the current version of PROC CPM, the procedure saves the schedule in an output data set.

In addition to time constraints on the activities in a project and an overall project time constraint, the CPM procedure has the ability to calculate a schedule around variable length holidays. This feature is not only useful for scheduling around vacations and national and religious holidays, but can be used to define non-standard work weeks (for example, five and a half day work week).

Reporting Resource Utilization

A new feature in PROC CPM is the ability to report resource usage when following a calculated schedule. The resource requirements of each activity in the project are included in the input data set. For example, suppose that three types

```sas
PROC CPM DATE='15MAR85'D;
ACTIVITY NODE;
SUCCESSOR SUC_NODE;
DURATION DURATION;
```
of personnel are used when introducing a new product: marketing personnel, art personnel, and research and development personnel. For each activity the daily requirement of each resource is specified. Figure 12 shows what the input data set might look like; for example, to perform the PRODUCT DESIGN activity one person from marketing is needed along with one person from the art department and five people from research and development.

<table>
<thead>
<tr>
<th>PRODUCT CONCEPT</th>
<th>PRODUCT DESIGN 25</th>
<th>3</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRODUCT CONCEPT</td>
<td>MARKET RESEARCH PLAN 25</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>PRODUCT DESIGN</td>
<td>PROTOTYPE DEVELOPMENT 70</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>PRODUCT DESIGN</td>
<td>MARKET SURVEY 70</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>MARKET RESEARCH PLAN</td>
<td>MARKET SURVEY 40</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>MARKET SURVEY</td>
<td>SALES/PRODUCTION PLAN 45</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>PROTOTYPE DEVELOPMENT</td>
<td>COST ESTIMATES 35</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>COST ESTIMATES</td>
<td>SALES/PRODUCTION PLAN 20</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>SALES/PRODUCTION PLAN</td>
<td>50</td>
<td>7</td>
<td>7</td>
</tr>
</tbody>
</table>

Figure 12. Resource Requirements in Input Data Set.

Notice that this data set no longer contains the time constraints on the several activities as discussed in the last section. These constraints could have been included but were not in order to simplify the example.

To schedule the project we invoke PROC CPM as before. However, in order to tell PROC CPM that we want to summarize the resource utilization resulting from the schedule we include the RESOURCEOUT= option on the PROC CPM statement and the RESOURCE statement. The RESOURCEOUT= option names the output data set that is to contain the resource summary and the RESOURCE statement names the variables in the input data set that are to be interpreted as resources.

```sas
PROC CPM DATE='1SMARBS' RESOURCEOUT=USAGE; ACTIVITY NODE; SUCCESSOR SUB_NODE; DURATION DURATION; RESOURCE MKT_PER ART_PER R&D_PER;
```

The USAGE data set saved by PROC CPM gives resource utilization for the early and late start schedules. Similarly, the EART_PER contains the daily use of ART_PER, art personnel, for the E, early start schedule. The _TIME_ variable labels each reporting interval in the reporting horizon. The user gains control over the size of the reporting interval and length of the reporting horizon through the use of several options.

```sas
_DATA_ SET USAGE;
IF _TIME_='07MAY85'D & _TIME_<='25MAY85'D THEN DO;
  DEPT='MARKET'; PERSON=EMKT_PER; OUTPUT;
  DEPT='ART'; PERSON=EART_PER; OUTPUT;
  DEPT='R&D'; PERSON=ERD_PER; OUTPUT;
END;
PROC CHART;
HBAR _TIME_ / DISCRETE SUMVAR=PERSON SUBGROUP=DEPT;
```

Figure 14 shows the chart produced by this program.
The RESOURCEOUT= data set USAGE can also be displayed on a pie chart. Figure 15 shows a pie chart that summarizes the total man hours devoted to the project by personnel from each of the departments. The chart is produced with the call to PROC GCHART.

Resource Constrained Scheduling

Scheduling subject to constraints on resource availability is also a new feature in Version 5 PROC CPM. Suppose that we learn from examining the RESOURCEOUT= data set that both the early and late start schedules require resources that are not available. We want to modify the schedule so that resource availabilities are not exceeded. To accomplish this we first specify the resource availabilities in a resource profile data set. For example, suppose that on March 1, 1985 there are 10 marketing personnel, 2 art personnel, and 5 research and development personnel available for work on the project. These levels of personnel continue to be available until April 1, 1985 when there are a total of 7 research and development personnel available. This information is specified in a data set as shown in Figure 16.
DATA AVAIL;
FORMAT DATE DATE7.;
INFORMAT DATE DATE7.;
INPUT DATE MKT_PER ART_PER RD_PER;
CARDS;
01MAR85 10 2 5
01APR85 7

Figure 16. The Resource Availability Data Set.

To schedule subject to the resource constraints we invoke PROC CPM naming the resource availabilities data set in the RESOURCEIN= option on the PROC CPM statement. For example,

PROC CPM DATA=PROJ RESOURCEIN=AVAIL;
   DATE='15MAR85'D DUT=SCH2;
   ACTIVITY NODE;
   SUCCESSOR SUC_NODE;
   DURATION DURATION;
   RESOURCE MKT_PER ART_PER RD_PER / PERIOD=DATE;

The procedure calculates the resource constrained schedule saving it in the data set named SCH2. This data set no longer contains the early and late start schedules by default. They can be obtained with options on the RESOURCE statement. The data set does contains the start and finish dates for the resource constrained schedule. They are given in the S_START and S_FINISH variables as shown in Figure 17.

SCHEDULING PRODUCT INTRODUCTION
RESOURCE CONSTRAINED SCHEDULE

<table>
<thead>
<tr>
<th>NODE</th>
<th>S_START</th>
<th>S_FINISH</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRODUCT CONCEPT</td>
<td>15MAR85</td>
<td>08APR85</td>
</tr>
<tr>
<td>PRODUCT DESIGN</td>
<td>09APR85</td>
<td>17JUN85</td>
</tr>
<tr>
<td>MARKET RESEARCH PLAN</td>
<td>09APR85</td>
<td>18MAY85</td>
</tr>
<tr>
<td>MARKET SURVEY</td>
<td>16JUN85</td>
<td>01JUL85</td>
</tr>
<tr>
<td>PROTOTYPE DEVELOPMENT</td>
<td>16JUN85</td>
<td>22JUL85</td>
</tr>
<tr>
<td>COST ESTIMATES</td>
<td>23JUL85</td>
<td>11AUG85</td>
</tr>
<tr>
<td>SALES/PRODUCTION PLAN</td>
<td>12AUG85</td>
<td>30SEP85</td>
</tr>
</tbody>
</table>

Figure 17. The Resource Constrained Schedule.

5. PROC GANTT: A New Procedure

PROC GANTT is a new procedure that provides for flexible charting on printers and on high resolution graphics devices. The user can control such features as chart size, labeling, symbols used, font and color.

The procedure is designed to process an output data set from PROC CPM which contains the schedule. There is no need for user intervention between calls to PROC CPM to schedule and PROC GANTT to report the schedule.

PROC GANTT is invoked as

PROC GANTT DATA=_GRAPHICS ID variable; BY variables; CHART variables / options;

A Gantt chart is produced for each CHART statement. If the GRAPHICS option is specified and the installation supports SAS/GRAPH, the chart is drawn on a graphics device. Otherwise, the chart is printed on the line printer. For example, the early and late start schedules for the product introduction project is displayed on a Gantt chart by invoking PROC GANTT as

PROC GANTT; ID NODE; CHART / NOJOBNUM;

The chart shown in Figure 18 displays the schedules and clearly indicates the critical path (those activities with zero total float).

Figure 18. A Gantt Chart Showing Product Introduction Schedule.
Figure 19 shows the resulting chart.

If you include the GRAPHICS option on the PROC GANTT statement the chart will be drawn on a high resolution graphics device. As with the line printer versions you have significant control of the chart appearance. Figure 21 shows one example in black and white.

Figure 19. A Gantt Chart Showing Milestones.

Elaborate modifications can be made to the chart using the options on the CHART statement. Here we show just a few of the possibilities.

PROC GANTT;
   NODE;
   CHART / NOJOBNUM SYMCHAR='ELLE;h'; JOINCHAR=' _';
   REF='O1APR85' 'O1MAY85' 'O1JUN85' 'O1JUL85' 'O1AUG85' 'O1SEP85' CRITFLAG;

Figure 20 contains the resulting chart.

Figure 20. A Modified Gantt Chart.
6. Summary

The SAS/OR software has experienced several significant enhancements in the Version 5 release. The project management capabilities have enlarged through the addition of time and resource constraint scheduling in PROC CPM. Reporting resource utilization has become easier with the addition of a resource output data set in PROC CPM. And PROC GANTT has added the ability to provide flexible, clear reporting of the schedules calculated by PROC CPM.

The mathematical programming capabilities of SAS/OR have also undergone a significant improvement. Now, PROC LP can solve integer and mixed integer programs in an interactive environment which is conducive to model exploration. Parametric programming and a sparse data format also provide tools that aid in model development and exploration. PROC TNETFLOW has given the means of efficiently solving network flow models with side constraints. Because of its efficiency, it becomes attractive to identify embedded networks in large linear programming models and then reformulate the model as a network flow model with side constraints for solution with PROC TNETFLOW.

SAS, SAS/GRAPH are registered trademarks of SAS Institute Inc., Cary, NC, USA.

SAS/OR is a trademark of SAS Institute Inc.