AN INTERACTIVE INTERFACE FOR SCHEDULING YOUR WORK FORCE

Michael J. Cybrynski
SAS Institute Inc.

INTRODUCTION

The technical support department at SAS Institute Inc. faces the problem of scheduling its consultants (currently 29!) every week. The department has outlined very specific criteria for staffing the phones: criteria dictated by the needs of our users and the needs of, and the considerations given to, our employees. These requirements must be fulfilled weekly.

Whenever you manage people, however, you must account for any holidays, vacations, meetings, appointments, and other conflicts that may or may not be anticipated. Rather than leaving the conflicts to be resolved by those involved, we have always tried to account for these "exceptions." Less "switching" tends to minimize confusion and the number of "missed" hours. However, this can lead to a very large combinatorial problem.

Until recently, this problem was always begrudgingly solved by hand, requiring two people for almost an entire day. Since the introduction of SAS/OR® software, this problem is solved using the SAS System. The SAS System is "black boxed" as an interactive tool that can be used by almost anyone. Only a minimal amount of data (primarily the "exceptions") need to be entered to solve the problem and generate the necessary reports.

GOALS

The following requirements the department must meet if it is to provide its current level of support are a result of the demands for our services as well as considerations given to our consultants.

§ assign 4 consultants during 12:00 to answer initial contact calls,
§ assign 5 consultants for each remaining hour,
§ limit the number of consecutive hours assigned to 2, and
§ reserve at least 12:00 or 1:00 for a lunch break.

To allow for organizational meetings we have set aside:

§ Monday at 9:00 for second level consultants, and
§ Monday at 11:00 for first level consultants.

If this program is to be beneficial and practical, it should also have the capability of:

§ preparing a schedule in advance,
§ differentiating between personnel,
§ assigning consultants randomly and uniformly,
§ accounting for weekly variations in:
  * responsibilities,
  * preferences,
  * conflicts,
  * holidays, and
  * vacations,
§ being understandable and usable by those unfamiliar with the SAS System and/or linear programming, and
§ being easily adaptable to changes in:
  * the length of a work day,
  * the number of days in a work week, and
  * the number of consultants needed during any particular hour.

THE INTERACTIVE INTERFACE

The interactive session is used to enter information that is variant from week to week (meetings, appointments, vacations, conflicts, etc). All other information is processed in batch so that there is only minimal interaction between the user and the actual scheduling process.

The sequence of events that makes up the interactive interface to the SAS System are as follows:

§ Enter a number from the list of dates.

The scheduler is prompted for this information with the following screen (figure 1).

Is this a schedule for:

1. December 31, 1984
2. January 7, 1985
3. January 14, 1985

--------

SAS® and SAS/FS® are registered trademarks of SAS Institute Inc., Cary, NC, USA.
SAS/OR® is a trademark for SAS Institute.
These dates represent the time period for which you are scheduling.

§ Enter the total, weekly, phone duty hours for each consultant in the space provided.

PROC FSEDIT in SAS/FSP software is used to add the information entered here into a SAS data set. This greatly enhances one's ability to enter sparse data from large problems in a form that is expedient and easily understood. In this instance we have been able to "disguise" the fact that the user is indeed generating LP constraints!

The "1" that appears next to the consultant's name is an indicator variable. See figures 2 and 3. When a "1" appears next to a consultant's name, the number you enter is the total phone duty responsibilities for that consultant. By scrolling through the SAS data set you can enter the weekly totals for all the consultants in the list. For example: figure 2 represents Margaret's hours for the week.

1. Margaret 1
2. Lynn
3. Mike C.
4. Steve
5. Gloria
6. Toby
7. Wayne
8. Jim
9. Joy
10. Patti

Total phone hours this week: 0

while, figure 3 represents Lynn's hours for the week.

1. Margaret
2. Lynn
3. Mike C.
4. Steve
5. Gloria
6. Toby
7. Wayne
8. Jim
9. Joy
10. Patti

Total phone hours this week: 2

A necessary (but not sufficient) condition for obtaining a solution to this problem is that you must have enough man-hours available to satisfy the demand during the week. If this condition is not met, you will NEVER solve this problem. Therefore, an error message is issued (alerting the user to this infeasibility condition) in an effort to save valuable time, money, and hardship. (See the figure below).

!!! ERROR !!!

You have assigned 194 man-hours to fill 195 time slots

AND

only 4 second level consultants to fill 5 time slots during 11:00 Mon.

You have the option of correcting the problem immediately (if you feel the error is merely a typographical mistake) by returning to the previous screen, or ending the session to recalculate your totals. All the information entered to this point is saved, so only modifications would be required when returning to the previous screen or returning from a previous session.

§ Enter any preferences, vacations, conflicts or any other times that need to be blocked or assigned.

PROC FSEDIT in SAS/FSP software is used again to add the information entered here into a SAS data set.

Each consultant has a time*day window where information can be entered. See figures 5 and 6. Indicate the time slot(s) of interest by placing a "1" in that position on the screen. To block the hours "indicated," enter a zero (0) in the lower-most variable field. See the figure below.

| Margaret |
|-----------------|-----------------|-----------------|-----------------|-----------------|
| Mon     | Tue   | Wed    | Thu    | Fri    |
| 9:00    | 1     |       |       |       |
| 10:00   |       |       |       |       |
| 11:00   | 1     |       |       |       |
| 12:00   | 1     |       |       |       |
| 1:00    | 1     |       |       |       |
| 2:00    | 1     |       |       |       |
| 3:00    | 1     |       |       |       |
| 4:00    |       |       |       |       |

Enter 0 to BLOCK, sum to ASSIGN

127
To assign the hours "indicated", enter the total number of "1"'s entered in the lower most variable field. See the figure below.

<table>
<thead>
<tr>
<th></th>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Thu</th>
<th>Fri</th>
</tr>
</thead>
<tbody>
<tr>
<td>9:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12:00</td>
<td>1</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4:00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

To enter 0 to BLOCK, sum to ASSIGN.

DO NOT mix "assignments" and "blocks" within the same observation.

$ Enter whether or not you wish to submit the schedule.

All the data entered to this point is saved. If you wish to wait and submit the schedule later, or add to the above information, you have that option.

**PROBLEM FORMULATION**

Figure 7 diagrams a subset of our scheduling problem.

**SUPPLY**

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>0</th>
<th>15</th>
<th>15</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lynn</td>
<td>Margaret</td>
<td></td>
<td>Jennifer</td>
<td>Lori</td>
<td>Sarah</td>
</tr>
</tbody>
</table>

**DEMAND**

<table>
<thead>
<tr>
<th></th>
<th>9:00 Monday</th>
<th>10:00 Monday</th>
<th>11:00 Monday</th>
<th>12:00 Monday</th>
<th>13:00 Monday</th>
<th>2:00 Monday</th>
<th>3:00 Monday</th>
<th>4:00 Monday</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

The objective is to "transport" our "supply" of man-hours across the "arcs" to "demand points" (or time slots), while minimizing costs.

Although the problem has a transportation formulation, we are unable to use PROC TRANS to solve this problem because of side constraints. Our side constraints are:

$ reserve at least 12:00 or 1:00 for lunch and
$ limit the maximum number of consecutive hours assigned to 2.

Transportation constraints define either supplies, demands or the conservation of flow only. However, the constraints listed above impose limitations on the "flow" across "neighboring" arcs.

In addition, we will restrict the optimal solution to only integer values from the set $ x \in [0,1]$. Therefore the problem will have to be formulated and solved as an integer linear program. If the optimal value is 1, the consultant is assigned to the time slot, otherwise the consultant is not assigned to the timeslot (implying a value of 0 in the optimal solution).

**NOTE:** If your problem does not have side constraints, it would be more economical to use PROC TRANS for solving your problem. PROC TRANS uses network algorithms (not the simplex algorithm) to arrive at a solution and this is generally quicker. Another advantage to using PROC TRANS, when possible, is that with integer supplies, demands, and capacities, the solution will always be integer.

**THE BATCH SESSION**

The remaining sections of this paper describe the batch portion of our program.

We have realized a great reduction in the amount effort needed to generate a schedule using the SAS System because:

$ Constraints that remain fixed from week to week are generated in batch.
$ Only the variant constraints are entered interactively.
$ The LP is solved in batch

The information entered during the interactive session is brought in as needed.

All of this drastically reduces the amount of time our secretary must spend preparing a schedule.

To formulate an LP in a SAS data set, label the "arcs" according to the following naming convention:

$$ X_{\text{employee}, \text{day}, \text{hour}} $$

where:

$ \text{employee}=1,2,3, \ldots ,29 $
$ \text{day}=1,2,3,4,5 $
$ \text{hour}=1,2,3, \ldots ,8 $

"Arrows," by convention, will represent the variables in the SAS data set. Figures 8 and 9 show the arcs labeled using this convention.
Observations in the SAS data set will represent constraints (described below).

Figure 10 shows the program data vector, in contracted form, for the input SAS data set to PROC LP.

PROGRAM DATA VECTOR

These variables can be generated very easily using the macro facility. As an example, the following macro is called within PROC MATRIX to define column name matrices for the output data sets. Since the size of the complete problem can very easily exceed the limitations of PROC MATRIX it is necessary, as well as helpful when the time comes to decipher the program, to chop the problem up. Standardizing the variable names allows one to program constraint blocks individually, and yet assures you that the constraints will be invariant to vertical concatenation of the data sets.

In fact, this was the primary purpose for selecting the naming convention described above (the second reason being that the optimal solution reported from PROC LP contains this information in a form that is very easy to subset).

The next order of business is to define our scheduling requirements in summation notation. This form will allow one to easily express these requirements as LP constraints.

§ Weekly total constraints.

\[ \sum_{i=1}^{n} \sum_{j=1}^{29} x_{ijk} = n \]

\[ n \geq 0 \]

\[ i \in \{1, 2, \ldots, 29\} \]

Figure 12 diagrams a partial listing of the variables as they are defined on the PROC FSEDIT screen used to enter this constraint.

1. Margaret X11H 11. Mike K. X11H
2. Lynn X12H 12. Roger X12H
5. Gloria X15H 15. Donna X15H
6. Toby X16H 16. Stacey X16H
7. Wayne X17H 17. Martin X17H
9. Jay X19H

Total phone hours this week: X22H

§ Blocking constraints.

\[ \sum_{i=1}^{n} \sum_{j=1}^{29} a_{ijk} x_{ijk} = 0 \]

\[ a < (0, 1) \]

\[ i \in \{1, 2, \ldots, 29\} \]
Figure 14 diagrams a partial listing of
the variables as they are defined on the
PROC FSEDIT screen used to enter this
constraint.

<table>
<thead>
<tr>
<th></th>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Thu</th>
<th>Fri</th>
</tr>
</thead>
<tbody>
<tr>
<td>9:00</td>
<td>xij1</td>
<td>xij12</td>
<td>xij13</td>
<td>xij42</td>
<td>xij51</td>
</tr>
<tr>
<td>10:00</td>
<td>xij12</td>
<td>xij122</td>
<td>xij132</td>
<td>xij422</td>
<td>xij52</td>
</tr>
<tr>
<td>11:00</td>
<td>xij13</td>
<td>xij133</td>
<td>xij134</td>
<td>xij434</td>
<td>xij53</td>
</tr>
<tr>
<td>12:00</td>
<td>xij14</td>
<td>xij124</td>
<td>xij135</td>
<td>xij445</td>
<td>xij54</td>
</tr>
<tr>
<td>1:00</td>
<td>xij15</td>
<td>xij125</td>
<td>xij136</td>
<td>xij456</td>
<td>xij55</td>
</tr>
<tr>
<td>2:00</td>
<td>xij16</td>
<td>xij126</td>
<td>xij137</td>
<td>xij457</td>
<td>xij56</td>
</tr>
<tr>
<td>3:00</td>
<td>xij17</td>
<td>xij127</td>
<td>xij138</td>
<td>xij458</td>
<td>xij57</td>
</tr>
<tr>
<td>4:00</td>
<td>xij18</td>
<td>xij128</td>
<td>xij139</td>
<td>xij459</td>
<td>xij58</td>
</tr>
</tbody>
</table>

RHS. Enter 0 to BLOCK, sum to ASSIGN

$4$ consultants / lunch hour /day.
$5$ consultants / non-lunch hour /day.

$$\sum_{i=1}^{29} x_{ijk} = \begin{cases} \frac{4}{5} & \text{when } j \in (1,2,3,4,5) \\ 5 & \text{when } k = 4 \end{cases}$$

§ Perform your duties one hour at a time.

$$X_{ijk} \leq 1$$

$X_{ij4} + X_{ij5} \leq 1$

$X_{ij4} \leq 1$

$X_{ij5} \leq 1$

$X_{ij4} + X_{ij5} \leq 1$

§ Reserve at least 12:00 or 1:00 for Lunch.

$X_{ij4} + X_{ij5} \leq 1$

$X_{ij4} \leq 1$

$X_{ij5} \leq 1$

$X_{ij4} + X_{ij5} \leq 1$

§ Reserve 9:00 Monday for a second level meeting.

$$\sum_{i=1}^{29} x_{ili} = 0$$

§ Reserve 11:00 Monday for a first level meeting.

$\sum_{i=1}^{29} x_{ili} = 0$

Do not schedule any consultant more than 2
hours in a row.

$X_{ijkl} + X_{ijkl+1} + X_{ijkl+2} \leq 2$

$i \in (1,2,\ldots,29)$

$j \in (1,2,3,4,5)$

$k \in (1,2,\ldots,6)$

Do not assign second level at 12:00 or
after 2:00.

$$\sum_{i=1}^{29} \sum_{j=1}^{4} (X_{ij4} + X_{ij7} + X_{ijn}) = 0$$

Uniformly distribute each consultant's
hours over the days they are in attendance.

$$\sum_{i=1}^{29} \sum_{j=1}^{5} x_{ijk} \leq u$$

$u \geq 0$

$i \in (1,2,\ldots,29)$

$j \in (1,2,3,4,5)$

To determine the days in attendance for each
employee, the program first checks to see if
there are any holidays during the present week
(one use for the date entered in the beginning;
the other is for report headings). Next, the
program checks to see if anyone has any vaca­
tions that week. The number of days in atten­
dance, is equal to the initial number of work
days minus the number of holidays (if any) minus
the number of vacation days (if any). To uni­
formly distribute each consultant's hours over
their days in attendance, the program calculates
as an upper limit, the smallest integer greater
than the total number of hours / number of days
in attendance (for days in attendance only; zero
otherwise). The program calculates, as a lower
limit, the largest integer less than the total
number of hours / number of days in attendance
(for days in attendance only; zero otherwise).
A means of transforming these constraints into observations in a SAS data set is through PROC MATRIX (although the data step is an alternative). This can be accomplished by defining a zero matrix of the appropriate size and then using do-loops to replace selectively specific cells with the appropriate non-zero values (as dictated by the above expressions). For example, to generate the constraint that blocks second level consultants for their 9:00 meeting on Monday:

```
PROC MATRIX;
  X=J.(1,29*8,.);
  DO I=1 TO 18;
    J=8*(I-1);
    X(1,1+J)=1;
  END;
RUN;
```

A temporary data set for each constraint block can then be generated (this is recommended because the problem size can very easily exceed the limitations of PROC MATRIX). Ultimately, these data sets are vertically concatenated to form one input data set to PROC LP.

**THE LINEAR PROGRAM**

The front end to PROC LP consists of the interactive session and the batch session. The interactive session utilizes PROC FSEDIT and the SAS DATA step as matrix-generating tools for the problem. In addition, the batch session introduced PROC MATRIX as another matrix-generating tool. In fact, PROC MATRIX allowed us to generate 95% of our constraints in batch. This represents a considerable savings in the amount of effort the user must supply to generate a schedule. All this is very indicative of the power and flexibility one has when using the SAS System to solve linear programs.

To solve the problem, all that remains is to call PROC LP:

```
PROC LP DATA=DSNAME;
  PRIMALOUT=POUT;
  X211 0
  X212 0
  X213 0
  X214 1
  X215 0
  X216 0
  X217 0
  X218 0
RUN;
```

### SUBSETTING THE SOLUTION

Once PROC LP has processed the data, it remains to be determined whether or not the problem is feasible (you may have expected too much). If the problem is infeasible, the Infeasible Information Summary will indicate which constraints are in conflict. At this point the infeasibilities must be corrected and the program rerun. If the problem is feasible optimal, then subset the PRIMALOUT data set so it contains only those observations where _VALUE_ = 1 (remember that a value of 2 indicates an assignment). It is necessary to keep only the _VAR_ variable at this point. All other output from PROC LP is superfluous to this problem. In essence we told PROC LP what we needed and it performed all the dirty work. Our scheduling headache has been reduced to 10 minutes of data entry!

To transform the solution so that it is meaningful, substring the indices from the values of _VAR_ according to the guidelines used when these names were created,

**PRIMALOUT DATA SET**

```
- VAR -    - VALUE -
  X211      0
  X212      0
  X213      0
  X214      1
  X215      0
  X216      0
  X217      0
  X218      0
```

and format these values using PROC FORMAT:

```
PROC FORMAT:
  VALUE _EMP_ 1 = MARGARET
              2 = LYNN
              3 = SARAH;
  VALUE _IL_   1 = MONDAY
              2 = TUESDAY
              3 = WEDNESDAY
              4 = THURSDAY
              5 = FRIDAY;
  VALUE _T_    1 = 9:00
              2 = 10:00
              3 = 11:00
              4 = 12:00
              5 = 13:00
              6 = 14:00;
```

Remember only the observations where _VALUE_ = 1 represent assignments. All other observations are discarded.
You will notice that the values of _VAR_ are the "arc" labels (but more correctly, the variable names from the input data set). The reasoning behind our labeling convention has now come full-circle:

From this data set, it is now possible to generate our scheduling reports. We generate two types of reports. Both reports, however, have the scheduling week printed at the top, and the time generated at the bottom. This allows for easy identification should there be any superseding reports generated for that week.

The figure below is our TIME*DAY report.

**Telephone Schedule**

**For the Week of December 31, 1984**

<table>
<thead>
<tr>
<th>MONDAY</th>
<th>TUESDAY</th>
<th>WEDNESDAY</th>
</tr>
</thead>
<tbody>
<tr>
<td>9:00</td>
<td>Gwen</td>
<td>Margaret</td>
</tr>
<tr>
<td></td>
<td>Tom</td>
<td>Heidi</td>
</tr>
<tr>
<td></td>
<td>Annette</td>
<td>Tom</td>
</tr>
<tr>
<td></td>
<td>Fritz</td>
<td>Annette</td>
</tr>
<tr>
<td></td>
<td>Sarah</td>
<td>Vicki</td>
</tr>
<tr>
<td>10:00</td>
<td>Gwen</td>
<td>Lori</td>
</tr>
<tr>
<td></td>
<td>Eddie</td>
<td>Heidi</td>
</tr>
<tr>
<td></td>
<td>Annette</td>
<td>Gwen</td>
</tr>
<tr>
<td></td>
<td>Vicki</td>
<td>Heidi</td>
</tr>
<tr>
<td></td>
<td>Fritz</td>
<td>Annette</td>
</tr>
</tbody>
</table>

The figure below is our EMPLOYEE*DAY report.

**Telephone Schedule**

**For the Week of December 31, 1984**

<table>
<thead>
<tr>
<th>MONDAY</th>
<th>TUESDAY</th>
<th>WEDNESDAY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Margaret</td>
<td>9:00</td>
<td></td>
</tr>
<tr>
<td>Lynn</td>
<td>9:00</td>
<td>10:00</td>
</tr>
<tr>
<td></td>
<td>11:00</td>
<td>12:00</td>
</tr>
<tr>
<td></td>
<td>1:00</td>
<td>2:00</td>
</tr>
<tr>
<td>Sarah</td>
<td>10:00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>11:00</td>
<td>12:00</td>
</tr>
<tr>
<td></td>
<td>1:00</td>
<td>2:00</td>
</tr>
</tbody>
</table>

**Summary**

The ability to use SAS/OR software to solve our scheduling reduced the problem from a task that consumed an entire day for our secretary and her scheduling coordinator, to a 10-minute interactive session and a batch job. Since the entire program is totally "black boxed," the users may not even realize that they are building linear programming constraints, solving an LP, or using the SAS System. In essence, we build a data set that contains all the information about our schedule for next week, and with a minimum of effort, we generate a schedule (no matter how complex) that is always, completely accurate.

This problem also demonstrates an area in which PROC LP can be used that may have been overlooked because linear programming is "too mathematical," "too theoretical," or "too hard to interpret". On the contrary, this type of problem is practical and very useful, it is very logically organized and can be very easily understood by experienced programmers, and only a very small segment of the output needs to be regarded. If these boundaries can be overcome, SAS/OR software can make your decision-making much easier.

**Acknowledgements**

I would like to thank Lynn Hanna, Kaye Dean, Marc-david Cohen, and Margaret Adair for their assistance in helping to make this project a success.

For a printed copy of this program, send your inquiry to:

SAS Institute Inc.
Technical Support
Box 8000
Cary, NC 27511