A PHOSPHATE PRODUCT MIX MODEL USING THE LP PROCEDURE

Agnes Eggleston
Texasgulf Inc.

Introduction

The Phosphate Production Model was originally conceived as an optimization model to determine the most profitable product mix in a multi-product environment. However, recent market conditions have necessitated operating below capacity and the emphasis of the model has shifted. At this time, the main purpose of the model is to assist the company accountants in projecting costs based on anticipated production scenarios. This is achieved by generating material balance tonnage information within the model and interfacing those tonnages with projected costing data. This paper will not address the costing aspects but will concentrate on generating the material balance data.

Design of Equations

1. Design Considerations

The Phosphate Production Model was designed to reflect the existing accounting system with which it was to interface. In the past, production tonnage information was calculated manually and entered into the system in card image format. The data was then processed by an existing Cobol program and a material balance report was produced. Because of the complexity of the Cobol program and the costing interface, it was decided not to rewrite the report program. It was instead decided to develop a series of equations which would relate shipping tonnages, production tonnages, inventory levels, and raw material consumption. The solution of these equations would consist of the material balance tonnages and would be formatted in such a way as to be acceptable to the current Cobol reporting program. To achieve this goal, the existing input to the report program had to be carefully considered and the model designed accordingly.

2. Origin of required variables

The existing final report was intended to remain in place, and to receive a file containing information generated by the Phosphate Production Model. The data needed is shown in Figure 1 and it can be seen that the required information is:

- Cost area - an accounting designation for the production area. For example, the first cost area is 416-Sulphuric Acid Storage.
- Product code - an accounting designation for the raw material, intermediate product, or end product. The product code 142 (93% Sulphuric) is shown for cost area 416.
- Beginning inventory - tons of material on hand at the beginning of the time period.
- Transfer In tons - material coming in from purchases or from another cost area. There can be as many as 5 areas from which the material is transferred.
- Production tons - quantity of production in that cost area.
- Transfer out tons - quantity transferred to another cost area. There can be as many as eight areas to which the product can be transferred.
- Losses - tonnage lost due to normal processing, such as evaporation, etc.
- Waste - tonnage lost due to production fluctuation, but for which an average percentage can be anticipated.
- Consumed - material consumed in the manufacture of another product.
- Ending Inventory - tonnage of product on hand at the end of the time period.
- Movement From and Movement To - These fields show cost areas between which products are moved. For example, looking at cost area 421, product code 142, we see the second Movement To is 421-142. Looking at cost area 421, product code 142, we see the Movement From is cost area 416, product code 142. Thus, it was necessary only to generate transfer out tons for each cost area, with the Cobol report program generating the associated transfer in tonnage. The actual derivation of the tonnages on Figure 1 will be addressed as the model design is discussed.

3. Relationships of Variables

During the development of the model, extensive discussions were held with production personnel at the mine site to determine typical tonnage movements between cost areas. Using cost area 421 in Figure 1 as an example, it can be shown that the following relationships exists between tonnage movements:

- Tonnage transferred out from 429-402 is equal to consumed tonnage in 421-402.
- Consumed tonnage in 421-402 is equal to 3.7 times transfer out tonnage in 421-144.
4. Column Naming Convention

Keeping in mind that the output from the phosphate production model was to replace a card image file with a predetermined design, the naming convention of the variables had to allow for a transformation of the model data into the existing format. For that reason the variables are constructed as follows:

- The first character of the variable represents the activity, i.e., P for production, C for consumed, W for waste, X for transfer out, etc.
- The second character represents a sequence number. This is necessary since there can be multiple transfers in and transfers out in a cost area.
- The next six characters represent the cost area and product code.

Thus, the activity, sequence, cost area, and product code are combined to create the decision variables which become the columns of the matrix. An example of such a variable is C1421402, which represents the consumed tonnage for cost area 421, product code 402.

5. Row Naming Convention

A naming convention for the rows was also adopted. This was desirable in order to facilitate the location of a particular row in the model. The row names, like the column names, reflect the cost area and product code with which they are associated. Thus, in equation form, cost area 421 becomes:

<table>
<thead>
<tr>
<th>Row</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>421402A</td>
<td>C1421402 - X1429402 = 0</td>
</tr>
<tr>
<td>421402B</td>
<td>C1421402 - 3.7 X1421144 = 0</td>
</tr>
<tr>
<td>421142A</td>
<td>C1421142 - X1418142 = 0</td>
</tr>
<tr>
<td>421142B</td>
<td>C1421142 - 2.85 X1421144 = 0</td>
</tr>
<tr>
<td>421144A</td>
<td>P1421144 - X1421144 - W1421144 = 0</td>
</tr>
<tr>
<td>421144B</td>
<td>W1421144 - 0.7 X1421144 = 0</td>
</tr>
</tbody>
</table>

6. The Objective Function

The objective function was designed to maximize profit. Therefore, the decision variables consisted of tons of product shipped and the coefficients consisted of the profit margin on those products. Thus, the row defining the objective function would take the following form:

_Profit = 1.10 X1615030 + 2.15 X1615032...

where 615 represents the cost area for shipping and 030 and 032 represent product codes.

Development of SAS® Data Sets

1. Input Format of Equations

At the completion of the model design, there were approximately 450 rows and 750 columns in the model. Because of the size of the model, it seemed impractical to use the equation input format of Var1...VarN, TYPE RHS. Instead, each column entry for each row became an observation to the input data set. Thus rows 421402A and 421402B were input as follows:

DATA EQUATION;
Input row $ function $ value;
cards:
  _421402A C1421402 1
  _421402A X1429402 -1
  _421402B C1421402 1
  _421402B X1421144 -3.7

The data set named EQUATION remains constant during periods for which there are no changes in the production process. It is periodically revised to reflect new products, changes in the components of existing products, and changes in the proportions between products.

2. Input to the model

Following input of all the equations, the values for the known tonnages for a particular production scenario had to be input to the model. These values were for such variables as beginning inventory tonnages, ending inventory tonnages, and tons of product shipped from the plant. The values assigned to these variables became the bounds on the model. Since the values vary from run to run, they are input into a separate data set. An example of some of the shipping tonnage entries is as follows:

DATA MONTH;
Input row $ function $ value;
cards:
  LO X1530038 100
  LO X1615045 22
  LO X1615048 333
Subsequent code also created upper bound entries on these variables.

The previous month's ending inventory tonnages were read from a disk file and assigned to the beginning inventory tonnages for the model. Ending inventory tonnages were assumed to be equal to beginning inventory tonnages unless they were overridden. By establishing beginning and ending inventory levels and shipping tonnages, the model could be transformed into a series of $N$ equations in $N$ unknowns.

3. Use of PROC TRANSPOSE

Following the generation of upper bounds to correspond to the lower bounds, the data sets named EQUATION and MONTH were concatenated and sorted. PROC TRANSPOSE was run against the output data set which resulted in a new data set having standard row and column format. Since the model consisted of a series of equalities, the right hand sides of the equations were set to zero and the keyword of 'EQ' was assigned to the variable. The keywords 'UPPERBD' and 'LOWERBD' were assigned to the bounded variables and the keyword of 'MAX' was assigned to the objective function. Following the execution of PROC TRANSPOSE and the subsequent SAS program, the data set named SASLP1B.MODINPUT contained observations containing the variables Row, Var1...VarN, TYPE, RHS.

1. Selection of PROC LP Options

In general, the default options were used in the execution of PROC LP. However, since the default on MAXIT (number of iterations) was 100, it was necessary to increase this parameter because of the size of the Phosphate Production Model. The execution of the model normally involves about 350 iterations. In addition, it was necessary to implement the PRIMALOUT option to have later access to the solution from a subsequent SAS program.

A point of interest to note is that the use of the DUALOUT option would have produced a data set which would have contained the value of each row at optimality and thus would have provided access to the value of the objective function. Since the value of the objective function was not needed for further processing in this application, the DUALOUT option was not selected.

2. Selection of PROC LP statements

In some cases, PROC LP uses the VAR statement. If this statement is used, there are two ways to enter the variables.

First, all the variables can be listed in the VAR statement. Secondly, the first and last variables can be listed separated by dashes, i.e. Var1-VarN. In a model as large as the Phosphate Production Model it was not convenient to list all the variables. Since the first and last variable can change as the model design is altered, the approach of listing the first and last variable was discarded. If the VAR statement is not used, then PROC LP uses all numeric variables not in other statement lists as structural variables. Since this was the desired result, the omission of the VAR statement caused no problems.

Since row names were assigned to the equations, the ID statement was needed. The variable named ROW was assigned to ID in order to preserve the row names in the model.

To summarize, the code used to execute PROC LP is as follows:

```
PROC LP DATA=SASLP1B.MODINPUT
MAXIT = 1000
PRIMALOUT=SASLP1B.SOLUTION;
```
2. Conversion from MPSX format to PROC LP format

About a year ago, we became an Alpha Test Site for SAS/OR. One of the features of SAS/OR is the SAS macro SASMPSX which allows the user to convert data from MPSX format to PROC LP format. This macro and PROC LP were used on two models with the following results:

<table>
<thead>
<tr>
<th>Model Size</th>
<th>CPU time for SASMPSX and PROC LP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row Col</td>
<td></td>
</tr>
<tr>
<td>4 4</td>
<td>7.02 secs</td>
</tr>
<tr>
<td>51 92</td>
<td>11.36 secs</td>
</tr>
<tr>
<td>CPU time for PROC LP only</td>
<td>CPU time</td>
</tr>
<tr>
<td>.25 secs</td>
<td>1.07 secs</td>
</tr>
<tr>
<td>1.66 secs</td>
<td>1.34 secs</td>
</tr>
</tbody>
</table>

From this data, it appears that PROC LP execution time is directly proportional to the size of the model, whereas the CPU times of MPSX solutions do not vary a great deal with problem size. It is also apparent that a considerable amount of CPU time in the SAS jobs is involved in the SASMPSX macro. Therefore, if the problem file is saved and modified within SAS, there is a considerable amount of CPU time saved.

3. Implementation of the Phosphate Production Model

When design began on the Phosphate Production Model, it was apparent that it would be necessary to computerize the front end and output portions of the model. This requirement was not only because of the size of the model, but also because we were interfacing with our existing accounting system. In view of the fact that we were familiar with SAS, the decision was made to use SAS for the entire application. Therefore, the data in the Phosphate Production Model was input to a SAS data set as demonstrated earlier in this paper.

During the development of the model, there was considerable testing and debugging. During this phase, it was noted that error messages generated by PROC LP were less descriptive than those generated by MPSX. At that point, a SAS program was written to convert the SAS data set to MPSX format and some debugging was done using MPSX. Following the debugging stage, the model was run using PROC LP because of the ease of accessing the solution data set with a SAS program for further processing. Since that time, PROC LP has been modified to contain more descriptive error messages. It is our intention in the future to use PROC LP exclusively for our linear programming applications.

To summarize, it should be emphasized that PROC LP has proven to be an excellent tool for this application. The ease of setting up the problem, the ability to use existing data sets in our system with minimal manipulation, and the ease of extracting the solution for further processing combine to greatly facilitate the implementation of the Phosphate Production Model.

References:

MPSX is a registered trademark of the International Business Machines Corporation.

For additional information on PROC LP, see SAS/OR USER'S GUIDE, 1983 Edition, SAS Institute, Inc., Cary, N. C.

Acknowledgements:

The author would like to thank Marc Cohen, Keith Daniel, Ellie Maines, and Lee West for technical suggestions and editorial assistance during preparation of this paper.

SAS is the registered trademark of SAS Institute Inc., Cary, NC, USA. SAS/OR is the trademark of SAS Institute Inc., Cary, NC, USA.
### Linear Programming Procedure

**Variable Summary**

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>STATUS</th>
<th>TYPE</th>
<th>PRICE</th>
<th>ACTIVITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>456 C1</td>
<td>BASIC</td>
<td>NON-NEG</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>451 P1</td>
<td>BASIC</td>
<td>NON-NEG</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>458 W1</td>
<td>BASIC</td>
<td>NON-NEG</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>459 C1</td>
<td>BASIC</td>
<td>NON-NEG</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

**Problem Summary**

- **Max Profit**
- **Objective Function Type**
- **Problem Structure**
- **Variable Type**
- **Constraint Type**
- **Solution Summary**

**SAS Output**

- **Objective Value**
- **Phase 1 Iterations**
- **Phase 2 Iterations**
- **Initial B.F. Variables**
- **Time Used (Secs)**
- **Number of Inversions**
- **Machine Epsilon**
- **Machine Infinity**
- **Invert Frequency**
- **Maximum Total Iterations**
- **Time Limit (Secs)**

**Figure 1**

```
<table>
<thead>
<tr>
<th>PRODUCT</th>
<th>PRODUCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>HAI UN</td>
<td>1000</td>
</tr>
<tr>
<td>HAI UN</td>
<td>1000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PRODUCT</th>
<th>PRODUCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>HAI UN</td>
<td>1000</td>
</tr>
<tr>
<td>HAI UN</td>
<td>1000</td>
</tr>
</tbody>
</table>
```

**Figure 2**

```
<table>
<thead>
<tr>
<th>PRODUCT</th>
<th>PRODUCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>HAI UN</td>
<td>1000</td>
</tr>
<tr>
<td>HAI UN</td>
<td>1000</td>
</tr>
</tbody>
</table>
```

**Figure 3**

```
<table>
<thead>
<tr>
<th>PRODUCT</th>
<th>PRODUCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>HAI UN</td>
<td>1000</td>
</tr>
<tr>
<td>HAI UN</td>
<td>1000</td>
</tr>
</tbody>
</table>
```