The SAS/OR®’s OPTMODEL Procedure: A Powerful Modeling Environment for Building, Solving, and Maintaining Mathematical Optimization Models

Maurice Djona

OASUS - Wednesday, November 19th, 2008
Agenda

- Context: Why Proc OPTMODEL?
- Defining Mathematical Optimization Problems
- Classification of Optimization Problems
- Examples of the use of Mathematical Optimization
- SAS Procedures for solving optimization problems
- Proc OPTMODEL – Introduction
- Proc OPTMODEL – Syntax
- Proc OPTMODEL – Examples
- Proc OPTMODEL vs. Proc OPTLP
- Conclusion
- How to Get Started?
Context: Why Proc OPTMODEL?

- Redevelopment of some survey processing systems that use mathematical optimization, at Statistics Canada
- Too costly to reinvent the wheel by developing our own optimization module
- Searched for an optimization software package taking into account:
  - Performance,
  - Cost,
  - Ease of use,
  - Support
  - Integration with existing systems
- SAS/OR®’s Proc OPTMODEL was selected among several possibilities
- This presentation focuses on mathematical optimization and Proc OPTMODEL
Defining Mathematical Optimization Problems

General form of optimization problems:

\[
\begin{align*}
\text{Min OR Max} & \quad f(x) \\
\text{Subject to} & \quad c_i(x) \{\leq, =, \geq\} b_i (i = 1, 2, \ldots, m) \\
& \quad l_j \leq x_j \leq u_j (j = 1, 2, \ldots, n)
\end{align*}
\]

Where:

- \( x \) is the set of decision variables \( x_1, x_2, \ldots, x_N \)
- \( f(x) \) is an objective function (ex. \( f = x_1 + 2x_2 \))
- \( c_i(x) \) are the constraints on the variables \( x \) (ex. \( x_1 + 3x_2 \leq 5 \))
- \( l_j \) and \( u_j \) are lower and upper bounds on variables \( x \) (ex. \( 2 \leq x_1 \leq 6 \))
Classification of Mathematical Optimization Problems

- **LP** (Linear Programming): \( f(x) \) and \( c_i(x) \) are linear functions, all \( x \) are real numbers

- **ILP** (Integer Linear Programming): \( f(x) \) and \( c_i(x) \) are linear functions, all \( x \) are integers

- **MILP** (Mixed-Integer Linear Programming): \( f(x) \) and \( c_i(x) \) are linear functions, at least some \( x \) must be integers, while other \( x \) may be real

- **NLP** (Non Linear Programming): \( f(x) \) and \( c_i(x) \) are continuous, but at least some of them must be non linear functions (ex. \( f = x_1 + x_2 \times x_3 \))
Examples of Use of Mathematical Optimization

Examples of typical use include:

- Facility Location
- Production Planning
- Workforce Planning
- Product Distribution
- Delivery Network Configuration
- Investment Planning
- Inventory Replenishment Planning
- Retail Pricing

SAS solutions using optimization:

- SAS Marketing Optimization
- SAS Revenue Optimization Suite
- SAS Size Optimization
- SAS Inventory Optimization,
- SAS Service Parts Optimization
- SAS Credit Risk Management

Two examples of particular use at Statistics Canada:

- **Banff system**: Edit and Imputation of survey data
- **CONFID system**: Minimization of information lost while suppressing sensitive survey data before publication
Examples of Optimization Problems

- **A LP (Linear Programming) problem:**
  \[
  \text{max } x_1 + x_2 + x_3 \\
  \text{subject to } 3x_1 + 2x_2 - x_3 \leq 1 \\
  -2x_1 - 3x_2 + 2x_3 \leq 1 \\
  x_1 \geq 0, x_2 \geq 0, x_3 \geq 0
  \]

- **A NLP (Non Linear Programming) problem:**
  \[
  \text{max } x_1 * x_2 * x_3 \\
  \text{subject to } 3x_1 + 2x_2 - x_3 \leq 1 \\
  -2x_1 - 3x_2 + 2x_3 \leq 1 \\
  x_1 \geq 0, x_2 \geq 0, x_3 \geq 0
  \]

- **LPs solved in CONFID at Statistics Canada:**
  - Number of variables: dozens to \( \approx 200,000 \)
  - Number of constraints: dozens to \( \approx 50,000 \)
  - Non-zero constraint coefficients: hundreds to \( \approx 300,000 \)
Software Packages for Solving Optimization Problems

- Commercial packages include:
  - CPLEX (ILOG)
  - XPRESS (Dash Optimization)
  - Many procedures of SAS/OR (SAS Institute)
  - etc.

- Free packages include:
  - GLPK (GNU Linear Programming Kit)
  - lp_solve
  - etc.
# SAS Procedures for Solving Optimization Problems

## Chapter 1

### Introduction to Optimization

### Chapter Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>OVERVIEW</td>
<td>3</td>
</tr>
<tr>
<td>LINEAR PROGRAMMING PROBLEMS</td>
<td>4</td>
</tr>
<tr>
<td>PROC OPTLP</td>
<td>4</td>
</tr>
<tr>
<td>PROC OPTMODEL</td>
<td>5</td>
</tr>
<tr>
<td>PROC LP</td>
<td>5</td>
</tr>
<tr>
<td>PROC INTPOINT</td>
<td>5</td>
</tr>
<tr>
<td>NETWORK PROBLEMS</td>
<td>6</td>
</tr>
<tr>
<td>PROC NETFLOW</td>
<td>6</td>
</tr>
<tr>
<td>PROC INTPOINT</td>
<td>7</td>
</tr>
<tr>
<td>MIXED INTEGER LINEAR PROBLEMS</td>
<td>7</td>
</tr>
<tr>
<td>PROC OPTMILP</td>
<td>7</td>
</tr>
<tr>
<td>PROC OPTMODEL</td>
<td>7</td>
</tr>
<tr>
<td>PROC LP</td>
<td>7</td>
</tr>
<tr>
<td>QUADRATIC PROGRAMMING PROBLEMS</td>
<td>8</td>
</tr>
<tr>
<td>PROC OPTQP</td>
<td>8</td>
</tr>
<tr>
<td>PROC OPTMODEL</td>
<td>8</td>
</tr>
<tr>
<td>NONLINEAR PROBLEMS</td>
<td>8</td>
</tr>
<tr>
<td>PROC OPTMODEL</td>
<td>8</td>
</tr>
<tr>
<td>PROC NLP</td>
<td>9</td>
</tr>
<tr>
<td>MODEL BUILDING</td>
<td>10</td>
</tr>
<tr>
<td>PROC LP</td>
<td>10</td>
</tr>
<tr>
<td>PROC NETFLOW</td>
<td>15</td>
</tr>
<tr>
<td>PROC OPTMODEL</td>
<td>19</td>
</tr>
</tbody>
</table>

(SAS/OR® 9.1.3 User’s Guide: Mathematical Programming 3.2)
OPTMODEL Procedure - Introduction

- Introduced with SAS/OR® version 9.1.3
- Provides a modeling environment tailored to building, solving, and maintaining optimization models
- Powerful modeling language mimics symbolic algebra
  ⇒ Building optimization models is “virtually transparent”
- Simplifies population of models by reading data from SAS data sets, and creates SAS data sets from the models
- Bottom line: with Proc OPTMODEL, models are
  - more easily built,
  - more easily inspected for completeness and correctness,
  - more easily corrected,
  - more easily modified, and
  - manipulated mainly in memory ⇒ less I/O ⇒ higher efficiency
OPTMODEL Procedure - Introduction (cont.)

OPTMODEL can be used to:

- Build and solve models
- Build models that are then solved using other SAS/OR® procedures (or non SAS solvers as CPLEX, XPRESS, GLPK, …)
- Seven solvers are available in OPTMODEL to solve models:

<table>
<thead>
<tr>
<th>Problem</th>
<th>Solver</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear Programming</td>
<td>LP</td>
</tr>
<tr>
<td>Mixed Integer Programming</td>
<td>MILP</td>
</tr>
<tr>
<td>Quadratic Programming</td>
<td>QP (experimental)</td>
</tr>
<tr>
<td>Nonlinear Programming, Unconstrained</td>
<td>NLPU</td>
</tr>
<tr>
<td>General Nonlinear Programming</td>
<td>NLPC</td>
</tr>
<tr>
<td>General Nonlinear Programming</td>
<td>SQP</td>
</tr>
<tr>
<td>General Nonlinear Programming</td>
<td>IPNLP (experimental)</td>
</tr>
</tbody>
</table>

OPTMODEL \(\approx\) more friendly interface to various optimization solvers
### OPTMODEL Procedure – Some Elements of Syntax

<table>
<thead>
<tr>
<th>Declaration Statements:</th>
<th>Programming Statements</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROC OPTMODEL options ;</td>
<td>FOR { index set } statement;</td>
</tr>
<tr>
<td></td>
<td>IF logic THEN statement ; [ ELSE statement ; ]</td>
</tr>
<tr>
<td></td>
<td>LEAVE ;</td>
</tr>
<tr>
<td></td>
<td>PRINT print items ;</td>
</tr>
<tr>
<td></td>
<td>PUT put items ;</td>
</tr>
<tr>
<td></td>
<td>CREATE DATA SAS-data-set INTO columns ;</td>
</tr>
<tr>
<td></td>
<td>READ DATA SAS-data-set INTO columns ;</td>
</tr>
<tr>
<td></td>
<td>RESET OPTIONS options ;</td>
</tr>
<tr>
<td></td>
<td>RESTORE constraint ;</td>
</tr>
<tr>
<td></td>
<td>SAVE MPS SAS-data-set ;</td>
</tr>
<tr>
<td></td>
<td>SAVE QPS SAS-data-set ;</td>
</tr>
<tr>
<td></td>
<td>SOLVE [ WITH solver ] [ OBJECTIVE name ] [ / options ;</td>
</tr>
<tr>
<td></td>
<td>STOP ;</td>
</tr>
<tr>
<td></td>
<td>UNFIX variable [ = expression ] ;</td>
</tr>
<tr>
<td></td>
<td>Plus: Powerful expressions for sets manipulation</td>
</tr>
</tbody>
</table>

#### Declaration Statements:
- CONSTRAINT constraints ;
- MAX objective ;
- MIN objective ;
- NUMBER parameter declarations ;
- STRING parameter declarations ;
- SET [ < types > ] parameter declarations ;
- VAR variable declarations ;

#### Programming Statements:
- parameter = expression ; (Assignment)
- CALL name [ ( expressions ) ] ;
- CLOSEFILE files ;
- CONTINUE ;
- CREATE DATA SAS-data-set FROM columns ;
- DO ; statements ; END ;
- DO variable = specifications; statements ; END;
- DO UNTIL ( logic ) ; statements ; END ;
- DO WHILE ( logic ) ; statements ; END ;
- DROP constraint ;
- EXPAND name [ / options ] ;
- FILE file ;
- FIX variable [ = expression ] ;

**OPTMODEL ≈ A complete programming environment, may be preferred to data step in some cases**
OPTMODEL Procedure: Example

A simple LP problem:
max \( x_1 + x_2 + x_3 \)
subject to \( 3x_1 + 2x_2 - x_3 \leq 1 \)
\( -2x_1 - 3x_2 + 2x_3 \leq 1 \)
\( x_1 \geq 0, x_2 \geq 0, x_3 \geq 0 \)

/* Linear Problem using named parameters and variables*/

PROC OPTMODEL;
   VAR x1 >= 0, x2 >= 0, x3 >= 0; /*Declare variables, & set bounds*/
   MAX f = x1 + x2 + x3; /* Objective function*/
   CONSTRAINT c1: 3*x1 + 2*x2 - x3 <= 1; /* Constraint*/
   CONSTRAINT c2: -2*x1 - 3*x2 + 2*x3 <= 1; /* Constraint*/
   SOLVE WITH LP OBJECTIVE f; /* Solve; */ /* Solve model using LP solver */
   PRINT x1 x2 x3; /* Print the solution */
QUIT;
The OPTMODEL Procedure

Problem Summary

<table>
<thead>
<tr>
<th>Objective Sense</th>
<th>Maximization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objective Function</td>
<td>f</td>
</tr>
<tr>
<td>Objective Type</td>
<td>Linear</td>
</tr>
<tr>
<td>Number of Variables</td>
<td>3</td>
</tr>
<tr>
<td>Bounded Above</td>
<td>0</td>
</tr>
<tr>
<td>Bounded Below</td>
<td>3</td>
</tr>
<tr>
<td>Bounded Below and Above</td>
<td>0</td>
</tr>
<tr>
<td>Free</td>
<td>0</td>
</tr>
<tr>
<td>Fixed</td>
<td>0</td>
</tr>
<tr>
<td>Number of Constraints</td>
<td>2</td>
</tr>
<tr>
<td>Linear LE (&lt;=)</td>
<td>2</td>
</tr>
<tr>
<td>Linear EQ (=)</td>
<td>0</td>
</tr>
<tr>
<td>Linear GE (&gt;=)</td>
<td>0</td>
</tr>
<tr>
<td>Linear Range</td>
<td>0</td>
</tr>
</tbody>
</table>

Solution Summary

<table>
<thead>
<tr>
<th>Solver</th>
<th>Dual Simplex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objective Function</td>
<td>f</td>
</tr>
<tr>
<td>Solution Status</td>
<td>Optimal</td>
</tr>
<tr>
<td>Objective Value</td>
<td>8</td>
</tr>
<tr>
<td>Iterations</td>
<td>3</td>
</tr>
<tr>
<td>Primal Infeasibility</td>
<td>0</td>
</tr>
<tr>
<td>Dual Infeasibility</td>
<td>0</td>
</tr>
<tr>
<td>Bound Infeasibility</td>
<td>0</td>
</tr>
</tbody>
</table>

\[
\begin{array}{ccc}
  x1 & x2 & x3 \\
  0 & 3 & 5 \\
\end{array}
\]
OPTMODEL Procedure: Example (cont.)
/* Non Linear Problem using named parameters and variables */

PROC OPTMODEL;
   VAR x1 >= 0, x2 >= 0, x3 >= 0; /*Declare variables, & set bounds*/
   MAX f = x1 * x2 * x3; /* Objective function: Non Linear*/
   CONSTRAINT c1: 3*x1 + 2*x2 - x3 <= 1; /* Constraint*/
   CONSTRAINT c2: -2*x1 - 3*x2 + 2*x3 <= 1; /* Constraint*/
   SOLVE WITH NLPC OBJECTIVE f; /* Solve; */ /* Solve model using NLPC solver */
   PRINT x1 x2 x3; /* Print the solution */
QUIT;

The OPTMODEL Procedure
Solution Summary

Solver               NLPC/Trust Region
Objective Function   f
Solution Status       Optimal
Objective Value       1.8951890412
Iterations           6
Absolute Optimality Error 1.2305578E-8
Relative Optimality Error 1.8366968E-9
Absolute Infeasibility 8.326673E-16
Relative Infeasibility 4.163336E-16

x1     x2     x3
0.28287 1.8685 3.5856
OPTMODEL Procedure: Example (cont.)

/* Linear problem using indexes for parameters and variables
   (Indexing \approx like arrays in the DATA STEP, but much more flexible) */

PROC OPTMODEL;
  NUMBER NbVar INIT 3; /*Number of variables*/
  VAR x{j IN 1.. NbVar} >= 0; /*Declare variables, & set bounds*/
  MAX f = SUM{j IN 1..NbVar} x[j]; /* Objective function*/
  SOLVE WITH LP OBJECTIVE f; /* Solve model using LP solver */
  PRINT x; /* Print the solution */
QUIT;

The OPTMODEL Procedure
Solution Summary

<table>
<thead>
<tr>
<th>Solver</th>
<th>Dual Simplex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objective Function</td>
<td>f</td>
</tr>
<tr>
<td>Solution Status</td>
<td>Optimal</td>
</tr>
<tr>
<td>Objective Value</td>
<td>8</td>
</tr>
<tr>
<td>Iterations</td>
<td>3</td>
</tr>
<tr>
<td>Primal Infeasibility</td>
<td>0</td>
</tr>
<tr>
<td>Dual Infeasibility</td>
<td>0</td>
</tr>
<tr>
<td>Bound Infeasibility</td>
<td>0</td>
</tr>
</tbody>
</table>

[1]  x
    1  0
    2  3
    3  5
Read model data from data sets

\[
\text{max} \quad x_1 + x_2 + x_3 \\
\text{subject to} \quad 3x_1 + 2x_2 - x_3 \leq 1 \quad \text{L}
\]

\[-2x_1 - 3x_2 + 2x_3 \leq 1 \quad \text{L}
\]

\[x_1 \geq 0, x_2 \geq 0, x_3 \geq 0\]

\[
/* \text{Objective coefficients data set} */
\]

DATA objCoefDS;
   INPUT varId objCoef; DATALINES;
   1 12
   2 1
   3 1
;  
RUN;

\[
/* \text{Constraint coefficients data set} */
\]

DATA constrDS;
   INPUT constrId varId ConstrCoef; DATALINES;
   1 1 3
   1 2 2
   1 3 -1
   2 1 -2
   2 2 -3
   2 3 2
;  
RUN;

\[
/* \text{Constraints comparison operator (L, E or H) & right hand side data set} */
\]

DATA rhsDS;
   INPUT constrId rhs comp $1.; DATALINES;
   1 1 L
   2 1 L
;  
RUN;
OPTMODEL Procedure: Example (cont.)

/* Read model data from data sets: more indexing*/

PROC OPTMODEL;
SET<NUMBER> varIndex; /*Set of indexes of variables*/
NUMBER mObjCoef{varIndex}; /*Array of objective function coefficients values*/
SET<NUMBER, NUMBER> constrCoefLoc; /*Set of indexes (tuples) of constraint coefficients*/
NUMBER mConstrCoef{constrCoefLoc}; /*Array of constraint coefficients values*/
SET<NUMBER> constrIndex; /*Set of indexes of constraints*/
NUMBER mrhs{constrIndex}; /*Array of right hand side values*/
STRING mcomp{constrIndex}; /*Array for comparison operators: L (lower), E(equal) or H (higher)*/

/* Read variables indexes and objective function coefficients from data set*/
READ DATA objCoefDS INTO varIndex = [varId] mObjCoef = objCoef;

/* Read constraint coefficients from data set*/
READ DATA constrDS INTO constrCoefLoc = [constrId varId] mConstrCoef = ConstrCoef;

/* Read rhs from data set*/
READ DATA rhsDS INTO constrIndex = [constrId] mcomp = comp mrhs = rhs;

/*Declare variables, objective function and constraints*/
VAR x{j IN varIndex} >= 0;
MAX f = SUM{j IN varIndex} mObjCoef[j]*x[j];
CONSTRAINT cl{i IN constrIndex: mcomp[i] = 'L'}: (SUM{<(i), j> IN constrCoefLoc} mConstrCoef[i, j]*x[j]) <= mrhs[i];
CONSTRAINT ce{i IN constrIndex: mcomp[i] = 'E'}: (SUM{<(i), j> IN constrCoefLoc} mConstrCoef[i, j]*x[j]) = mrhs[i];
CONSTRAINT ch{i IN constrIndex: mcomp[i] = 'H'}: (SUM{<(i), j> IN constrCoefLoc} mConstrCoef[i, j]*x[j]) >= mrhs[i];
EXPAND/SOLVE; /*Print model in algebraic form*/
SOLVE WITH LP OBJECTIVE f; /*Solve model using LP solver*/
PRINT x; /*Print solution*/
CREATE DATA results FROM [var]=varIndex value=x; /*Write solution to data set*/
QUIT;

NB: Possibility to modify and re-solve the problem in memory without leaving OPTMODEL (loops).
The OPTMODEL Procedure

Var x[1] >= 0
Var x[2] >= 0
Var x[3] >= 0

Solution Summary

Solver                  Dual Simplex
Objective Function     f
Solution Status        Optimal
Objective Value        8
Iterations              3
Primal Infeasibility   0
Dual Infeasibility     0
Bound Infeasibility    0

[1]     x
       1  0
       2  3
       3  5
OPTMODEL Procedure: Example (cont.)

Creating data sets from the model

/* Write solution to data set*/
CREATE DATA results FROM [var]=varIndex value=x;

PROC PRINT DATA=results;
   TITLE "Results data set created in Proc OPTMODEL";
RUN;

Results data set created in Proc OPTMODEL

<table>
<thead>
<tr>
<th>Obs</th>
<th>var</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>5</td>
</tr>
</tbody>
</table>
OPTMODEL vs. OPTLP: MPS Format

- Proc OPTLP requires models in SAS data set, in MPS format
- MPS format: an industry standard, but not user friendly
- Use Proc OPTMODEL to create MPS format data set required by OPTLP:

```sas
PROC OPTMODEL;
  VAR x1 >= 0, x2 >= 0, x3 >= 0;  /*Declare variables, and set bounds*/
  MAX f   = x1 + x2 +   x3 ;  /* Objective function*/
  CON c1: 3*x1 + 2*x2 -   x3 <= 1;  /* Constraint*/
  CON c2: -2*x1 - 3*x2 + 2*x3 <= 1;  /* Constraint*/
  / *Save model in MPS format to be solved outside OPTMODEL */
  SAVE MPS model_mps;
QUIT;

PROC PRINT DATA=model_mps;
  TITLE "Model in MPS format";
RUN;
```
### OPTMODEL vs. OPTLP: MPS Format

Model in MPS format data set required by Proc OPTLP

<table>
<thead>
<tr>
<th>Obs</th>
<th>FIELD1</th>
<th>FIELD2</th>
<th>FIELD3</th>
<th>FIELD4</th>
<th>FIELD5</th>
<th>FIELD6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NAME</td>
<td></td>
<td>model_mp</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>ROWS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>MAX</td>
<td>f</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>L</td>
<td>c1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>L</td>
<td>c2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>COLUMNS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>x1</td>
<td>f</td>
<td>1</td>
<td>c1</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>x1</td>
<td>c2</td>
<td>-2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>x2</td>
<td>f</td>
<td>1</td>
<td>c1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>x2</td>
<td>c2</td>
<td>-3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>x3</td>
<td>f</td>
<td>1</td>
<td>c1</td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>x3</td>
<td>c2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>RHS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>.RHS.</td>
<td>c1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>.RHS.</td>
<td>c2</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>ENDATA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**NB:** Models in MPS format (text file) can also be solved by non-SAS solvers such as CPLEX, XPRESS, GLPK, etc.
OPTMODEL vs. OPTLP: MPS Format

Solve the model using Proc OPTLP:

PROC OPTLP DATA=model_mps
   PRIMALOUT = optlp_sol; /*Solution data set*/
RUN;

PROC PRINT DATA=optlp_sol;
   TITLE "Solution from OPTLP";
RUN;

Solution data set created by Proc OPTLP (selected fields):

<table>
<thead>
<tr>
<th>Obs</th>
<th><em>OBJ_ID</em></th>
<th><em>VAR</em></th>
<th><em>TYPE</em></th>
<th><em>OBJCOEF</em></th>
<th><em>VALUE</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>f</td>
<td>x1</td>
<td>N</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>f</td>
<td>x2</td>
<td>N</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>f</td>
<td>x3</td>
<td>N</td>
<td>1</td>
<td>5</td>
</tr>
</tbody>
</table>
Conclusion

Proc OPTMODEL:

- Thanks to its powerful programming language, facilitates building and maintaining optimization models
- Models data reside in memory and can be more easily manipulated ⇒ less I/O ⇒ higher efficiency
- Has access to many SAS/OR optimization solvers
- Facilitates building models to be solved outside OPTMODEL, including by non-SAS solvers.
How to Get Started?

- **By yourself:**
  SAS/OR® 9.1.3 User’s Guide:
  Mathematical Programming 3.2
  Chapter 6: The OPTMODEL Procedure

- **SAS Course:**
  “Building and Solving Optimization Models with SAS/OR®”
  (Mainly focused on Proc OPTMODEL)
  I found this course very useful!
Questions / Comments

Maurice Djona
Programmer/Analyst,
SAS Technology Centre
System Development Division
R.H. Coats Building, 14th Floor, Section Q

Ottawa, Ontario, Canada K1A 0T6
(613) 951-5331  Fax (613) 951-0607
Maurice.Djona@statcan.gc.ca