Statistics Canada
www.statcan.gc.ca

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

Statistics Canada Statistique Canada

Canada

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:

Min OR Max $f(x)$

Subject to $c_i(x)\ \{\leq, =, \geq\}\ b_i\ (i = 1, 2, \ldots, m)$

$l_j \leq x_j \leq u_j\ (j = 1, 2, \ldots, n)$

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. $f = x_1 + x_2 * x_3$)

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:**

$$
\begin{aligned}
\text{max} \quad & x_1 + x_2 + x_3 \\
\text{subject to} \quad & 3x_1 + 2x_2 - x_3 \leq 1 \\
& -2x_1 - 3x_2 + 2x_3 \leq 1 \\
& x_1 \geq 0,\ x_2 \geq 0,\ x_3 \geq 0
\end{aligned}
$$

- **A NLP (Non Linear Programming) problem:**

$$
\begin{aligned}
\text{max} \quad & x_1 * x_2 * x_3 \\
\text{subject to} \quad & 3x_1 + 2x_2 - x_3 \leq 1 \\
& -2x_1 - 3x_2 + 2x_3 \leq 1 \\
& x_1 \geq 0,\ x_2 \geq 0,\ x_3 \geq 0
\end{aligned}
$$

- **LPs solved in *CONFID* at Statistics Canada:**
  - Number of variables: dozens to ≈ 200 000
  - Number of constraints: dozens to ≈ 50 000
  - Non-zero constraint coefficients: hundreds to ≈ 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

Proc OPTMODEL

(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
  $\Rightarrow$ 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 $\Rightarrow$ less I/O $\Rightarrow$ 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:

Problem	**Solver**
Linear Programming	LP
Mixed Integer Programming	MILP
Quadratic Programming	QP (experimental)
Nonlinear Programming, Unconstrained	NLPU
General Nonlinear Programming	NLPC
General Nonlinear Programming	SQP
General Nonlinear Programming	IPNLP (experimental)

OPTMODEL ≈ more friendly interface to various optimization solvers

OPTMODEL Procedure – Some Elements of Syntax

PROC OPTMODEL options **;**

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] **;**
- **FOR {** index set **}** statement;
- **IF** logic **THEN** statement **;** [**ELSE** statement **;**]
- **LEAVE ;**
- **PRINT** print items **;**
- **PUT** put items **;**
- **READ DATA** SAS-data-set **INTO** columns **;**
- **RESET OPTIONS** options **;**
- **RESTORE** constraint **;**
- **SAVE MPS** SAS-data-set **;**
- **SAVE QPS** SAS-data-set **;**
- **SOLVE** [**WITH** solver] [**OBJECTIVE** name] [**/** options **;**
- **STOP ;**
- **UNFIX** variable [**=** expression] **;**

Quit;

Plus: Powerful expressions for sets manipulation

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

OPTMODEL Procedure: Example

- **A simple LP problem:**

$$\begin{aligned} \max \quad & x_1 + x_2 + x_3 \\ \text{subject to} \quad & 3x_1 + 2x_2 - x_3 \leq 1 \\ & -2x_1 - 3x_2 + 2x_3 \leq 1 \\ & x_1 \geq 0,\ x_2 \geq 0,\ x_3 \geq 0 \end{aligned}$$

/* Linear Problem using <u>named</u> 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;
```

OPTMODEL Procedure: Example (cont.)

SAS output windows

The OPTMODEL Procedure

Problem Summary

Objective Sense	Maximization
Objective Function	f
Objective Type	Linear
Number of Variables	3
Bounded Above	0
Bounded Below	3
Bounded Below and Above	0
Free	0
Fixed	0
Number of Constraints	2
Linear LE (<=)	2
Linear EQ (=)	0
Linear GE (>=)	0
Linear Range	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

x1	x2	x3
0	3	5

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 <u>indexes</u> for parameters and variables (Indexing ≈ 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*/
    CONSTRAINT c1:  3*x[1] + 2*x[2] -     x[3] <= 1;      /* Constraint*/
    CONSTRAINT c2: -2*x[1] -  3*x[2] + 2*x[3] <= 1;       /* Constraint*/
    SOLVE WITH LP OBJECTIVE f;                            /* Solve model using LP solver */
    PRINT x;                                              /* Print the solution */
QUIT;
```

The OPTMODEL Procedure

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.)

- **Read model data from data sets**

$$
\begin{array}{ll}
\text{max} & x_1 + x_2 + x_3 \\
\text{subject to} & 3x_1 + 2x_2 - x_3 \le 1 \\
& -2x_1 - 3x_2 + 2x_3 \le 1 \\
& x_1 \ge 0,\ x_2 \ge 0,\ x_3 \ge 0
\end{array}
$$

```
/* Objective coefficients data set */
DATA objCoefDS;
 INPUT varId objCoef; DATALINES;
            1      1
            2      1
            3      1
      ;
RUN;
```

```
/* 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;
```

```
/* 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;
```

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;
```

Data set's variable

OPTMODEL's array

NB: Possibility to modify and re-solve the problem <u>in memory</u> without leaving OPTMODEL (loops).

OPTMODEL Procedure: Example (cont.)

The OPTMODEL Procedure

```
Var x[1] >= 0
Var x[2] >= 0
Var x[3] >= 0
Maximize f=x[1] + x[2] + x[3]
Constraint c[1]: 3*x[1] + 2*x[2] - x[3] <= 1
Constraint c[2]: - 2*x[1] - 3*x[2] + 2*x[3] <= 1
```

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

Arrays in OPTMODEL

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

Data Set Variables

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

Results data set created in Proc OPTMODEL

Obs	var	value
1	1	0
2	2	3
3	3	5

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:

```
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

Model in MPS format

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

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):

Solution from OPTLP

Obs	_OBJ_ID_	_VAR_	_TYPE_	_OBJCOEF_	_VALUE_
1	f	x1	N	1	0
2	f	x2	N	1	3
3	f	x3	N	1	5

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 $\Rightarrow$ less I/O $\Rightarrow$ 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

Statistics Canada
Statistique Canada

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

Canada