

A Goal-Programming Macro

David Sommer, Optimal Systems

Introduction

This paper reports on a new goal-programming macro, MSGOAL. MSGOAL is a multi stage goal programming method. It provides a natural, efficient, robust, and flexible approach to goal programming.

Problem Formulation

The "classic" formulation of goal programming is: given a $g \times n$ matrix G and a $n \times 1$ solution vector X , find that value of X which sequentially maximizes $G_i X$, as $i = 1, 2, \dots, g$. Here G_i represents row i of G . In words: find the maximum possible value of $G_i X$. Impose that value as a constraint as $G_2 X$ is maximized. Then impose both values as constraints as $G_3 X$ is maximized, etc.

This formulation, in practice, has few applications. This is because of the absolute nature of the precedence of goal 1 over goal 2, goal 2 over goal 3, etc. For example, goal 1 could be "maximize cancer treatment effectiveness for the President", goal 2 "maximize the President's comfort while treatment is given", and goal 3 "maximize efficiency".

The classic formulation also has a practical drawback in terms of linear programming formulation. Since the goal programming problem is solved as a sequence of LP's (applying each goal as a constraint after it is solved), it turns out that digital problems often arise because the problem becomes slightly overconstrained.

The classic formulation is also overly restrictive in allowing only maximization of goals. Sometimes minimization is wanted. Sometimes it is desired to bring the goal function below, equal to, or greater than a specified constant.

Also, real problems often have "side constraints": linear constraints that are imposed from the beginning as the goals are sequentially solved. Since an LP algorithm is used to solve the goal problems, such side constraints are not difficult to include.

Finally, the classic formulation is often stated in terms that may be convenient for mathematical clarity, but are unnatural for those used to standard LP algorithms. Typically, such formulations require the user to specify an "object" and "priorities" datasets. Making the "priorities" dataset is a very non-intuitive business!

MSGOAL Parameters

MSGOAL is a SAS macro that addresses the problems discussed above. After creating a SAS* dataset representing the goals and side constraints of the problem in standard PROC LP format (not the SPARSEDATA form) the user simply enters:

```
%MSGOAL(DATA=dataset_name,  
          G1=goal_1_name ...,  
          G2=goal_2_name ..., ...)
```

Here, the ellipses (...) represent additional information or parameters, details to follow. But first, note the simplicity of formulation and use: the problem is set up like a normal LP problem for solution by PROC LP of SAS/OR*, and the call to MSGOAL states in a direct way the necessary information.

MSGOAL is a Multi Stage Goal Programming macro. It permits detailed control over the goal programming process. For example, the goals may be implemented in any order (regardless of their position in the problem dataset). The goals can be to maximize, minimize, make equal to a given value, make less than a given value, or greater than a given value. After attaining (or moving as close as possible towards) a goal, that goal then becomes fixed, but with a user specified relaxation allowed. A particular row in the problem dataset may be used more than once in goal statements. For example, one could "try to attain \$1000 profit" and then later "maximize profit".

Parameters are:

DATA=datasetname (REQUIRED)

The name of the dataset containing the problem data. This may be any acceptable LP format dataset, with the stipulations that the RHS is named RHS, the Type is TYPE, and that all rows that are wanted for goals are of type MIN, MAX, or FREE. If ranges are wanted, they should be expressed in column RANGE.

G1=id type u v (REQUIRED).

This is an expression of the first goal to be applied. It consists of 2, 3, or 4 parts separated by spaces. These are:

id Name of the goal row in the DATA dataset.

type Any of MAX, MIN, EQ, LE, or GE. This expresses the kind of goal. MAX and MIN are obvious. EQ means "get as close as possible to value u". LE (GE) mean "try to get LE (GE) value u".

u The value (numeric) for EQ, LE, and GE goals. May also be used as the relaxation value for MAX and MIN type of goals. For MAX, this means: after reaching the maximum possible value, set this goal as a constraint using that value LESS the amount u. For MIN, the idea is similar.

v The relaxation for goals of type EQ, LE, or GE.

Note: relaxation values are optional. If absent, the relaxation is 0.01.

G2=, G3=, ... (OPTIONAL).

Similar in format to G1. There should not be any "gaps" in the goals given. For example, one could use G1, G2, and G3, but not G1, G3, and G4.

FC=lpcontrols (OPTIONAL).

This may be any permissible set of controls for PROC LP. They will be applied in solving for the First goal (expressed by G1). ("FC" stands for First Control.) For example, one might want to give FC=PRIMALIN=dname, where dname is the name of a previous PRIMALOUT for this problem (thus giving a starting basis). One could also include other controls such as PRINTLEVEL=n, etc. Default is FC=PRINTLEVEL=-1.

IC=lpcontrols (OPTIONAL).

This is the Intermediate Controls parameter. Similar in format to FC, but PRIMALIN is not permitted because MSGOAL automatically uses the basis from the solution of the previous goal, and this is a good feasible basis. The controls given in IC are used for all PROC LP goal solutions except the first and last. Default is IC=PRINTLEVEL=-1.

LC=lpcontrols (OPTIONAL).

The Last Controls parameter, used for the PROC LP call on the last goal. Note: if there is only one goal, then FC will be used for the PROC LP controls (rather than LC). Default is LC=PRINTLEVEL=1.

Some examples of invocations of MSGOAL:

```
%MSGOAL(DATA=MYPROB, G1=COST MIN 10,  
G2=HAPPINES MAX)
```

This specifies the SAS dataset MYPROB as containing the problem data. MSGOAL is to minimize the COST row, then make COST into a LE constraint whose RHS value is 10 units greater than the minimum attained while the row HAPPINES is maximized.

```
%MSGOAL(DATA=ANOTHER, G1=FRED EQ 123 7,  
G2=JERRY LE 50 5, G3=EVIL MIN,  
FC=PRINTLEVEL=1, IC=PRINTLEVEL=1)
```

SAS dataset ANOTHER contains the problem data. MSGOAL is to try to bring row FRED to the value 123. After doing so, it will make FRED into a constraint that holds that linear expression to within 7 units of the best value attained. Then JERRY is brought to under 50, if possible. JERRY then becomes a constraint of the form LE, with RHS 5 units greater than the value attained. Finally, the row EVIL is minimized. The FC and IC parameters set the PRINTLEVEL to 1, so that full printouts occur at each optimization.

Stochastic Requirements

Sometimes a problem has a degree of randomness in either the LP tableau or in the goals that are to be given in the solution process. In either case, the following steps could be followed:

1. Define a macro consisting of a %DO I = 1 %TO (limit), and necessary datasteps plus invocations of MSGOAL. The resulting solutions would be kept as PRIMALOUT, using the FC parameter of MSGOAL to specify the PRIMALOUT datasets.
2. Use clustering or regression techniques to discover groupings or relationships involving the randomized requirements and the MSGOAL solutions.

A technical point: the CALL SYMPUT capability of SAS is a very useful method to make the content of data step variables (including values returned from the random number functions), into macrovariable contents. Macrovariables can, in turn, be used very easily in calls to MSGOAL or in datasteps that modify the LP tableau.

The FUNPROF Problem

The Fun and Profit goal programming problem is intended to illustrate the use of the MSGOAL macro.

The Dorn Construction Company does home improvement projects. In assigning workers to tasks, owner Ken Dorn takes three factors into account:

Cost: workers have different hourly costs

Danger: some dangerous tasks are done with greater care by some workers.

Fun: workers have preferences as to the jobs they do.

On one job, there is roofing, siding, and insulation to be done. The workers available are Bill, Fred, and Jim. Variables BILROOF, BILSIDE, etc. represent the use (in hours) of a worker on a task. The COST, DANGER, and FUN rows represent minimization, minimization, and maximization of cost, danger, and fun respectively.

There are three side constraints: ROOFING, SIDING, and INSUL that represent the requirement in

"standard" work hours for each of the tasks. (Workers are more or less effective at these tasks. For example, Bill gets 1.2 "standard" hours of work done on roofing for each clock hour he works.)

Another two side constraints enforce the requirement that all three get the same number of hours. (See BILFRD and FRDJIM rows.)

Given the formulation (see problem tableau below), any number of experiments can be done. For example, using the MSGOAL macro, one can first try to bring cost down to a given level, then bring danger below an acceptable point, then bring fun up to a satisfactory level, and finally return to cost and attempt to minimize it. Since the goals are actually expressed in the %MSGOAL statement, it is easy for a user to try several different goal statements and orderings.

	BIL	BIL	BIL	FRD	FRD	FRD	JIM	JIM	JIM	
	ROOF	INSL	FRD	FRD	FRD	INSL	ROOF	INSL	INSL	
COST	9	9	9	8	8	8	6	6	6	(MIN)
FUN	-4	2	6	2	4	0	8	2	-1	(MAX)
DANGER	3	1	2	5	1	0	2	4	0	(MIN)
ROOFING	1.2	.	.	1.1	.	.	0.9	.	.	≤ 20
SIDING	.	1.1	.	.	1.0	.	.	0.7	.	≤ 36
INSUL	.	.	1.1	.	.	1.2	.	.	0.8	≤ 14
BILFRD	1	1	1	-1	-1	-1	.	.	.	= 0
FRDJIM	.	.	.	1	1	1	-1	-1	-1	= 0

The Dorn Construction Problem is listed below:

```
DATA FUNPROF;
  INPUT ID $ TYPE $ RHS BIL ROOF BIL SIDE
        BIL INSL FRD ROOF FRD SIDE FRD INSL JIM ROOF
        JIM SIDE JIM INSL;
  CARDS;
  COST MIN . 9 9 9 8 8 8 6 6 6
  FUN MAX . -4 2 6 2 4 0 8 2 -1
  DANGER MIN . 3 1 2 5 1 0 2 4 0
  ROOFING EQ 20 1.2 . . 1.1 . . 0.9 . .
  SIDING EQ 36 . . 1.1 . . 1.0 . . 0.7 .
  INSUL EQ 14 . . 1.1 . . 1.2 . . 0.8
  BIL FRD EQ 0 1 1 1 -1 -1 -1 . . .
  FRD JIM EQ 0 . . . 1 1 1 -1 -1 -1
;
```

```
RUN;
%INCLUDE MSGOAL;
%MSGOAL(DATA=FUNPROF, G1=COST LE 530 10,
        G2=FUN GE 350 15,
        G3=DANGER LE 100,
        G4=COST MIN )
```

The SASLOG and final PRINT are given below.

```
1  OPTIONS LS=66 PS=55;
2  %INCLUDE FUNPROF;
NOTE: The data set WORK.FUNPROF has 8 observations and
12 variables.
NOTE: The DATA statement used 2.00 seconds.
50  %INCLUDE MSGOAL;
320 %MSGOAL(DATA=FUNPROF, G1=COST LE 530 10,
G2=FUN GE 350 15,
321 G3=DANGER LE 100,
G4=COST MIN )
NOTE: The data set WORK._CURROBJ has 1 observations
and 1 variables.
NOTE: The DATA statement used 1.00 seconds.
NOTE: The data set WORK._GOALP has 9 observations and
14 variables.
NOTE: The DATA statement used 5.00 seconds.
NOTE: The data set WORK._PRIMSOL has 15 observations
and 10 variables.
NOTE: The PROCEDURE LP used 12.00 seconds.
```

```
GOAL 1: 'COST LE 530 10' HAS BEEN COMPLETED WITH
SOLUTION VALUE = 530
NOTE: The DATA statement used 3.00 seconds.
NOTE: The data set WORK._GOALP has 9 observations and
15 variables.
NOTE: The DATA statement used 6.00 seconds.
NOTE: The data set WORK._PRIMSOL has 16 observations
and 10 variables.
NOTE: The PROCEDURE LP used 11.00 seconds.
GOAL 2: 'FUN GE 350 15' HAS BEEN COMPLETED WITH
SOLUTION VALUE = 350
NOTE: The DATA statement used 3.00 seconds.
NOTE: The data set WORK._GOALP has 9 observations and
15 variables.
NOTE: The DATA statement used 6.00 seconds.
NOTE: The data set WORK._PRIMSOL has 15 observations
and 10 variables.
NOTE: The PROCEDURE LP used 12.00 seconds.
GOAL 3: 'DANGER LE 100' HAS BEEN COMPLETED WITH
SOLUTION VALUE = 100
NOTE: The DATA statement used 3.00 seconds.
NOTE: The data set WORK._GOALP has 9 observations and
15 variables.
NOTE: The DATA statement used 5.00 seconds.
NOTE: The data set WORK._PRIMSOL has 15 observations
and 10 variables.
NOTE: The PROCEDURE LP used 12.00 seconds.
GOAL 4: 'COST MIN' HAS BEEN COMPLETED WITH SOLUTION
VALUE = 531.772852
NOTE: The DATA statement used 3.00 seconds.
```

```
SAS 19:47 Monday, December 4, 1989
LINEAR PROGRAMMING PROCEDURE
VARIABLE SUMMARY
```

Variable Name	Status	Type	Price	Activity	Reduced Cost
BIL ROOF	NON-NEG		0	0.000000	3.855620
BIL SIDE	BASIC NON-NEG		0	14.855055	0.000000
BIL INSL	BASIC NON-NEG		0	8.265504	0.000000
FRD ROOF	NON-NEG		0	0.000000	2.826712
FRD SIDE	BASIC NON-NEG		0	19.030604	0.000000
FRD INSL	BASIC NON-NEG		0	4.089955	0.000000
JIM ROOF	BASIC NON-NEG		0	22.222222	0.000000
JIM SIDE	BASIC NON-NEG		0	0.898337	0.000000
JIM INSL	NON-NEG		0	0.000000	0.317587
_ABOVE	BASIC NON-NEG		1	531.773	0.000000
_BELOW	ALTER NON-NEG		-1	0.000000	0.000000
FUN	SURPLUS			0.000000	0.183866
DANGER	BASIC SLACK			1.555543	0.000000

```
CONSTRAINT SUMMARY
```

Constraint Name	Type	S/S Col	Rhs	Activity	Dual Activity
COST	EQ		0	0.000000	-1.000000
FUN	GE	12	335	335.000	0.183866
DANGER	LE	13	100.01	98.454457	0.000000
ROOFING	EQ		20	20.000000	4.754522
SIDING	EQ		36	36.000000	7.688953
INSUL	EQ		14	14.000000	7.020349
BIL FRD	EQ		0	0.000000	0.174419
FRD JIM	EQ		0	0.000000	-0.250000
_CURROBJ	OBJECT			531.773	-1.000000

MSGOAL Listing

```
OPTIONS NOSOURCE2 CHARCODE NOCENTER;
%MACRO MKSET; %* A few system specific
initializations.;
%IF &SYSVER<6 %THEN %DO;
  OPTIONS DQUOTE; %* Needed in CMS and TSO;
%END;
%XEND MKSET;
%MKSET
%MACRO
MSGOAL(DATA=G1,G2,G3,G4,G5,G6,G7,G8,G9,G10,
FC=,IC=,LC=);
%* This is a Multi Stage Goal Programming macro. It
permits detailed control over the goal programming
```

process. For example, the goals may be implemented in any order (regardless of their position in the problem dataset). The goals can be to maximize, minimize, make equal to a given value, make less than a given value, or greater than a given value. After attaining (or moving as close as possible towards) a goal, that goal then becomes fixed, but with a user specified relaxation allowed. A particular row in the problem dataset may be used more than once in goal statements. For example, one could "try to attain \$1000 profit" and then later "maximize profit".

Parameters are:

DATA=datasetname (REQUIRED) The name of the dataset containing the problem data. This may be any acceptable LP format dataset, with the stipulations that the RHS is named **RHS**, the type is **_TYPE**, and that all rows that are wanted for goals are of type MIN or MAX. If ranges are wanted, they should be expressed in column **RANGE**.

G1=id type u v (REQUIRED). This is an expression of the first goal to be applied. It consists of 2, 3, or 4 parts separated by spaces. These are:

id Name of the goal row in the DATA dataset. **type** Any of MAX, MIN, EQ, LE, or GE. This expresses the kind of goal. MAX and MIN are obvious. EQ means "get as close as possible to value u". LE (GE) mean "try to get LE (GE) value u".

u The value (numeric) for EQ, LE, and GE goals. May also be used as the relaxation value for MAX and MIN type of goals. For MAX, this means: after reaching the maximum possible value, set this goal value LESS the amount u. For MIN, the idea is similar.

v The relaxation for goals of type EQ, LE, or GE. Note: relaxation values are optional. If absent, the relaxation is 0.01.

G2=, G3=, ... G10 (OPTIONAL). Similar in format to G1.

FC=lpcontrols (OPTIONAL). This may be any permissible set of controls for PROC LP. They will be applied in solving for the FIRST goal (expressed by G1). For example, one might want to give

FC=PRIMALIN=dsname, where dsname is the name of a previous PRIMALOUT for this problem (thus giving a starting basis). One could also include other controls such as **PRINTLEVEL=n**, etc. Default is **FC=PRINTLEVEL=-1**.

IC=lpcontrols (OPTIONAL). This is the Intermediate controls parameter. Similar in format to FC, but PRIMALIN would not be recommended because MSGOAL automatically uses the basis from the solution of the previous goal, and this is a good feasible basis. The controls given in IC are used for all PROC LP goal solutions except the first and last. Default is **IC=PRINTLEVEL=-1**.

LC=lpcontrols (OPTIONAL). Controls used for the PROC LP call on the LAST goal. Note: if there is only one goal, then FC will be used for the PROC LP controls (rather than LC). Default is **LC=PRINTLEVEL=1**.

DATA_CURROBJ; * Build a silly little ds with Current Obj as the single observation;

_ID = '_CURROBJ';

RUN;

* Determine number of last Goal used (as &L);

%DO I = 1 %TO 10;

%IF &&G&I?== %THEN %LET L = &I;

%END;

* This is the main loop that goes over all goals, solving each in turn and using results of preceding goal to fix values. Due to chokeability of SAS macro processor in PC-SAS 6.03, this is broken up into three smaller macros.;

%DO I = 1 %TO &L;

%IF &&G&I = %THEN %GOTO SKIP;

%MSGOAL1;

%MSGOAL2;

%MSGOAL3;

%SKIP;;

%END;

RUN;

%MEND MSGOAL;

* MSGOAL1 first cracks the G1 goal parameter into &ID, &TYPE, &RHV, and &TOL. Then if this is the goal G1 (first goal), it concatenates the little **_CURROBJ** dataset with the users dataset, producing the working **_GOALP** dataset, which is used hereafter. It also sets all MAX and MIN rows to FREE to prepare for subsequent work. But if this is G2 ... G&L, it first gets solution values for the **_ABOVE** and **_BELOW** variables from the previous solution, and saves them as **&ABOVEP** and **&BELOWP**, since they will be needed later to 'fix' the previous goal as a constraint. Note: **_ABOVE** represents the amount by which the goal function is above 0 (for MAX or MIN goals) or above the 'u' value (for LE, GE, or EQ goals). Similarly, **_BELOW** is amount by which the goal function is below 0 or u. After getting the solution values, initiate the main datasetp.

```

;
%MACRO MSGOAL1;
  %GLOBAL ID TYPE TOL RHV ABOVEP BELOWP;
  %GLOBAL IDP TYPEP RHVP TOLP;
  %LET ID = %SCAN(&&G&I,1,%STR( ));
  %LET TYPE = %SCAN(&&G&I,2,%STR( ));
  %IF %TYPE=XMAX ?/ %TYPE=XMIN %THEN %DO;
    %LET TOL = %SCAN(&&G&I,3,%STR( ));
    %LET RHV = 0;
  %END;
  %ELSE %DO;
    %LET RHV = %SCAN(&&G&I,3,%STR( ));
    %IF %RHV= %THEN %LET RHV = 0;
    %LET TOL = %SCAN(&&G&I,4,%STR( ));
  %END;
  %IF %TOL= %THEN %LET TOL = 0.01;
  %IF %I=1 %THEN %DO;
    DATA _GOALP;
      SET &DATA _CURROBJ;
      IF _TYPE_='MAX' OR _TYPE_='MIN' THEN
        _TYPE_='FREE';
    %END;
    %IF %I>1 %THEN %DO;
      DATA _GOALP; SET _GOALP;
    %END;
  %MEND MSGOAL1;

```

* This macro continues with the main datasetp. It alters three observations of **_GOALP**:

1. Converts the current goal observation into one of the form **SUM(AiXi) - _ABOVE + _BELOW = RHSVALUE** (where RHSVALUE is 'u' for EQ, LE, and GE goals, 0 otherwise.

2. Changes the **_CURROBJ** observation into one of the following forms, depending on the 'type' in the users goal specification:

type	_TYPE	function form	comment
MAX	MAX	_ABOVE - _BELOW	Maximize goal function
MIN	MIN	_ABOVE - _BELOW	Minimize goal function
EQ	MIN	_ABOVE + _BELOW	Minimize dev. from goal
LE	MIN	_ABOVE	Minimize positive dev.
GE	MIN	_BELOW	Minimize negative dev.

3. If this is other than solution of the first goal, changes the previous goal "balance" equation (set in preceding step as in 1. above), so that it now becomes a constraint. Changes MAX, MIN, LE, and GE goal balance equations into GE, LE, LE, and GE constraints, using the relaxation factor (if any) supplied by the user. The EQ type of goal, if completely attained, requires a **_RANGE** entry to handle the relaxation correctly.

;

```

XMACRO MSGOAL2;
  IF _ID_="&ID" THEN DO;  %* Make current goal
                           observation into;
                           %* a balance equation.;
    _TYPE_ = 'EQ';
    _ABOVE_ = -1;
    _BELOW_ = 1;
    _RHS_ = &RHV;
  END;
  IF _ID_='_CURROBJ_' THEN DO;  %* Adjust _CURROBJ_.;
    _ABOVE_ = 1;
    _TYPE_ = 'MIN';
    %IF &XTYPE=XMAX ?/ &XTYPE=XMIN XTHEN XDO;
      _BELOW_ = -1;
      _TYPE_ = "&TYPE";
    XEND;
    %ELSE XSTR( _BELOW_ = 1;);
    %IF &XTYPE=XLE XTHEN XSTR( _BELOW_ = 0;);
    %IF &XTYPE=XGE XTHEN XSTR( _ABOVE_ = 0;);
  END;
  %IF &I>1 XTHEN XDO;
    IF _ID_="&IDP" THEN DO;  %* Make previous goal
                              balance equation;
                              %* into a constraint
                              'holding' the goal;
      _ABOVE_ = .;
      _BELOW_ = .;
      _RANGE_ = .;
      %IF &XTYPE=XMAX XTHEN XDO;
        _TYPE_ = 'GE';
        _RHS_ = &ABOVEP - &BELOWP - &TOLP;
      XEND;
      %IF &XTYPE=XMIN XTHEN XDO;
        _TYPE_ = 'LE';
        _RHS_ = &ABOVEP - &BELOWP + &TOLP;
      XEND;
      %IF &XTYPE=XEQ XTHEN XDO;
        %IF &ABOVEP?==0 XTHEN XDO;
          _TYPE_ = 'LE';
          _RHS_ = &RHVP + &ABOVEP + &TOLP;
        XEND;
        %IF &BELOWP?==0 XTHEN XDO;
          _TYPE_ = 'GE';
          _RHS_ = &RHVP - &BELOWP - &TOLP;
        XEND;
        %IF &ABOVEP=0 & &BELOWP=0 XTHEN XDO;
          _TYPE_ = 'GE';
          _RHS_ = &RHVP - &TOLP;
        %IF &TOLP?==0 XTHEN XSTR( _RANGE_ = 2 *
          &TOLP;);
      XEND;
    XEND;
    %IF &XTYPE=XLE XTHEN XDO;
      _TYPE_ = 'LE';
      _RHS_ = &RHVP + &ABOVEP + &TOLP;
    XEND;
    %IF &XTYPE=XGE XTHEN XDO;
      _TYPE_ = 'GE';
      _RHS_ = &RHVP - &BELOWP - &TOLP;
    XEND;
  END;
XEND;
RUN;
XMEMD MSGOAL2;

```

* MSGOAL3 concludes the main dataset for _GOALP and then calls on PROC LP to solve the problem. First, it saves the current goals id, type, rhsvalue, and tolerance since they will be needed to modify in the next step. Then it calls PROC LP with the appropriate parameters. It searches the PRIMSOL dataset (output by PROC LP) to get the solution values, needed for the next iteration.

```

/*      %IF &I>1 & &SYSVER<6 XTHEN
        XSTR( PRIMALIN=PRIMSOL );*/

```

* The above line SHOULD work in MSGOAL3 (old version of PROC LP) but now does not seem to, so it is pulled out. I think the primal solution dataset has been

redefined in current versions. Should be some way to accomplish the same thing.;

```

XMACRO MSGOAL3;
  PROC LP DATA= GOALP PRIMALOUT=_PRIMSOL
    %IF &I=1 XTHEN XDO;
      %IF %QUOTE(&FC)= XTHEN XSTR( PRINTLEVEL=-1);
      %ELSE XSTR( &FC);
    XEND;
    %IF &I>1 & &I<=L XTHEN XDO;
      %IF %QUOTE(&IC)= XTHEN XSTR( PRINTLEVEL=-1);
      %ELSE XSTR( &IC);
    XEND;
    %IF &I>1 & &I<=L XTHEN XDO;
      %IF %QUOTE(&LC)= XTHEN XSTR( PRINTLEVEL=1);
      %ELSE XSTR( &LC);
    XEND;
  ;
  RUN;
  %LET IDP = &ID;
  %LET TYPEP = &TYPE;
  %LET RHVP = &RHV;
  %LET TOLP = &TOL;
  %LET ABOVEP = 0 ;
  %LET BELOWP = 0 ;
  DATA NULL ; SET _PRIMSOL END=LAST;
  RETAIN SOLN &RHVP;
  IF _VAR_='_ABOVE_' & _STATUS_='_BASIC_' THEN DO;
    CALL
    SYMPUT('ABOVEP',LEFT(PUT(_VALUE_,BEST12.)));
    SOLN + _VALUE_;
  END;
  IF _VAR_='_BELOW_' & _STATUS_='_BASIC_' THEN DO;
    CALL
    SYMPUT('BELOWP',LEFT(PUT(_VALUE_,BEST12.)));
    SOLN + (-_VALUE_);
  END;
  IF LAST THEN DO;
    Y = LEFT(PUT(SOLN,BEST10.));
    PUT "GOAL &I: '&&G&I' HAS BEEN COMPLETED "
        'WITH SOLUTION VALUE = ' Y;
  END;
  RUN;
XMEMD MSGOAL3;

* MPUT is a simple debugging utility for macros to
show values of macro variables.;
XMACRO MPUT(N1,N2,N3,N4,N5,N6,N7,N8,N9);
  %LOCAL I MPUT;
  %LET _MPUT_ = ;
  %DO I = 1 %TO 9;
    %IF &&N&I?== XTHEN %LET _MPUT_ = &_MPUT
    /&&N&I/&&&&&&N&I/;
  XEND;
  %PUT &_MPUT_;
XMEMD;

```

Trademark Notice

*SAS and SAS/OR are registered trademarks of SAS Institute Inc., Cary, NC, USA.

Author

David Sommer
Optimal Systems, Inc.
10338 Colorado Road
Bloomington, Minnesota 55438
Voice: 612-831-5724
FAX: 612-831-2765