SAS/OR® 14.3 User's Guide
Mathematical Programming
The OPTLP Procedure

This document is an individual chapter from *SAS/OR® 14.3 User's Guide: Mathematical Programming*.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2017. *SAS/OR® 14.3 User's Guide: Mathematical Programming*. Cary, NC: SAS Institute Inc.

SAS/OR® 14.3 User's Guide: Mathematical Programming

SAS Institute Inc., SAS Campus Drive, Cary, NC 27513-2414

September 2017

Chapter 12
The OPTLP Procedure

Contents

Overview: OPTLP Procedure

The OPTLP procedure provides four methods of solving linear programs (LPs). A linear program has the following formulation:

$$\begin{array}{rl} \min & \mathbf{c}^{\mathrm{T}}\mathbf{x} \\ \text{subject to} & \mathbf{A}\mathbf{x}\ \{\geq, =, \leq\}\ \mathbf{b} \\ & \mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \end{array}$$

where

$\mathbf{x}$	$\in$	$\mathbb{R}^n$	is the vector of decision variables
$\mathbf{A}$	$\in$	$\mathbb{R}^{m \times n}$	is the matrix of constraints
$\mathbf{c}$	$\in$	$\mathbb{R}^n$	is the vector of objective function coefficients
$\mathbf{b}$	$\in$	$\mathbb{R}^m$	is the vector of constraints' right-hand sides (RHS)
$\mathbf{l}$	$\in$	$\mathbb{R}^n$	is the vector of lower bounds on variables
$\mathbf{u}$	$\in$	$\mathbb{R}^n$	is the vector of upper bounds on variables

The following LP algorithms are available in the OPTLP procedure:

- primal simplex algorithm
- dual simplex algorithm
- network simplex algorithm
- interior point algorithm

The primal and dual simplex algorithms implement the two-phase simplex method. In phase I, the algorithm tries to find a feasible solution. If no feasible solution is found, the LP is infeasible; otherwise, the algorithm enters phase II to solve the original LP. The network simplex algorithm extracts a network substructure, solves this using network simplex, and then constructs an advanced basis to feed to either primal or dual simplex. The interior point algorithm implements a primal-dual predictor-corrector interior point algorithm.

PROC OPTLP requires a linear program to be specified using a SAS data set that adheres to the MPS format, a widely accepted format in the optimization community. For details about the MPS format see Chapter 17, "The MPS-Format SAS Data Set."

You can use the MPSOUT= option to convert typical PROC LP format data sets into MPS-format SAS data sets. The option is available in the LP, INTPOINT, and NETFLOW procedures. For details about this option, see Chapter 4, "The LP Procedure" (*SAS/OR User's Guide: Mathematical Programming Legacy Procedures*), Chapter 3, "The INTPOINT Procedure" (*SAS/OR User's Guide: Mathematical Programming Legacy Procedures*), and Chapter 5, "The NETFLOW Procedure" (*SAS/OR User's Guide: Mathematical Programming Legacy Procedures*).

Getting Started: OPTLP Procedure

The following example illustrates how you can use the OPTLP procedure to solve linear programs. Suppose you want to solve the following problem:

$$
\begin{array}{rrcrcrcrl}
\min & 2x_1 & - & 3x_2 & - & 4x_3 & & & \\
\text{subject to} & & - & 2x_2 & - & 3x_3 & \geq & -5 & \text{(R1)} \\
& x_1 & + & x_2 & + & 2x_3 & \leq & 4 & \text{(R2)} \\
& x_1 & + & 2x_2 & + & 3x_3 & \leq & 7 & \text{(R3)} \\
& & & x_1, & & x_2, \; x_3 & \geq & 0 &
\end{array}
$$

The corresponding MPS-format SAS data set is as follows:

```
data example;
   input field1 $ field2 $ field3 $ field4 field5 $ field6;
   datalines;
NAME            .       EXAMPLE    .     .      .
ROWS            .       .          .     .      .
N               COST    .          .     .      .
G               R1      .          .     .      .
L               R2      .          .     .      .
L               R3      .          .     .      .
COLUMNS         .       .          .     .      .
.               X1      COST       2     R2     1
.               X1      R3         1     .      .
.               X2      COST      -3     R1    -2
.               X2      R2         1     R3     2
.               X3      COST      -4     R1    -3
.               X3      R2         2     R3     3
RHS             .       .          .     .      .
.               RHS     R1        -5     R2     4
.               RHS     R3         7     .      .
ENDATA          .       .          .     .      .
;
```

You can also create this data set from an MPS-format flat file (examp.mps) by using the following SAS macro:

```
%mps2sasd(mpsfile = "examp.mps", outdata = example);
```

NOTE: The SAS macro %MPS2SASD is provided in SAS/OR software. See "Converting an MPS/QPS-Format File: %MPS2SASD" on page 824 for details.

You can use the following statement to call the OPTLP procedure:

```
title1 'The OPTLP Procedure';
proc optlp data = example
  objsense  = min
  presolver = automatic
  algorithm = primal
  primalout = expout
  dualout   = exdout;
run;
```

NOTE: The "N" designation for "COST" in the rows section of the data set example also specifies a minimization problem. See the section "ROWS Section" on page 817 for details.

The optimal primal and dual solutions are stored in the data sets expout and exdout, respectively, and are displayed in Figure 12.1.

```
title2 'Primal Solution';
proc print data=expout label;
run;

title2 'Dual Solution';
proc print data=exdout label;
run;
```

Figure 12.1 Primal and Dual Solution Output

The OPTLP Procedure
Primal Solution

Obs	Objective Function ID	RHS ID	Variable Name	Variable Type	Objective Coefficient	Lower Bound	Upper Bound	Variable Value	Variable Status	Reduced Cost
1	COST	RHS	X1	N	2	0	1.7977E308	0.0	L	2.0
2	COST	RHS	X2	N	-3	0	1.7977E308	2.5	B	0.0
3	COST	RHS	X3	N	-4	0	1.7977E308	0.0	L	0.5

The OPTLP Procedure
Dual Solution

Obs	Objective Function ID	RHS ID	Constraint Name	Constraint Type	Constraint RHS	Constraint Lower Bound	Constraint Upper Bound	Dual Variable Value	Constraint Status	Constraint Activity
1	COST	RHS	R1	G	-5	.	.	1.5	U	-5.0
2	COST	RHS	R2	L	4	.	.	0.0	B	2.5
3	COST	RHS	R3	L	7	.	.	0.0	B	5.0

For details about the type and status codes displayed for variables and constraints, see the section "Data Input and Output" on page 570.

Syntax: OPTLP Procedure

The following statements are available in the OPTLP procedure:

PROC OPTLP < *options* > ;
DECOMP < *options* > ;
DECOMP_MASTER < *options* > ;
DECOMP_SUBPROB < *options* > ;

Functional Summary

Table 12.1 summarizes the list of options available for the OPTLP procedure, classified by function.

Table 12.1 Options for the OPTLP Procedure

Description	Option
Data Set Options	
Specifies the input data set	DATA=
Specifies the dual input data set for warm start	DUALIN=
Specifies the dual solution output data set	DUALOUT=
Specifies whether the LP model is a maximization or minimization problem	OBJSENSE=
Specifies the primal input data set for warm start	PRIMALIN=
Specifies the primal solution output data set	PRIMALOUT=
Solver Options	
Enables or disables IIS detection	IIS=
Specifies the type of algorithm	ALGORITHM=
Specifies the type of algorithm called after network simplex	ALGORITHM2=
Presolve Option	
Specifies the type of presolve	PRESOLVER=
Controls the dualization of the problem	DUALIZE=
Control Options	
Specifies the feasibility tolerance	FEASTOL=
Specifies the frequency of printing solution progress	LOGFREQ=
Specifies the detail of solution progress printed in log	LOGLEVEL=
Specifies the maximum number of iterations	MAXITER=
Specifies the time limit for the optimization process	MAXTIME=
Specifies the optimality tolerance	OPTTOL=
Enables or disables printing summary	PRINTLEVEL=
Specifies units of CPU time or real time	TIMETYPE=
Simplex Algorithm Options	
Specifies the type of initial basis	BASIS=
Specifies the type of pricing strategy	PRICETYPE=
Specifies the queue size for pricing	QUEUESIZE=
Enables or disables scaling of the problem	SCALE=

Table 12.1 (continued)

Description	Option
Specifies the initial seed for the random number generator	SEED=
Interior Point Algorithm Options	
Enables or disables interior crossover	CROSSOVER=
Specifies the stopping criterion based on duality gap	DUALITYGAP=
Parallel Options	
Enables the OPTLP procedure to run deterministically	DETERMINISTIC=
Specifies number of threads for the parallel OPTLP procedure to use	NTHREADS=

PROC OPTLP Statement

PROC OPTLP < *options* > **;**

You can specify the following options in the PROC OPTLP statement.

Data Set Options

DATA=*SAS-data-set*

specifies the input data set corresponding to the LP model. If this option is not specified, PROC OPTLP will use the most recently created SAS data set. See Chapter 17, "The MPS-Format SAS Data Set," for more details about the input data set.

DUALIN=*SAS-data-set*

DIN=*SAS-data-set*

specifies the input data set corresponding to the dual solution that is required for warm starting the primal and dual simplex algorithms. See the section "Data Input and Output" on page 570 for details.

DUALOUT=*SAS-data-set*

DOUT=*SAS-data-set*

specifies the output data set for the dual solution. This data set contains the dual solution information. See the section "Data Input and Output" on page 570 for details.

OBJSENSE=*option*

specifies whether the LP model is a minimization or a maximization problem. You specify OBJSENSE=MIN for a minimization problem and OBJSENSE=MAX for a maximization problem. Alternatively, you can specify the objective sense in the input data set; see the section "ROWS Section" on page 817 for details. If for some reason the objective sense is specified differently in these two places, this option supersedes the objective sense specified in the input data set. If the objective sense is not specified anywhere, then PROC OPTLP interprets and solves the linear program as a minimization problem.

PRIMALIN=*SAS-data-set*

PIN=*SAS-data-set*

specifies the input data set corresponding to the primal solution that is required for warm starting the primal and dual simplex algorithms. See the section "Data Input and Output" on page 570 for details.

PRIMALOUT=*SAS-data-set*

POUT=*SAS-data-set*

specifies the output data set for the primal solution. This data set contains the primal solution information. See the section "Data Input and Output" on page 570 for details.

Solver Options

ALGORITHM=*option*

SOLVER=*option*

SOL=*option*

specifies an LP algorithm. You can specify the following *options*:

PRIMAL (PS)	uses the primal simplex algorithm.
DUAL (DS)	uses the dual simplex algorithm.
NETWORK (NS)	uses the network simplex algorithm.
INTERIORPOINT (IP)	uses the interior point algorithm.
CONCURRENT (CON)	uses several different algorithms in parallel.

The valid abbreviated value for each option is indicated in parentheses. By default, ALGORITHM=DUAL.

ALGORITHM2=*option*

SOLVER2=*option*

specifies an LP algorithm if ALGORITHM=NS. You can specify the following values:

PRIMAL (PS)	uses the primal simplex algorithm (after network simplex).
DUAL (DS)	uses the dual simplex algorithm (after network simplex).

The valid abbreviated value for each option is indicated in parentheses. By default, the OPTLP procedure decides which algorithm is best to use after calling the network simplex algorithm on the extracted network.

IIS=FALSE | TRUE

specifies whether PROC OPTLP attempts to identify a set of constraints and variables that form an irreducible infeasible set (IIS). The following values are valid for the IIS= option.

FALSE	disables IIS detection.
TRUE	enables IIS detection.

If an IIS is found, information about infeasible constraints or variable bounds can be found in the DUALOUT= and PRIMALOUT= data sets. The default value of this option is OFF. See the section "Irreducible Infeasible Set" on page 584 for details.

Presolve Options

DUALIZE=AUTOMATIC | OFF | ON

controls the dualization of the problem. You can specify the following values:

AUTOMATIC specifies that the presolver use a heuristic to decide whether to dualize the problem or not.

OFF disables dualization. The optimization problem is solved in the form that you specify.

ON specifies that the presolver formulate the dual of the linear optimization problem.

Dualization is usually helpful for problems that have many more constraints than variables. You can use this option with all simplex algorithms in PROC OPTLP, but it is most effective with the primal and dual simplex algorithms.

By default, DUALIZE=AUTOMATIC.

PRESOLVER=AUTOMATIC | NONE | BASIC | MODERATE | AGGRESSIVE

PRESOL=AUTOMATIC | NONE | BASIC | MODERATE | AGGRESSIVE

specifies the presolve option. You can specify the following values:

AUTOMATIC applies the presolver by using the default settings.

NONE disables the presolver.

BASIC performs a basic presolve, such as removing empty rows, columns, and fixed variables.

MODERATE performs a basic presolve and applies other inexpensive presolve techniques.

AGGRESSIVE performs a moderate presolve and applies other aggressive (but computationally expensive) presolve techniques.

By default, PRESOLVER=AUTOMATIC, which is somewhere between the MODERATE and AGGRESSIVE settings. For more information, see the section "Presolve" on page 574.

Control Options

FEASTOL=ϵ

specifies the feasibility tolerance $\epsilon \in$[1E–9, 1E–4] for determining the feasibility of a variable value. The default value is 1E–6. Simplex algorithms use the absolute error and interior point algorithms use the relative error for the computation of feasibility tolerance.

LOGFREQ=k

PRINTFREQ=k

specifies that the printing of the solution progress to the iteration log is to occur after every k iterations. The print frequency, k, is an integer between zero and the largest four-byte signed integer, which is $2^{31} - 1$.

The value $k = 0$ disables the printing of the progress of the solution.

If the LOGFREQ= option is not specified, then PROC OPTLP displays the iteration log with a dynamic frequency according to the problem size if the primal or dual simplex algorithm is used, with frequency

10,000 if the network simplex algorithm is used, or with frequency 1 if the interior point algorithm is used.

LOGLEVEL=NONE | BASIC | MODERATE | AGGRESSIVE

controls the amount of information displayed in the SAS log by the LP solver, from a short description of presolve information and summary to details at each iteration. The following values are valid for this option.

NONE	turns off all solver-related messages in SAS log.
BASIC	displays a solver summary after stopping.
MODERATE	prints a solver summary and an iteration log by using the interval dictated by the LOGFREQ= option.
AGGRESSIVE	prints a detailed solver summary and an iteration log by using the interval dictated by the LOGFREQ= option.

The default value is MODERATE.

MAXITER=*k*

specifies the maximum number of iterations. The value k can be any integer between one and the largest four-byte signed integer, which is $2^{31} - 1$. If you do not specify this option, the procedure does not stop based on the number of iterations performed. For network simplex, this iteration limit corresponds to the algorithm called after network simplex (either primal or dual simplex).

MAXTIME=*t*

specifies an upper limit of t seconds of time for reading in the data and performing the optimization process. The value of the TIMETYPE= option determines the type of units used. If you do not specify this option, the procedure does not stop based on the amount of time elapsed. The value of t can be any positive number; the default value is the positive number that has the largest absolute value that can be represented in your operating environment.

OPTTOL=ϵ

specifies the optimality tolerance $\epsilon \in$[1E–9, 1E–4] for declaring optimality. The default value is 1E–6. Simplex algorithms use the absolute error and interior point algorithms use the relative error for the computation of feasibility tolerance.

PRINTLEVEL=0 | 1 | 2

specifies whether a summary of the problem and solution should be printed. If PRINTLEVEL=1, then the Output Delivery System (ODS) tables ProblemSummary and SolutionSummary are produced and printed. If PRINTLEVEL=2, then the same tables are produced and printed along with an additional table called ProblemStatistics. If PRINTLEVEL=0, then no ODS tables are produced or printed. The default value is 1.

For details about the ODS tables created by PROC OPTLP, see the section "ODS Tables" on page 581.

TIMETYPE=CPU | REAL

specifies whether CPU time or real time is used for the MAXTIME= option and the _OROPTLP_ macro variable in a PROC OPTLP call. The following values are valid of the TIMETYPE= option.

CPU specifies units of CPU time.

REAL specifies units of real time.

The default value of the TIMETYPE= option depends on the value of the NTHREADS= option.

If you specify a value greater than 1 for the NTHREADS= option, the default value of the TIMETYPE= option is REAL, otherwise the default value of the TIMETYPE= option is CPU.

Simplex Algorithm Options

BASIS=CRASH | SLACK | WARMSTART

specifies the option for generating an initial basis. You can specify the following values:

CRASH generates an initial basis by using crash techniques (Maros 2003). The procedure creates a triangular basic matrix consisting of both decision variables and slack variables.

SLACK generates an initial basis by using all slack variables.

WARMSTART starts the primal and dual simplex algorithms with a user-specified initial basis. The PRIMALIN= and DUALIN= data sets are required to specify an initial basis.

The default option is determined automatically based on the problem structure. For network simplex, this option has no effect.

PRICETYPE=HYBRID | PARTIAL | FULL | DEVEX | STEEPESTEDGE

specifies the pricing strategy for the primal and dual simplex algorithms. You can specify the following values:

HYBRID uses a hybrid of the Devex and steepest-edge pricing strategies. This strategy is available for the primal simplex algorithm only.

PARTIAL uses Dantzig's rule on a queue of decision variables. Optionally, you can specify QUEUESIZE=. This strategy is available for the primal simplex algorithm only.

FULL uses Dantzig's rule on all decision variables.

DEVEX uses the Devex pricing strategy.

STEEPESTEDGE uses the steepest-edge pricing strategy.

The default option is determined automatically based on the problem structure. For the network simplex algorithm, this option applies only to the algorithm specified by the ALGORITHM2= option. See the section "Pricing Strategies for the Primal and Dual Simplex Algorithms" on page 574 for details.

QUEUESIZE=*k*

specifies the queue size $k \in [1, n]$, where n is the number of decision variables. This queue is used for finding an entering variable in the simplex iteration. The default value is chosen adaptively based on the number of decision variables. This option is used only when PRICETYPE=PARTIAL.

SCALE=NONE | AUTOMATIC

specifies one of the following scaling options:

NONE	disables scaling.
AUTOMATIC	automatically applies scaling procedure if necessary.

The default option is AUTOMATIC.

SEED=*number*

specifies the initial seed for the random number generator. Because the seed affects the perturbation in the simplex algorithms, the result might be a different optimal solution and a different solver path, but the effect is usually negligible. The value of *number* can be any positive integer up to the largest four-byte signed integer, which is $2^{31} - 1$. By default, SEED=100.

Interior Point Algorithm Options

CROSSOVER=FALSE | TRUE

specifies whether to convert the interior point solution to a basic simplex solution. If the interior point algorithm terminates with a solution, the crossover algorithm uses the interior point solution to create an initial basic solution. After performing primal fixing and dual fixing, the crossover algorithm calls a simplex algorithm to locate an optimal basic solution.

The default value of the CROSSOVER= option is TRUE.

DUALITYGAP=δ

specifies the desired relative duality gap $\delta \in$[1E–9, 1E–4]. This is the relative difference between the primal and dual objective function values and is the primary solution quality parameter. The default value is 1E–6. See the section "The Interior Point Algorithm" on page 576 for details.

Parallel Options

DETERMINISTIC=TRUE | FALSE

specifies whether PROC OPTLP should run parallel in deterministic or nondeterministic mode. You can specify the following values:

TRUE	runs PROC OPTLP in deterministic parallel mode.
FALSE	runs PROC OPTLP in nondeterministic parallel mode.

By default, DETERMINISTIC=TRUE.

NTHREADS=*number*

specifies the maximum number of threads for PROC OPTLP to use for multithreaded processing. The value of *number* can be any integer between 1 and 256, inclusive. The default is the number of cores on the machine that executes the process or the number of cores permissible based on your installation (whichever is less). The number of simultaneously active CPUs is limited by your installation and license configuration.

Decomposition Algorithm Statements

The following statements are available for the decomposition algorithm in the OPTLP procedure:

DECOMP < *options* > **;**

DECOMP_MASTER < *options* > **;**

DECOMP_SUBPROB < *options* > **;**

For more information about these statements, see Chapter 15, "The Decomposition Algorithm."

Details: OPTLP Procedure

Data Input and Output

This subsection describes the PRIMALIN= and DUALIN= data sets required to warm start the primal and dual simplex algorithms, and the PRIMALOUT= and DUALOUT= output data sets.

Definitions of Variables in the PRIMALIN= Data Set

The PRIMALIN= data set has two required variables defined as follows:

VAR
specifies the name of the decision variable.

STATUS
specifies the status of the decision variable. It can take one of the following values:

B	basic variable
L	nonbasic variable at its lower bound
U	nonbasic variable at its upper bound
F	free variable
A	newly added variable in the modified LP model when using the BASIS=WARMSTART option

NOTE: The PRIMALIN= data set is created from the PRIMALOUT= data set that is obtained from a previous "normal" run of PROC OPTLP (one that uses only the DATA= data set as the input).

Definitions of Variables in the DUALIN= Data Set

The DUALIN= data set also has two required variables defined as follows:

ROW
specifies the name of the constraint.

STATUS

specifies the status of the slack variable for a given constraint. It can take one of the following values:

- B basic variable
- L nonbasic variable at its lower bound
- U nonbasic variable at its upper bound
- F free variable
- A newly added variable in the modified LP model when using the BASIS=WARMSTART option

NOTE: The DUALIN= data set is created from the DUALOUT= data set that is obtained from a previous "normal" run of PROC OPTLP (one that uses only the DATA= data set as the input).

Definitions of Variables in the PRIMALOUT= Data Set

The PRIMALOUT= data set contains the primal solution to the LP model; each observation corresponds to a variable of the LP problem. The PRIMALOUT= data set can contain an intermediate solution, if one is available. See Example 12.1 for an example of the PRIMALOUT= data set. The variables in the data set have the following names and meanings.

_OBJ_ID_

specifies the name of the objective function. This is particularly useful when there are multiple objective functions, in which case each objective function has a unique name.

NOTE: PROC OPTLP does not support simultaneous optimization of multiple objective functions in this release.

_RHS_ID_

specifies the name of the variable that contains the right-hand-side value of each constraint.

VAR

specifies the name of the decision variable.

TYPE

specifies the type of the decision variable. _TYPE_ can take one of the following values:

- N nonnegative
- D bounded (with both lower and upper bound)
- F free
- X fixed
- O other (with either lower or upper bound)

OBJCOEF

specifies the coefficient of the decision variable in the objective function.

LBOUND

specifies the lower bound on the decision variable.

UBOUND
specifies the upper bound on the decision variable.

VALUE
specifies the value of the decision variable.

STATUS
specifies the status of the decision variable. _STATUS_ can take one of the following values:

B	basic variable
L	nonbasic variable at its lower bound
U	nonbasic variable at its upper bound
F	free variable
A	superbasic variable (a nonbasic variable that has a value strictly between its bounds)
I	LP model infeasible (all decision variables have _STATUS_ equal to I)

For the interior point algorithm with IIS= OFF, _STATUS_ is blank.

The following values can appear only if IIS= ON. See the section "Irreducible Infeasible Set" on page 584 for details.

I_L	the lower bound of the variable is needed for the IIS
I_U	the upper bound of the variable is needed for the IIS
I_F	both bounds of the variable needed for the IIS (the variable is fixed or has conflicting bounds)

_R_COST_
specifies the reduced cost of the decision variable, which is the amount by which the objective function is increased per unit increase in the decision variable. The reduced cost associated with the *i*th variable is the *i*th entry of the following vector:

$$\left(\mathbf{c}^{\mathrm{T}} - \mathbf{c}_{\boldsymbol{B}}^{\mathrm{T}}\mathbf{B}^{-1}\mathbf{A}\right)$$

where $\mathbf{B} \in \mathbb{R}^{m \times m}$ denotes the basis (matrix composed of *basic* columns of the constraints matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$), $\mathbf{c} \in \mathbb{R}^{n}$ is the vector of objective function coefficients, and $\mathbf{c}_{\boldsymbol{B}} \in \mathbb{R}^{m}$ is the vector of objective coefficients of the variables in the basis.

Definitions of Variables in the DUALOUT= Data Set

The DUALOUT= data set contains the dual solution to the LP model; each observation corresponds to a constraint of the LP problem. The DUALOUT= data set can contain an intermediate solution, if one is available. Information about the objective rows of the LP problems is not included. See Example 12.1 for an example of the DUALOUT= data set. The variables in the data set have the following names and meanings.

_OBJ_ID_
specifies the name of the objective function. This is particularly useful when there are multiple objective functions, in which case each objective function has a unique name.

NOTE: PROC OPTLP does not support simultaneous optimization of multiple objective functions in this release.

_RHS_ID_
specifies the name of the variable that contains the right-hand-side value of each constraint.

ROW
specifies the name of the constraint.

TYPE
specifies the type of the constraint. _TYPE_ can take one of the following values:

L "less than or equals" constraint

E equality constraint

G "greater than or equals" constraint

R ranged constraint (both "less than or equals" and "greater than or equals")

RHS
specifies the value of the right-hand side of the constraint. It takes a missing value for a ranged constraint.

_L_RHS_
specifies the lower bound of a ranged constraint. It takes a missing value for a non-ranged constraint.

_U_RHS_
specifies the upper bound of a ranged constraint. It takes a missing value for a non-ranged constraint.

VALUE
specifies the value of the dual variable associated with the constraint.

STATUS
specifies the status of the slack variable for the constraint. _STATUS_ can take one of the following values:

B basic variable

L nonbasic variable at its lower bound

U nonbasic variable at its upper bound

F free variable

A superbasic variable (a nonbasic variable that has a value strictly between its bounds)

I LP model infeasible (all decision variables have _STATUS_ equal to I)

The following values can appear only if option IIS= ON. See the section "Irreducible Infeasible Set" on page 584 for details.

I_L the "GE" ($\geq$) condition of the constraint is needed for the IIS

I_U the "LE" ($\leq$) condition of the constraint is needed for the IIS

I_F both conditions of the constraint are needed for the IIS (the constraint is an equality or a range constraint with conflicting bounds)

ACTIVITY

specifies the left-hand-side value of a constraint. In other words, the value of _ACTIVITY_ for the ith constraint would be equal to $\mathbf{a}_i^{\mathrm{T}}\mathbf{x}$, where $\mathbf{a}_i$ refers to the ith row of the constraints matrix and $\mathbf{x}$ denotes the vector of current decision variable values.

Presolve

Presolve in PROC OPTLP uses a variety of techniques to reduce the problem size, improve numerical stability, and detect infeasibility or unboundedness (Andersen and Andersen 1995; Gondzio 1997). During presolve, redundant constraints and variables are identified and removed. Presolve can further reduce the problem size by substituting variables. Variable substitution is a very effective technique, but it might occasionally increase the number of nonzero entries in the constraint matrix.

In most cases, using presolve is very helpful in reducing solution times. You can enable presolve at different levels or disable it by specifying the PRESOLVER= option.

Pricing Strategies for the Primal and Dual Simplex Algorithms

Several pricing strategies for the primal and dual simplex algorithms are available. Pricing strategies determine which variable enters the basis at each simplex pivot. They can be controlled by specifying the PRICETYPE= option.

The primal simplex algorithm has the following five pricing strategies:

PARTIAL	uses Dantzig's most violated reduced cost rule (Dantzig 1963). It scans a queue of decision variables and selects the variable with the most violated reduced cost as the entering variable. You can optionally specify the QUEUESIZE= option to control the length of this queue.
FULL	uses Dantzig's most violated reduced cost rule. It compares the reduced costs of all decision variables and selects the variable with the most violated reduced cost as the entering variable.
DEVEX	implements the Devex pricing strategy developed by Harris (1973).
STEEPESTEDGE	uses the steepest-edge pricing strategy developed by Forrest and Goldfarb (1992).
HYBRID	uses a hybrid of the Devex and steepest-edge pricing strategies.

The dual simplex algorithm has only three pricing strategies available: FULL, DEVEX, and STEEPEST-EDGE.

Warm Start for the Primal and Dual Simplex Algorithms

You can warm start the primal and dual simplex algorithms by specifying the option BASIS=WARMSTART. Additionally you need to specify the PRIMALIN= and DUALIN= data sets. The primal and dual simplex algorithms start with the basis thus provided. If the given basis cannot form a valid basis, the algorithms use the basis generated using their *crash* techniques.

After an LP model is solved using the primal and dual simplex algorithms, the BASIS=WARMSTART option enables you to perform sensitivity analysis such as modifying the objective function, changing the right-hand sides of the constraints, adding or deleting constraints or decision variables, and combinations of these cases. A faster solution to such a modified LP model can be obtained by starting with the basis in the optimal solution to the original LP model. This can be done by using the BASIS=WARMSTART option, modifying the DATA= input data set, and specifying the PRIMALIN= and DUALIN= data sets. Example 12.4 and Example 12.5 illustrate how to reoptimize an LP problem with a modified objective function and a modified right-hand side by using this technique. Example 12.6 shows how to reoptimize an LP problem after adding a new constraint.

The network simplex algorithm ignores the option BASIS=WARMSTART.

CAUTION: Since the presolver uses the objective function and/or right-hand-side information, the basis provided by you might not be valid for the presolved model. It is therefore recommended that you turn the PRESOLVER= option off when using BASIS=WARMSTART.

The Network Simplex Algorithm

The network simplex algorithm in PROC OPTLP attempts to leverage the speed of the network simplex algorithm to more efficiently solve linear programs by using the following process:

1. It heuristically extracts the largest possible network substructure from the original problem.
2. It uses the network simplex algorithm to solve for an optimal solution to this substructure.
3. It uses this solution to construct an advanced basis to warm-start either the primal or dual simplex algorithm on the original linear programming problem.

The network simplex algorithm is a specialized version of the simplex algorithm that uses spanning-tree bases to more efficiently solve linear programming problems that have a pure network form. Such LPs can be modeled using a formulation over a directed graph, as a minimum-cost flow problem. Let $G = (N, A)$ be a directed graph, where N denotes the nodes and A denotes the arcs of the graph. The decision variable x_{ij} denotes the amount of flow sent from node i to node j. The cost per unit of flow on the arcs is designated by c_{ij}, and the amount of flow sent across each arc is bounded to be within $[l_{ij}, u_{ij}]$. The demand (or supply) at each node is designated as b_i, where $b_i > 0$ denotes a supply node and $b_i < 0$ denotes a demand node. The corresponding linear programming problem is as follows:

$$
\begin{array}{rlcll}
\min & \sum_{(i,j)\in A} c_{ij} x_{ij} & & & \\
\text{subject to} & \sum_{(i,j)\in A} x_{ij} - \sum_{(j,i)\in A} x_{ji} & = & b_i & \forall i \in N \\
 & x_{ij} & \leq & u_{ij} & \forall (i,j) \in A \\
 & x_{ij} & \geq & l_{ij} & \forall (i,j) \in A
\end{array}
$$

The network simplex algorithm used in PROC OPTLP is the primal network simplex algorithm. This algorithm finds the optimal primal feasible solution and a dual solution that satisfies complementary slackness. Sometimes the directed graph G is disconnected. In this case, the problem can be decomposed into its weakly connected components and each minimum-cost flow problem can be solved separately. After solving each component, the optimal basis for the network substructure is augmented with the non-network variables and constraints from the original problem. This advanced basis is then used as a starting point for the primal or dual simplex method. The solver automatically selects the algorithm to use after network simplex. However, you can override this selection with the ALGORITHM2= option.

The network simplex algorithm can be more efficient than the other algorithms on problems with a large network substructure. You can view the size of the network structure in the log.

The Interior Point Algorithm

The interior point algorithm in PROC OPTLP implements an infeasible primal-dual predictor-corrector interior point algorithm. To illustrate the algorithm and the concepts of duality and dual infeasibility, consider the following LP formulation (the primal):

$$\begin{array}{rl} \text{min} & \mathbf{c}^{\mathrm{T}}\mathbf{x} \\ \text{subject to} & \mathbf{A}\mathbf{x} \geq \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{array}$$

The corresponding dual formulation is as follows:

$$\begin{array}{rlclcl} \text{max} & \mathbf{b}^{\mathrm{T}}\mathbf{y} & & & & \\ \text{subject to} & \mathbf{A}^{\mathrm{T}}\mathbf{y} & + & \mathbf{w} & = & \mathbf{c} \\ & & & \mathbf{y} & \geq & \mathbf{0} \\ & & & \mathbf{w} & \geq & \mathbf{0} \end{array}$$

where $\mathbf{y} \in \mathbb{R}^m$ refers to the vector of dual variables and $\mathbf{w} \in \mathbb{R}^n$ refers to the vector of dual slack variables.

The dual formulation makes an important contribution to the certificate of optimality for the primal formulation. The primal and dual constraints combined with complementarity conditions define the first-order optimality conditions, also known as KKT (Karush-Kuhn-Tucker) conditions, which can be stated as follows:

$$\begin{array}{rcll} \mathbf{A}\mathbf{x} - \mathbf{s} & = & \mathbf{b} & \text{(primal feasibility)} \\ \mathbf{A}^{\mathrm{T}}\mathbf{y} + \mathbf{w} & = & \mathbf{c} & \text{(dual feasibility)} \\ \mathbf{W}\mathbf{X}\mathbf{e} & = & \mathbf{0} & \text{(complementarity)} \\ \mathbf{S}\mathbf{Y}\mathbf{e} & = & \mathbf{0} & \text{(complementarity)} \\ \mathbf{x},\ \mathbf{y},\ \mathbf{w},\ \mathbf{s} & \geq & \mathbf{0} & \end{array}$$

where $\mathbf{e} \equiv (1, \ldots, 1)^{\mathrm{T}}$ of appropriate dimension and $\mathbf{s} \in \mathbb{R}^m$ is the vector of primal *slack* variables.

NOTE: Slack variables (the s vector) are automatically introduced by the algorithm when necessary; it is therefore recommended that you not introduce any slack variables explicitly. This enables the algorithm to handle slack variables much more efficiently.

The letters $\mathbf{X}$, $\mathbf{Y}$, $\mathbf{W}$, and $\mathbf{S}$ denote matrices with corresponding x, y, w, and s on the main diagonal and zero elsewhere, as in the following example:

$$\mathbf{X} \equiv \begin{bmatrix} x_1 & 0 & \cdots & 0 \\ 0 & x_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & x_n \end{bmatrix}$$

If $(\mathbf{x}^*, \mathbf{y}^*, \mathbf{w}^*, \mathbf{s}^*)$ is a solution of the previously defined system of equations that represent the KKT conditions, then $\mathbf{x}^*$ is also an optimal solution to the original LP model.

At each iteration the interior point algorithm solves a large, sparse system of linear equations,

$$\begin{bmatrix} \mathbf{Y}^{-1}\mathbf{S} & \mathbf{A} \\ \mathbf{A}^{\mathrm{T}} & -\mathbf{X}^{-1}\mathbf{W} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{y} \\ \Delta\mathbf{x} \end{bmatrix} = \begin{bmatrix} \Xi \\ \Theta \end{bmatrix}$$

where $\Delta\mathbf{x}$ and $\Delta\mathbf{y}$ denote the vector of *search directions* in the primal and dual spaces, respectively, and Θ and Ξ constitute the vector of the right-hand sides.

The preceding system is known as the reduced KKT system. PROC OPTLP uses a preconditioned quasi-minimum residual algorithm to solve this system of equations efficiently.

An important feature of the interior point algorithm is that it takes full advantage of the sparsity in the constraint matrix, thereby enabling it to efficiently solve large-scale linear programs.

The interior point algorithm works simultaneously in the primal and dual spaces. It attains optimality when both primal and dual feasibility are achieved and when complementarity conditions hold. Therefore, it is of interest to observe the following four measures where $\|v\|_2$ is the Euclidean norm of the vector v:

- relative primal infeasibility measure α:

$$\alpha = \frac{\|\mathbf{A}\mathbf{x} - \mathbf{b} - \mathbf{s}\|_2}{\|\mathbf{b}\|_2 + 1}$$

- relative dual infeasibility measure β:

$$\beta = \frac{\|\mathbf{c} - \mathbf{A}^{\mathrm{T}}\mathbf{y} - \mathbf{w}\|_2}{\|\mathbf{c}\|_2 + 1}$$

- relative duality gap δ:

$$\delta = \frac{|\mathbf{c}^{\mathrm{T}}\mathbf{x} - \mathbf{b}^{\mathrm{T}}\mathbf{y}|}{|\mathbf{c}^{\mathrm{T}}\mathbf{x}| + 1}$$

- absolute complementarity γ:

$$\gamma = \sum_{i=1}^{n} x_i w_i + \sum_{i=1}^{m} y_i s_i$$

These measures are displayed in the iteration log.

Iteration Log for the Primal and Dual Simplex Algorithms

The primal and dual simplex algorithms implement a two-phase simplex algorithm. Phase I finds a feasible solution, which phase II improves to an optimal solution.

When LOGFREQ=1, the following information is printed in the iteration log:

Algorithm	indicates which simplex method is running by printing the letter P (primal) or D (dual).
Phase	indicates whether the algorithm is in phase I or phase II of the simplex method.
Iteration	indicates the iteration number.
Objective Value	indicates the current amount of infeasibility in phase I and the primal objective value of the current solution in phase II.
Time	indicates the time elapsed (in seconds).
Entering Variable	indicates the entering pivot variable. A slack variable that enters the basis is indicated by the corresponding row name followed by "(S)". If the entering nonbasic variable has distinct and finite lower and upper bounds, then a "bound swap" can take place in the primal simplex method.
Leaving Variable	indicates the leaving pivot variable. A slack variable that leaves the basis is indicated by the corresponding row name followed by "(S)". The leaving variable is the same as the entering variable if a bound swap has taken place.

When you omit the LOGFREQ= option or specify a value greater than 1, only the algorithm, phase, iteration, objective value, and time information is printed in the iteration log.

The behavior of objective values in the iteration log depends on both the current phase and the chosen algorithm. In phase I, both simplex methods have artificial objective values that decrease to 0 when a feasible solution is found. For the dual simplex method, phase II maintains a dual feasible solution, so a minimization problem has increasing objective values in the iteration log. For the primal simplex method, phase II maintains a primal feasible solution, so a minimization problem has decreasing objective values in the iteration log.

During the solution process, some elements of the LP model might be perturbed to improve performance. In this case the objective values that are printed correspond to the perturbed problem. After reaching optimality for the perturbed problem, PROC OPTLP solves the original problem by switching from the primal simplex method to the dual simplex method (or from the dual to the primal simplex method). Because the problem might be perturbed again, this process can result in several changes between the two algorithms.

Iteration Log for the Network Simplex Algorithm

After finding the embedded network and formulating the appropriate relaxation, the network simplex algorithm uses a primal network simplex algorithm. In the case of a connected network, with one (weakly connected) component, the log shows the progress of the simplex algorithm. The following information is displayed in the iteration log:

Iteration	indicates the iteration number.
PrimalObj	indicates the primal objective value of the current solution.
Primal Infeas	indicates the maximum primal infeasibility of the current solution.
Time	indicates the time spent on the current component by network simplex.

The frequency of the simplex iteration log is controlled by the LOGFREQ= option. The default value of the LOGFREQ= option is 10,000.

If the network relaxation is disconnected, the information in the iteration log shows progress at the component level. The following information is displayed in the iteration log:

Component	indicates the component number being processed.
Nodes	indicates the number of nodes in this component.
Arcs	indicates the number of arcs in this component.
Iterations	indicates the number of simplex iterations needed to solve this component.
Time	indicates the time spent so far in network simplex.

The frequency of the component iteration log is controlled by the LOGFREQ= option. In this case, the default value of the LOGFREQ= option is determined by the size of the network.

The LOGLEVEL= option adjusts the amount of detail shown. By default, LOGLEVEL= is set to MODERATE and reports as described previously. If set to NONE, no information is shown. If set to BASIC, the only information shown is a summary of the network relaxation and the time spent solving the relaxation. If set to AGGRESSIVE, in the case of one component, the log displays as described previously; in the case of multiple components, for each component, a separate simplex iteration log is displayed.

Iteration Log for the Interior Point Algorithm

The interior point algorithm implements an infeasible primal-dual predictor-corrector interior point algorithm. The following information is displayed in the iteration log:

Iter	indicates the iteration number.
Complement	indicates the (absolute) complementarity.
Duality Gap	indicates the (relative) duality gap.
Primal Infeas	indicates the (relative) primal infeasibility measure.
Bound Infeas	indicates the (relative) bound infeasibility measure.
Dual Infeas	indicates the (relative) dual infeasibility measure.
Time	indicates the time elapsed (in seconds).

If the sequence of solutions converges to an optimal solution of the problem, you should see all columns in the iteration log converge to zero or very close to zero. If they do not, it can be the result of insufficient iterations being performed to reach optimality. In this case, you might need to increase the value specified in the MAXITER= or MAXTIME= options. If the complementarity or the duality gap do not converge, the problem might be infeasible or unbounded. If the infeasibility columns do not converge, the problem might be infeasible.

Iteration Log for the Crossover Algorithm

The crossover algorithm takes an optimal solution from the interior point algorithm and transforms it into an optimal basic solution. The iterations of the crossover algorithm are similar to simplex iterations; this similarity is reflected in the format of the iteration logs.

When LOGFREQ=1, the following information is printed in the iteration log:

Phase	indicates whether the primal crossover (PC) or dual crossover (DC) technique is used.
Iteration	indicates the iteration number.
Objective Value	indicates the total amount by which the superbasic variables are off their bound. This value decreases to 0 as the crossover algorithm progresses.
Time	indicates the time elapsed (in seconds).
Entering Variable	indicates the entering pivot variable. A slack variable that enters the basis is indicated by the corresponding row name followed by "(S)."
Leaving Variable	indicates the leaving pivot variable. A slack variable that leaves the basis is indicated by the corresponding row name followed by "(S)."

When you omit the LOGFREQ= option or specify a value greater than 1, only the phase, iteration, objective value, and time information is printed in the iteration log.

After all the superbasic variables have been eliminated, the crossover algorithm continues with regular primal or dual simplex iterations.

Concurrent LP

The ALGORITHM=CON option starts several different linear optimization algorithms in parallel in a single-machine mode. The OPTLP procedure automatically determines which algorithms to run and how many threads to assign to each algorithm. If sufficient resources are available, the procedure runs all four standard algorithms. When the first algorithm ends, the procedure returns the results from that algorithm and terminates any other algorithms that are still running. If you specify a value of TRUE for the DETERMINISTIC option, the algorithm for which the results are returned is not necessarily the one that finished first. The OPTLP procedure deterministically selects the algorithm for which the results are returned. Regardless of which mode (deterministic or nondeterministic) is in effect, terminating algorithms that are still running might take a significant amount of time.

During concurrent optimization, the procedure displays the iteration log for the dual simplex algorithm. For more information about this iteration log, see the section "Iteration Log for the Primal and Dual Simplex Algorithms" on page 577. Upon termination, the procedure displays the iteration log for the algorithm that finishes first, unless the dual simplex algorithm finishes first. If you specify LOGLEVEL=AGGRESSIVE, the OPTLP procedure displays the iteration logs for all algorithms that are run concurrently.

If you specify PRINTLEVEL=2 and ALGORITHM=CON, the OPTLP procedure produces an ODS table called ConcurrentSummary. This table contains a summary of the solution statuses of all algorithms that are run concurrently.

Parallel Processing

The interior point and concurrent LP algorithms can be run in single-machine mode (in single-machine mode, the computation is executed by multiple threads on a single computer). The decomposition algorithm can be run in either single-machine or distributed mode (in distributed mode, the computation is executed on multiple computing nodes in a distributed computing environment).

NOTE: Distributed mode requires the SAS Optimization product on the SAS Viya platform.

ODS Tables

PROC OPTLP creates two Output Delivery System (ODS) tables by default. The first table, ProblemSummary, is a summary of the input LP problem. The second table, SolutionSummary, is a brief summary of the solution status. You can use ODS table names to select tables and create output data sets. For more information about ODS, see *SAS Output Delivery System: Procedures Guide*.

If you specify a value of 2 for the PRINTLEVEL= option, then the ProblemStatistics table is produced. This table contains information about the problem data. For more information, see the section "Problem Statistics" on page 583. If you specify PRINTLEVEL=2 and ALGORITHM=CON, the ConcurrentSummary table is produced. This table contains solution status information for all algorithms that are run concurrently. For more information, see the section "Concurrent LP" on page 580.

Table 12.2 lists all the ODS tables that can be produced by the OPTLP procedure, along with the statement and option specifications required to produce each table.

Table 12.2 ODS Tables Produced by PROC OPTLP

ODS Table Name	Description	Statement	Option
ProblemSummary	Summary of the input LP problem	PROC OPTLP	PRINTLEVEL=1 (default)
SolutionSummary	Summary of the solution status	PROC OPTLP	PRINTLEVEL=1 (default)
ProblemStatistics	Description of input problem data	PROC OPTLP	PRINTLEVEL=2
ConcurrentSummary	Summary of the solution status for all algorithms run concurrently	PROC OPTLP	PRINTLEVEL=2, ALGORITHM=CON

A typical output of PROC OPTLP is shown in Figure 12.2.

Figure 12.2 Typical OPTLP Output

The OPTLP Procedure

Problem Summary	
Problem Name	ADLITTLE
Objective Sense	Minimization
Objective Function	.Z....
RHS	ZZZZ0001
Number of Variables	97
Bounded Above	0
Bounded Below	97
Bounded Above and Below	0
Free	0
Fixed	0
Number of Constraints	56
LE (<=)	40
EQ (=)	15
GE (>=)	1
Range	0
Constraint Coefficients	383

Solution Summary	
Solver	LP
Algorithm	Dual Simplex
Objective Function	.Z....
Solution Status	Optimal
Objective Value	225494.96316
Primal Infeasibility	2.842171E-13
Dual Infeasibility	4.263256E-13
Bound Infeasibility	0
Iterations	73
Presolve Time	0.00
Solution Time	0.00

You can create output data sets from these tables by using the ODS OUTPUT statement. This can be useful, for example, when you want to create a report to summarize multiple PROC OPTLP runs. The output data sets corresponding to the preceding output are shown in Figure 12.3, where you can also find (at the row following the heading of each data set in display) the variable names that are used in the table definition (template) of each table.

Figure 12.3 ODS Output Data Sets

Problem Summary

Obs	Label1	cValue1	nValue1
1	Problem Name	ADLITTLE	.
2	Objective Sense	Minimization	.
3	Objective Function	.Z....	.
4	RHS	ZZZZ0001	.
5			.
6	Number of Variables	97	97.000000
7	Bounded Above	0	0
8	Bounded Below	97	97.000000
9	Bounded Above and Below	0	0
10	Free	0	0
11	Fixed	0	0
12			.
13	Number of Constraints	56	56.000000
14	LE (<=)	40	40.000000
15	EQ (=)	15	15.000000
16	GE (>=)	1	1.000000
17	Range	0	0
18			.
19	Constraint Coefficients	383	383.000000

Solution Summary

Obs	Label1	cValue1	nValue1
1	Solver	LP	.
2	Algorithm	Dual Simplex	.
3	Objective Function	.Z....	.
4	Solution Status	Optimal	.
5	Objective Value	225494.96316	225495
6			.
7	Primal Infeasibility	2.842171E-13	2.842171E-13
8	Dual Infeasibility	4.263256E-13	4.263256E-13
9	Bound Infeasibility	0	0
10			.
11	Iterations	73	73.000000
12	Presolve Time	0.00	0
13	Solution Time	0.00	0

Problem Statistics

Optimizers can encounter difficulty when solving poorly formulated models. Information about data magnitude provides a simple gauge to determine how well a model is formulated. For example, a model whose constraint matrix contains one very large entry (on the order of 10^9) can cause difficulty when the remaining entries are single-digit numbers. The PRINTLEVEL=2 option in the OPTLP procedure causes the ODS table ProblemStatistics to be generated. This table provides basic data magnitude information that enables you to improve the formulation of your models.

The example output in Figure 12.4 demonstrates the contents of the ODS table ProblemStatistics.

Figure 12.4 ODS Table ProblemStatistics

The OPTLP Procedure

Problem Statistics	
Number of Constraint Matrix Nonzeros	8
Maximum Constraint Matrix Coefficient	3
Minimum Constraint Matrix Coefficient	1
Average Constraint Matrix Coefficient	1.875
Number of Objective Nonzeros	3
Maximum Objective Coefficient	4
Minimum Objective Coefficient	2
Average Objective Coefficient	3
Number of RHS Nonzeros	3
Maximum RHS	7
Minimum RHS	4
Average RHS	5.3333333333
Maximum Number of Nonzeros per Column	3
Minimum Number of Nonzeros per Column	2
Average Number of Nonzeros per Column	2.67
Maximum Number of Nonzeros per Row	3
Minimum Number of Nonzeros per Row	2
Average Number of Nonzeros per Row	2.67

Irreducible Infeasible Set

For a linear programming problem, an irreducible infeasible set (IIS) is an infeasible subset of constraints and variable bounds that will become feasible if any single constraint or variable bound is removed. It is possible to have more than one IIS in an infeasible LP. Identifying an IIS can help isolate the structural infeasibility in an LP.

The presolver in the OPTLP procedure can detect infeasibility, but it identifies only the variable bound or constraint that triggers the infeasibility.

The IIS=ON option directs the OPTLP procedure to search for an IIS in a specified LP. The OPTLP procedure does not apply the presolver to the problem during the IIS search. If PROC OPTLP detects an IIS, it first outputs the IIS to the data sets that are specified by the PRIMALOUT= and DUALOUT= options, and then it stops. The number of iterations that are reported in the macro variable and the ODS table is the total number of simplex iterations. This total includes the initial LP solve and all subsequent iterations during the constraint deletion phase.

The IIS= option can add special values to the _STATUS_ variables in the output data sets. (For more information, see the section "Data Input and Output" on page 570.) For constraints, a status of "I_L", "I_U", or "I_F" indicates that the "GE" ($\geq$), "LE" ($\leq$), or "EQ" (=) constraint, respectively, is part of the IIS. For range constraints, a status of "I_L" or "I_U" indicates that the lower or upper bound of the constraint, respectively, is needed for the IIS, and "I_F" indicates that the bounds in the constraint are conflicting. For variables, a status of "I_L", "I_U", or "I_F" indicates that the lower, upper, or both bounds of the variable, respectively, are needed for the IIS. From this information, you can identify both the names of the constraints (variables) in the IIS and the corresponding bound where infeasibility occurs.

Making any one of the constraints or variable bounds in the IIS nonbinding removes the infeasibility from the IIS. In some cases, changing a right-hand side or bound by a finite amount removes the infeasibility. However, the only way to guarantee removal of the infeasibility is to set the appropriate right-hand side or bound to ∞ or $-\infty$. Because it is possible for an LP to have multiple irreducible infeasible sets, simply removing the infeasibility from one set might not make the entire problem feasible. To make the entire problem feasible, you can specify IIS=ON and rerun PROC OPTLP after removing the infeasibility from an IIS. Repeating this process until the LP solver no longer detects an IIS results in a feasible problem. This approach to infeasibility repair can produce different end problems depending on which right-hand sides and bounds you choose to relax.

Changing different constraints and bounds can require considerably different changes to the MPS-format SAS data set. For example, if you use the default lower bound of 0 for a variable but you want to relax the lower bound to $-\infty$, you might need to add an MI row to the BOUNDS section of the data set. For more information about changing variable and constraint bounds, see Chapter 17, "The MPS-Format SAS Data Set."

The IIS= option in PROC OPTLP uses two different methods to identify an IIS:

1. Based on the result of the initial solve, the *sensitivity filter* removes several constraints and variable bounds immediately while still maintaining infeasibility. This phase is quick and dramatically reduces the size of the IIS.

2. Next, the *deletion filter* removes each remaining constraint and variable bound one by one to check which of them are needed to obtain an infeasible system. This second phase is more time consuming, but it ensures that the IIS set that PROC OPTLP returns is indeed irreducible. The progress of the deletion filter is reported at regular intervals. Occasionally, the sensitivity filter might be called again during the deletion filter to improve performance.

See Example 12.7 for an example that demonstrates the use of the IIS= option in locating and removing infeasibilities. You can find more details about IIS algorithms in Chinneck (2008).

Macro Variable _OROPTLP_

The OPTLP procedure defines a macro variable named _OROPTLP_. This variable contains a character string that indicates the status of the OPTLP procedure upon termination. The various terms of the variable are interpreted as follows.

STATUS

indicates the solver status at termination. It can take one of the following values:

OK	The procedure terminated normally.
SYNTAX_ERROR	Incorrect syntax was used.
DATA_ERROR	The input data were inconsistent.
OUT_OF_MEMORY	Insufficient memory was allocated to the procedure.
IO_ERROR	A problem occurred in reading or writing data.
ERROR	The status cannot be classified into any of the preceding categories.

ALGORITHM

indicates the algorithm that produces the solution data in the macro variable. This term appears only when STATUS=OK. It can take one of the following values:

PS	The primal simplex algorithm produced the solution data.
DS	The dual simplex algorithm produced the solution data.
NS	The network simplex algorithm produced the solution data.
IP	The interior point algorithm produced the solution data.
DECOMP	The decomposition algorithm produced the solution data.

When you run algorithms concurrently (ALGORITHM=CON), this term indicates which algorithm is the first to terminate.

SOLUTION_STATUS

indicates the solution status at termination. It can take one of the following values:

OPTIMAL	The solution is optimal.
CONDITIONAL_OPTIMAL	The solution is optimal, but some infeasibilities (primal, dual or bound) exceed tolerances due to scaling or preprocessing.
FEASIBLE	The problem is feasible.
INFEASIBLE	The problem is infeasible.
UNBOUNDED	The problem is unbounded.
INFEASIBLE_OR_UNBOUNDED	The problem is infeasible or unbounded.
ITERATION_LIMIT_REACHED	The maximum allowable number of iterations was reached.
TIME_LIMIT_REACHED	The solver reached its execution time limit.
ABORTED	The solver was interrupted externally.
FAILED	The solver failed to converge, possibly due to numerical issues.

OBJECTIVE

indicates the objective value obtained by the solver at termination.

PRIMAL_INFEASIBILITY
indicates, for the primal simplex and dual simplex algorithms, the maximum (absolute) violation of the primal constraints by the primal solution. For the interior point algorithm, this term indicates the relative violation of the primal constraints by the primal solution.

DUAL_INFEASIBILITY
indicates, for the primal simplex and dual simplex algorithms, the maximum (absolute) violation of the dual constraints by the dual solution. For the interior point algorithm, this term indicates the relative violation of the dual constraints by the dual solution.

BOUND_INFEASIBILITY
indicates, for the primal simplex and dual simplex algorithms, the maximum (absolute) violation of the lower or upper bounds (or both) by the primal solution. For the interior point algorithm, this term indicates the relative violation of the lower or upper bounds (or both) by the primal solution.

DUALITY_GAP
indicates the (relative) duality gap. This term appears only if the interior point algorithm is used.

COMPLEMENTARITY
indicates the (absolute) complementarity. This term appears only if the interior point algorithm is used.

ITERATIONS
indicates the number of iterations taken to solve the problem. When the network simplex algorithm is used, this term indicates the number of network simplex iterations taken to solve the network relaxation. When crossover is enabled, this term indicates the number of interior point iterations taken to solve the problem.

ITERATIONS2
indicates the number of simplex iterations performed by the secondary algorithm. In network simplex, the secondary algorithm is selected automatically, unless a value has been specified for the ALGORITHM2= option. When crossover is enabled, the secondary algorithm is selected automatically. This term appears only if the network simplex algorithm is used or if crossover is enabled.

PRESOLVE_TIME
indicates the time (in seconds) used in preprocessing.

SOLUTION_TIME
indicates the time (in seconds) taken to solve the problem, including preprocessing time.

NOTE: The time reported in PRESOLVE_TIME and SOLUTION_TIME is either CPU time or real time. The type is determined by the TIMETYPE= option.

When SOLUTION_STATUS has a value of OPTIMAL, CONDITIONAL_OPTIMAL, ITERATION_LIMIT_REACHED, or TIME_LIMIT_REACHED, all terms of the _OROPTLP_ macro variable are present; for other values of SOLUTION_STATUS, some terms do not appear.

Examples: OPTLP Procedure

Example 12.1: Oil Refinery Problem

Consider an oil refinery scenario. A step in refining crude oil into finished oil products involves a distillation process that splits crude into various streams. Suppose there are three types of crude available: Arabian light (a_l), Arabian heavy (a_h), and Brega (br). These crudes are distilled into light naphtha (na_l), intermediate naphtha (na_i), and heating oil (h_o). These in turn are blended into two types of jet fuel. Jet fuel j_1 is made up of 30% intermediate naphtha and 70% heating oil, and jet fuel j_2 is made up of 20% light naphtha and 80% heating oil. What amounts of the three crudes maximize the profit from producing jet fuel (j_1, j_2)? This problem can be formulated as the following linear program:

$$\max \quad -175\text{a\_l} - 165\text{a\_h} - 205\text{br} + 350\text{j\_1} + 350\text{j\_2}$$

subject to

$$\begin{array}{lrcrcrcl}
\text{(napha_l)} & 0.035\,\text{a_l} & + & 0.03\,\text{a_h} & + & 0.045\,\text{br} & = & \text{na_l} \\
\text{(napha_i)} & 0.1\,\text{a_l} & + & 0.075\,\text{a_h} & + & 0.135\,\text{br} & = & \text{na_i} \\
\text{(htg_oil)} & 0.39\,\text{a_l} & + & 0.3\,\text{a_h} & + & 0.43\,\text{br} & = & \text{h_o} \\
\text{(blend1)} & & & 0.3\,\text{j_1} & & & \le & \text{na_i} \\
\text{(blend2)} & & & & & 0.2\,\text{j_2} & \le & \text{na_l} \\
\text{(blend3)} & & & 0.7\,\text{j_1} & + & 0.8\,\text{j_2} & \le & \text{h_o} \\
 & \text{a_l} & & & & & \le & 110 \\
 & & & \text{a_h} & & & \le & 165 \\
 & & & & & \text{br} & \le & 80
\end{array}$$

and

$$\text{a\_l}, \text{a\_h}, \text{br}, \text{na\_1}, \text{na\_i}, \text{h\_o}, \text{j\_1}, \text{j\_2} \ge 0$$

The constraints "blend1" and "blend2" ensure that j_1 and j_2 are made with the specified amounts of na_i and na_l, respectively. The constraint "blend3" is actually the reduced form of the following constraints:

$$\begin{array}{rcrcl}
\text{h_o1} & & & \ge & 0.7\,\text{j_1} \\
 & & \text{h_o2} & \ge & 0.8\,\text{j_2} \\
\text{h_o1} & + & \text{h_o2} & \le & \text{h_o}
\end{array}$$

where h_o1 and h_o2 are dummy variables.

You can use the following SAS code to create the input data set ex1:

```
data ex1;
   input field1 $ field2 $ field3 $ field4 field5 $ field6;
   datalines;
NAME        .         EX1      .     .        .
ROWS        .         .        .     .        .
 N          profit    .        .     .        .
 E          napha_l   .        .     .        .
 E          napha_i   .        .     .        .
 E          htg_oil   .        .     .        .
 L          blend1    .        .     .        .
 L          blend2    .        .     .        .
 L          blend3    .        .     .        .
COLUMNS     .         .        .     .        .
.           a_l       profit   -175  napha_l  .035
.           a_l       napha_i  .100  htg_oil  .390
.           a_h       profit   -165  napha_l  .030
.           a_h       napha_i  .075  htg_oil  .300
.           br        profit   -205  napha_l  .045
.           br        napha_i  .135  htg_oil  .430
.           na_l      napha_l  -1    blend2   -1
.           na_i      napha_i  -1    blend1   -1
.           h_o       htg_oil  -1    blend3   -1
.           j_1       profit   350   blend1   .3
.           j_1       blend3   .7    .        .
.           j_2       profit   350   blend2   .2
.           j_2       blend3   .8    .        .
BOUNDS      .         .        .     .        .
UP          .         a_l      110   .        .
UP          .         a_h      165   .        .
UP          .         br       80    .        .
ENDATA      .         .        .     .        .
;
```

You can use the following call to PROC OPTLP to solve the LP problem:

```
proc optlp data=ex1
   objsense  = max
   algorithm = primal
   primalout = ex1pout
   dualout   = ex1dout
   logfreq   = 1;
run;
%put &_OROPTLP_;
```

Note that the OBJSENSE=MAX option is used to indicate that the objective function is to be maximized.

The primal and dual solutions are displayed in Output 12.1.1.

Output 12.1.1 Example 1: Primal and Dual Solution Output

Primal Solution

Obs	Objective Function ID	RHS ID	Variable Name	Variable Type	Objective Coefficient	Lower Bound	Upper Bound	Variable Value	Variable Status	Reduced Cost
1	profit		a_l	D	-175	0	110	110.000	U	10.2083
2	profit		a_h	D	-165	0	165	0.000	L	-22.8125
3	profit		br	D	-205	0	80	80.000	U	2.8125
4	profit		na_l	N	0	0	1.7977E308	7.450	B	0.0000
5	profit		na_i	N	0	0	1.7977E308	21.800	B	0.0000
6	profit		h_o	N	0	0	1.7977E308	77.300	B	0.0000
7	profit		j_1	N	350	0	1.7977E308	72.667	B	0.0000
8	profit		j_2	N	350	0	1.7977E308	33.042	B	0.0000

Dual Solution

Obs	Objective Function ID	RHS ID	Constraint Name	Constraint Type	Constraint RHS	Constraint Lower Bound	Constraint Upper Bound	Dual Variable Value	Constraint Status	Constraint Activity
1	profit		napha_l	E	0	.	.	0.000	L	0.00000
2	profit		napha_i	E	0	.	.	-145.833	U	0.00000
3	profit		htg_oil	E	0	.	.	-437.500	U	0.00000
4	profit		blend1	L	0	.	.	145.833	L	0.00000
5	profit		blend2	L	0	.	.	0.000	B	-0.84167
6	profit		blend3	L	0	.	.	437.500	L	-0.00000

The progress of the solution is printed to the log as follows.

Output 12.1.2 Log: Solution Progress

```
NOTE: The problem EX1 has 8 variables (0 free, 0 fixed).
NOTE: The problem has 6 constraints (3 LE, 3 EQ, 0 GE, 0 range).
NOTE: The problem has 19 constraint coefficients.
WARNING: The objective sense has been changed to maximization.
NOTE: The LP presolver value AUTOMATIC is applied.
NOTE: The LP presolver removed 3 variables and 3 constraints.
NOTE: The LP presolver removed 6 constraint coefficients.
NOTE: The presolved problem has 5 variables, 3 constraints, and 13 constraint
      coefficients.
NOTE: The LP solver is called.
NOTE: The Primal Simplex algorithm is used.
                           Objective                 Entering      Leaving
      Phase Iteration        Value         Time      Variable      Variable
       P 2           1    0.000000E+00        0         j_1          blend1 (S)
       P 2           2    2.022784E-03        0         j_2          blend2 (S)
       P 2           3    3.902347E-03        0          br          blend3 (S)
       P 2           4    4.025073E-03        0         a_l            a_l
       P 2           5    1.202248E+03        0      blend2 (S)         br
       P 2           6    1.347921E+03        0
       D 2           7    1.347917E+03        0
NOTE: Optimal.
NOTE: Objective = 1347.9166667.
NOTE: The Primal Simplex solve time is 0.01 seconds.
NOTE: The data set WORK.EX1POUT has 8 observations and 10 variables.
NOTE: The data set WORK.EX1DOUT has 6 observations and 10 variables.
```

Note that the %put statement immediately after the OPTLP procedure prints value of the macro variable _OROPTLP_ to the log as follows.

Output 12.1.3 Log: Value of the Macro Variable _OROPTLP_

```
STATUS=OK ALGORITHM=PS SOLUTION_STATUS=OPTIMAL OBJECTIVE=1347.9166667
PRIMAL_INFEASIBILITY=0 DUAL_INFEASIBILITY=0 BOUND_INFEASIBILITY=0 ITERATIONS=7
PRESOLVE_TIME=0.00 SOLUTION_TIME=0.01
```

The value briefly summarizes the status of the OPTLP procedure upon termination.

Example 12.2: Using the Interior Point Algorithm

You can also solve the oil refinery problem described in Example 12.1 by using the interior point algorithm. You can create the input data set from an external MPS-format flat file by using the SAS macro %MPS2SASD or SAS DATA step code, both of which are described in "Getting Started: OPTLP Procedure" on page 561. You can use the following SAS code to solve the problem:

```
proc optlp data=ex1
  objsense  = max
  algorithm = ip
  primalout = ex1ipout
  dualout   = ex1idout
  logfreq   = 1;
run;
```

The optimal solution is displayed in Output 12.2.1.

Output 12.2.1 Interior Point Algorithm: Primal Solution Output

Primal Solution

Obs	Objective Function ID	RHS ID	Variable Name	Variable Type	Objective Coefficient	Lower Bound	Upper Bound	Variable Value	Variable Status	Reduced Cost
1	profit		a_l	D	-175	0	110	110.000	U	10.2083
2	profit		a_h	D	-165	0	165	0.000	L	-22.8125
3	profit		br	D	-205	0	80	80.000	U	2.8125
4	profit		na_l	N	0	0	1.7977E308	7.450	B	0.0000
5	profit		na_i	N	0	0	1.7977E308	21.800	B	0.0000
6	profit		h_o	N	0	0	1.7977E308	77.300	B	0.0000
7	profit		j_1	N	350	0	1.7977E308	72.667	B	0.0000
8	profit		j_2	N	350	0	1.7977E308	33.042	B	0.0000

The iteration log is displayed in Output 12.2.2.

Output 12.2.2 Log: Solution Progress

```
NOTE: The problem EX1 has 8 variables (0 free, 0 fixed).
NOTE: The problem has 6 constraints (3 LE, 3 EQ, 0 GE, 0 range).
NOTE: The problem has 19 constraint coefficients.
WARNING: The objective sense has been changed to maximization.
NOTE: The LP presolver value AUTOMATIC is applied.
NOTE: The LP presolver removed 3 variables and 3 constraints.
NOTE: The LP presolver removed 6 constraint coefficients.
NOTE: The presolved problem has 5 variables, 3 constraints, and 13 constraint
      coefficients.
NOTE: The LP solver is called.
NOTE: The Interior Point algorithm is used.
NOTE: The deterministic parallel mode is enabled.
NOTE: The Interior Point algorithm is using up to 2 threads.
                                          Primal        Bound         Dual
      Iter  Complement Duality Gap        Infeas       Infeas       Infeas    Time
         0  3.3251E+01  2.3083E+00  8.6736E+00  1.1031E-01  4.6594E-02      0
         1  1.0404E+01  6.5073E+00  1.9572E+00  2.4891E-02  1.9783E-02      0
         2  9.3768E+00  7.3001E-01  1.8505E+00  2.3535E-02  1.9154E-02      0
         3  1.9049E+00  2.1685E-01  1.8505E-02  2.3535E-04  4.0512E-03      0
         4  7.3941E-01  9.7599E-02  8.9080E-03  1.1329E-04  7.4274E-04      0
         5  1.3936E-02  1.5210E-03  8.9080E-05  1.1329E-06  1.9471E-05      0
         6  1.3942E-04  1.5202E-05  8.9114E-07  1.1333E-08  1.9471E-07      0
         7  3.7730E-04  1.8243E-07  2.6709E-09  1.7235E-10  4.2406E-07      0
         8  0.0000E+00  4.3239E-09  1.3361E-08  2.8326E-13  1.6412E-08      0
NOTE: The Interior Point solve time is 0.00 seconds.
NOTE: The CROSSOVER option is enabled.
NOTE: The crossover basis contains 0 primal and 0 dual superbasic variables.
                           Objective
      Phase Iteration          Value         Time
       P C          1    0.000000E+00           0
       P 2          2    1.347917E+03           0
       D 2          3    1.347917E+03           0
NOTE: The Crossover time is 0.00 seconds.
NOTE: Optimal.
NOTE: Objective = 1347.9166667.
NOTE: The data set WORK.EX1IPOUT has 8 observations and 10 variables.
NOTE: The data set WORK.EX1IDOUT has 6 observations and 10 variables.
```

Example 12.3: The Diet Problem

Consider the problem of diet optimization. There are six different foods: bread, milk, cheese, potato, fish, and yogurt. The cost and nutrition values per unit are displayed in Table 12.3.

Table 12.3 Cost and Nutrition Values

	Bread	Milk	Cheese	Potato	Fish	Yogurt
Cost	2.0	3.5	8.0	1.5	11.0	1.0

Table 12.3 (continued)

	Bread	Milk	Cheese	Potato	Fish	Yogurt
Protein, g	4.0	8.0	7.0	1.3	8.0	9.2
Fat, g	1.0	5.0	9.0	0.1	7.0	1.0
Carbohydrates, g	15.0	11.7	0.4	22.6	0.0	17.0
Calories	90	120	106	97	130	180

The objective is to find a minimum-cost diet that contains at least 300 calories, not more than 10 grams of protein, not less than 10 grams of carbohydrates, and not less than 8 grams of fat. In addition, the diet should contain at least 0.5 unit of fish and no more than 1 unit of milk.

You can use the following SAS code to create the MPS-format input data set:

```
data ex3;
   input field1 $ field2 $ field3 $ field4 field5 $ field6;
   datalines;
NAME        .          EX3      .      .         .
ROWS        .          .        .      .         .
 N          diet       .        .      .         .
 G          calories   .        .      .         .
 L          protein    .        .      .         .
 G          fat        .        .      .         .
 G          carbs      .        .      .         .
COLUMNS     .          .        .      .         .
.           br         diet     2      calories  90
.           br         protein  4      fat       1
.           br         carbs    15     .         .
.           mi         diet     3.5    calories  120
.           mi         protein  8      fat       5
.           mi         carbs    11.7   .         .
.           ch         diet     8      calories  106
.           ch         protein  7      fat       9
.           ch         carbs    .4     .         .
.           po         diet     1.5    calories  97
.           po         protein  1.3    fat       .1
.           po         carbs    22.6   .         .
.           fi         diet     11     calories  130
.           fi         protein  8      fat       7
.           fi         carbs    0      .         .
.           yo         diet     1      calories  180
.           yo         protein  9.2    fat       1
.           yo         carbs    17     .         .
RHS         .          .        .      .         .
.           .          calories 300    protein   10
.           .          fat      8      carbs     10
BOUNDS      .          .        .      .         .
UP          .          mi       1      .         .
LO          .          fi       .5     .         .
ENDATA      .          .        .      .         .
;
```

You can solve the diet problem by using PROC OPTLP as follows:

```
proc optlp data=ex3
   presolver = none
   algorithm = ps
   primalout = ex3pout
   dualout   = ex3dout
   logfreq   = 1;
run;
```

The solution summary and the optimal primal solution are displayed in Output 12.3.1.

Output 12.3.1 Diet Problem: Solution Summary and Optimal Primal Solution

Solution Summary

Obs	Label1	cValue1	nValue1
1	Solver	LP	.
2	Algorithm	Primal Simplex	.
3	Objective Function	diet	.
4	Solution Status	Optimal	.
5	Objective Value	12.081337881	12.081338
6			.
7	Primal Infeasibility	0	0
8	Dual Infeasibility	0	0
9	Bound Infeasibility	0	0
10			.
11	Iterations	6	6.000000
12	Presolve Time	0.00	0
13	Solution Time	0.00	0

Primal Solution

Obs	Objective Function ID	RHS ID	Variable Name	Variable Type	Objective Coefficient	Lower Bound	Upper Bound	Variable Value	Variable Status	Reduced Cost
1	diet		br	N	2.0	0.0	1.7977E308	0.00000	L	1.19066
2	diet		mi	D	3.5	0.0	1	0.05360	B	0.00000
3	diet		ch	N	8.0	0.0	1.7977E308	0.44950	B	0.00000
4	diet		po	N	1.5	0.0	1.7977E308	1.86517	B	0.00000
5	diet		fi	O	11.0	0.5	1.7977E308	0.50000	L	5.15641
6	diet		yo	N	1.0	0.0	1.7977E308	0.00000	L	1.10849

The cost of the optimal diet is 12.08 units.

Example 12.4: Reoptimizing after Modifying the Objective Function

Using the diet problem described in Example 12.3, this example illustrates how to reoptimize an LP problem after modifying the objective function.

Assume that the optimal solution of the diet problem is found and the optimal solutions are stored in the data sets ex3pout and ex3dout.

Suppose the cost of cheese increases from 8 to 10 per unit and the cost of fish decreases from 11 to 7 per serving unit. The COLUMNS section in the input data set ex3 is updated (and the data set is saved as ex4) as follows:

```
COLUMNS      .          .         .      .          .
   ...
.            ch         diet      10     calories   106
   ...
.            fi         diet      7      calories   130
   ...
RHS          .          .         .      .          .
   ...

ENDATA
;
```

You can use the following DATA step to create the data set ex4:

```
data ex4;
   input field1 $ field2 $ field3 $ field4 field5 $ field6;
   datalines;
NAME         .          EX4       .      .          .
ROWS         .          .         .      .          .
 N           diet       .         .      .          .
 G           calories   .         .      .          .
 L           protein    .         .      .          .
 G           fat        .         .      .          .
 G           carbs      .         .      .          .
COLUMNS      .          .         .      .          .
.            br         diet      2      calories   90
.            br         protein   4      fat        1
.            br         carbs     15     .          .
.            mi         diet      3.5    calories   120
.            mi         protein   8      fat        5
.            mi         carbs     11.7   .          .
.            ch         diet      10     calories   106
.            ch         protein   7      fat        9
.            ch         carbs     .4     .          .
.            po         diet      1.5    calories   97
.            po         protein   1.3    fat        .1
.            po         carbs     22.6   .          .
.            fi         diet       7     calories   130
.            fi         protein   8      fat        7
```

```
.           fi          carbs     0        .          .
.           yo          diet      1        calories   180
.           yo          protein   9.2      fat        1
.           yo          carbs     17       .          .
RHS         .           .         .        .          .
.           .           calories  300      protein    10
.           .           fat       8        carbs      10
BOUNDS      .           .         .        .          .
UP          .           mi        1        .          .
LO          .           fi        .5       .          .
ENDATA      .           .         .        .          .
;
```

You can use the BASIS=WARMSTART option (and the ex3pout and ex3dout data sets from Example 12.3) in the following call to PROC OPTLP to solve the modified problem:

```
proc optlp data=ex4
   presolver = none
   basis     = warmstart
   primalin  = ex3pout
   dualin    = ex3dout
   algorithm = primal
   primalout = ex4pout
   dualout   = ex4dout
   logfreq   = 1;
run;
```

The following iteration log indicates that it takes the primal simplex algorithm no extra iterations to solve the modified problem by using BASIS=WARMSTART, since the optimal solution to the LP problem in Example 12.3 remains optimal after the objective function is changed.

Output 12.4.1 Iteration Log

```
NOTE: The problem EX4 has 6 variables (0 free, 0 fixed).
NOTE: The problem has 4 constraints (1 LE, 0 EQ, 3 GE, 0 range).
NOTE: The problem has 23 constraint coefficients.
NOTE: The LP presolver value NONE is applied.
NOTE: The LP solver is called.
NOTE: The Primal Simplex algorithm is used.
                           Objective                   Entering      Leaving
      Phase Iteration          Value          Time     Variable      Variable
       P 2          1    1.098034E+01          0
NOTE: Optimal.
NOTE: Objective = 10.980335514.
NOTE: The Primal Simplex solve time is 0.00 seconds.
NOTE: The data set WORK.EX4POUT has 6 observations and 10 variables.
NOTE: The data set WORK.EX4DOUT has 4 observations and 10 variables.
```

Note that the primal simplex algorithm is preferred because the primal solution to the original LP is still feasible for the modified problem in this case.

Example 12.5: Reoptimizing after Modifying the Right-Hand Side

You can also modify the right-hand side of your problem and use the BASIS=WARMSTART option to obtain an optimal solution more quickly. Since the dual solution to the original LP is still feasible for the modified problem in this case, the dual simplex algorithm is preferred. This case is illustrated by using the same diet problem as in Example 12.3. Assume that you now need a diet that supplies at least 150 calories. The RHS section in the input data set ex3 is updated (and the data set is saved as ex5) as follows:

```
    ...
RHS         .          .        .     .        .
.           .          calories 150   protein  10
.           .          fat      8     carbs    10
BOUNDS      .          .        .     .        .
    ...
```

You can use the following DATA step to create the data set ex5:

```
data ex5;
   input field1 $ field2 $ field3 $ field4 field5 $ field6;
   datalines;
NAME        .          EX5      .     .        .
ROWS        .          .        .     .        .
 N          diet       .        .     .        .
 G          calories   .        .     .        .
 L          protein    .        .     .        .
 G          fat        .        .     .        .
 G          carbs      .        .     .        .
COLUMNS     .          .        .     .        .
.           br         diet     2     calories 90
.           br         protein  4     fat      1
.           br         carbs    15    .        .
.           mi         diet     3.5   calories 120
.           mi         protein  8     fat      5
.           mi         carbs    11.7  .        .
.           ch         diet     8     calories 106
.           ch         protein  7     fat      9
.           ch         carbs    .4    .        .
.           po         diet     1.5   calories 97
.           po         protein  1.3   fat      .1
.           po         carbs    22.6  .        .
.           fi         diet     11    calories 130
.           fi         protein  8     fat      7
.           fi         carbs    0     .        .
.           yo         diet     1     calories 180
.           yo         protein  9.2   fat      1
.           yo         carbs    17    .        .
RHS         .          .        .     .        .
.           .          calories 150   protein  10
.           .          fat      8     carbs    10
BOUNDS      .          .        .     .        .
UP          .          mi       1     .        .
LO          .          fi       .5    .        .
ENDATA      .          .        .     .        .
;
```

You can use the BASIS=WARMSTART option in the following call to PROC OPTLP to solve the modified problem:

```
proc optlp data=ex5
   presolver = none
   basis     = warmstart
   primalin  = ex3pout
   dualin    = ex3dout
   algorithm = dual
   primalout = ex5pout
   dualout   = ex5dout
   logfreq   = 1;
run;
```

Note that the dual simplex algorithm is preferred because the dual solution to the last solved LP is still feasible for the modified problem in this case.

The following iteration log indicates that it takes the dual simplex algorithm just one more phase II iteration to solve the modified problem by using BASIS=WARMSTART.

Output 12.5.1 Iteration Log

```
NOTE: The problem EX5 has 6 variables (0 free, 0 fixed).
NOTE: The problem has 4 constraints (1 LE, 0 EQ, 3 GE, 0 range).
NOTE: The problem has 23 constraint coefficients.
NOTE: The LP presolver value NONE is applied.
NOTE: The LP solver is called.
NOTE: The Dual Simplex algorithm is used.
                           Objective                Entering      Leaving
      Phase Iteration        Value          Time     Variable      Variable
       D 2          1     8.813205E+00         0    calories (S)     carbs (S)
       D 2          2     9.174413E+00         0
NOTE: Optimal.
NOTE: Objective = 9.1744131985.
NOTE: The Dual Simplex solve time is 0.00 seconds.
NOTE: The data set WORK.EX5POUT has 6 observations and 10 variables.
NOTE: The data set WORK.EX5DOUT has 4 observations and 10 variables.
```

Compare this with the following call to PROC OPTLP:

```
proc optlp data=ex5
   presolver = none
   algorithm = dual
   logfreq   = 1;
run;
```

This call to PROC OPTLP solves the modified problem "from scratch" (without using the BASIS=WARMSTART option) and produces the following iteration log.

Output 12.5.2 Iteration Log

```
NOTE: The problem EX5 has 6 variables (0 free, 0 fixed).
NOTE: The problem has 4 constraints (1 LE, 0 EQ, 3 GE, 0 range).
NOTE: The problem has 23 constraint coefficients.
NOTE: The LP presolver value NONE is applied.
NOTE: The LP solver is called.
NOTE: The Dual Simplex algorithm is used.
                          Objective                   Entering      Leaving
      Phase Iteration        Value           Time      Variable      Variable
       D 2          1     5.500000E+00         0          mi             fat (S)
       D 2          2     8.650000E+00         0          ch         protein (S)
       D 2          3     8.925676E+00         0          po           carbs (S)
       D 2          4     9.174413E+00         0
NOTE: Optimal.
NOTE: Objective = 9.1744131985.
NOTE: The Dual Simplex solve time is 0.00 seconds.
```

It is clear that using the BASIS=WARMSTART option saves computation time. For larger or more complex examples, the benefits of using this option are more pronounced.

Example 12.6: Reoptimizing after Adding a New Constraint

Assume that after solving the diet problem in Example 12.3 you need to add a new constraint on sodium intake of no more than 550 mg/day for adults. The updated nutrition data are given in Table 12.4.

Table 12.4 Updated Cost and Nutrition Values

	Bread	Milk	Cheese	Potato	Fish	Yogurt
Cost	2.0	3.5	8.0	1.5	11.0	1.0
Protein, g	4.0	8.0	7.0	1.3	8.0	9.2
Fat, g	1.0	5.0	9.0	0.1	7.0	1.0
Carbohydrates, g	15.0	11.7	0.4	22.6	0.0	17.0
Calories, Cal	90	120	106	97	130	180
sodium, mg	148	122	337	186	56	132

The input data set ex3 is updated (and the data set is saved as ex6) as follows:

```
/* added a new constraint to the diet problem */
data ex6;
   input field1 $ field2 $ field3 $ field4 field5 $ field6;
   datalines;
NAME        .          EX6      .     .         .
ROWS        .          .        .     .         .
 N          diet       .        .     .         .
 G          calories   .        .     .         .
 L          protein    .        .     .         .
 G          fat        .        .     .         .
```

```
 G          carbs      .         .     .         .
 L          sodium     .         .     .         .
COLUMNS     .          .         .     .         .
.           br         diet      2     calories  90
.           br         protein   4     fat       1
.           br         carbs     15    sodium    148
.           mi         diet      3.5   calories  120
.           mi         protein   8     fat       5
.           mi         carbs     11.7  sodium    122
.           ch         diet      8     calories  106
.           ch         protein   7     fat       9
.           ch         carbs     .4    sodium    337
.           po         diet      1.5   calories  97
.           po         protein   1.3   fat       .1
.           po         carbs     22.6  sodium    186
.           fi         diet      11    calories  130
.           fi         protein   8     fat       7
.           fi         carbs     0     sodium    56
.           yo         diet      1     calories  180
.           yo         protein   9.2   fat       1
.           yo         carbs     17    sodium    132
RHS         .          .         .     .         .
.           .          calories  300   protein   10
.           .          fat       8     carbs     10
.           .          sodium    550   .         .
BOUNDS      .          .         .     .         .
UP          .          mi        1     .         .
LO          .          fi        .5    .         .
ENDATA      .          .         .     .         .
;
```

For the modified problem you can warm start the primal and dual simplex algorithms to get a solution faster. The dual simplex algorithm is preferred because a dual feasible solution can be readily constructed from the optimal solution to the diet optimization problem.

Since there is a new constraint in the modified problem, you can use the following SAS code to create a new DUALIN= data set ex6din with this information:

```
data ex6newcon;
   _ROW_='sodium  '; _STATUS_='A';
   output;
run;

/* create a new DUALIN= data set to include the new constraint */
data ex6din;
   set ex3dout ex6newcon;
run;
```

Note that this step is optional. In this example, you can still use the data set ex3dout as the DUALIN= data set to solve the modified LP problem by using the BASIS=WARMSTART option. PROC OPTLP validates the PRIMALIN= and DUALIN= data sets against the input model. Any new variable (or constraint) in the model is added to the PRIMALIN= (or DUALIN=) data set, and its status is assigned to be ‘A’. The primal and dual simplex algorithms decide its corresponding status internally. Any variable in the PRIMALIN= and DUALIN= data sets but not in the input model is removed.

The _ROW_ and _STATUS_ columns of the DUALIN= data set ex6din are shown in Output 12.6.1.

Output 12.6.1 DUALIN= Data Set with a Newly Added Constraint

Obs	_ROW_	_STATUS_
1	calories	U
2	protein	L
3	fat	U
4	carbs	B
5	sodium	A

The dual simplex algorithm is called to solve the modified diet optimization problem more quickly with the following SAS code:

```
proc optlp data=ex6
   objsense=min
   presolver=none
   algorithm=ds
   primalout=ex6pout
   dualout=ex6dout
   scale=none
   logfreq=1
   basis=warmstart
   primalin=ex3pout
   dualin=ex6din;
run;
```

The optimal primal and dual solutions of the modified problem are displayed in Output 12.6.2.

Output 12.6.2 Primal and Dual Solution Output

Primal Solution

Obs	Objective Function ID	RHS ID	Variable Name	Variable Type	Objective Coefficient	Lower Bound	Upper Bound	Variable Value	Variable Status	Reduced Cost
1	diet		br	N	2.0	0.0	1.7977E308	0.00000	L	1.19066
2	diet		mi	D	3.5	0.0	1	0.05360	B	0.00000
3	diet		ch	N	8.0	0.0	1.7977E308	0.44950	B	0.00000
4	diet		po	N	1.5	0.0	1.7977E308	1.86517	B	0.00000
5	diet		fi	O	11.0	0.5	1.7977E308	0.50000	L	5.15641
6	diet		yo	N	1.0	0.0	1.7977E308	0.00000	L	1.10849

Output 12.6.2 *continued*

Dual Solution

Obs	Objective Function ID	RHS ID	Constraint Name	Constraint Type	Constraint RHS	Constraint Lower Bound	Constraint Upper Bound	Dual Variable Value	Constraint Status	Constraint Activity
1	diet		calories	G	300	.	.	0.02179	U	300.000
2	diet		protein	L	10	.	.	-0.55360	L	10.000
3	diet		fat	G	8	.	.	1.06286	U	8.000
4	diet		carbs	G	10	.	.	0.00000	B	42.960
5	diet		sodium	L	550	.	.	0.00000	B	532.941

The iteration log shown in Output 12.6.3 indicates that it takes the dual simplex algorithm no more iterations to solve the modified problem by using the BASIS=WARMSTART option, since the optimal solution to the original problem remains optimal after one more constraint is added.

Output 12.6.3 Iteration Log

```
NOTE: The problem EX6 has 6 variables (0 free, 0 fixed).
NOTE: The problem has 5 constraints (2 LE, 0 EQ, 3 GE, 0 range).
NOTE: The problem has 29 constraint coefficients.
NOTE: The LP presolver value NONE is applied.
NOTE: The LP solver is called.
NOTE: The Dual Simplex algorithm is used.
                           Objective                Entering      Leaving
      Phase Iteration        Value          Time    Variable      Variable
       D 2          1    1.208134E+01         0
NOTE: Optimal.
NOTE: Objective = 12.081337881.
NOTE: The Dual Simplex solve time is 0.00 seconds.
NOTE: The data set WORK.EX6POUT has 6 observations and 10 variables.
NOTE: The data set WORK.EX6DOUT has 5 observations and 10 variables.
```

Both this example and Example 12.4 illustrate the situation in which the optimal solution does not change after some perturbation of the parameters of the LP problem. The simplex algorithm starts from an optimal solution and quickly verifies the optimality. Usually the optimal solution of the slightly perturbed problem can be obtained after performing relatively small number of iterations if starting with the optimal solution of the original problem. In such cases you can expect a dramatic reduction of computation time, for instance, if you want to solve a large LP problem and a slightly perturbed version of this problem by using the BASIS=WARMSTART option rather than solving both problems from scratch.

Example 12.7: Finding an Irreducible Infeasible Set

This example demonstrates the use of the IIS= option to locate an irreducible infeasible set. Suppose you want to solve a linear program that has the following simple formulation:

$$
\begin{array}{rrcrcrcrl}
\min & x_1 & + & x_2 & + & x_3 & & & \text{(cost)} \\
\text{subject to} & x_1 & + & x_2 & & & \geq & 10 & \text{(con1)} \\
 & x_1 & & & + & x_3 & \leq & 4 & \text{(con2)} \\
4 \leq & & & x_2 & + & x_3 & \leq & 5 & \text{(con3)} \\
 & & & & & x_1, \; x_2 & \geq & 0 & \\
 & & & 0 & \leq & x_3 & \leq & 3 &
\end{array}
$$

The corresponding MPS-format SAS data set is as follows:

```
/* infeasible */
data exiis;
   input field1 $ field2 $ field3 $ field4 field5 $ field6;
datalines;
NAME      .        .        .        .        .
ROWS      .        .        .        .        .
 N        cost     .        .        .        .
 G        con1     .        .        .        .
 L        con2     .        .        .        .
 G        con3     .        .        .        .
COLUMNS   .        .        .        .        .
.         x1       cost     1        con1     1
.         x1       con2     1        .        .
.         x2       cost     1        con1     1
.         x2       con3     1        .        .
.         x3       cost     1        con2     1
.         x3       con3     1        .        .
RHS       .        .        .        .        .
.         rhs      con1     10       con2     4
.         rhs      con3     4        .        .
RANGES    .        .        .        .        .
.         r1       con3     1        .        .
BOUNDS    .        .        .        .        .
UP        b1       x3       3        .        .
ENDATA    .        .        .        .        .
;
```

It is easy to verify that the following three constraints (or rows) and one variable (or column) bound form an IIS for this problem.

$$
\begin{array}{rcrcrcrl}
x_1 & + & x_2 & & & \geq & 10 & \text{(con1)} \\
x_1 & & & + & x_3 & \leq & 4 & \text{(con2)} \\
 & & x_2 & + & x_3 & \leq & 5 & \text{(con3)} \\
 & & & & x_3 & \geq & 0 &
\end{array}
$$

You can use the IIS=ON option to detect this IIS by using the following statements:

```
proc optlp data=exiis
   iis=on
   primalout=iis_vars
   dualout=iis_cons
   logfreq=1;
run;
```

The OPTLP procedure outputs the detected IIS to the data sets specified by the PRIMALOUT= and DUALOUT= options, then stops. The notes shown in Output 12.7.1 are printed to the log.

Output 12.7.1 The IIS= Option: Log

```
NOTE: The problem has 3 variables (0 free, 0 fixed).
NOTE: The problem has 3 constraints (1 LE, 0 EQ, 1 GE, 1 range).
NOTE: The problem has 6 constraint coefficients.
NOTE: The LP solver is called.
NOTE: The IIS= option is enabled.
                           Objective                Entering      Leaving
      Phase Iteration        Value          Time    Variable      Variable
       P 1          1    6.000000E+00          0      con3 (S)       con3 (S)
       P 1          2    5.000000E+00          0        x1           con2 (S)
       P 1          3    1.000000E+00          0
NOTE: Applying the IIS sensitivity filter.
NOTE: The sensitivity filter removed 1 constraints and 3 variable bounds.
NOTE: Applying the IIS deletion filter.
NOTE: Processing constraints.
      Processed      Removed       Time
              0            0          0
              1            0          0
              2            0          0
              3            0          0
NOTE: Processing variable bounds.
      Processed      Removed       Time
              0            0          0
              1            0          0
              2            0          0
              3            0          0
NOTE: The deletion filter removed 0 constraints and 0 variable bounds.
NOTE: The IIS= option found this problem to be infeasible.
NOTE: The IIS= option found an irreducible infeasible set with 1 variables and
      3 constraints.
NOTE: The IIS solve time is 0.00 seconds.
NOTE: The data set WORK.IIS_VARS has 3 observations and 10 variables.
NOTE: The data set WORK.IIS_CONS has 3 observations and 10 variables.
```

The data sets iis_cons and iis_vars are shown in Output 12.7.2.

Output 12.7.2 Identify Rows and Columns in the IIS

Constraints in the IIS

Obs	Objective Function ID	RHS ID	Constraint Name	Constraint Type	Constraint RHS	Constraint Lower Bound	Constraint Upper Bound	Dual Variable Value	Constraint Status	Constraint Activity
1	cost	rhs	con1	G	10	.	.	.	I_L	.
2	cost	rhs	con2	L	4	.	.	.	I_U	.
3	cost	rhs	con3	R	.	4	5	.	I_U	.

Variables in the IIS

Obs	Objective Function ID	RHS ID	Variable Name	Variable Type	Objective Coefficient	Lower Bound	Upper Bound	Variable Value	Variable Status	Reduced Cost
1	cost	rhs	x1	N	1	0	1.7977E308	.		.
2	cost	rhs	x2	N	1	0	1.7977E308	.		.
3	cost	rhs	x3	D	1	0	3	.	I_L	.

The constraint $x_2 + x_3 \leq 5$, which is an element of the IIS, is created by the RANGES section. The original constraint is con3, a "$\geq$" constraint with an RHS value of 4. If you choose to remove the constraint $x_2 + x_3 \leq 5$, you can accomplish this by removing con3 from the RANGES section in the MPS-format SAS data set exiis. Since con3 is the only observation in the section, the identifier observation can also be removed. The modified LP problem is specified in the following SAS statements:

```
 /* dropping con3, feasible */
data exiisf;
   input field1 $ field2 $ field3 $ field4 field5 $ field6;
datalines;
NAME     .        .        .        .        .
ROWS     .        .        .        .        .
 N       cost     .        .        .        .
 G       con1     .        .        .        .
 L       con2     .        .        .        .
 G       con3     .        .        .        .
COLUMNS  .        .        .        .        .
.        x1       cost     1        con1     1
.        x1       con2     1        .        .
.        x2       cost     1        con1     1
.        x2       con3     1        .        .
.        x3       cost     1        con2     1
.        x3       con3     1        .        .
RHS      .        .        .        .        .
.        rhs      con1     10       con2     4
.        rhs      con3     4        .        .
BOUNDS   .        .        .        .        .
UP       b1       x3       3        .        .
ENDATA   .        .        .        .        .
;
```

Since one element of the IIS has been removed, the modified LP problem should no longer contain the infeasible set. Due to the size of this problem, there should be no additional irreducible infeasible sets. You can confirm this by submitting the following SAS statements:

```
proc optlp data=exiisf
   iis=on;
run;
```

The notes shown in Output 12.7.3 are printed to the log.

Output 12.7.3 The IIS= Option: Log

```
NOTE: The problem has 3 variables (0 free, 0 fixed).
NOTE: The problem has 3 constraints (1 LE, 0 EQ, 2 GE, 0 range).
NOTE: The problem has 6 constraint coefficients.
NOTE: The LP solver is called.
NOTE: The IIS= option is enabled.
                           Objective
      Phase Iteration        Value         Time
       P 1          1     1.400000E+01        0
       P 1          3     0.000000E+00        0
NOTE: The IIS= option found this problem to be feasible.
NOTE: The IIS solve time is 0.00 seconds.
NOTE: The data set WORK.EXSS has 8 observations and 3 variables.
```

The solution summary is displayed in Output 12.7.4.

Output 12.7.4 Infeasibility Removed

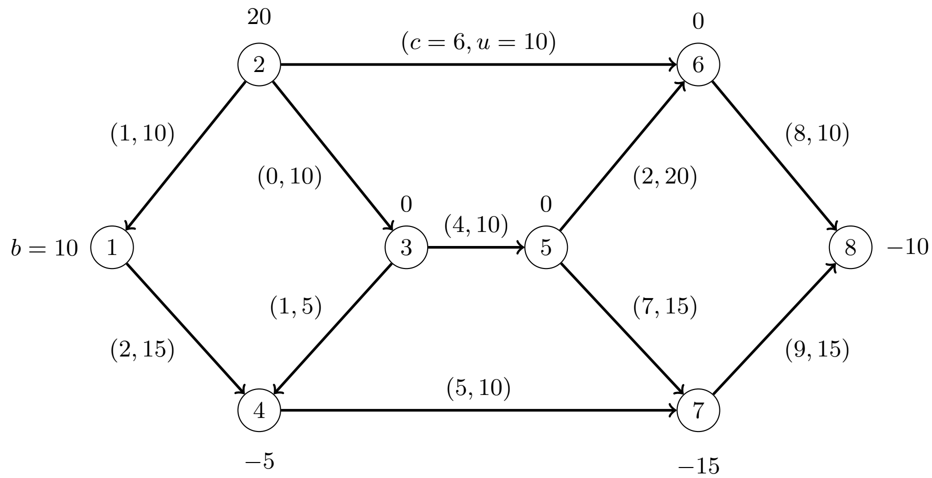
Solution Summary

Obs	Label1	cValue1	nValue1
1	Solver	LP	.
2	Algorithm	IIS	.
3	Objective Function	cost	.
4	Solution Status	Feasible	.
5			.
6	Iterations	3	3.000000
7	Presolve Time	0.00	0
8	Solution Time	0.00	0

Example 12.8: Using the Network Simplex Algorithm

This example demonstrates how to use the network simplex algorithm to find the minimum-cost flow in a directed graph. Consider the directed graph in Figure 12.5, which appears in Ahuja, Magnanti, and Orlin (1993).

Figure 12.5 Minimum-Cost Network Flow Problem: Data



You can use the following SAS statements to create the input data set ex8:

```
data ex8;
   input field1 $8. field2 $13. @25 field3 $13. field4  @53 field5 $13. field6;
   datalines;
NAME     .              .                    .      .                .
ROWS     .              .                    .      .                .
N        obj            .                    .      .                .
E        balance['1']   .                    .      .                .
E        balance['2']   .                    .      .                .
E        balance['3']   .                    .      .                .
E        balance['4']   .                    .      .                .
E        balance['5']   .                    .      .                .
E        balance['6']   .                    .      .                .
E        balance['7']   .                    .      .                .
E        balance['8']   .                    .      .                .
COLUMNS  .              .                    .      .                .
.        x['1','4']     obj                  2      balance['1']     1
.        x['1','4']     balance['4']        -1      .                .
.        x['2','1']     obj                  1      balance['1']    -1
.        x['2','1']     balance['2']         1      .                .
.        x['2','3']     balance['2']         1      balance['3']    -1
.        x['2','6']     obj                  6      balance['2']     1
.        x['2','6']     balance['6']        -1      .                .
.        x['3','4']     obj                  1      balance['3']     1
.        x['3','4']     balance['4']        -1      .                .
.        x['3','5']     obj                  4      balance['3']     1
.        x['3','5']     balance['5']        -1      .                .
.        x['4','7']     obj                  5      balance['4']     1
.        x['4','7']     balance['7']        -1      .                .
.        x['5','6']     obj                  2      balance['5']     1
.        x['5','6']     balance['6']        -1      .                .
.        x['5','7']     obj                  7      balance['5']     1
.        x['5','7']     balance['7']        -1      .                .
.        x['6','8']     obj                  8      balance['6']     1
.        x['6','8']     balance['8']        -1      .                .
```

```
.       x['7','8']      obj             9       balance['7']        1
.       x['7','8']      balance['8']   -1       .                   .
RHS     .               .               .       .                   .
.       .RHS.           balance['1']   10       .                   .
.       .RHS.           balance['2']   20       .                   .
.       .RHS.           balance['4']   -5       .                   .
.       .RHS.           balance['7']  -15       .                   .
.       .RHS.           balance['8']  -10       .                   .
BOUNDS  .               .               .       .                   .
UP      .BOUNDS.        x['1','4']     15       .                   .
UP      .BOUNDS.        x['2','1']     10       .                   .
UP      .BOUNDS.        x['2','3']     10       .                   .
UP      .BOUNDS.        x['2','6']     10       .                   .
UP      .BOUNDS.        x['3','4']      5       .                   .
UP      .BOUNDS.        x['3','5']     10       .                   .
UP      .BOUNDS.        x['4','7']     10       .                   .
UP      .BOUNDS.        x['5','6']     20       .                   .
UP      .BOUNDS.        x['5','7']     15       .                   .
UP      .BOUNDS.        x['6','8']     10       .                   .
UP      .BOUNDS.        x['7','8']     15       .                   .
ENDATA  .               .               .       .                   .
;
```

You can use the following call to PROC OPTLP to find the minimum-cost flow:

```
proc optlp
  presolver  = none
  printlevel = 2
  logfreq    = 1
  data       = ex8
  primalout  = ex8out
  algorithm  = ns;
run;
```

The optimal solution is displayed in Output 12.8.1.

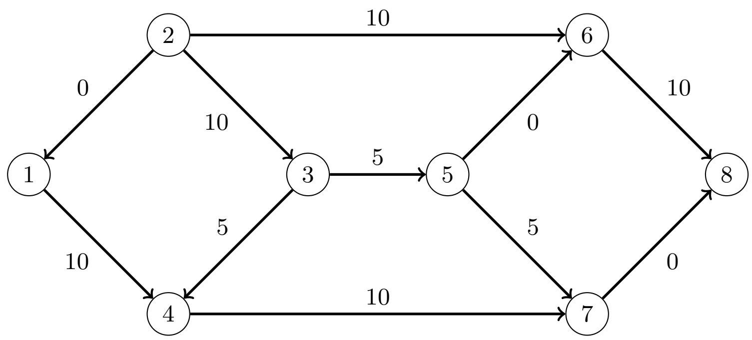
Output 12.8.1 Network Simplex Algorithm: Primal Solution Output

Primal Solution

Obs	Objective Function ID	RHS ID	Variable Name	Variable Type	Objective Coefficient	Lower Bound	Upper Bound	Variable Value	Variable Status	Reduced Cost
1	obj	.RHS.	x['1','4']	D	2	0	15	10	B	0
2	obj	.RHS.	x['2','1']	D	1	0	10	0	L	2
3	obj	.RHS.	x['2','3']	D	0	0	10	10	B	0
4	obj	.RHS.	x['2','6']	D	6	0	10	10	B	0
5	obj	.RHS.	x['3','4']	D	1	0	5	5	B	0
6	obj	.RHS.	x['3','5']	D	4	0	10	5	B	0
7	obj	.RHS.	x['4','7']	D	5	0	10	10	U	-5
8	obj	.RHS.	x['5','6']	D	2	0	20	0	L	0
9	obj	.RHS.	x['5','7']	D	7	0	15	5	B	0
10	obj	.RHS.	x['6','8']	D	8	0	10	10	B	0
11	obj	.RHS.	x['7','8']	D	9	0	15	0	L	6

The optimal solution is represented graphically in Figure 12.6.

Figure 12.6 Minimum-Cost Network Flow Problem: Optimal Solution



The iteration log is displayed in Output 12.8.2.

Output 12.8.2 Log: Solution Progress

```
NOTE: The problem has 11 variables (0 free, 0 fixed).
NOTE: The problem has 8 constraints (0 LE, 8 EQ, 0 GE, 0 range).
NOTE: The problem has 22 constraint coefficients.
NOTE: The LP presolver value NONE is applied.
NOTE: The LP solver is called.
NOTE: The Network Simplex algorithm is used.
NOTE: The network has 8 rows (100.00%), 11 columns (100.00%), and 1 component.
NOTE: The network extraction and setup time is 0.00 seconds.
                        Primal          Primal            Dual
      Iteration      Objective  Infeasibility  Infeasibility      Time
              1   0.000000E+00   2.000000E+01   8.900000E+01      0.00
              2   0.000000E+00   2.000000E+01   8.900000E+01      0.00
              3   5.000000E+00   1.500000E+01   8.400000E+01      0.00
              4   5.000000E+00   1.500000E+01   8.300000E+01      0.00
              5   7.500000E+01   1.500000E+01   8.300000E+01      0.00
              6   7.500000E+01   1.500000E+01   7.900000E+01      0.00
              7   1.300000E+02   1.000000E+01   7.600000E+01      0.00
              8   2.700000E+02   0.000000E+00   0.000000E+00      0.00
NOTE: The Network Simplex solve time is 0.00 seconds.
NOTE: The total Network Simplex solve time is 0.00 seconds.
NOTE: Optimal.
NOTE: Objective = 270.
NOTE: The data set WORK.EX8OUT has 11 observations and 10 variables.
```

References

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (1993). *Network Flows: Theory, Algorithms, and Applications*. Englewood Cliffs, NJ: Prentice-Hall.

Andersen, E. D., and Andersen, K. D. (1995). "Presolving in Linear Programming." *Mathematical Programming* 71:221–245.

Chinneck, J. W. (2008). *Feasibility and Infeasibility in Optimization: Algorithms and Computational Methods.* Vol. 118 of International Series in Operations Research and Management Science. New York: Springer.

Dantzig, G. B. (1963). *Linear Programming and Extensions*. Princeton, NJ: Princeton University Press.

Forrest, J. J., and Goldfarb, D. (1992). "Steepest-Edge Simplex Algorithms for Linear Programming." *Mathematical Programming* 5:1–28.

Gondzio, J. (1997). "Presolve Analysis of Linear Programs Prior to Applying an Interior Point Method." *INFORMS Journal on Computing* 9:73–91.

Harris, P. M. J. (1973). "Pivot Selection Methods in the Devex LP Code." *Mathematical Programming* 57:341–374.

Maros, I. (2003). *Computational Techniques of the Simplex Method*. Boston: Kluwer Academic.

Subject Index

Syntax Index