# Chapter 13
The OPTLP Procedure

## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview: OPTLP Procedure</td>
<td>620</td>
</tr>
<tr>
<td>Getting Started: OPTLP Procedure</td>
<td>621</td>
</tr>
<tr>
<td>Syntax: OPTLP Procedure</td>
<td>623</td>
</tr>
<tr>
<td>Functional Summary</td>
<td>623</td>
</tr>
<tr>
<td>PROC OPTLP Statement</td>
<td>624</td>
</tr>
<tr>
<td>Decomposition Algorithm Statements</td>
<td>630</td>
</tr>
<tr>
<td>Details: OPTLP Procedure</td>
<td>631</td>
</tr>
<tr>
<td>Data Input and Output</td>
<td>631</td>
</tr>
<tr>
<td>Presolve</td>
<td>634</td>
</tr>
<tr>
<td>Pricing Strategies for the Primal and Dual Simplex Algorithms</td>
<td>635</td>
</tr>
<tr>
<td>Warm Start for the Primal and Dual Simplex Algorithms</td>
<td>635</td>
</tr>
<tr>
<td>The Network Simplex Algorithm</td>
<td>635</td>
</tr>
<tr>
<td>The Interior Point Algorithm</td>
<td>636</td>
</tr>
<tr>
<td>Iteration Log for the Primal and Dual Simplex Algorithms</td>
<td>638</td>
</tr>
<tr>
<td>Iteration Log for the Network Simplex Algorithm</td>
<td>639</td>
</tr>
<tr>
<td>Iteration Log for the Interior Point Algorithm</td>
<td>640</td>
</tr>
<tr>
<td>Iteration Log for the Crossover Algorithm</td>
<td>640</td>
</tr>
<tr>
<td>Concurrent LP</td>
<td>641</td>
</tr>
<tr>
<td>Parallel Processing</td>
<td>641</td>
</tr>
<tr>
<td>ODS Tables</td>
<td>641</td>
</tr>
<tr>
<td>Irreducible Infeasible Set</td>
<td>645</td>
</tr>
<tr>
<td>Macro Variable <em>OROPTLP</em></td>
<td>646</td>
</tr>
<tr>
<td>Examples: OPTLP Procedure</td>
<td>649</td>
</tr>
<tr>
<td>Example 13.1: Oil Refinery Problem</td>
<td>649</td>
</tr>
<tr>
<td>Example 13.2: Using the Interior Point Algorithm</td>
<td>654</td>
</tr>
<tr>
<td>Example 13.3: The Diet Problem</td>
<td>655</td>
</tr>
<tr>
<td>Example 13.4: Reoptimizing after Modifying the Objective Function</td>
<td>658</td>
</tr>
<tr>
<td>Example 13.5: Reoptimizing after Modifying the Right-Hand Side</td>
<td>660</td>
</tr>
<tr>
<td>Example 13.6: Reoptimizing after Adding a New Constraint</td>
<td>662</td>
</tr>
<tr>
<td>Example 13.7: Finding an Irreducible Infeasible Set</td>
<td>666</td>
</tr>
<tr>
<td>Example 13.8: Using the Network Simplex Algorithm</td>
<td>669</td>
</tr>
<tr>
<td>References</td>
<td>673</td>
</tr>
</tbody>
</table>
Overview: OPTLP Procedure

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

\[
\begin{align*}
\text{min} & \quad c^T x \\
\text{subject to} & \quad Ax \{\geq, =, \leq\} b \\
& \quad l \leq x \leq u
\end{align*}
\]

where

- \( x \in \mathbb{R}^n \) is the vector of decision variables
- \( A \in \mathbb{R}^{m \times n} \) is the matrix of constraints
- \( c \in \mathbb{R}^n \) is the vector of objective function coefficients
- \( b \in \mathbb{R}^m \) is the vector of constraints’ right-hand sides (RHS)
- \( l \in \mathbb{R}^n \) is the vector of lower bounds on variables
- \( u \in \mathbb{R}^n \) 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 18, “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).
The following example illustrates how you can use the OPTLP procedure to solve linear programs. Suppose you want to solve the following problem:

\[
\begin{align*}
\text{min} & \quad 2x_1 - 3x_2 - 4x_3 \\
\text{subject to} & \quad -2x_2 - 3x_3 \geq -5 \quad (R1) \\
& \quad x_1 + x_2 + 2x_3 \leq 4 \quad (R2) \\
& \quad x_1 + 2x_2 + 3x_3 \leq 7 \quad (R3) \\
& \quad x_1, x_2, x_3 \geq 0
\end{align*}
\]

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

```sas
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:

```sas
%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 891 for details.
You can use the following statement to call the OPTLP procedure:

```plaintext
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 884 for details.

The optimal primal and dual solutions are stored in the data sets expout and exdout, respectively, and are displayed in Figure 13.1.

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

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

**Figure 13.1 Primal and Dual Solution Output**

The **OPTLP Procedure**

Primal Solution

<table>
<thead>
<tr>
<th>Obs</th>
<th>Objective Function</th>
<th>RHS Variable</th>
<th>Type</th>
<th>Variable Name</th>
<th>Objective Coefficient</th>
<th>Lower Bound</th>
<th>Upper Bound</th>
<th>Variable Value</th>
<th>Status</th>
<th>Reduced Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>COST</td>
<td>RHS X1</td>
<td>N</td>
<td></td>
<td>2</td>
<td>0</td>
<td>1.7977E308</td>
<td>0.0</td>
<td>L</td>
<td>2.0</td>
</tr>
<tr>
<td>2</td>
<td>COST</td>
<td>RHS X2</td>
<td>N</td>
<td></td>
<td>-3</td>
<td>0</td>
<td>1.7977E308</td>
<td>2.5</td>
<td>B</td>
<td>0.0</td>
</tr>
<tr>
<td>3</td>
<td>COST</td>
<td>RHS X3</td>
<td>N</td>
<td></td>
<td>-4</td>
<td>0</td>
<td>1.7977E308</td>
<td>0.0</td>
<td>L</td>
<td>0.5</td>
</tr>
</tbody>
</table>

The **OPTLP Procedure**

Dual Solution

<table>
<thead>
<tr>
<th>Obs</th>
<th>Objective Function</th>
<th>RHS Constraint</th>
<th>Constraint Type</th>
<th>Constraint Value</th>
<th>Constraint Lower Bound</th>
<th>Constraint Upper Bound</th>
<th>Dual Variable Value</th>
<th>Constraint Status</th>
<th>Constraint Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>COST</td>
<td>RHS R1</td>
<td>G</td>
<td>-5</td>
<td></td>
<td></td>
<td>1.5</td>
<td>U</td>
<td>-5.0</td>
</tr>
<tr>
<td>2</td>
<td>COST</td>
<td>RHS R2</td>
<td>L</td>
<td>4</td>
<td></td>
<td></td>
<td>0.0</td>
<td>B</td>
<td>2.5</td>
</tr>
<tr>
<td>3</td>
<td>COST</td>
<td>RHS R3</td>
<td>L</td>
<td>7</td>
<td></td>
<td></td>
<td>0.0</td>
<td>B</td>
<td>5.0</td>
</tr>
</tbody>
</table>

For details about the type and status codes displayed for variables and constraints, see the section “Data Input and Output” on page 631.
Syntax: OPTLP Procedure

The following statements are available in the OPTLP procedure:

```
PROC OPTLP <options> ;
DECOMP <options> ;
DECOMPMaster <options> ;
DECOMPSUBPROB <options> ;
```

Functional Summary

Table 13.1 summarizes the list of options available for the OPTLP procedure, classified by function.

<table>
<thead>
<tr>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input and Output Options</strong></td>
<td></td>
</tr>
<tr>
<td>Specifies the input data set</td>
<td>DATA=</td>
</tr>
<tr>
<td>Specifies the dual input data set for warm start</td>
<td>DUALIN=</td>
</tr>
<tr>
<td>Specifies the dual solution output data set</td>
<td>DUALOUT=</td>
</tr>
<tr>
<td>Specifies the input MPS file format</td>
<td>FORMAT=</td>
</tr>
<tr>
<td>Specifies the input MPS file</td>
<td>MPSFILE=</td>
</tr>
<tr>
<td>Specifies the constant part of the objective</td>
<td>OBJCONSTANT=</td>
</tr>
<tr>
<td>Specifies whether the LP model is a maximization or minimization problem</td>
<td>OBJSENSE=</td>
</tr>
<tr>
<td>Specifies the primal input data set for warm start</td>
<td>PRIMALIN=</td>
</tr>
<tr>
<td>Specifies the primal solution output data set</td>
<td>PRIMALOUT=</td>
</tr>
<tr>
<td>Saves output data sets only if optimal</td>
<td>SAVE_ONLY_IF_OPTIMAL</td>
</tr>
<tr>
<td><strong>Solver Options</strong></td>
<td></td>
</tr>
<tr>
<td>Enables or disables IIS detection</td>
<td>IIS=</td>
</tr>
<tr>
<td>Specifies the type of algorithm</td>
<td>ALGORITHM=</td>
</tr>
<tr>
<td>Specifies the type of algorithm called after network simplex</td>
<td>ALGORITHM2=</td>
</tr>
<tr>
<td><strong>Presolve Option</strong></td>
<td></td>
</tr>
<tr>
<td>Specifies the type of presolve</td>
<td>PRESOLVER=</td>
</tr>
<tr>
<td>Controls the dualization of the problem</td>
<td>DUALIZE=</td>
</tr>
<tr>
<td><strong>Control Options</strong></td>
<td></td>
</tr>
<tr>
<td>Specifies the feasibility tolerance</td>
<td>FEASTOL=</td>
</tr>
<tr>
<td>Specifies the frequency of printing solution progress</td>
<td>LOGFREQ=</td>
</tr>
<tr>
<td>Specifies the detail of solution progress printed in log</td>
<td>LOGLEVEL=</td>
</tr>
<tr>
<td>Specifies the maximum number of iterations</td>
<td>MAXITER=</td>
</tr>
<tr>
<td>Specifies the time limit for the optimization process</td>
<td>MAXTIME=</td>
</tr>
<tr>
<td>Specifies the optimality tolerance</td>
<td>OPTTOL=</td>
</tr>
<tr>
<td>Enables or disables printing summary</td>
<td>PRINTLEVEL=</td>
</tr>
<tr>
<td>Specifies units of CPU time or real time</td>
<td>TIMETYPE=</td>
</tr>
<tr>
<td><strong>Simplex Algorithm Options</strong></td>
<td></td>
</tr>
<tr>
<td>Specifies the type of initial basis</td>
<td>BASIS=</td>
</tr>
</tbody>
</table>

Table 13.1 (continued)

<table>
<thead>
<tr>
<th>Description</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specifies the type of pricing strategy</td>
<td>PRICETYPE=</td>
</tr>
<tr>
<td>Specifies the queue size for pricing</td>
<td>QUEUESIZE=</td>
</tr>
<tr>
<td>Enables or disables scaling of the problem</td>
<td>SCALE=</td>
</tr>
<tr>
<td>Specifies the initial seed for the random number generator</td>
<td>SEED=</td>
</tr>
</tbody>
</table>

**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**

```latex
PROC OPTLP <options> ;
```

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

**Input and Output Options**

- **DATA=SAS-data-set**
  - Specifies the input data set that corresponds to the LP model. If neither the DATA= option nor the MPSFILE= option is specified, PROC OPTLP uses the most recently created SAS data set. For more information about the input data set, see Chapter 18, “The MPS-Format SAS Data Set.”

- **DUALIN=SAS-data-set**
  - Specifies the input data set for the dual solution that is required for warm starting the primal and dual simplex algorithms. For more information, see the section “Data Input and Output” on page 631.

- **DUALOUT=SAS-data-set**
  - Specifies the output data set for the dual solution. This data set contains the dual solution information. For more information, see the section “Data Input and Output” on page 631.

- **DOUT=SAS-data-set**
  - Specifies the format of the MPS file that is specified in the MPSFILE= option. You can specify the following values:

  - **FREE**
    - Specifies that the fields of a data record are separated by a space.
  - **FIXED**
    - Specifies that each field of a data record occurs in specific columns.
This option is used only when the MPSFILE= option is specified. For more information about the free and fixed formats of MPS-format files, see Chapter 18, “The MPS-Format SAS Data Set.”

By default, FORMAT=FREE.

**MPSFILE=** *string*

specifies the input MPS-format file that corresponds to the LP model. This option cannot be used with the DATA= option. If neither the DATA= option nor the MPSFILE= option is specified, PROC OPTLP uses the most recently created SAS data set.

**OBJCONSTANT=** *number*

**OBJECTIVECONSTANT=** *number*

specifies the constant part of the objective, where *number* can be any number. This option supersedes the objective right-hand side that is specified in the input data set. By default, OBJCONSTANT=0.

**OBJSENSE=** MIN | MAX

specifies whether the LP model is a minimization or a maximization problem. 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; for more information, see the section “ROWS Section” on page 884. If the objective sense is specified differently in this option and in the input data set, this option supersedes the objective sense that is 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 for the primal solution that is required for warm starting the primal and dual simplex algorithms. For more information, see the section “Data Input and Output” on page 631.

**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. For more information, see the section “Data Input and Output” on page 631.

**SAVE_ONLY_IF_OPTIMAL**

specifies that the PRIMALOUT= and DUALOUT= data sets be saved only if the final solution obtained by the solver at termination is optimal. If the PRIMALOUT= and DUALOUT= options are specified, then by default (that is, omitting the SAVE_ONLY_IF_OPTIMAL option), PROC OPTLP always saves the solutions obtained at termination, regardless of the final status. If the SAVE_ONLY_IF_OPTIMAL option is not specified, the output data sets can contain an intermediate solution, if one is available.

**Solver Options**

**ALGORITHM=** *option*

**SOLVER=** *option*

**SOL=** *option*

specifies an LP algorithm. You can specify the following options:
Chapter 13: The OPTLP Procedure

**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). You can specify the following values:

**FALSE** disable 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. For more information, see the section “Irreducible Infeasible Set” on page 645. By default, IIS=FALSE.

**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.
### PROC OPTLP Statement

**PRESOLVER=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 634.

### Control Options

**FEASTOL=\( \epsilon \)**

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 that is displayed in the SAS log by the LP solver, from a short description of presolve information and summary to details at each iteration. You can specify the following values:

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

By default, LOGLEVEL=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**=$\epsilon$

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 to print a summary of the problem and solution. You can specify the following values:

- $0$ neither produces nor prints any ODS tables.
- $1$ prints the Output Delivery System (ODS) tables ProblemSummary, SolutionSummary, and PerformanceInfo.
- $2$ prints the same tables as PRINTLEVEL=1 along with an additional table called ProblemStatistics.

For more information about the ODS tables created by PROC OPTLP, see the section “ODS Tables” on page 641. By default, PRINTLEVEL=1.

**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. You can specify the following values:

- CPU specifies units of CPU time.
- REAL specifies units of real time.

By default, TIMETYPE=REAL.

**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= DEVEX | FULL | HYBRID | PARTIAL | STEEPESTEDGE specifies the pricing strategy for the primal and dual simplex algorithms. You can specify the following values:

DEVEX uses the Devex pricing strategy.

FULL uses Dantzig’s rule on all decision variables.

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

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

STEEPESTEDGE uses the steepest-edge pricing strategy.

The default option is determined automatically according to the problem structure. For the network simplex algorithm, this option applies only to the algorithm that is specified in the ALGORITHM2= option. For more information, see the section “Pricing Strategies for the Primal and Dual Simplex Algorithms” on page 635.

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=AUTOMATIC | NONE specifies a scaling option. You can specify the following values:

AUTOMATIC automatically applies scaling procedure if necessary.

NONE disables scaling.

By default, SCALE=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.
Chapter 13: The OPTLP Procedure

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.

You can specify the following values:

- **FALSE** does not convert the interior point solution to a basic simplex solution.
- **TRUE** converts the interior point solution to a basic simplex solution.

By default, CROSSOVER=TRUE.

**DUALITYGAP=δ**
specifies the desired relative duality gap $\delta \in [1E^{-9}, 1E^{-4}]$, which is the relative difference between the primal and dual objective function values and is the primary solution quality parameter. For more information, see the section “The Interior Point Algorithm” on page 636. The default value is $1E^{-6}$.

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> ;**
- **DECOMPMASTER <options> ;**
- **DECOMPSUBPROB <options> ;**

For more information about these statements, see Chapter 16, “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 must contain the following variables:

_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 must contain the following variables:

_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).
Chapter 13: The OPTLP Procedure

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. If the SAVE_ONLY_IF_OPTIMAL option is not specified, the PRIMALOUT= data set can contain an intermediate solution, if one is available. See Example 13.1 for an example of the PRIMALOUT= data set. The variables in the data set have the following names and meanings:

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

For the interior point algorithm when IIS=FALSE, _STATUS_ is blank.

The following values can appear only if IIS=TRUE. For more information, see the section “Irreducible Infeasible Set” on page 645.

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:

\[
(c^T - c_B^T B^{-1} A)
\]

where \( B \in \mathbb{R}^{m \times m} \) denotes the basis (matrix composed of basic columns of the constraints matrix \( A \in \mathbb{R}^{m \times n} \)), \( c \in \mathbb{R}^n \) is the vector of objective function coefficients, and \( c_B \in \mathbb{R}^m \) 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. If the SAVE_ONLY_IF_OPTIMAL option is not specified, 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 13.1 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”)
Chapter 13: The OPTLP Procedure

_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 IIS=TRUE. For more information, see the section “Irreducible Infeasible Set” on page 645.

- I_L: the “GE” (≥) condition of the constraint is needed for the IIS
- I_U: the “LE” (≤) 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 i-th constraint would be equal to a_i^T x, where a_i refers to the i-th row of the constraints matrix and 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:

- **DEVEX** implements the Devex pricing strategy developed by Harris (1973).
- **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.
- **HYBRID** uses a hybrid of the Devex and steepest-edge 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.
- **STEEPESTEDGE** uses the steepest-edge pricing strategy developed by Forrest and Goldfarb (1992).

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

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 13.4 and Example 13.5 illustrate how to reoptimize an LP problem with a modified objective function and a modified right-hand side by using this technique. Example 13.6 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:

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

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):

$$\begin{align*}
\min & \quad c^T x \\
\text{subject to} & \quad Ax \geq b \\
& \quad x \geq 0
\end{align*}$$
The corresponding dual formulation is as follows:

$$\begin{align*}
\max & \quad b^T y \\
\text{subject to} & \quad A^T y + w = c \\
& \quad y \geq 0 \\
& \quad w \geq 0
\end{align*}$$

where \( y \in \mathbb{R}^m \) refers to the vector of dual variables and \( w \in \mathbb{R}^n \) 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:

\[
\begin{align*}
Ax - s &= b \quad (\text{primal feasibility}) \\
A^Ty + w &= c \quad (\text{dual feasibility}) \\
WXe &= 0 \quad (\text{complementarity}) \\
SYe &= 0 \quad (\text{complementarity}) \\
x, y, w, s &\geq 0
\end{align*}
\]

where \( e \equiv (1, \ldots, 1)^T \) of appropriate dimension and \( s \in \mathbb{R}^m \) 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 \( X, Y, W, \) and \( S \) denote matrices with corresponding \( x, y, w, \) and \( s \) on the main diagonal and zero elsewhere, as in the following example:

\[
X = \begin{bmatrix}
  x_1 & 0 & \cdots & 0 \\
  0 & x_2 & \cdots & 0 \\
  \vdots & \vdots & \ddots & \vdots \\
  0 & 0 & \cdots & x_n
\end{bmatrix}
\]

If \((x^*, y^*, w^*, s^*)\) is a solution of the previously defined system of equations that represent the KKT conditions, then \( x^* \) 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,

\[
\begin{bmatrix}
  Y^{-1}S \\
  A^T \\
  -X^{-1}W
\end{bmatrix}
\begin{bmatrix}
  \Delta y \\
  \Delta x
\end{bmatrix} = \begin{bmatrix}
  \Xi \\
  \Theta
\end{bmatrix}
\]

where \( \Delta x \) and \( \Delta y \) denote the vector of search directions in the primal and dual spaces, respectively, and \( \Theta \) and \( \Xi \) 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 $\alpha$:
  \[
  \alpha = \frac{\|Ax - b - s\|_2}{\|b\|_2 + 1}
  \]

- relative dual infeasibility measure $\beta$:
  \[
  \beta = \frac{\|c - ATy - w\|_2}{\|c\|_2 + 1}
  \]

- relative duality gap $\delta$:
  \[
  \delta = \frac{|c^Tx - b^Ty|}{|c^Tx| + 1}
  \]

- absolute complementarity $\gamma$:
  \[
  \gamma = \sum_{i=1}^{n} x_i w_i + \sum_{i=1}^{m} y_i s_i
  \]

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.
Chapter 13: The OPTLP Procedure

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 638. 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, concurrent LP, and decomposition algorithms can be run in single-machine mode; in this mode, the computation is executed by multiple threads on a single computer.

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 PRINTLEVEL=2, then the ProblemStatistics table and the Timing table are produced. The ProblemStatistics table contains information about the problem data. For more information, see the section “Problem Statistics” on page 644. The Timing table contains detailed information about the solution time. 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 641.

Table 13.2 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 13.2  ODS Tables Produced by PROC OPTLP

<table>
<thead>
<tr>
<th>ODS Table Name</th>
<th>Description</th>
<th>Statement</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProblemSummary</td>
<td>Summary of the input LP problem</td>
<td>PROC OPTLP</td>
<td>PRINTLEVEL=1 (default)</td>
</tr>
<tr>
<td>SolutionSummary</td>
<td>Summary of the solution status</td>
<td>PROC OPTLP</td>
<td>PRINTLEVEL=1 (default)</td>
</tr>
<tr>
<td>ProblemStatistics</td>
<td>Description of input problem data</td>
<td>PROC OPTLP</td>
<td>PRINTLEVEL=2</td>
</tr>
<tr>
<td>ConcurrentSummary</td>
<td>Summary of the solution status for all algorithms run concurrently</td>
<td>PROC OPTLP, ALGORITHM=CON</td>
<td>PRINTLEVEL=2</td>
</tr>
<tr>
<td>Timing</td>
<td>Detailed solution timing</td>
<td>PROC OPTLP</td>
<td>PRINTLEVEL=2</td>
</tr>
</tbody>
</table>

A typical output of PROC OPTLP is shown in Figure 13.2.

**Figure 13.2**  Typical OPTLP Output

**The OPTLP Procedure**

<table>
<thead>
<tr>
<th>Problem Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem Name</td>
</tr>
<tr>
<td>Objective Sense</td>
</tr>
<tr>
<td>Objective Function</td>
</tr>
<tr>
<td>RHS</td>
</tr>
<tr>
<td>Number of Variables</td>
</tr>
<tr>
<td>Bounded Above</td>
</tr>
<tr>
<td>Bounded Below</td>
</tr>
<tr>
<td>Bounded Above and Below</td>
</tr>
<tr>
<td>Free</td>
</tr>
<tr>
<td>Fixed</td>
</tr>
<tr>
<td>Number of Constraints</td>
</tr>
<tr>
<td>LE (&lt;=)</td>
</tr>
<tr>
<td>EQ (=)</td>
</tr>
<tr>
<td>GE (&gt;=)</td>
</tr>
<tr>
<td>Range</td>
</tr>
<tr>
<td>Constraint Coefficients</td>
</tr>
</tbody>
</table>
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 that correspond to the preceding output are shown in Figure 13.3, 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 13.3** ODS Output Data Sets

**Problem Summary**

<table>
<thead>
<tr>
<th>Obs</th>
<th>Label1</th>
<th>cValue1</th>
<th>nValue1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Problem Name</td>
<td>ADLITTLE</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Objective Sense</td>
<td>Minimization</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Objective Function</td>
<td>.Z....</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>RHS</td>
<td>ZZZZ0001</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Number of Variables</td>
<td>97</td>
<td>97.000000</td>
</tr>
<tr>
<td>6</td>
<td>Bounded Above</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>Bounded Below</td>
<td>97</td>
<td>97.000000</td>
</tr>
<tr>
<td>8</td>
<td>Bounded Above and Below</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>Free</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>Fixed</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Number of Constraints</td>
<td>56</td>
<td>56.000000</td>
</tr>
<tr>
<td>13</td>
<td>LE (&lt;=)</td>
<td>40</td>
<td>40.000000</td>
</tr>
<tr>
<td>14</td>
<td>EQ (=)</td>
<td>15</td>
<td>15.000000</td>
</tr>
<tr>
<td>15</td>
<td>GE (&gt;=)</td>
<td>1</td>
<td>1.000000</td>
</tr>
<tr>
<td>16</td>
<td>Range</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Constraint Coefficients</td>
<td>383</td>
<td>383.000000</td>
</tr>
</tbody>
</table>
### Figure 13.3  continued

**Solution Summary**

<table>
<thead>
<tr>
<th>Obs</th>
<th>Label1</th>
<th>cValue1</th>
<th>nValue1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Solver</td>
<td>LP</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Algorithm</td>
<td>Dual Simplex</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Objective Function</td>
<td>Z.....</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Solution Status</td>
<td>Optimal</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Objective Value</td>
<td>225494.96316</td>
<td>225495</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Primal Infeasibility</td>
<td>2.593481E-13</td>
<td>2.593481E-13</td>
</tr>
<tr>
<td>8</td>
<td>Dual Infeasibility</td>
<td>3.453238E-12</td>
<td>3.453238E-12</td>
</tr>
<tr>
<td>9</td>
<td>Bound Infeasibility</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Iterations</td>
<td>92</td>
<td>92.000000</td>
</tr>
<tr>
<td>12</td>
<td>Presolve Time</td>
<td>0.00</td>
<td>0.000797</td>
</tr>
<tr>
<td>13</td>
<td>Solution Time</td>
<td>0.00</td>
<td>0.002831</td>
</tr>
</tbody>
</table>

### 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 13.4 demonstrates the contents of the ODS table ProblemStatistics.
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=TRUE** 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 631.) For constraints, a status of “I_L”, “I_U”, or “I_F” indicates that the “GE” (≥), “LE” (≤), 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 −∞. 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=TRUE 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 −∞, 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 18, “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 13.7 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.
Chapter 13: The OPTLP Procedure

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.
Example 13.1: 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:

\[
\begin{align*}
\text{max} & \quad -175a_l - 165a_h - 205br + 350j_1 + 350j_2 \\
\text{subject to} & \\
(napha_l) & \quad 0.035a_l + 0.03a_h + 0.045br = na_l \\
(napha_i) & \quad 0.1a_l + 0.075a_h + 0.135br = na_i \\
(htg_oil) & \quad 0.39a_l + 0.3a_h + 0.43br = h_o \\
(blend1) & \quad 0.3j_1 \leq na_i \\
(blend2) & \quad 0.2j_2 \leq na_l \\
(blend3) & \quad 0.7j_1 + 0.8j_2 \leq h_o \\
\text{and} & \quad a_l \leq 110 \\
& \quad a_h \leq 165 \\
& \quad br \leq 80 \\
\end{align*}
\]

and

\[a_l, a_h, br, na_l, na_i, h_o, j_1, j_2 \geq 0\]

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:

\[
\begin{align*}
h_o1 & \quad \geq 0.7j_1 \\
h_o2 & \quad \geq 0.8j_2 \\
h_o1 + h_o2 & \quad \leq h_o \\
\end{align*}
\]

where h_o1 and h_o2 are dummy variables.
You can use the following SAS code to create the input data set `ex1`:

```sas
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 .
  ENDDATA . . . .
;
```

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

```sas
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 13.1.1.
Output 13.1.1  Example 1: Primal and Dual Solution Output

Primal Solution

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

Dual Solution

<table>
<thead>
<tr>
<th>Obs</th>
<th>Objective Function</th>
<th>RHS ID</th>
<th>Constraint Name</th>
<th>Constraint Type</th>
<th>Constraint RHS</th>
<th>Constraint Lower Bound</th>
<th>Constraint Upper Bound</th>
<th>Dual Variable Value</th>
<th>Constraint Status</th>
<th>Constraint Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>profit</td>
<td>napha_l</td>
<td>E</td>
<td>0</td>
<td>.</td>
<td>.</td>
<td>0.000</td>
<td>U</td>
<td>0.0000</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>profit</td>
<td>napha_i</td>
<td>E</td>
<td>0</td>
<td>.</td>
<td>-145.833</td>
<td>.</td>
<td>U</td>
<td>0.0000</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>profit</td>
<td>htg_oil</td>
<td>E</td>
<td>0</td>
<td>.</td>
<td>-437.500</td>
<td>.</td>
<td>U</td>
<td>0.0000</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>profit</td>
<td>blend1</td>
<td>L</td>
<td>0</td>
<td>.</td>
<td>145.833</td>
<td>.</td>
<td>U</td>
<td>0.0000</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>profit</td>
<td>blend2</td>
<td>L</td>
<td>0</td>
<td>.</td>
<td>0.000</td>
<td>.</td>
<td>B</td>
<td>-0.84167</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>profit</td>
<td>blend3</td>
<td>L</td>
<td>0</td>
<td>.</td>
<td>437.500</td>
<td>.</td>
<td>L</td>
<td>-0.0000</td>
<td></td>
</tr>
</tbody>
</table>

The progress of the solution is printed to the log as follows.
Output 13.1.2 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.
NOTE: The MPS read time is 0.01 seconds.
WARNING: The objective sense has been changed to maximization.
NOTE: The LP presolver value AUTOMATIC is applied.
NOTE: The LP presolver time is 0.00 seconds.
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.

<table>
<thead>
<tr>
<th>Phase</th>
<th>Iteration</th>
<th>Objective</th>
<th>Entering Variable</th>
<th>Leaving Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>P 2</td>
<td>1</td>
<td>0.000000E+00</td>
<td>j_1</td>
<td>blend1 (S)</td>
</tr>
<tr>
<td>P 2</td>
<td>2</td>
<td>2.022784E-03</td>
<td>j_2</td>
<td>blend2 (S)</td>
</tr>
<tr>
<td>P 2</td>
<td>3</td>
<td>3.902347E-03</td>
<td>br</td>
<td>blend3 (S)</td>
</tr>
<tr>
<td>P 2</td>
<td>4</td>
<td>4.025073E-03</td>
<td>a_l</td>
<td>a_l</td>
</tr>
<tr>
<td>P 2</td>
<td>5</td>
<td>1.202248E+03</td>
<td>blend2 (S)</td>
<td>br</td>
</tr>
<tr>
<td>P 2</td>
<td>6</td>
<td>1.347921E+03</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D 2</td>
<td>7</td>
<td>1.347917E+03</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NOTE: Optimal.
NOTE: Objective = 1347.9166667.
NOTE: The Primal Simplex solve time is 0.00 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 13.1.3 Log: Value of the Macro Variable _OROPTLP_

STATUS=OK ALGORITHM=PS SOLUTION_STATUS=OPTIMAL OBJECTIVE=1347.9166667
PRIMAL_INFEASIBILITY=3.552714E-15 DUAL_INFEASIBILITY=0 BOUND_INFEASIBILITY=0
ITERATIONS=7 PRESOLVE_TIME=0.00 SOLUTION_TIME=0.00

The value briefly summarizes the status of the OPTLP procedure upon termination.
Alternatively, you can use an MPS-format file instead of a data set. Using the MPS-format file is typically faster than using the data set for large instances. You can use the following file `ex1.mps`:

```
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:

```latex
proc optlp mpsfile='ex1.mps'
obsense = max
algorithm = primal
primalout = exlpout
dualout = exldout
printlevel = 0
logfreq = 1;
run;
%put &_OROPTLP_;
```

The output is the same as when you use the data set for input.
Example 13.2: Using the Interior Point Algorithm

You can also solve the oil refinery problem described in Example 13.1 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 621. You can use the following SAS code to solve the problem:

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

The optimal solution is displayed in Output 13.2.1.

**Output 13.2.1** Interior Point Algorithm: Primal Solution Output

**Primal Solution**

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

The iteration log is displayed in Output 13.2.2.
Example 13.3: 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 13.3.
Table 13.3  Cost and Nutrition Values

<table>
<thead>
<tr>
<th></th>
<th>Bread</th>
<th>Milk</th>
<th>Cheese</th>
<th>Potato</th>
<th>Fish</th>
<th>Yogurt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>2.0</td>
<td>3.5</td>
<td>8.0</td>
<td>1.5</td>
<td>11.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Protein, g</td>
<td>4.0</td>
<td>8.0</td>
<td>7.0</td>
<td>1.3</td>
<td>8.0</td>
<td>9.2</td>
</tr>
<tr>
<td>Fat, g</td>
<td>1.0</td>
<td>5.0</td>
<td>9.0</td>
<td>0.1</td>
<td>7.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Carbohydrates, g</td>
<td>15.0</td>
<td>11.7</td>
<td>0.4</td>
<td>22.6</td>
<td>0.0</td>
<td>17.0</td>
</tr>
<tr>
<td>Calories</td>
<td>90</td>
<td>120</td>
<td>106</td>
<td>97</td>
<td>130</td>
<td>180</td>
</tr>
</tbody>
</table>

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  1.4  .  .
  .  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:

```latex
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 13.3.1.

**Output 13.3.1** Diet Problem: Solution Summary and Optimal Primal Solution

### Solution Summary

<table>
<thead>
<tr>
<th>Obs</th>
<th>Label1</th>
<th>cValue1</th>
<th>nValue1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Solver</td>
<td>LP</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Algorithm</td>
<td>Primal Simplex</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Objective Function</td>
<td>diet</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Solution Status</td>
<td>Optimal</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Objective Value</td>
<td>12.081337881</td>
<td>12.081338</td>
</tr>
<tr>
<td>6</td>
<td>Primal Infeasibility</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>Dual Infeasibility</td>
<td>3.608225E-16</td>
<td>3.608225E-16</td>
</tr>
<tr>
<td>8</td>
<td>Bound Infeasibility</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>Iterations</td>
<td>6</td>
<td>6.000000</td>
</tr>
<tr>
<td>10</td>
<td>Presolve Time</td>
<td>0.00</td>
<td>0.001369</td>
</tr>
</tbody>
</table>

### Primal Solution

<table>
<thead>
<tr>
<th>Objective Function</th>
<th>RHS ID</th>
<th>Variable</th>
<th>Type</th>
<th>Objective Coefficient</th>
<th>Lower Bound</th>
<th>Upper Bound</th>
<th>Variable Value</th>
<th>Variable Status</th>
<th>Reduced Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>diet</td>
<td>br</td>
<td>N</td>
<td>2.0</td>
<td>0.0</td>
<td>1.7977E308</td>
<td>0.00000</td>
<td>L</td>
<td>1.19066</td>
<td></td>
</tr>
<tr>
<td>diet</td>
<td>mi</td>
<td>D</td>
<td>3.5</td>
<td>0.0</td>
<td>1.05360</td>
<td>0.05360</td>
<td>B</td>
<td>0.00000</td>
<td></td>
</tr>
<tr>
<td>diet</td>
<td>ch</td>
<td>N</td>
<td>8.0</td>
<td>0.0</td>
<td>1.7977E308</td>
<td>0.44950</td>
<td>B</td>
<td>0.00000</td>
<td></td>
</tr>
<tr>
<td>diet</td>
<td>po</td>
<td>N</td>
<td>1.5</td>
<td>0.0</td>
<td>1.7977E308</td>
<td>1.86517</td>
<td>B</td>
<td>-0.00000</td>
<td></td>
</tr>
<tr>
<td>diet</td>
<td>fi</td>
<td>O</td>
<td>11.0</td>
<td>0.5</td>
<td>1.7977E308</td>
<td>0.50000</td>
<td>L</td>
<td>5.15641</td>
<td></td>
</tr>
<tr>
<td>diet</td>
<td>yo</td>
<td>N</td>
<td>1.0</td>
<td>0.0</td>
<td>1.7977E308</td>
<td>0.00000</td>
<td>L</td>
<td>1.10849</td>
<td></td>
</tr>
</tbody>
</table>

The cost of the optimal diet is 12.08 units.
Example 13.4: Reoptimizing after Modifying the Objective Function

Using the diet problem described in Example 13.3, 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:

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

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

```plaintext
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
```

Example 13.4: Reoptimizing after Modifying the Objective Function

You can use the BASIS=WARMSTART option (and the ex3pout and ex3dout data sets from Example 13.3) in the following call to PROC OPTLP to solve the modified problem:

```latex
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 13.3 remains optimal after the objective function is changed.

### Output 13.4.1 Iteration Log

<table>
<thead>
<tr>
<th>Phase</th>
<th>Iteration</th>
<th>Entering</th>
<th>Leaving</th>
<th>Objective</th>
<th>Value</th>
<th>Time</th>
<th>Variable</th>
<th>Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>P 2</td>
<td>1</td>
<td>1.098034E+01</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 13.5: 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 13.3. 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:

```plaintext
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 . .  
  ENDDATA . . . . .  
;```

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

```plaintext
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 13.5.1 Iteration Log

<table>
<thead>
<tr>
<th>Phase</th>
<th>Iteration</th>
<th>Objective Value</th>
<th>Entering</th>
<th>Leaving</th>
</tr>
</thead>
<tbody>
<tr>
<td>D 2</td>
<td>1</td>
<td>8.813205E+00</td>
<td>0</td>
<td>calories (S)</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>9.174413E+00</td>
<td>0</td>
<td>carbs (S)</td>
</tr>
</tbody>
</table>

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:

```plaintext
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 13.5.2 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 MPS read time is 0.00 seconds.
NOTE: The LP presolver value NONE is applied.
NOTE: The LP solver is called.
NOTE: The Dual Simplex algorithm is used.

<table>
<thead>
<tr>
<th>Phase</th>
<th>Iteration</th>
<th>Objective Value</th>
<th>Time</th>
<th>Entering Variable</th>
<th>Leaving Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>D 2</td>
<td>1</td>
<td>5.500000E+00</td>
<td>0</td>
<td>mi</td>
<td>fat (S)</td>
</tr>
<tr>
<td>D 2</td>
<td>2</td>
<td>8.650000E+00</td>
<td>0</td>
<td>ch</td>
<td>protein (S)</td>
</tr>
<tr>
<td>D 2</td>
<td>3</td>
<td>8.925676E+00</td>
<td>0</td>
<td>po</td>
<td>carbs (S)</td>
</tr>
<tr>
<td>D 2</td>
<td>4</td>
<td>9.174413E+00</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 13.6: Reoptimizing after Adding a New Constraint

Assume that after solving the diet problem in Example 13.3 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 13.4.

Table 13.4 Updated Cost and Nutrition Values

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

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 . . . .```
Example 13.6: Reoptimizing after Adding a New Constraint

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:

```sas
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;
```
Chapter 13: The OPTLP Procedure

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 13.6.1.

**Output 13.6.1** DUALIN= Data Set with a Newly Added Constraint

<table>
<thead>
<tr>
<th>Obs</th>
<th><em>ROW</em></th>
<th><em>STATUS</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>calories</td>
<td>U</td>
</tr>
<tr>
<td>2</td>
<td>protein</td>
<td>L</td>
</tr>
<tr>
<td>3</td>
<td>fat</td>
<td>U</td>
</tr>
<tr>
<td>4</td>
<td>carbs</td>
<td>B</td>
</tr>
<tr>
<td>5</td>
<td>sodium</td>
<td>A</td>
</tr>
</tbody>
</table>

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

```sas
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 13.6.2.

**Output 13.6.2** Primal and Dual Solution Output

<table>
<thead>
<tr>
<th>Primal Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obs</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
</tbody>
</table>
Output 13.6.2 continued

Dual Solution

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

The iteration log shown in Output 13.6.3 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 13.6.3 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 MPS read time is 0.00 seconds.
NOTE: The LP presolver value NONE is applied.
NOTE: The LP solver is called.
NOTE: The Dual Simplex algorithm is used.

<table>
<thead>
<tr>
<th>Objective</th>
<th>Entering</th>
<th>Leaving</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase Iteration</td>
<td>Value</td>
<td>Time</td>
</tr>
<tr>
<td>D 2</td>
<td>1</td>
<td>1.208134E+01</td>
</tr>
</tbody>
</table>

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 13.4 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 13.7: 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:

\[
\begin{align*}
\text{min} & \quad x_1 + x_2 + x_3 \quad \text{(cost)} \\
\text{subject to} & \quad x_1 + x_2 \geq 10 \quad \text{(con1)} \\
& \quad x_1 + x_3 \leq 4 \quad \text{(con2)} \\
& \quad 4 \leq x_2 + x_3 \leq 5 \quad \text{(con3)} \\
& \quad x_1, x_2 \geq 0 \\
& \quad 0 \leq x_3 \leq 3
\end{align*}
\]

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

```sas
/* infeasible */
data exiis;
  input field1 $ field2 $ field3 $ field4 field5 $ field6;
datelines;
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.

\[
\begin{align*}
x_1 + x_2 & \geq 10 \quad \text{(con1)} \\
x_1 + x_3 & \leq 4 \quad \text{(con2)} \\
x_2 + x_3 & \leq 5 \quad \text{(con3)} \\
x_3 & \geq 0
\end{align*}
\]
You can use the `IIS=TRUE` option to detect this IIS by using the following statements:

```plaintext
proc optlp data=exiis
   iis=true
   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 13.7.1 are printed to the log.

**Output 13.7.1  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 MPS read time is 0.00 seconds.
NOTE: The LP solver is called.
NOTE: The IIS= option is enabled.

<table>
<thead>
<tr>
<th>Phase</th>
<th>Iteration</th>
<th>Objective Value</th>
<th>Time</th>
<th>Entering Variable</th>
<th>Leaving Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>P 1</td>
<td>1</td>
<td>6.000000E+00</td>
<td>0</td>
<td>con3 (S)</td>
<td>con3 (S)</td>
</tr>
<tr>
<td>P 1</td>
<td>2</td>
<td>5.000000E+00</td>
<td>0</td>
<td>x1</td>
<td>con2 (S)</td>
</tr>
<tr>
<td>P 1</td>
<td>3</td>
<td>1.000000E+00</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Processed</th>
<th>Removed</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

NOTE: Processing variable bounds.

<table>
<thead>
<tr>
<th>Processed</th>
<th>Removed</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

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 13.7.2.
Output 13.7.2 Identify Rows and Columns in the IIS

### Constraints in the IIS

<table>
<thead>
<tr>
<th>Objective Function ID</th>
<th>RHS ID</th>
<th>Constraint Name</th>
<th>Constraint Type</th>
<th>Constraint RHS</th>
<th>Constraint Lower Bound</th>
<th>Constraint Upper Bound</th>
<th>Dual Variable Value</th>
<th>Constraint Status</th>
<th>Constraint Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>cost</td>
<td>rhs</td>
<td>con1</td>
<td>G</td>
<td>10</td>
<td>.</td>
<td>.</td>
<td>. I_L</td>
<td>.</td>
</tr>
<tr>
<td>2</td>
<td>cost</td>
<td>rhs</td>
<td>con2</td>
<td>L</td>
<td>4</td>
<td>.</td>
<td>.</td>
<td>. I_U</td>
<td>.</td>
</tr>
<tr>
<td>3</td>
<td>cost</td>
<td>rhs</td>
<td>con3</td>
<td>R</td>
<td>4</td>
<td>5</td>
<td>.</td>
<td>. I_U</td>
<td>.</td>
</tr>
</tbody>
</table>

### Variables in the IIS

<table>
<thead>
<tr>
<th>Objective Function ID</th>
<th>RHS ID</th>
<th>Variable Name</th>
<th>Variable Type</th>
<th>Objective Coefficient</th>
<th>Objective Lower Bound</th>
<th>Objective Upper Bound</th>
<th>Reduced Cost</th>
<th>Variable Status</th>
<th>Reduced Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>cost</td>
<td>x1</td>
<td>N</td>
<td>1</td>
<td>0</td>
<td>1.7977E308</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>2</td>
<td>cost</td>
<td>x2</td>
<td>N</td>
<td>1</td>
<td>0</td>
<td>1.7977E308</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>3</td>
<td>cost</td>
<td>x3</td>
<td>D</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>. I_L</td>
<td>.</td>
<td>.</td>
</tr>
</tbody>
</table>

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:

```sas
/* 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 . .
  ENDDATA ;
```
Example 13.8: Using the Network Simplex Algorithm

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:

```sas
proc optlp data=exiisf
  iis=true;
run;
```

The notes shown in Output 13.7.3 are printed to the log.

Output 13.7.3  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 MPS read time is 0.00 seconds.
NOTE: The LP solver is called.
NOTE: The IIS= option is enabled.

<table>
<thead>
<tr>
<th>Objective</th>
<th>Phase</th>
<th>Iteration</th>
<th>Value</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>P 1</td>
<td>1</td>
<td>1</td>
<td>1.4E+01</td>
<td>0</td>
</tr>
<tr>
<td>P 1</td>
<td>2</td>
<td>0.0E+00</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

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 13.7.4.

Output 13.7.4  Infeasibility Removed

<table>
<thead>
<tr>
<th>Solution Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obs</td>
</tr>
<tr>
<td>-----</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
</tbody>
</table>

Example 13.8: 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 13.5, which appears in Ahuja, Magnanti, and Orlin (1993).
You can use the following SAS statements to create the input data set ex8:

```sas
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 . .
```
Example 13.8: Using the Network Simplex Algorithm

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

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

The optimal solution is displayed in **Output 13.8.1**.

**Output 13.8.1** Network Simplex Algorithm: Primal Solution Output

### Primal Solution

<table>
<thead>
<tr>
<th>Obs</th>
<th>Objective Function ID</th>
<th>RHS Name</th>
<th>Variable Name</th>
<th>Variable Type</th>
<th>Objective Coefficient</th>
<th>Lower Bound</th>
<th>Upper Bound</th>
<th>Variable Value</th>
<th>Variable Status</th>
<th>Reduced Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>obj</td>
<td>RHS</td>
<td>x['7','8']</td>
<td>obj</td>
<td>9</td>
<td>balance['7']</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>obj</td>
<td>RHS</td>
<td>balance['8']</td>
<td></td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>RHS</td>
<td>RHS</td>
<td>balance['1']</td>
<td></td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>RHS</td>
<td>RHS</td>
<td>balance['2']</td>
<td></td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>RHS</td>
<td>RHS</td>
<td>balance['4']</td>
<td></td>
<td>-5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>RHS</td>
<td>RHS</td>
<td>balance['7']</td>
<td></td>
<td>-15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>RHS</td>
<td>RHS</td>
<td>balance['8']</td>
<td></td>
<td>-10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>BOUNDS</td>
<td>BOUNDS</td>
<td>x['1','4']</td>
<td>D</td>
<td>2</td>
<td>0</td>
<td>15</td>
<td>10</td>
<td>B</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>BOUNDS</td>
<td>BOUNDS</td>
<td>x['2','1']</td>
<td>D</td>
<td>1</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>L</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>BOUNDS</td>
<td>BOUNDS</td>
<td>x['2','3']</td>
<td>D</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>B</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>BOUNDS</td>
<td>BOUNDS</td>
<td>x['2','6']</td>
<td>D</td>
<td>6</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>B</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>BOUNDS</td>
<td>BOUNDS</td>
<td>x['3','4']</td>
<td>D</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>B</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>BOUNDS</td>
<td>BOUNDS</td>
<td>x['3','5']</td>
<td>D</td>
<td>4</td>
<td>0</td>
<td>10</td>
<td>5</td>
<td>B</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>BOUNDS</td>
<td>BOUNDS</td>
<td>x['4','7']</td>
<td>D</td>
<td>5</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>U</td>
<td>-5</td>
</tr>
<tr>
<td>15</td>
<td>BOUNDS</td>
<td>BOUNDS</td>
<td>x['5','6']</td>
<td>D</td>
<td>2</td>
<td>0</td>
<td>20</td>
<td>0</td>
<td>L</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>BOUNDS</td>
<td>BOUNDS</td>
<td>x['5','7']</td>
<td>D</td>
<td>7</td>
<td>0</td>
<td>15</td>
<td>5</td>
<td>B</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>BOUNDS</td>
<td>BOUNDS</td>
<td>x['6','8']</td>
<td>D</td>
<td>8</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>B</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>BOUNDS</td>
<td>BOUNDS</td>
<td>x['7','8']</td>
<td>D</td>
<td>9</td>
<td>0</td>
<td>15</td>
<td>0</td>
<td>L</td>
<td>6</td>
</tr>
</tbody>
</table>
The optimal solution is represented graphically in Figure 13.6.

**Figure 13.6** Minimum-Cost Network Flow Problem: Optimal Solution

![Minimum-Cost Network Flow Problem: Optimal Solution](image)

The iteration log is displayed in Output 13.8.2.

**Output 13.8.2** 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 MPS read time is 0.00 seconds.
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.

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Primal Objective</th>
<th>Primal Infeasibility</th>
<th>Dual Infeasibility</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0000000E+00</td>
<td>2.0000000E+00</td>
<td>8.9000000E+01</td>
<td>0.00</td>
</tr>
<tr>
<td>2</td>
<td>0.0000000E+00</td>
<td>2.0000000E+00</td>
<td>8.9000000E+01</td>
<td>0.00</td>
</tr>
<tr>
<td>3</td>
<td>5.0000000E+00</td>
<td>1.5000000E+01</td>
<td>8.4000000E+01</td>
<td>0.00</td>
</tr>
<tr>
<td>4</td>
<td>5.0000000E+00</td>
<td>1.5000000E+01</td>
<td>8.3000000E+01</td>
<td>0.00</td>
</tr>
<tr>
<td>5</td>
<td>7.5000000E+01</td>
<td>1.5000000E+01</td>
<td>8.3000000E+01</td>
<td>0.00</td>
</tr>
<tr>
<td>6</td>
<td>7.5000000E+01</td>
<td>1.5000000E+01</td>
<td>7.9000000E+01</td>
<td>0.00</td>
</tr>
<tr>
<td>7</td>
<td>1.3000000E+02</td>
<td>1.0000000E+01</td>
<td>7.6000000E+01</td>
<td>0.00</td>
</tr>
<tr>
<td>8</td>
<td>2.7000000E+02</td>
<td>0.0000000E+00</td>
<td>0.0000000E+00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

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


Subject Index

.ACTIVITY_ variable
   DUALOUT= data set, 634

basis, 628

concurrent LP, 641

data, 624

decomposition algorithm
   OPTLP procedure, 630
   DUALIN= data set
      OPTLP procedure, 631
      variables, 631
   dualization, 626
   DUALOUT= data set
      OPTLP procedure, 633, 634
      variables, 633, 634

feasibility tolerance, 627

IIS option
   OPTLP procedure, 645
irreducible infeasible set
   OPTLP procedure, 645
iteration log
   crossover algorithm, 640
   interior point algorithm, 640
   network simplex algorithm, 639
   OPTLP procedure, 638–640
   primal and dual simplex algorithms, 638

.LBOUND_ variable
   PRIMALOUT= data set, 632

linear programming, see also OPTLP procedure

.L_RHS_ variable
   DUALOUT= data set, 634

OROPTLP
   _OROPTLP_, 646

number of threads, 630

.VAR_ variable
   PRIMALOUT= data set, 632

_OBJ_ID_ variable
   DUALOUT= data set, 633
   PRIMALOUT= data set, 632

ODS table names
   OPTLP procedure, 641
   OPTLP examples

diet optimization problem, 655
finding an irreducible infeasible set, 666
oil refinery problem, 649
reoptimizing after adding a new constraint, 662
reoptimizing after modifying the objective function, 658
reoptimizing after modifying the right-hand side, 660
using the interior point algorithm, 654
using the network simplex algorithm, 669

OPTLP procedure
   algorithm, 625
   basis, 628
   concurrent LP, 641
   crossover, 630
   data, 624
   decomposition algorithm, 630
   definitions of DUALIN= data set variables, 631
   definitions of DUALOUT= data set variables, 633, 634
   definitions of DUALOUT=data set variables, 633, 634
   definitions of PRIMALIN data set variables, 631
   definitions of PRIMALIN= data set variables, 631
   definitions of PRIMALOUT= data set variables, 632, 633
   DUALIN= data set, 631
   duality gap, 630
dualization, 626
   DUALOUT= data set, 633, 634
   feasibility tolerance, 627
   functional summary, 623
   IIS option, 645
   interior point algorithm, 636
   introductory example, 621
   iteration log, 638–640
   network simplex algorithm, 636
   number of threads, 630
   ODS table names, 641
   _OROPTLP_ macro variable, 646
   preprocessing, 627
   presolver, 627
   pricing, 629
   PRIMALIN= data set, 631
   PRIMALOUT= data set, 632, 633
   problem statistics, 644
   queue size, 629
   random seed, 629
scaling, 629

presolver, 627

pricing, 629

PRIMALIN= data set

   OPTLP procedure, 631
   variables, 631

PRIMALOUT= data set

   OPTLP procedure, 632, 633
   variables, 632, 633

queue size, 629

random seed, 629

   _R_COST_ variable
   PRIMALOUT= data set, 633

   _RHS_ variable
   DUALOUT= data set, 634

   _RHS_ID_ variable
   DUALOUT= data set, 633
   PRIMALOUT= data set, 632

   _ROW_ variable
   DUALIN= data set, 631
   DUALOUT= data set, 633

scaling, 629

   _STATUS_ variable
   DUALIN= data set, 631
   DUALOUT= data set, 634
   PRIMALIN= data set, 631
   PRIMALOUT= data set, 632

   _TYPE_ variable
   DUALOUT= data set, 633
   PRIMALOUT= data set, 632

   _UBOUND_ variable
   PRIMALOUT= data set, 632

   _U_RHS_ variable
   DUALOUT= data set, 634

   _VALUE_ variable
   DUALOUT= data set, 634
   PRIMALOUT= data set, 632

   _VAR_ variable
   PRIMALIN= data set, 631
   PRIMALOUT= data set, 632
Syntax Index

ALGORITHM2= option
   PROC OPTLP statement, 626
ALGORITHM= option
   PROC OPTLP statement, 625
BASIS= option
   PROC OPTLP statement, 628
CROSSOVER= option
   PROC OPTLP statement, 630
DATA= option
   PROC OPTLP statement, 624
DECOMPMASTER statement
   OPTLP procedure, 630
DECOMP statement
   OPTLP procedure, 630
DECOMPSUBPROB statement
   OPTLP procedure, 630
DETERMINISTIC= option
   PROC OPTLP statement, 626
DUALIN= option
   PROC OPTLP statement, 624
DUALIZE= option
   PROC OPTLP statement, 626
DUALOUT= option
   PROC OPTLP statement, 624
FEASTOL= option
   PROC OPTLP statement, 627
FORMAT= option
   PROC OPTLP statement, 624
IIS= option
   PROC OPTLP statement, 626
LOGFREQ= option
   PROC OPTLP statement, 627
LOGLEVEL= option
   PROC OPTLP statement, 627
MAXITER= option
   PROC OPTLP statement, 628
MAXTIME= option
   PROC OPTLP statement, 628
MPSFILE= option
   PROC OPTLP statement, 625
NTHREADS= option
   PROC OPTLP statement, 630
OBJCONSTANT option
   PROC OPTLP statement, 625
OBJSENSE= option
   PROC OPTLP statement, 625
OPTLP procedure, 623
   DECOMPMASTER statement, 630
   DECOMP statement, 630
   DECOMPSUBPROB statement, 630
OPTTOL= option
   PROC OPTLP statement, 628
PRESOLVER= option
   PROC OPTLP statement, 627
PRICETYPE= option
   PROC OPTLP statement, 629
PRIMALIN= option
   PROC OPTLP statement, 625
PRIMALOUT= option
   PROC OPTLP statement, 625
PRINTFREQ= option
   PROC OPTLP statement, 627
PRINTLEVEL= option
   PROC OPTLP statement, 628
PROC OPTLP statement
   ALGORITHM2= option, 626
   ALGORITHM= option, 625
   BASIS= option, 628
   CROSSOVER= option, 630
   DATA= option, 624
   DETERMINISTIC= option, 630
   DUALIN= option, 624
   DUALIZE= option, 626
   DUALOUT= option, 624
   FEASTOL= option, 627
   FORMAT= option, 624
   IIS= option, 626
   LOGFREQ= option, 627
   LOGLEVEL= option, 627
   MAXITER= option, 628
   MAXTIME= option, 628
   MPSFILE= option, 625
   NTHREADS= option, 630
   OBJCONSTANT option, 625
   OBJSENSE= option, 625
   OPTTOL= option, 628
   PRESOLVER= option, 627
   PRICETYPE= option, 629
   PRIMALIN= option, 625
PRIMALOUT= option, 625
PRINTFREQ= option, 627
PRINTLEVEL= option, 628
QUEUESIZE= option, 629
SAVE_ONLY_IF_OPTIMAL option, 625
SCALE= option, 629
SEED= option, 629
SOL= option, 625
SOLVER2= option, 626
SOLVER= option, 625
STOP_DG= option, 630
TIMETYPE= option, 628

QUEUESIZE= option
  PROC OPTLP statement, 629

SAVE_ONLY_IF_OPTIMAL option
  PROC OPTLP statement, 625
SCALE= option
  PROC OPTLP statement, 629
SEED= option
  PROC OPTLP statement, 629
SOL= option
  PROC OPTLP statement, 625
SOLVER2= option
  PROC OPTLP statement, 626
SOLVER= option
  PROC OPTLP statement, 625
STOP_DG= option
  PROC OPTLP statement, 630
TIMETYPE= option
  PROC OPTLP statement, 628