LP file format: algebraic representation

Version 12.4.0

CPLEX conforms to these rules for the LP file format.

CPLEX provides a facility for entering a problem in a natural, algebraic LP formulation from the keyboard. The problem can be modified and saved from within CPLEX. This procedure is one way to create a file in a format that CPLEX can read. An alternative technique is to create a similar file using a standard text editor and to read it into CPLEX.

The CPLEX LP format is provided as an input alternative to the MPS file format. An LP format file may be easier to generate than an MPS file if your problem already exists in an algebraic format or if you have an application which generates the problem file more readily in algebraic format (such as a C application). Working with LP files in the CPLEX User's Manual in the chapter about managing input and output explains the implications of using LP format rather than MPS format.

CPLEX accepts any problem saved in an ASCII file provided that it adheres to the following syntax rules.

Comments

Anything that follows a backslash (\) is a comment and is ignored until a return is encountered. Blank lines are also ignored. Blank lines and comment lines may be placed anywhere and as frequently as you want in the file.

White space and line length

In general, white space between characters is irrelevant as it is skipped when a file is read. However, white space is not allowed in the keywords used to introduce a new section, such as `MAX` , `MIN` , `ST` , or `BOUNDS` . Also the keywords must be separated by white space from the rest of the file and must be at the beginning of a line. The maximum length for any name is 255. The maximum length of any line of input is 560.

Skipping spaces may cause CPLEX to misinterpret (and accept) an invalid entry, such as the following:

```
x1 x2 = 0
```

If the user intended to enter that example as a nonlinear constraint, CPLEX would instead interpret it as a constraint specifying that one variable named `x1x2` must be equal to zero.

To indicate a quadratic constraint in this section, use explicit notation for multiplication and exponentiation (not space).

Problem sense

The problem statement must begin with the word `MINIMIZE` or `MAXIMIZE`, `MINIMUM` or `MAXIMUM`, or the abbreviations `MIN` or `MAX` , in any combination of upper- and lowercase characters. The word introduces the objective function section.

Variables

Variables can be named anything provided that the name does not exceed 255 characters, all of which must be alphanumeric (a-z, A-Z, 0-9) or one of these symbols: ! " # $ % & ( ) , . ; ? @ _ ‘ ’ { } ~. Longer names are truncated to 255 characters. A variable name can not begin with a number or a period.

The letter `E` or `e` , alone or followed by other valid symbols, or followed by another `E` or `e` , should be avoided as this notation is reserved for exponential entries. Thus, variables can not be named `e9` , `E-24` , `E8cats` , or other names that could be interpreted as an exponent. Even variable names such as `eels` or `example` can cause a read error, depending on their placement in an input line.

Objective function

The objective function definition must follow `MINIMIZE` or `MAXIMIZE` . It may be entered on multiple lines as long as no variable, constant, or sense indicator is split by a return. For example, this objective function `1x1 + 2x2 +3x3` can be entered like this:

```
1x1 + 2x2
+ 3x3
```

but not like this:

```
1x1 + 2x
2 + 3x3         \ a bad idea
```

because the second style splits the variable name `x2` with a return.

The objective function may be named by typing a name and a colon before the objective function. The objective function name and the colon must appear on the same line. Objective function names must conform to the same guidelines as variable names. (See the rule about Variables). If no objective function name is specified, CPLEX assigns the name `obj`.

An objective function may be quadratic. For an example and details about formatting a quadratic objective function, see the rule about Quadratic terms.

Constraints

The constraints section is introduced by the keyword `subject to`. This expression can also appear as `such that`, `st`, `S.T.`, or `ST.` in any mix of upper- and lowercase characters. One of these expressions must precede the first constraint and be separated from it by at least one space.

Each constraint definition must begin on a new line. A constraint may be named by typing a name and a colon before the constraint. The constraint name and the colon must appear on the same line. Constraint names must adhere to the same guidelines as variable names. (See the rule about names of Variables.) If no constraint names are specified, CPLEX assigns the names `c1`, `c2`, `c3`, etc.

The constraints are entered in the same way as the objective function; however, a constraint must be followed by an indication of its sense and a righthand side coefficient. The righthand side coefficient must be typed on the same line as the sense indicator. Acceptable sense indicators are <, <=, =<, >, >=, =>, and =. These are interpreted as ≤, ≤, ≤, ≥, ≥, ≥, and =, respectively.

For example, here is a named constraint:

```
time: x1 + x2 <= 10
```

Quadratic constraints are allowed in this section. Quadratic terms are specified inside square brackets `[]` as detailed in the rule about Quadratic terms. The specification of a quadratic constraint differs from the specification of a quadratic objective in one important way: in a quadratic constraint, the terms are not divided by two; that is, they are not multiplied by 1/2, as they must be in a quadratic objective.

Indicator constraints are also allowed in this section. The rule about MIP indicator constraints explains how to specify indicator constraints.

Bounds

The optional `bounds` section follows the mandatory constraint section. It is preceded by the word `bounds` or `bound` in any mix of lower- and uppercase characters.

Each bound definition must begin on a new line. The format for a bound is $l_n \leq x_n \leq u_n$ except in the following cases.

Upper and lower bounds may also be entered separately as

$l_n \leq x_n$

$x_n \leq u_n$

with the default lower bound of 0 (zero) and the default upper bound of $+\infty$ remaining in effect until the bound is explicitly changed.

Bounds that fix a variable can be entered as simple equalities. For example, `x5 = 5.6` is equivalent to `5.6 <= x5 <= 5.6.`

The bounds $+\infty$ (positive infinity) and $-\infty$ (negative infinity) must be entered as words: `+infinity`, `-infinity`, `+inf`, `-inf`.

A variable with a negative infinity lower bound and positive infinity upper bound may be entered as `free`, in any mix of upper- and lowercase characters, with a space separating the variable name and the word `free`. For example, `x7 free` is equivalent to `- infinity <= x7 <= + infinity.`

The last bound entered takes precedence over previously entered bounds.

End of file

The file must end with the word `end` in any combination of upper- and lowercase characters, alone on a line, when it is created with the `enter` command. This word is not required for files that are read in to CPLEX, but it is a good practice to use it. Files that have been corrupted can frequently be detected by a missing last line.

MIP integer variables

This rule applies to the CPLEX MIP optimizer.

To specify any of the variables as general integer variables, add a `GENERAL` section; to specify any of the variables as binary integer variables, add a `BINARY` section. The `GENERAL` and `BINARY` sections follow the `BOUNDS` section, if one is present; otherwise, they follow the constraints section. Either of the `GENERAL` or `BINARY` sections can precede the other. The `GENERAL` section is preceded by the word `GENERAL` , `GENERALS` , or `GEN` in any mix of upper- and lowercase characters which must appear alone on a line. The following line or lines should list the names of all variables which are to be restricted to general integer values, separated by at least one space. The `BINARY` section is preceded by the word `BINARY` , `BINARIES` , or `BIN` in any mix of upper- and lowercase characters which must appear alone on a line. The following line or lines should list the names of all variables which are to be restricted to binary integer values, separated by at least one space. Binary variables are automatically given bounds of 0 (zero) and 1 (one), unless alternative bounds are specified in the `BOUNDS` section, in which case a warning message is issued.

Here is an example of a problem formulation in LP format where `x4` is a general integer:

```
Maximize
 obj: x1 + 2 x2 + 3 x3 + x4
Subject To
 c1: - x1 + x2 + x3 + 10 x4 <= 20
 c2: x1 - 3 x2 + x3 <= 30
 c3: x2 - 3.5 x4 = 0
Bounds
 0 <= x1 <= 40
 2 <= x4 <= 3
```

```
General
 x4
End
```

If branching priorities or branching directions exist, enter this information through ORD files, as documented in ORD file format: priorities and branching orders.

MIP semi-continuous variables

This rule applies to the CPLEX MIP optimizer.

To specify any of the variables as semi-continuous variables, that is as variables that may take the value `0` or values between the specified lower and upper bounds, use a `SEMI-CONTINUOUS` section. This section must follow the `BOUNDS` , `GENERALS` , and `BINARIES` sections. The `SEMI-CONTINUOUS` section is preceded by the keyword `SEMI-CONTINUOUS` , `SEMI` , or `SEMIS` . The following line or lines should list the names of all the variables which are to be declared semi-continuous, separated by at least one space.

```
Semi-continuous

x1 x2 x3
```

MIP special ordered sets

This rule applies to the CPLEX MIP optimizer. To specify special ordered sets, use an SOS section, which is preceded by the SOS keyword. The SOS section should follow the Bounds, General, Binaries and Semi-Continuous sections. Special ordered sets of type 1 require that, of the variables in the set, one at most may be nonzero. Special ordered sets of type 2 require that at most two variables in the set may be nonzero, and if there are two nonzeros, they must be adjacent. Adjacency is defined by the weights, which must be unique within a set given to the variables. The sorted weights define the order of the special ordered set. For MIP branch and cut, the order is used to decide how the variables are branched upon. See the CPLEX User’s Manual for more information. The set is specified by an optional set name followed by a colon and then either of the S1 or S2 keywords (specifying the type) followed by a double colon. The set member names are listed on this line or lines, with their weights. Variable names and weights are separated by a colon, for example:

```
SOS

set1: S1:: x1:10 x2:13
```

MIP indicator constraints

This rule applies to CPLEX MIP optimizer.

To specify an indicator constraint, enter it among any other constraints in the model, like this:

```
[constraintname:]  binaryvariable = value  ->  linear constraint
```

The constraint name, followed by a colon, is optional. The hyphen followed by the greater-than symbol (`->`), separates the indicator variable and its value from the linear constraint that is controlled. The indicator variable must be declared as a binary variable, and the value it is compared to must be either 0 (zero) or 1 (one).

Quadratic terms

This rule applies to applications that solve problems with quadratic terms in them, that is, quadratic programming problems and quadratically constrained programs (QPs and QCPs). Quadratic coefficients may appear in the objective function. Quadratic coefficients may also appear in constraints under certain conditions. If there are quadratically constrained variables in the problem, see also rules about Variables, Constraints, and Solving problems with quadratic constraints (QCP) in the CPLEX User's Manual.

The algebraic coefficients of the function x'Qx are specified inside square brackets `[]`. The square brackets must be followed by a divide sign followed by the number 2. This convention denotes that all coefficients inside the square brackets will be divided by 2 in evaluating the quadratic terms of the objective function. All quadratic coefficients must appear inside square brackets. Multiple square bracket sections may be specified.

Inside of the square brackets, two variables are multiplied by an asterisk (`*`). For example, `[4x*y]` indicates that the coefficients of both of the off-diagonal terms of Q, corresponding to the variables `x` and `y` in the model are 2, since `4x*y` equals `2x*y + 2x*y`. Each pair of off-diagonal terms of Q is specified only once. CPLEX automatically creates both off-diagonal entries of Q. Diagonal terms in Q (that is, terms with an exponent of 2) are indicated by the caret (`^`) followed by 2. For example, `4x^2` indicates that the coefficient of the diagonal term of Q corresponding to the variable `x` in the model is 4.

For example, this problem

Minimize $a + b + 1/2(a^2 + 4ab + 7b^2)$

subject to $a + b \geq 10$ and $a, b \geq 0$

in LP format looks like this:

```
Minimize
obj: a + b + [ a^2 + 4 a * b + 7 b^2 ]/2
Subject To
c1: a + b >= 10
End
```

Pools of lazy constraints and user-defined cuts

This rule is of interest only to advanced users.

It is possible to include pools of lazy constraints and user defined cuts in an LP file. A pool of lazy constraints or of user-defined cuts must not contain any quadratic constraints. For more about these concepts, see User-cut and lazy-constraint pools in the CPLEX User's Manual.

Parent topic:

File Formats Reference Manual