GNU Linear Programming Kit

Reference Manual

for GLPK Version 4.45

(DRAFT, December 2010)

Appendix C

CPLEX LP Format

C.1 Prelude

The CPLEX LP format[1] is intended for coding LP/MIP problem data. It is a row-oriented format that assumes the formulation of LP/MIP problem (1.1)—(1.3) (see Section 1.1, page 12).

CPLEX LP file is a plain text file written in CPLEX LP format. Each text line of this file may contain up to 255 characters[2]. Blank lines are ignored. If a line contains the backslash character (\), this character and everything that follows it until the end of line are considered as a comment and also ignored.

An LP file is coded by the user using the following elements:

- keywords;
- symbolic names;
- numeric constants;
- delimiters;
- blanks.

[1] The CPLEX LP format was developed in the end of 1980's by CPLEX Optimization, Inc. as an input format for the CPLEX linear programming system. Although the CPLEX LP format is not as widely used as the MPS format, being row-oriented it is more convenient for coding mathematical programming models by human. This appendix describes only the features of the CPLEX LP format which are implemented in the GLPK package.

[2] GLPK allows text lines of arbitrary length.

Keywords which may be used in the LP file are the following:

```
minimize        minimum         min
maximize        maximum         max
subject to      such that       s.t.      st.      st
bounds          bound
general         generals        gen
integer         integers        int
binary          binaries        bin
infinity        inf
free
end
```

All the keywords are case insensitive. Keywords given above on the same line are equivalent. Any keyword (except `infinity`, `inf`, and `free`) being used in the LP file must start at the beginning of a text line.

Symbolic names are used to identify the objective function, constraints (rows), and variables (columns). All symbolic names are case sensitive and may contain up to 16 alphanumeric characters[3] (`a`, ..., `z`, `A`, ..., `Z`, `0`, ..., `9`) as well as the following characters:

```
!  "  #  $  %  &  (  )  /  ,  .  ;  ?  @  _  `  '  {  }  |  ~
```

with exception that no symbolic name can begin with a digit or a period.

Numeric constants are used to denote constraint and objective coefficients, right-hand sides of constraints, and bounds of variables. They are coded in the standard form xx`E`syy, where xx is a real number with optional decimal point, s is a sign (`+` or `-`), yy is an integer decimal exponent. Numeric constants may contain arbitrary number of characters. The exponent part is optional. The letter '`E`' can be coded as '`e`'. If the sign s is omitted, plus is assumed.

Delimiters that may be used in the LP file are the following:

```
:
+
-
<   <=   =<
>   >=   =>
=
```

[3] GLPK allows symbolic names having up to 255 characters.

Delimiters given above on the same line are equivalent. The meaning of the delimiters will be explained below.

Blanks are non-significant characters. They may be used freely to improve readability of the LP file. Besides, blanks should be used to separate elements from each other if there is no other way to do that (for example, to separate a keyword from a following symbolic name).

The order of an LP file is:

- objective function definition;
- constraints section;
- bounds section;
- general, integer, and binary sections (can appear in arbitrary order);
- end keyword.

These components are discussed in following sections.

C.2 Objective function definition

The objective function definition must appear first in the LP file. It defines the objective function and specifies the optimization direction.

The objective function definition has the following form:

$$\left\{\begin{matrix}\texttt{minimize}\\ \texttt{maximize}\end{matrix}\right\}\ f : s\ c\ x\ s\ c\ x\ \ldots\ s\ c\ x$$

where f is a symbolic name of the objective function, s is a sign `+` or `-`, c is a numeric constant that denotes an objective coefficient, x is a symbolic name of a variable.

If necessary, the objective function definition can be continued on as many text lines as desired.

The name of the objective function is optional and may be omitted (together with the semicolon that follows it). In this case the default name '`obj`' is assigned to the objective function.

If the very first sign s is omitted, the sign plus is assumed. Other signs cannot be omitted.

If some objective coefficient c is omitted, 1 is assumed.

Symbolic names x used to denote variables are recognized by context and therefore needn't to be declared somewhere else.

Here is an example of the objective function definition:

```
Minimize Z : - x1 + 2 x2 - 3.5 x3 + 4.997e3x(4) + x5 + x6 +
   x7 - .01x8
```

C.3 Constraints section

The constraints section must follow the objective function definition. It defines a system of equality and/or inequality constraints.

The constraint section has the following form:

`subject to`
constraint$_1$
constraint$_2$
...
constraint$_m$

where *constraint*$_i, i = 1, \ldots, m$, is a particular constraint definition.

Each constraint definition can be continued on as many text lines as desired. However, each constraint definition must begin on a new line except the very first constraint definition which can begin on the same line as the keyword '`subject to`'.

Constraint definitions have the following form:

$$r : s\ c\ x\ s\ c\ x\ \ldots\ s\ c\ x\ \left\{ \begin{array}{c} \texttt{<=} \\ \texttt{>=} \\ \texttt{=} \end{array} \right\}\ b$$

where r is a symbolic name of a constraint, s is a sign `+` or `-`, c is a numeric constant that denotes a constraint coefficient, x is a symbolic name of a variable, b is a right-hand side.

The name r of a constraint (which is the name of the corresponding auxiliary variable) is optional and may be omitted (together with the semicolon that follows it). In this case the default names like '`r.nnn`' are assigned to unnamed constraints.

The linear form $s\ c\ x\ s\ c\ x\ \ldots\ s\ c\ x$ in the left-hand side of a constraint definition has exactly the same meaning as in the case of the objective function definition (see above).

After the linear form one of the following delimiters that indicate the constraint sense must be specified:

`<=` means 'less than or equal to'
`>=` means 'greater than or equal to'
`=` means 'equal to'

The right hand side b is a numeric constant with an optional sign.

Here is an example of the constraints section:

```
Subject To
   one: y1 + 3 a1 - a2 - b >= 1.5
   y2 + 2 a3 + 2
      a4 - b >= -1.5
   two : y4 + 3 a1 + 4 a5 - b <= +1
   .20y5 + 5 a2 - b = 0
   1.7 y6 - a6 + 5 a777 - b >= 1
```

(Should note that it is impossible to express ranged constraints in the CPLEX LP format. Each a ranged constraint can be coded as two constraints with identical linear forms in the left-hand side, one of which specifies a lower bound and other does an upper one of the original ranged constraint.)

C.4 Bounds section

The bounds section is intended to define bounds of variables. This section is optional; if it is specified, it must follow the constraints section. If the bound section is omitted, all variables are assumed to be non-negative (i.e. that they have zero lower bound and no upper bound).

The bounds section has the following form:

`bounds`
definition$_1$
definition$_2$
...
definition$_p$

where *definition*$_k, k = 1, \ldots, p$, is a particular bound definition.

Each bound definition must begin on a new line[4] except the very first bound definition which can begin on the same line as the keyword '`bounds`'.

Syntactically constraint definitions can have one of the following six forms:

x `>=` l	specifies a lower bound
l `<=` x	specifies a lower bound
x `<=` u	specifies an upper bound
l `<=` x `<=` u	specifies both lower and upper bounds
x `=` t	specifies a fixed value
x `free`	specifies free variable

[4]The GLPK implementation allows several bound definitions to be placed on the same line.

where x is a symbolic name of a variable, l is a numeric constant with an optional sign that defines a lower bound of the variable or `-inf` that means that the variable has no lower bound, u is a numeric constant with an optional sign that defines an upper bound of the variable or `+inf` that means that the variable has no upper bound, t is a numeric constant with an optional sign that defines a fixed value of the variable.

By default all variables are non-negative, i.e. have zero lower bound and no upper bound. Therefore definitions of these default bounds can be omitted in the bounds section.

Here is an example of the bounds section:

```
Bounds
   -inf <= a1 <= 100
   -100 <= a2
   b <= 100
   x2 = +123.456
   x3 free
```

C.5 General, integer, and binary sections

The general, integer, and binary sections are intended to define some variables as integer or binary. All these sections are optional and needed only in case of MIP problems. If they are specified, they must follow the bounds section or, if the latter is omitted, the constraints section.

All the general, integer, and binary sections have the same form as follows:

$$\left\{\begin{matrix}\texttt{general}\\ \texttt{integer}\\ \texttt{binary}\end{matrix}\right\}$$
$$x_1$$
$$x_2$$
$$\dots$$
$$x_q$$

where x_k is a symbolic name of variable, $k = 1, \dots, q$.

Each symbolic name must begin on a new line[5] except the very first symbolic name which can begin on the same line as the keyword '`general`', '`integer`', or '`binary`'.

[5] The GLPK implementation allows several symbolic names to be placed on the same line.

If a variable appears in the general or the integer section, it is assumed to be general integer variable. If a variable appears in the binary section, it is assumed to be binary variable, i.e. an integer variable whose lower bound is zero and upper bound is one. (Note that if bounds of a variable are specified in the bounds section and then the variable appears in the binary section, its previously specified bounds are ignored.)

Here is an example of the integer section:

```
Integer
   z12
   z22
   z35
```

C.6 End keyword

The keyword '`end`' is intended to end the LP file. It must begin on a separate line and no other elements (except comments and blank lines) must follow it. Although this keyword is optional, it is strongly recommended to include it in the LP file.

C.7 Example of CPLEX LP file

Here is a complete example of CPLEX LP file that corresponds to the example given in Section B.11, page 198.

```
\* plan.lp *\

Minimize
   value: .03 bin1 + .08 bin2 + .17 bin3 + .12 bin4 + .15 bin5 +
          .21 alum + .38 silicon

Subject To
   yield:     bin1 +     bin2 +     bin3 +     bin4 +     bin5 +
              alum +     silicon                                 =  2000

   fe:    .15 bin1 + .04 bin2 + .02 bin3 + .04 bin4 + .02 bin5 +
          .01 alum + .03 silicon                                 <=   60

   cu:    .03 bin1 + .05 bin2 + .08 bin3 + .02 bin4 + .06 bin5 +
          .01 alum                                               <=  100

   mn:    .02 bin1 + .04 bin2 + .01 bin3 + .02 bin4 + .02 bin5   <=   40
```

```
   mg:    .02 bin1 + .03 bin2                      + .01 bin5   <=   30

   al:    .70 bin1 + .75 bin2 + .80 bin3 + .75 bin4 + .80 bin5 +
          .97 alum                                              >= 1500

   si1:   .02 bin1 + .06 bin2 + .08 bin3 + .12 bin4 + .02 bin5 +
          .01 alum + .97 silicon                                >=  250

   si2:   .02 bin1 + .06 bin2 + .08 bin3 + .12 bin4 + .02 bin5 +
          .01 alum + .97 silicon                                <=  300

Bounds
          bin1 <=  200
          bin2 <= 2500
   400 <= bin3 <=  800
   100 <= bin4 <=  700
          bin5 <= 1500

End

\* eof *\
```