Using CPLEX

CPLEX is optimization software developed and sold by ILOG, Inc. It can be used to solve a variety of different optimization problems in a variety of computing environments. Here we will discuss only its use to solve "linear programs" and will discuss only its use in interactive mode.

Getting Started: Solving a Linear Program using CPLEX

Here is an example of a simple *linear program* (LP):

$$
\begin{aligned}
&\text{Maximize} \\
&\quad 20\, x_1 + 30\, x_2 \\
&\text{Subject To} \\
&\quad 4\, x_1 + 5\, x_2 \le 30 \\
&\quad 6\, x_1 + 4\, x_2 \le 40 \\
&\quad x_1 + 2\, x_2 \le 10 \\
&\quad x_1 \ge 0 \\
&\quad x_2 \ge 0
\end{aligned}
$$

This LP has two *variables* and five *constraints*[1]. The constraints limit the values that can be taken on by the variables. A *feasible solution* is an assignment of the values of the variables such that the constraints are satisfied. An example of a feasible solutions is $x_1=x_2=0$. Another example of a feasible solution is $x_1=0$ and $x_2=5$. The expression "$20x_1+30x_2$" is called the *objective function*. An *optimal solution* is a feasible solution that maximizes the objective function. The unique optimal solution for this problem is $x_1=x_2=3•$. This solution yields an objective value of **166•**. By comparison, the feasible solution $x_1=x_2=0$ has objective value **0**, and the feasible solution $x_1=0$ and $x_2=5$ has objective value **150.**

Note that two of the constraints in this problem are particularly simple, limiting the values of single variables: $x_1 \ge 0$ and $x_2 \ge 0$. Such constraints are called *bounds*. Bounds play a special role in linear programming, and are usually treated separately by the underlying solution methods. The particular bounds in this simple example are quite common in linear programming: The variables are restricted to be *nonnegative*.

CPLEX can be used to solve the above problem. There are several ways to do so. The simplest is to create a text file that defines the LP (using a text editor such as *Microsoft Notepad*). This file can then be read by CPLEX and the associated LP solved by issuing the appropriate CPLEX command. The text format used to express such LPs is quite simple, and attempts to closely mirror the way in which the problem has been

[1] One normally refers to this problem as having three constraints in addition to the "bounds on the variables". This distinction is explained later in the text.

expressed above. In this case, you could create a file with the name "example.lp"[2], the contents of which are the following:

```
Maximize
   20 x1 + 30 x2
Subject To
   4 x1 + 5 x2 <= 30
   6 x1 + 4 x2 <= 40
     x1 + 2 x2 <= 10
            x1 >= 0
            x2 >= 0
End
```

This file format is quite forgiving. For example, the above file could also be written in the following form:

```
Max  20 x1
  + 30
   x2
st  4 x1 + 5 x2 < 30
  6 x1
  + 4 x2 <= 40
x1 + 2 x2 <= 10
```

It is beyond the scope of this document to give a complete description of the allowed boundaries of this file format. See the CPLEX documentation for more details. For our purposes, the following short list of conditions should be sufficient:

1. Variable names can be any length but may contain no white space. They must begin with a letter or "symbol" (the allowed symbols are listed in the documentation -- using a letter will always be safe). The remaining characters can be any letter or number (and most symbols).

2. Coefficients multiplying a variable must come before the variable. Where no coefficient is given, it is assumed to be1.

3. The objective must be specified before any constraints are specified.

4. No variables can occur on the right-hand side of a constraint.

5. ‘<’ is interpreted as ‘<=’ and ‘>’ is interpreted as ‘>=’: Strict inequalities such as ‘<’ and ‘>’ are not allowed in LPs.

6. Allowed senses for the objective are: Maximize and Minimize. Any variation of these designations with capitals and non-capitals and using at least the first three letters of the given choice will suffice.

---

[2] The “.lp” file-name extension is important here. CPLEX can read LPs in several different formats. By specifying “.lp” you are telling CPLEX to expect the format being discussed here.

7. "Subject To" demarks the beginning of the constraints. "ST" can also be used. Again, capitalization is not significant.

8. If no bounds are specified for a variable, then it is assumed that the variable is nonnegative.

Let us assume that the linear program specified above has been stored in a file named "`example.lp`". Let us also suppose that the name of the CPLEX executable is "`cplex.exe`". Begin by opening a command window: From the start menu

`All Programs → Accessories → Command Prompt`

or from the start menu

`Run`

and enter the command "**cmd**". Then "**cd**" to a directory from which the cplex.exe command will be recognized and that contains the "`example.lp`" file. Enter the command

`CPLEX`

at the command line. The following is the session that followed when I solved "`example.lp`" (the bold-face text is what I typed in):

```
Welcome to CPLEX Interactive Optimizer 8.1.0
  with Simplex, Mixed Integer & Barrier Optimizers
Copyright (c) ILOG 1997-2002
CPLEX is a registered trademark of ILOG

Type 'help' for a list of available commands.
Type 'help' followed by a command name for more
information on commands.

CPLEX>  read example.lp
Problem 'example.lp' read.
Read time =    0.00 sec.
CPLEX> display problem all
Maximize
 obj: 20 x1 + 30 x2
Subject To
 c1: 4 x1 + 5 x2 <= 30
 c2: 6 x1 + 4 x2 <= 40
 c3: x1 + 2 x2 <= 10
Bounds
 All variables are >= 0.
CPLEX> optimize
Tried aggregator 1 time.
No LP presolve or aggregator reductions.
Presolve time =    0.02 sec.

Iteration log . . .
Iteration:     1   Dual infeasibility =             0.000000
```

```
Iteration:     2   Dual objective     =          180.000000

Dual simplex - Optimal:  Objective =   1.6666666667e+002
Solution time =    0.09 sec.  Iterations = 3 (1)

CPLEX>display solution variables -
Variable Name           Solution Value
x1                            3.333333
x2                            3.333333
CPLEX>quit
```

At least one of the steps above could have been skipped: "**`display problem all`**". In fact, this is a dangerous command, since it will display the entire resident LP. Since real linear programs often have thousands of variables and constraints, such a display would be of little value (and very time consuming). Here I have used this command simply to verify that what CPLEX read in is what I intended to be read in. Note that when the problem is displayed, the objective is given a name "`obj:`", and the constraints are also given names. For example, the first constraint is named "`c1:`". If we had specified names in the input file, these would have been used instead.

The command "**`optimize`**" tells CPLEX to solve the model. Here it was solved using the "dual simplex method". This method required three "iterations" and 0.09 seconds to solve the problem. The messages

```
Tried aggregator 1 time.
No LP presolve or aggregator reductions.
Presolve time =    0.02 sec.
```

refer to "problem reductions" that CPLEX attempted to apply prior to solving this problem in order to simplify its solution. The details of this step need not concern us at this point.

Finally, we note that, whenever CPLEX is run, a log file with default name "`cplex.log`" is written to the disk in the directory where CPLEX was run. This file can be read with *Notepad*. For the session above, the contents of the CPLEX log file are the following:

```
Log started (V8.1.0) Sat Feb 22 17:16:25 2003

Problem 'example.lp' read.
Read time =    0.00 sec.
Maximize
 obj: 20 x1 + 30 x2
Subject To
 c1: 4 x1 + 5 x2 <= 30
 c2: 6 x1 + 4 x2 <= 40
 c3: x1 + 2 x2 <= 10
Bounds
 All variables are >= 0.
Tried aggregator 1 time.
No LP presolve or aggregator reductions.
Presolve time =    0.00 sec.

Iteration log . . .
```

```
Iteration:     1   Dual infeasibility =            0.000000
Iteration:     2   Dual objective     =          180.000000

Dual simplex - Optimal:  Objective =   1.6666666667e+002
Solution time =    0.02 sec.  Iterations = 3 (1)

Variable Name           Solution Value
x1                            3.333333
x2                            3.333333
```

CPLEX: Additional Commands

CPLEX provides a modest, interactive help facility. At the command prompt simply type the command "**help**" and the following will be displayed:

```
CPLEX> help

add             add constraints to the problem
baropt          solve using barrier algorithm
change          change the problem
display         display problem, solution, or parameter settings
enter           enter a new problem
help            provide information on CPLEX commands
mipopt          solve a mixed integer program
netopt          solve the problem using network method
optimize        solve the problem
primopt         solve using the primal method
quit            leave CPLEX
read            read problem or basis information from a file
set             set parameters
tranopt         solve using the dual method
write           write problem or solution info. to a file
xecute          execute a command from the operating system

Enter enough characters to uniquely identify commands & options.
Commands can be entered partially (CPLEX will prompt you for
further information) or as a whole.
```

No attempt will be made here to explain all of these commands. Only a subset of them will be be used in this course. Among the commands most likely to be used are: **`baropt, display, mipopt, optimize, primopt, quit, read, set, tranopt,`** and **`xecute`**. For example, the commands **`baropt, primopt,`** and **`tranopt`** all refer to different algorithms for solving LPs. It is not the purpose of this course to discuss these algorithms in any detail, but high-level explanations will be given. All three algorithms find significant use in practice, and are each particularly useful in particular applications.   As a simple exercise, it is instructive to apply each of these algorithms to the example in the previous section – be sure to reread the file before each attempt to solve "`example.lp`"; otherwise, CPLEX will remember that the problem has already been solved and nothing will happen.

The **`display`** command has many subcommands the will be used throughout the course. Simply entering the command **`diplay`** will yield a list of additional options:

```
CPLEX> display
```

```
Display Options:

iis                 display infeasibility diagnostics (IIS
constraints)
problem             display problem characteristics
sensitivity         display sensitivity analysis
settings            display parameter settings
solution            display existing solution

Display what:
```

If you now enter, say, **`solution`** (assuming that a problem is resident and has been solved) you will see the display

```
Display what: solution

Display Solution Options:

basis               display a range of basic constraints or
variables
bestbound           display the current MIP best bound
dual                display a set of solution dual values
kappa               display the condition number of the basis matrix
objective           display solution objective value
quality             display quality of solution
reduced             display a set of solution reduced costs
slacks              display a set of solution slack values
variables           display a set of solution variable values

Display which part of the solution:
```

In the example what we used was "variables –", where the symbol "-" indicates that the values of all variables should be displayed.

CPLEX: A more Complex Example

Consider the LP specified by the following file contents (we have named the file "`alldiet.lp`"):

```
\Problem name: alldiet.lp

Minimize
 DOLLARS: 10.89 pizza + 0.79 FRFries + 2.89 M.cherry + 2.59 MILK + 2.69 C.Milk
 + 0.47 P.towels + 1.49 !SUGARS! + 1.27 cereal + 3.29 SixPackX + 3.29 SixPackY
Subject To
 aluminum: 137 pizza + 71.4 SixPackX + 71.4 SixPackY >= 27230
 Vacuous:  >= 0
 RED#2:    1.9 pizza + 18 M.cherry + 1.9 P.towels + 1.6 !SUGARS! + 1.6 cereal
           + 1.7 SixPackX + 1.7 SixPackY - RgRED#2  = 18390
 salt:     67.2 pizza + 36 FRFries + 2.6 M.cherry + 3.4 P.towels + 4.3 !SUGARS!
           + 4.3 cereal + 31.4 SixPackX + 31.4 SixPackY + 0.000067 Air >= 21740
 fat:      41.8 pizza + 210.6 FRFries + 243 MILK + 223 C.Milk + 1.3 SixPackX
           + 1.2 SixPackY - Rgfat  = 30270
 Fiber:    3.4 pizza + 9.3 FRFries + 0.084 M.cherry + 0.45 P.towels
           + 1.78 !SUGARS! + 1.78 cereal - RgFiber  = 4789
 CALCIUM:  45.2 MILK + 43.2 C.Milk - RgCALCIUM  = 11460
 Sparkle:  - pizza - FRFries <= 10000
 Dirt:     pizza + FRFries >= -10000
Bounds
      FRFries >= 20
 500 <= M.cherry <= 800
 20 <= MILK <= 5000
 0 <= C.Milk <= 8888
```

```
      P.towels >= -27000
      !SUGARS! >= 144
 0 <= SixPackX <= 99999999
      SixPackY = 24
      Air Free
-10 <= RgRED#2 <= 0
 0 <= Rgfat <= 99999
-34789 <= RgFiber <= 0
 0 <= RgCALCIUM <= 240
End
```

We will leave it to interested readers to interpret this LP. We use it here simply to illustrate several additional features of CPLEX. First, note the use of the comment character '\' in the file: All entries on any line containing this character and following the character, including the character itself, are ignored when the file is read.

Reading in the model and displaying statistics about its size, we obtain:

```
CPLEX> read alldiet.lp
Problem 'alldiet.lp' read.
Read time =    0.00 sec.
CPLEX> display problem stats
Problem name: alldiet.lp
Constraints          :       9  [Less: 1,  Greater: 4,  Equal: 4]
Variables            :      15  [Nneg: 2,  Fix: 1,  Box: 8,  Free: 1,
                                 Other: 3]
Constraint nonzeros  :      41
Objective  nonzeros  :      10
RHS        nonzeros  :       8
CPLEX>
```

Thus, this LP has nine constraints and 15 variables. Among these 15 variables, two are nonnegative, one is *fixed* ('`SixPackY`'), eight are *boxed*, meaning that they have finite lower and upper bounds (a nonnegative variable has a finite lower bound, 0, but no upper bound), one is *free* ('`Air`' – with no upper or lower bound), and the three other variables do not fit any of these classifications (for example, '`P.towels`'). Solving this model using the **`primopt`** command, and displaying two kinds of solutions information, we obtain:

```
CPLEX> primopt
Tried aggregator 1 time.
LP Presolve eliminated 4 rows and 4 columns.
Reduced LP has 5 rows, 11 columns, and 22 nonzeros.
Presolve time =    0.00 sec.

Iteration log . . .
Iteration:     1    Scaled infeas =           186.251095
Switched to devex.
Iteration:     3    Objective     =          5744.600764

Primal simplex - Optimal:  Objective =   5.2208992665e+003
Solution time =    0.00 sec.  Iterations = 6 (2)

CPLEX> display solution variables -
Variable Name          Solution Value
FRFries                     20.000000
M.cherry                   800.000000
MILK                       253.539823
P.towels                  1632.245614
!SUGARS!                   144.000000
SixPackX                   357.372549
SixPackY                    24.000000
```

```
RgRED#2                        -10.000000
Air                      11881595.120373
Rgfat                       36045.561305
RgFiber                     -3544.969474
All other variables in the range 1-15 are zero.
CPLEX> display solution slacks -
Constraint Name                     Slack Value
slack Sparkle                      10020.000000
slack Dirt                        -10020.000000
All other slacks in the range 1-9 are zero.
CPLEX>
```

Note that four constraints and four variables were logically removed before the "primal simplex algorithm", which is the algorithm used by the **`primopt`** command, was applied to solve the problem[3]. The algorithm took six iterations[4] to solve the model, two of which occurring before the first feasible solution was found. Eleven of the 15 variables take nonzero values in the optimal solution. Finally, the values of the *slacks* means that the optimal solution satisfies all constraints at equality except for the final two, labeled `Sparkle` and `Dirt`[5].

[3] In spite of the fact that the algorithm solved a reduced problem, the answer to the original problem is what is provided.

[4] The line "`Switched to devex`." in the output refers to a very technical feature of the primal simplex implementation in CPLEX.

[5] The observant reader will note that these two constraints are actually equivalent. It will therefore come as no surprise that one of these two constraints was among those eliminated before the solution algorithms started.