cplexAPI – Quick Start

Gabriel Gelius-Dietrich

August 22, 2011

1 Introduction

The package *cplexAPI* provides a low level interface to the C API of IBM® ILOG® CPLEX®[1]. The package *cplexAPI* requires a working installation of IBM® ILOG® CPLEX®.

2 Installation

See `INSTALL` for installation instructions and platform specific details.

3 Usage

3.1 Creating and solving a linear optimization problem

In the following, an example lp-problem will be created and solved:

maximize

$$z = 5x_1 + 4x_2 + 3x_3$$

subject to

$$\begin{aligned} 2x_1 + 3x_2 + \ x_3 &\leq 5 \\ 4x_1 + \ x_2 + 2x_3 &\leq 11 \\ 3x_1 + 4x_2 + 2x_3 &\leq 8 \end{aligned}$$

where all variables are non-negative

$$x_1 \geq 0,\ x_2 \geq 0,\ x_3 \geq 0$$

Load the library.

```
> library(cplexAPI)
```

Open a IBM® ILOG® CPLEX® environment.

[1] IBM® ILOG® CPLEX® version $\geq$ 12.1 from the IBM Academic Initiative
`https://www.ibm.com/developerworks/university/academicinitiative/`

```
> env <- openEnvCPLEX()
```

Create a problem object.

```
> prob <- initProbCPLEX(env)
```

Assign a name to the problem object.

```
> chgProbNameCPLEX(env, prob, "sample")
```

```
[1] 0
```

Prepare data structures for the problem object. Number of columns and rows.

```
> nc <- 3
> nr <- 3
```

Objective function.

```
> obj <- c(5, 4, 3)
```

Right hand side.

```
> rhs <- c(5, 11, 8)
```

Sense of the right hand side.

```
> sense <- rep("L", 3)
```

Variable lower bounds.

```
> lb <- rep(0, 3)
```

Variable upper bounds.

```
> ub <- rep(CPX_INFBOUND, 3)
```

Column and row names.

```
> cn <- c("x1", "x2", "x3")
> rn <- c("q1", "q2", "q3")
```

The constraint matrix is passed in column major order format. **Be careful here:** all indices start with 0! Begin indices of rows.

```
> beg <- c(0, 3, 6)
```

Number of non-zero elements per row.

```
> cnt <- rep(3, 3)
```

Column indices.

```
> ind <- c(0, 1, 2, 0, 1, 2, 0, 1, 2)
```

Non-zero elements.

```
> val <- c(2, 4, 3, 3, 1, 4, 1, 2, 2)
```

Load problem data.

```
> copyLpwNamesCPLEX(env, prob, nc, nr, CPX_MAX, obj, rhs, sense,
+     beg, cnt, ind, val, lb, ub, NULL, cn, rn)

[1] 0
```

Solve the problem using the simplex algorithm.

```
> lpoptCPLEX(env, prob)

[1] 0
```

Retrieve solution after optimization.

```
> solutionCPLEX(env, prob)

$lpstat
[1] 1

$objval
[1] 13

$x
[1] 2 0 1

$pi
[1] 1 0 1

$slack
[1] 0 1 0

$dj
[1]  0 -3  0
```

Write the problem to file `prob.lp` in lp format.

```
> writeProbCPLEX(env, prob, "prob.lp")

[1] 0
```

Read problem from file `prob.lp` in lp format.

```
> lp <- initProbCPLEX(env)
> readCopyProbCPLEX(env, lp, "prob.lp")
[1] 0
```

Free memory, allacated to the problem object.

```
> delProbCPLEX(env, prob)
[1] 0
> closeEnvCPLEX(env)
[1] 0
```

3.2 Setting control prarmeters

Open a new environment.

```
> pe <- openEnvCPLEX()
```

All parameters and possible values are described in the IBM® ILOG® CPLEX® documentation. All parameters can be set in *cplexAPI*; the parameters names are the same as in IBM® ILOG® CPLEX®. For example, if one wants to use the debugging routines, the 'messages to screen switch' must be set to 1.

```
> setIntParmCPLEX(pe, CPX_PARAM_SCRIND, CPX_ON)
[1] 0
```

Do not use advanced start information.

```
> setIntParmCPLEX(pe, CPX_PARAM_ADVIND, 0)
[1] 0
```

Lower the feasibility tolerance.

```
> setDblParmCPLEX(pe, CPX_PARAM_EPRHS, 1e-09)
[1] 0
```

Retrieve parameters which are not set at their default values.

```
> (parm <- getChgParmCPLEX(pe))
[1] 1001 1016 1035
```

Retrieve names of theese parameters.

```
> sapply(parm, getParmNameCPLEX, env = pe)
[1] "CPX_PARAM_ADVIND" "CPX_PARAM_EPRHS"  "CPX_PARAM_SCRIND"
```

Close the envoronment.

```
> closeEnvCPLEX(pe)
[1] 0
```