AMPL – CPLEX para resolver problemas lineales enteros de optimización

Víctor Bucarey López

IN3701 – Modelamiento y Optimización

Otoño 2014

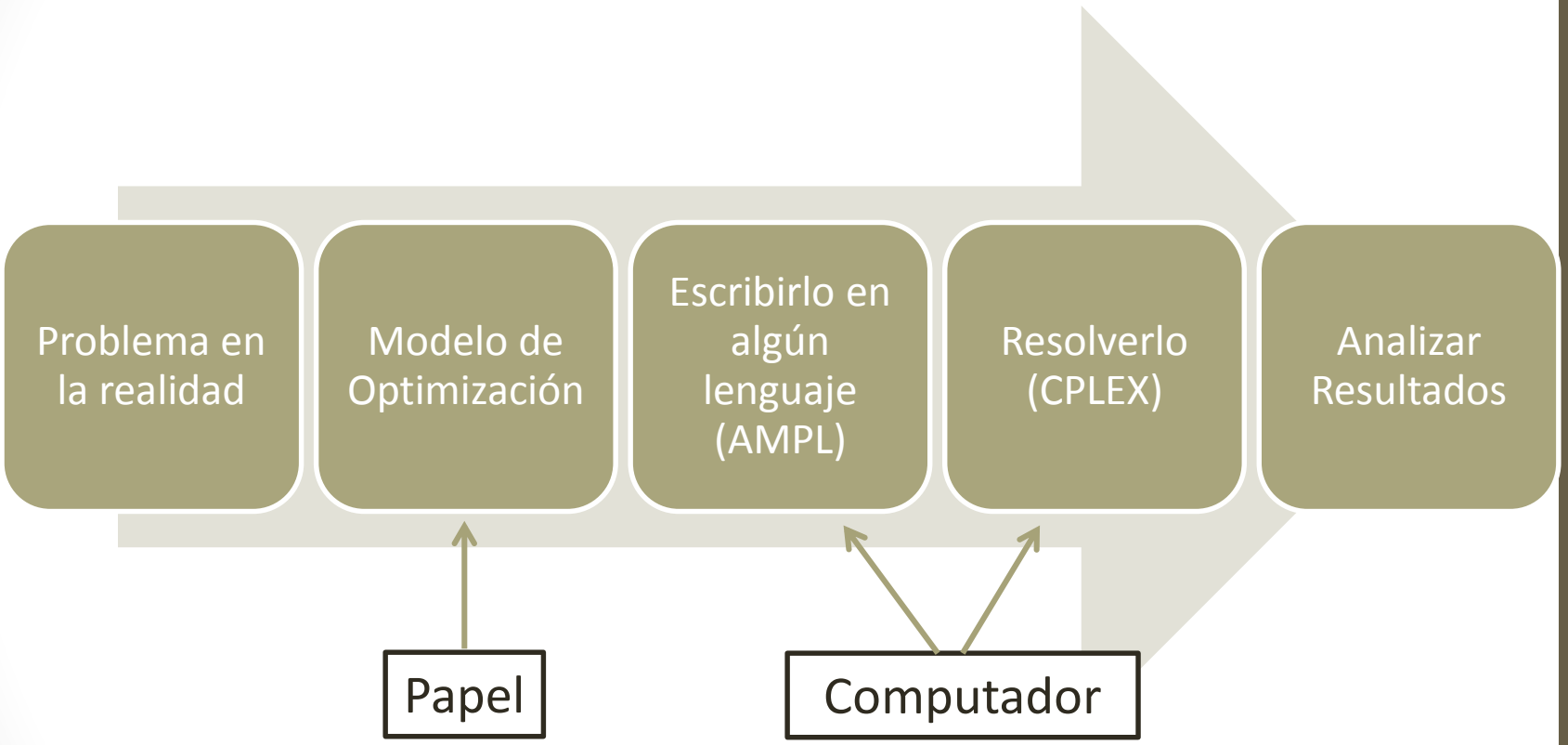
Introducción

- **“AMPL** is a comprehensive and powerful algebraic modeling **language for linear and nonlinear optimization problems**, in discrete or continuous variables.”

  Página Web: http://www.ampl.com

- **CPLEX** es un paquete de optimización que tiene implementado algoritmos tales como SIMPLEX y Branch and Bound para la resolución de problemas lineales y lineales enteros.

Introducción

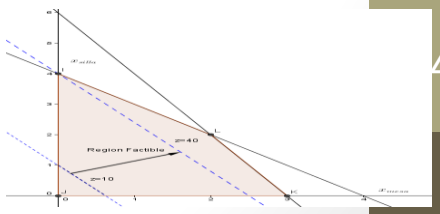


Problemas de programación lineal entera

- Recordamos que los problemas de programación lineal (y lineal entera) se componen de cinco elementos:

  - Conjuntos: Son los conjuntos que se utilizarán durante la modelación. Sirven para escribir el modelo de forma más compacta y más entendible, y son especialmente útiles para problemas muy grandes.

  - Parámetros: Son todos las partes exógenas del problema, es decir, que no pueden ser cambiadas, o no dependen de las decisiones tomadas.

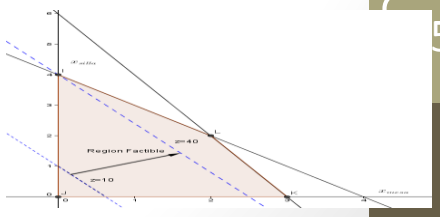


Problemas de programación lineal entera

- Variables de decisión: Corresponden a todos los aspectos del problema, sobre los cuales se pueden tomar decisiones. Se intenta determinar sus valores mediante la resolución del problema.

- Restricciones: Son todas las características que cualquier solución debe tener, en particular la solución óptima. Estas ecuaciones son las que nos definirán el **"conjunto"** de soluciones que nos interesan.

- Función Objetivo:Corresponde al valor que se quiere optimizar, que es análogo a nuestro criterio para decidir cuál solución es mejor o peor.



Ejemplo

Suponga que usted es dueño de una fábrica de muebles. Usted posee dos tipos de piezas de madera, piezas grandes y piezas pequeñas. Usted puede hacer dos tipos de productos: mesas y sillas de la siguiente forma:

- Una mesa se hace con dos piezas pequeñas y dos piezas grandes.
- Una silla se hace con dos piezas pequeñas y una pieza grande.
- El costo de una pieza pequeña es de $3 y el de una pieza grande es de $5.
- El precio de una mesa es de $32 y el de una silla $21.

Usted posee 6 piezas grandes y 8 piezas pequeñas. Como dueño de la fabrica lo que ustedes quieren hacer es maximizar sus ganancias (los ingresos por las ventas menos lo que le cuesta hacer lo que vendieron.)

Escribiendo AMPL

Conjuntos de Índices

$$I = \text{Conjunto de Productos} = \{\text{mesa, silla}\}$$
$$J = \text{Conjuntos de Materiales} = \{\text{Pieza Grande, Pieza Pequeña}\}$$

Parámetros

	Cantidad
Piezas Pequeñas	8
Piezas Grandes	6

	Costo
Piezas Pequeñas	3
Piezas Grandes	5

Cantidad de piezas necesarias para hacer los productos productos		
	Mesa	Silla
Piezas Pequeñas	2	2
Piezas Grandes	2	1

	Precio
Mesa	32
Silla	21

Escribiendo AMPL

Variables de Decisión

$$x_{mesa} = \text{Cantidad de mesas a producir}$$
$$x_{silla} = \text{Cantidad de sillas a producir}$$

$$x_i = \text{Cantidad del Producto i a producir}, \qquad \forall\, i \in I$$

Restricciones

$$2x_{mesa} + 2x_{silla} \leq 8$$
$$2x_{mesa} + 1x_{silla} \leq 6$$
$$x_{mesa}, x_{silla} \geq 0$$

Función Objetivo

$$\max Ganancia: \; 16x_{mesa} + 10x_{silla}$$

Escribiendo AMPL

Por lo que debemos escribir:

$$\max \text{Ganancia: } 16x_{mesa} + 10x_{silla}$$

$$2x_{mesa} + 2x_{silla} \leq 8$$
$$2x_{mesa} + 1x_{silla} \leq 6$$
$$x_{mesa}, x_{silla} \geq 0$$

$$\max \text{Ganancia: } \sum_{i \in Productos} g_i x_i$$

s.a

$$\sum_{i \in Productos} a_{ij} x_i \leq c_j \qquad \forall j \in Piezas$$

$$x_i \geq 0 \quad \forall i \in Productos$$

Escribiendo AMPL

Los modelos de AMPL se hacen con 3 archivos, los cuales pueden ser escritos en cualquier editor de texto, por ejemplo en el block de notas:

- Archivos del Modelo ***.mod**
- Archivos de Datos ***.dat**
- Archivos de Rutinas ***.run**

Escribiendo AMPL

Archivo *.mod:

En este archivo se programará el modelo, tal como lo escribimos en las secciones anteriores. Se declararán los parámetros y variables, pero no se les asignará valores. Este archivo se puede escribir en cualquier editor de texto y se debe guardar con la extensión ***.mod.**

Escribiendo AMPL

Archivo *.dat:

En este archivo se asignarán **valores a los parámetros del problema.** Lo potente de utilizar esta modalidad, es que un mismo modelo se puede ejecutar con múltiples instancias solo llamando a archivos de datos diferentes, sin tener que modificar el archivo .mod. Este archivo también se escribe en cualquier editor de texto y se debe guardar con la extensión .dat.

Escribiendo AMPL

Archivo*.run:

Este archivo es opcional pero se recomienda su uso, en él se escribe la secuencia de comandos que se aplicará en la pantalla de AMPL. También se escribe en cualquier editor de texto y se guarda bajo la extensión .run.

Escribiendo AMPL

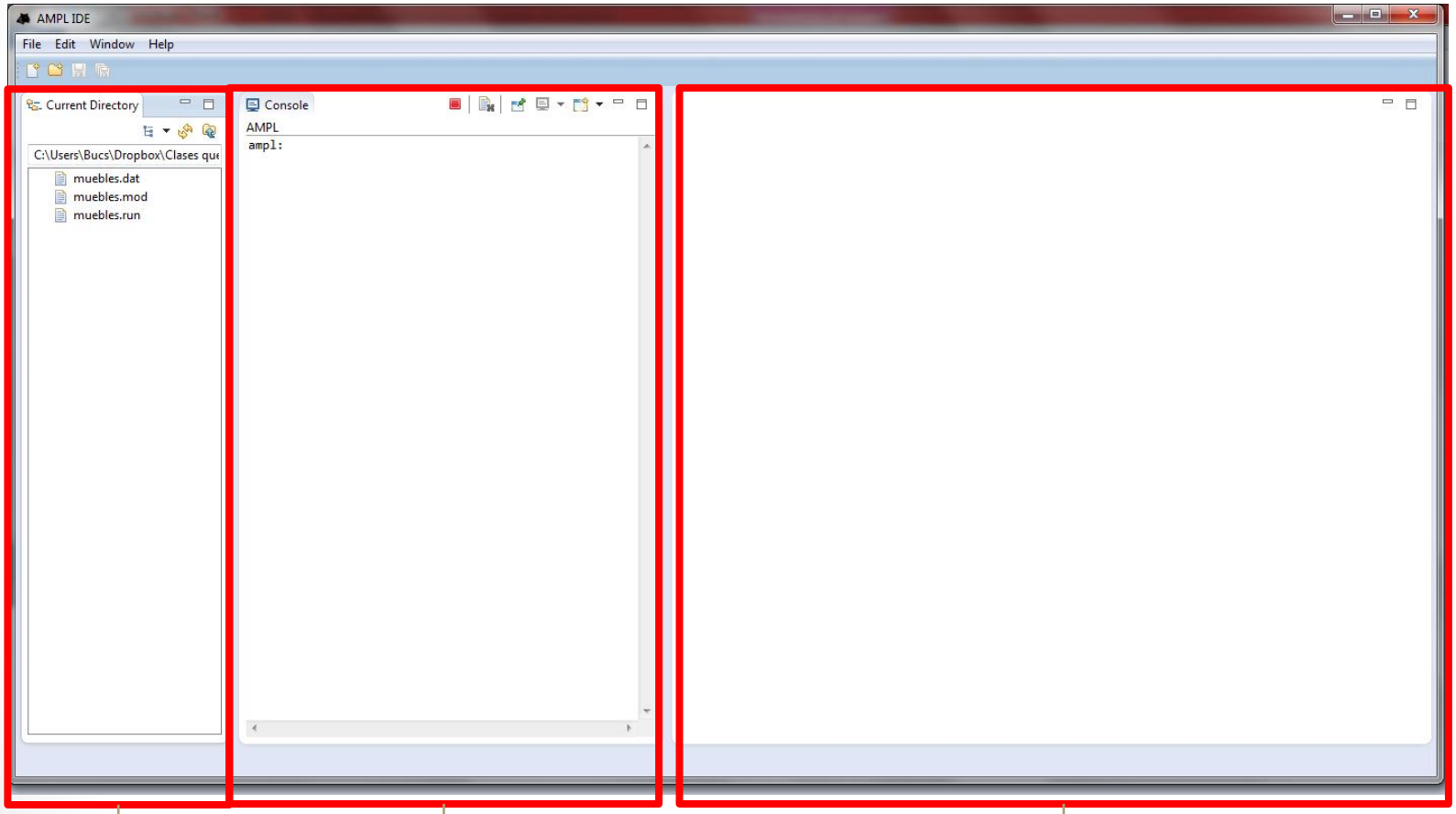
Para guardar en el block de notas archivos con alguna extensión debes poner el menú Archivo, luego "**Guardar como…"** y poner en el nombre de archivo el nombre de tu archivo con la extensión que corresponda al final y poner todo entre comillas, por ejemplo *.mod :

"FabricaDeMueble.mod"

Si utilizan la interfaz de AMPL, este crea los archivos automáticamente con su extensión.

Escribiendo AMPL

AMPL IDE: Entorno de Desarrollo Integrado



Escribiendo AMPL

Paso1: Escribir el archivo *.mod:

1. Declaramos los conjuntos que están en nuestro modelo:

set Productos;

set Piezas;

En general:

set *nombre_conjunto*;

Escribiendo AMPL

2. Ahora declararemos que parámetros están en el modelo. Para eso ocupamos el comando param y el nombre del parámetro.
En este modelo:

param g {Productos};

param c {Piezas};

param a {Productos, Piezas};

Cantidad de piezas necesarias para hacer los productos productos		
	Mesa	Silla
Piezas Pequeñas	2	2
Piezas Grandes	2	1

Entre medio de los paréntesis de llave se encuentran los conjuntos en los cuales están definidos los parámetros.

Escribiendo AMPL

3. Ahora declaramos nuestra variable. Para ello ponemos el comando **var** y después el nombre de nuestra variable, que en este caso será x, y luego entre paréntesis el conjunto al que pertenecen.

```
var x{Productos} >=0;
```

x_i = Cantidad del Producto i a producir, $\forall\ i \in I$

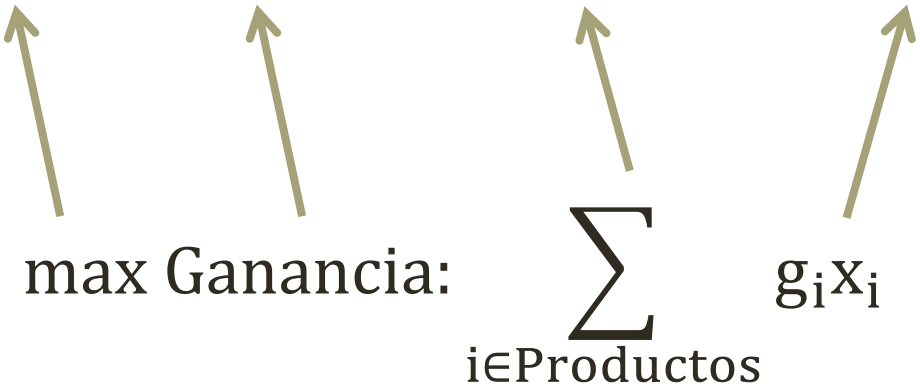
Otros Ejemplos:

```
var y{A,B} binary;
var z{A} integer >= 0;
```

Escribiendo AMPL

4. La **función objetivo** puede ser de minimización, en tal caso se pone la palabra ***minimize***, o de maximización como este ejemplo, en tal caso se coloca la palabra ***maximize***. Luego se coloca un nombre a la función objetivo, en este caso le pusimos Ganancia, y la expresión que representa la función objetivo:

```
maximize Ganancia: sum{i in Productos} g[i]*x[i];
```



Escribiendo AMPL

5. Ahora solo hace falta escribir las restricciones

subject to CantidadDePiezas{j in Piezas}:

sum{i in Productos} a[i,j]*x[i] <= c[j] ;

$$\sum_{i \in \text{Productos}} a_{ij} x_i \leq c_j \qquad \forall j \in \text{Piezas}$$

En general:

subject to nombreRest{conjunto}:

expresion1 (<= o = o >=) expresión2;

Escribiendo AMPL

muebles.dat | muebles.mod | muebles.run

```
#Conjuntos
set Productos;
set Piezas;

#Parámetros
param g {Productos};
param b {Piezas};
param c {Piezas};
param a {Productos, Piezas};

#Variables
var x{Productos} >=0;

#Funcion Objetivo
maximize Ganancia: sum{i in Productos} g[i]*x[i];

#Restricciones
subject to CantidadDePiezas{j in Piezas}: sum{i in Productos} a[i,j]*x[i] <= c[j] ;
```

Con el símbolo # se crean líneas de comentario, las cuales son muy útiles para ordenar la programación.

Escribiendo AMPL

Paso 2: Ahora debemos escribir el archivo de datos, o *.dat. En el le diremos al programa que datos y conjuntos están en el archivo.

1. Primero nombraremos los conjuntos que hay:

```
set Productos:= "mesa", "silla";
set Piezas := "pequena","grande";
```

Ejemplo importante: **Conjuntos numéricos**

```
set N = 12 .. 100;
set YEARS= 1900 .. 2020 by 5;
```

Escribiendo AMPL

2. La sintaxis para los parámetros con un índice es la siguiente:

```
param g:=
"mesa" 16
"silla"    10 ;

param c:=
"pequena"         8
 "grande"         6;
```

Escribiendo AMPL

2. La sintaxis para los parámetros con un índice es la siguiente:

```
param g:=
"mesa" 16
"silla"    10 ;

param c:=
"pequena"         8
 "grande"         6;
```

Escribiendo AMPL

Para los parámetros con dos índices se escriben en forma de matriz, muy parecido a como se hace en Excel.

```
param a: "pequena" "grande":=
"mesa"    2      2
"silla"      2      1 ;
```

Notar que esto es equivalente a decir $a_{\text{silla,grande}} = 1$, que es lo mismo que decir que las sillas ocupan una pieza grande.

Escribiendo AMPL

muebles.dat | muebles.mod | muebles.run

```
set Productos:= "mesa", "silla";
set Piezas := "pequena","grande";

param g:=
      "mesa"    16
      "silla"   10 ;

param c:=
    "pequena"   8
    "grande"    6;

param a: "pequena" "grande":=
"mesa"       2        2
"silla"      2        1
;
```

Con esto ya se puede resolver el problema.

Resolviendo con AMPL- CPLEX

Para resolver solo se deben ingresar en la consola los siguientes comandos:

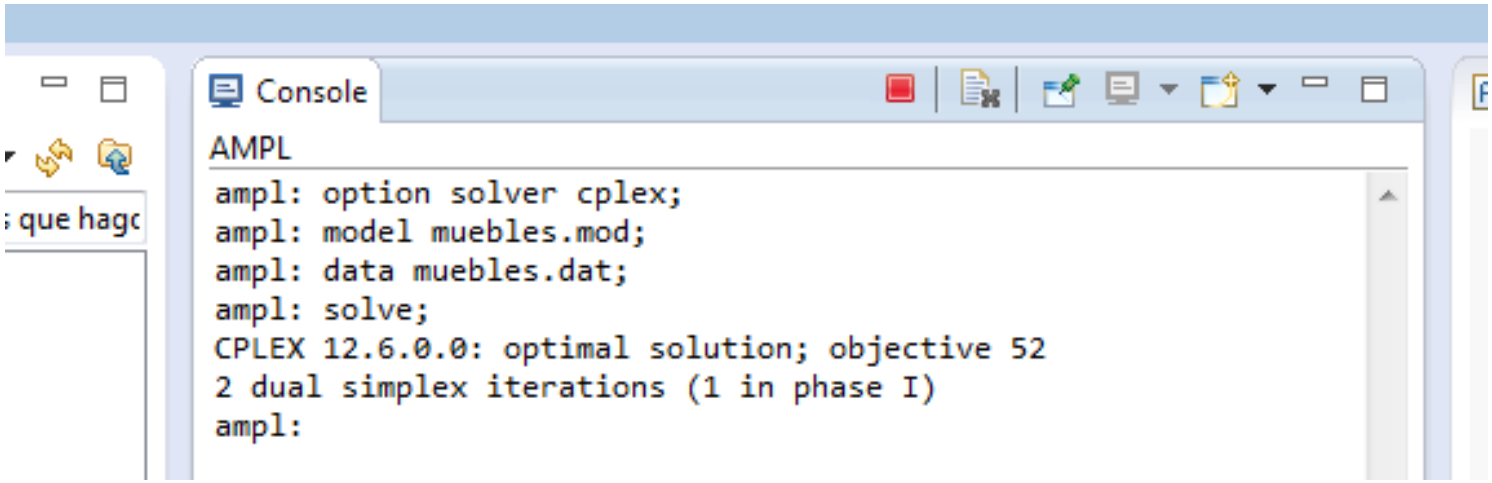
option solver cplex; → Llamamos al solver (CPLEX)

model muebles.mod; → Indicamos el modelo

data muebles.dat; → Indicamos los datos

solve; → Resuelve!

Resolviendo con AMPL- CPLEX



La máxima ganancia que se puede obtener es de 52.

¿Y cómo sabemos los valores de las variables de decisión?

Resolviendo con AMPL- CPLEX

Para esto utilizamos el comando ***display:***

display x;

```
ampl: display x;
x [*] :=
 mesa  2
silla  2
;
```

Otras veces queremos guardar los resultados en algún archivo:

display x > resultado.txt;

display x >> resultado.txt;

La diferencia está en que el primer código borra lo que estaba escrito anteriormente

Resolviendo con AMPL- CPLEX

¿Y el archivo ***.run**?

Este archivo tiene todos los scripts que colocamos en la consola:

```
option solver cplex;
model muebles.mod;
data muebles.dat;
solve;
display x > resultado.txt;
```

Y se llama con el comando ***include:***

AMPL

```
ampl: include muebles.run;
CPLEX 12.6.0.0: optimal solution; objective 52
2 dual simplex iterations (1 in phase I)
ampl:
```

Resolviendo con AMPL- CPLEX

Si queremos resolver una instancia más grande, de por ejemplo 100 articulos y 50 tipos de piezas:

No debemos cambiar el modelo! Es más podemos resolver los dos problemas en un mismo script.

muebles.mod | muebles.run | muebles.dat

```
reset;
option solver cplex;
model muebles.mod;
data muebles.dat;
solve;
display x > sol1.txt;
reset data;
data muebles2.dat;
update data;
solve;
display x > sol2.txt;
```

Comando	Acción
display _ncons display _nvars	Muestra la cantidad de variables y restricciones que tiene un problema.
display _solve_elapsed_time	Muestra el tiempo de resolución del último problema resuelto.
option cplex_options 'time=TT‘	Setea el tiempo de resolución máximo en TT segundos y reporta la mejor solución encontrada hasta ese entonces.
option cplex_options 'mipgap=A‘	Para problemas con variables enteras indica que tan lejos del óptimo queremos llegar (para un 10% colocamos A=0.1)
option cplex_options mipdisplay =X‘	Frecuencia con que muestra la información de Branch and Bound. 0 (default) = nunca 1 = cada solucion entera factible 2 = cada cierto tiempo .. ETCETERA. Recomendado colocar opción 4.

Y muchas más en archivo README.cplex