2 La herramienta ILOG CPLEX

2.1 Introducción a la herramienta ILOG CPLEX

ILOG CPLEX es una herramienta que permite la resolución de diversos tipos de problemas. Entre ellos cabe destacar:

1) Problemas de optimización lineal o también llamados problemas de programación lineal (LP), en los que todas las variables son continuas. La estructura general de este tipo de problemas es la siguiente:

$$\begin{aligned} &Max(o\ Min) && \sum_{i=1}^{n} c_i x_i && \\ &s.a && \sum_{j=1}^{n} a_{ij} x_j \le b_i && \forall i \\ &límites && 0 \le x_i \le d_i && \forall i \end{aligned} \qquad (2.1)$$

Los datos del problema son:

- Coeficientes de la función objetivo: c_i
- Coeficientes de la matriz de restricciones: a_{ij}
- Límites de las restricciones: b_i
- Límites de las variables: d_i

Tras la resolución del modelo, ILOG CPLEX obtiene los valores óptimos de las variables del problema: x_i

2) Problemas de flujo de red *(network)*. Son un tipo especial de problemas LP con una estructura tipo *network*, que ILOG CPLEX es capaz de resolver con mayor rapidez. La estructura *network* es un conjunto de N nodos y A arcos que conectan los nodos entre sí. El nodo de partida de un arco se denomina *tail* y el de llegada *head*. El problema puede también tener tan sólo una parte de estructura *network*. En este caso ILOG CPLEX extraerá y resolverá de forma más eficiente esta parte del problema y usará esta solución parcial como punto de partida para obtener la solución del problema global. La estructura del problema es la siguiente:

$$\begin{aligned} &Max(o\ Min) && \sum_{a\in A} c_a x_a \\ &s.a && \sum_{a\in T_n} x_a - \sum_{a\in H_n} x_a = S_n \quad \forall(n\in N) \\ &límites && I_a \le x_a \le u_a \quad \forall(a\in A) \end{aligned} \qquad (2.2)$$

donde:

- T_n es el conjunto de arcos que salen del nodo n.
- H_n es el conjunto de arcos que llegan al nodo n.

Los datos del problema son:

- Coeficientes de la función objetivo: c_a
- Valor de la fuente o sumidero de cada nodo: S_n (será positivo si se genera flujo y negativo en caso de que se consuma)
- Límites superiores de las variables: u_a
- Límites inferiores de las variables: I_a

Tras la resolución del modelo, ILOG CPLEX obtiene los valores óptimos de las variables: x_a, que representan el valor del flujo que circula por cada arco a. Los valores pueden ser también negativos, lo que significa que el flujo va en sentido contrario.

Además de problemas de tipo LP y *network*, ILOG CPLEX puede resolver otro tipo de problemas:

3) Problemas con términos cuadráticos (QP), en los que la función objetivo puede incluir términos cuadráticos.

4) Problemas que mezclan variables continuas y variables discretas (MIP). Las variables discretas podrán ser tanto enteras como binarias (limitadas a los valores 0 o 1). Este tipo de problemas pueden tener la función objetivo lineal, conociéndose como MILPs o programas lineales que mezclan variables continuas y discretas. También pueden tener términos cuadráticos en la función objetivo, en cuyo caso son conocidos como MIQPs o programas cuadráticos que mezclan variables continuas y discretas. La estructura de un problema MILP es la siguiente:

$$
\begin{aligned}
&\text{Max(o Min)} && \sum_{i=1}^{n} c_i x_i + \sum_{k=1}^{m} \mu_k y_k && \\
&\text{s.a.} && \sum_{j=1}^{n} a_{ij} x_j + \sum_{k=1}^{m} f_{ik} y_k \le b_i && \forall \text{i} \\
&\text{limites} && 0 \le x_i \le d_i && (2.3) \\
& && x_i \text{ continua} && \\
& && y_k \text{ discreta} &&
\end{aligned}
$$

2.1.1 Las tecnologías ILOG CPLEX

ILOG CPLEX utiliza tres tecnologías diferentes para facilitar la implementación y resolución de las tipologías de problemas vistas:

- *ILOG CPLEX Interactive Optimizer*: es un programa ejecutable que puede leer un problema que se introduzca por pantalla de forma interactiva o bien de ficheros con ciertos formatos estándar, resolver el problema y mostrar la solución por pantalla o bien guardarla en ficheros de texto. El programa se puede correr en plataforma Windows o UNIX.

- *Concert Technology*: son una serie de librerías escritas en C++ o en Java que permiten al programador utilizar las funciones de ILOG CPLEX en aplicaciones escritas en estos dos lenguajes. Estas librerías hacen uso de la *Callable Library* (se describe a continuación).

- *ILOG CPLEX Callable Library*: es una librería escrita en lenguaje C que permite al programador utilizar las funciones de la librería ILOG CPLEX en aplicaciones escritas en lenguaje C, Visual Basic, FORTRAM o cualquier otro lenguaje que sea capaz de realizar llamadas a funciones escritas en C.

2.1.2 Algoritmos de optimización

Para la resolución de los distintos tipos de problemas ILOG CPLEX ofrece varios algoritmos (ver *tabla 2.1*). Estos algoritmos se pueden seleccionar a través del correspondiente parámetro en las distintas tecnologías. Este parámetro es conocido de manera genérica como el LPMethod.

	LP	Network	QP	MIP
Dual Optimizer	X		X	
Primal Optimizer	X		X	
Barrier Optimizer	X		X	
Mixed Integer Optimizer				X
Network Optimizer	Nota 1	X	Nota 1	
Nota 1: el problema debe tener una parte de estructura Network				

Tabla 2.1 Algoritmos de optimización ILOG CPLEX

La elección del algoritmo adecuado puede tener un efecto bastante importante en la velocidad de resolución del problema en cuestión. A continuación se describen detalladamente cada uno de los algoritmos:

- *Dual Simplex Optimizer*: se basa en la resolución del problema usando el método *simplex dual*. Puesto que cualquier problema de optimización lineal se puede plantear a través de su forma *primal* o a través de su forma *dual*, y puesto que la solución óptima (de existir) de ambos problemas está directamente relacionada, este algoritmo resuelve el problema *dual*, y utiliza dicha relación para reportar la solución del modelo *primal*.

- *Primal Simplex Optimizer*: se basa en la resolución del problema a través del método *simplex primal*. Este algoritmo ha dejado de ser la primera opción a la hora de resolver un problema de optimización lineal a causa de los recientes avances en el método *dual*. No obstante, este método es más eficiente en problemas en los que el número de variables excede significativamente al número de restricciones o en problemas con poca variabilidad en los coeficientes de la función objetivo. Hay pocos problemas en los que

ninguno de los dos métodos (*primal* o *dual*) sea eficiente, por lo que si aparecen problemas a la hora de resolver con el *dual*, lo lógico es que se resuelva con el *primal*.

- *Barrier Optimizer*: este algoritmo trabaja de una forma muy diferente a como lo hacen el *primal* o el *dual* y genera secuencias de soluciones estrictamente positivas del problema. El *barrier* es muy eficiente en problemas que poseen un gran número de filas o columnas (más de 1000) pero en los que hay significativamente pocos elementos distintos de cero.

- *Network Optimizer*: este algoritmo presenta una gran eficiencia cuando los problemas tienen una estructura de flujo de red *(network)* o al menos una parte del mismo presenta dicha estructura. El *Network* busca la solución del problema que presenta estructura de red y ésta será la base de la que parte ILOG CPLEX para seguir realizando iteraciones hasta encontrar la solución del problema completo.

- *Mip Optimizer*: este algoritmo se usa en la resolución de problemas que presentan variables continuas y discretas (ya sean enteras o binarias).

2.2 Formato del problema

A la hora de crear un nuevo problema en ILOG CPLEX, es necesario seguir una serie de reglas básicas con el objeto de establecer el formato de los datos del mismo. Estas reglas se denominan formato LP de ILOG CPLEX.

2.2.1 Nombre de las variables

Existen una serie de limitaciones a la hora de nombrar las variables al seguir el formato LP. Son las siguientes:

- No se puede sobrepasar los 255 caracteres de longitud y los únicos caracteres que se permiten son los alfanuméricos, (a-z, A-Z, 0-9) y determinados símbolos (! “ # $ % & ( ) . , ; ? @ _ ‘ { } ~). En caso de que se introduzca una línea con más de 510 caracteres será truncada automáticamente.

- Los nombres no podrán comenzar por un número ni por un espacio en blanco.

- Existe una combinación de caracteres que no se podrá usar: la letra “e” o “E” solas o seguidas por un número (como “e24”) u otra e (como “eels”). Esta notación está reservada para entradas exponenciales.

- No es válida la utilización de los siguientes caracteres: ^ * [ ] ya que están reservados para poder definir la función objetivo con términos cuadráticos.

Por simplicidad en este proyecto se van a utilizar variables con la siguiente notación: x_{ijk}.

2.2.2 Límites de las variables

A la hora de fijar los límites inferiores y superiores de las variables, hay que tener en cuenta que ILOG CPLEX establece, por defecto, el límite inferior en 0 y el límite superior en infinito. Por lo tanto, será necesario fijar explícitamente los límites de aquellas variables que difieren de estos valores por defecto.

2.3 Resolución de problemas de programación lineal

Una vez que se han introducido correctamente todos los datos del problema y que se ha seleccionado el algoritmo adecuado, ILOG CPLEX está en disposición de abordar la resolución del problema.

En primer lugar, ILOG CPLEX intenta simplificar o reducir el problema mediante el uso del *presolver* y el *aggregator*. En esta etapa, ILOG CPLEX trata de simplificar restricciones, reducir el tamaño del problema y eliminar filas redundantes. Se intenta reducir el tamaño del problema eliminando filas o variables y sustituyéndolas. En un gran número de problemas el *presolver* consigue mejorar la velocidad final de resolución. ILOG CPLEX muestra los resultados del problema en términos de la formulación original del mismo y no del modelo reducido. No obstante, siempre es posible acceder al modelo reducido al que ILOG CPLEX llegó y que utilizó para la obtención de las soluciones.

Tras esta etapa inicial, ILOG CPLEX realiza la resolución del problema en dos fases:

Fase I: en ella ILOG CPLEX busca una solución factible. Durante esta fase se puede visualizar el número de la iteración y el grado de infactibilidad del problema.

Fase II: ILOG CPLEX busca la solución factible óptima del problema. Durante esta fase se mostrará el valor que va tomando la función objetivo.

Toda esta información es útil para conocer el grado de avance del proceso.

Una vez que ILOG CPLEX alcanza la solución óptima, se muestra por pantalla el valor de la función objetivo, el tiempo total de resolución y el número total de iteraciones necesarias para alcanzarla (con las iteraciones de la fase I entre paréntesis).

La información que aparece por pantalla se puede modificar usando los comandos apropiados en cada tecnología (en el *interactive optimizer* el parámetro es *set simplex display*) con el objeto de conocer diferentes cantidades de datos a medida que se está resolviendo el problema. En la *tabla 2.2* se muestran los posibles valores de este parámetro.

Simplex display	0	No se muestra nada hasta alcanzar la solución
	1	Se muestra información después de cada ciclo de iteraciones (valor por defecto)
	2	Se muestra información después de cada iteración

Tabla 2.2 Opciones de visualización por pantalla

Adicionalmente, ILOG CPLEX permite modificar una amplia gama de parámetros que permiten configurar la herramienta en función de las necesidades del usuario. En la *tabla 2.3* se muestran los principales parámetros del *interactive optimizer* acompañados de su significado (existen parámetros equivalentes para el resto de tecnologías).

Parámetro	Significado
Avance	Establece el indicador para una inicialización avanzada cuando se parte de una solución inicial
Barrier	Establece los parámetros para el algoritmo Barrier
Clocktype	Establece el tipo de reloj para medir el tiempo
Defaults	Establece los valores por defecto de todos los parámetros
Logfile	Establece el fichero en el que se guardan los resultados
Lpmethod	Establece el método usado para la optimización lineal
Mip	Establece los parámetros para el algoritmo MIP
Network	Establece los parámetros para el algoritmo Network
Output	Establece los destinos de salida
Preprocessing	Establece los parámetros para la etapa de preprocesado
Qpmethod	Establece el método para la optimización cuadrática
Read	Establece los parámetros de lectura del problema
Simplex	Establece los parámetros para los algoritmos Primal y Dual
Timelimit	Establece el tiempo límite en segundos
Workdir	Establece el directorio de ficheros de trabajo
Workmem	Establece la memoria disponible para almacenar el trabajo (en megabytes)

Tabla 2.3 Parámetros ILOG CPLEX

En caso de que el programa ILOG CPLEX se ejecute en un ordenador que no posea demasiado espacio libre en la memoria, se podría desactivar la opción de realizar un análisis previo al problema para intentar reducirlo, en cuyo caso, se disminuiría el tiempo de resolución. Sin embargo, independientemente de que este parámetro esté o no activado, ILOG CPLEX es capaz de reconocer cuándo se está procesando en un entorno de limitada memoria disponible y automáticamente trata de compensar este inconveniente, lo que reducirá obviamente la velocidad de la optimización.

La memoria que se debe tener disponible deberá ser mayor a la necesaria para que se puedan indicar todos los datos requeridos para la definición del problema. Puede ocurrir que, en el caso

de aproximarse al límite de la memoria, se muestre un mensaje advirtiendo que no se debería llevar a cabo una determinada operación de ILOG CPLEX. Sin embargo, el programa continuará con la ejecución. También puede aparecer el aviso de que se está llevando a cabo una compresión para aprovechar al máximo la escasa memoria disponible. Estos avisos indican indirectamente que existe insuficiente memoria disponible, por lo que se está resolviendo el problema por un camino no tan óptimo y deseable.

2.4 La herramienta ILOG CPLEX para C++

2.4.1 Diseño conceptual de ILOG CPLEX

Para usar la *Concert Technology* con la herramienta ILOG CPLEX y poder construir y resolver modelos de optimización, es fundamental conocer perfectamente los objetos que se usan en lenguaje C++. Estos objetos se pueden dividir en dos categorías:

- *Modeling objects*: se usan para definir el problema de optimización. Generalmente en una aplicación se crean muchos de estos objetos para especificar un problema de optimización. Todos ellos se agrupan luego en un *IloModel object* en el que se define al problema completo.

- *IloCplex objects*: se usan para resolver los problemas que se han creado con los *modeling objects*. Estos objetos leen el modelo, extraen su información y la representan de la forma apropiada para que el algoritmo de optimización de ILOG CPLEX sea capaz de interpretarla. En este momento el *IloCplex object* estará listo para resolver el modelo y mostrar la información del resultado.

De este modo, las fases de implementación y optimización de un programa escrito por el usuario se pueden representar a través de un grupo de objetos de C++ que interaccionan entre sí y que han sido creados y son controlados a través de la aplicación. En la *figura 2.1* se muestra el esquema de una aplicación que usa la *Concert Technology* de ILOG CPLEX para resolver un problema de optimización.

La base de datos de ILOG CPLEX incluye el entorno, los canales y el problema.

En este proyecto se ha creado una aplicación haciendo uso de la tecnología *Concert Technology* en lenguaje Visual C++ 6.0. En la *figura 2.1* aparece como 'aplicación escrita por el usuario'. En esta figura se puede apreciar como a través de esta aplicación se crean y se controlan los objetos necesarios para resolver el problema implementado y obtener los resultados. Para ello se usa en todo momento la información y las herramientas disponibles en la base de datos de ILOG CPLEX.

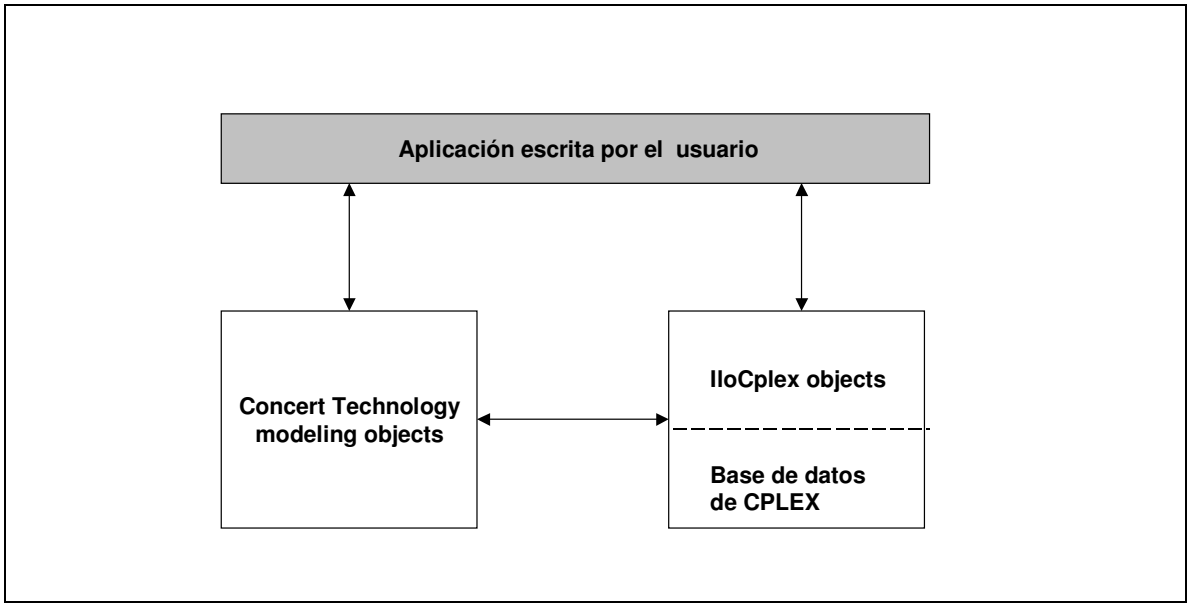

Figura 2.1 Esquema de la aplicación

2.5 Pasos para la resolución de un problema en C++

En este apartado se describen las distintas etapas que son necesarias para la implementación y posterior resolución de un problema de optimización lineal haciendo uso de una aplicación escrita en C++.

2.5.1 Construcción del modelo

El primer objeto que es necesario crear en la *Concert Technology* es el entorno. Para ello se declara una variable del tipo *IloEnv*. Este objeto es muy importante para la construcción del resto del modelo, puesto que es el encargado de optimizar la gestión de la memoria necesaria para crear los restantes objetos. Esta optimización es más eficiente que la que se consigue con el gestor de memoria del sistema operativo. El objeto *IloEnv* se declara como un puntero a objeto,

por lo que al final de la aplicación es necesario destruírlo para liberar la memoria reservada en su declaración, siendo esta generalmente la última sentencia escrita en la aplicación.

Una vez que se ha creado el entorno, se pueden comenzar a definir uno o más modelos de optimización. Para cada uno de esos modelos se crearán varios *modeling objects* que se agruparán luego en un objeto del tipo *IloModel object* que es el que define el modelo completo. El *IloModel object* se declara pasándole como parámetro el entorno definido anteriormente. Una vez construído el *IloModel object* se puede comenzar a implementar el modelo haciendo uso de los siguientes elementos:

- *IloNumVar*: representan las variables del problema.
- *IloRange*: describen las restricciones del modelo.
- *IloObjetive*: representa la función objetivo.

Con estos elementos se pueden construir tanto las restricciones como la función objetivo que posteriormente se incorporarán al modelo global.

2.5.2 Resolución del modelo y obtención de resultados

Tras la construcción del modelo completo en el *IloModel object*, se habrá de crear un objeto del tipo *IloCplex* que será el que permita la resolución del problema. Este objeto se construye declarando una variable a la que se le pasa como parámetro el entorno. El *IloCplex object* se usará para la extracción del modelo como paso previo a su resolución. La función que permite resolver el modelo es:

- cplex.solve()

Esta función puede devolver dos valores: verdadero, en caso de haber encontrado una solución al problema (aunque no necesariamente la óptima), y falso, en caso de que no se haya encontrado ninguna solución. Para obtener más información sobre cómo finalizó la resolución del modelo, se puede hacer uso de las siguientes funciones:

- cplex.getStatus(): devuelve información sobre la solución alcanzada o bien las razones por las que ésta no se ha podido obtener.

- cplex.getCplexStatus(): da una información mucho más detallada que la función anterior sobre la solución obtenida o los motivos por los que no se llegó a alcanzarla.

Una vez que el modelo se ha resuelto de manera satisfactoria, será necesario acceder a los resultados obtenidos. Las funciones más importantes de las que se dispone para recopilar esta información son las siguientes:

- cplex.getValue(x1): esta función devuelve el valor de la variable que se le pasa como argumento.
- cplex.getValues(x, var): devuelve los valores de todas las variables y las almacena en la variable que se le pasa como primer argumento.
- cplex.getObjValue(): devuelve el valor de la función objetivo.
- cplex.getDuals(): obtiene los valores de las variables duales de las restricciones del modelo.

La última sentencia de la aplicación en la *Concert Technology* suele ser, como se mencionó anteriormente, la liberación del entorno previamente declarado. Para ello se usa la siguiente expresión:

```
env.end()
```

Estas funciones y todas las restantes de las que dispone la herramienta ILOG CPLEX serán ampliamente descritas en el siguiente apartado.

2.5.3 Esquema general

Se presenta a continuación, a modo de resumen, un diagrama con los pasos generales vistos y que se han de seguir para la resolución de un problema de programación lineal en una aplicación escrita en C++ (*figura 2.2*).

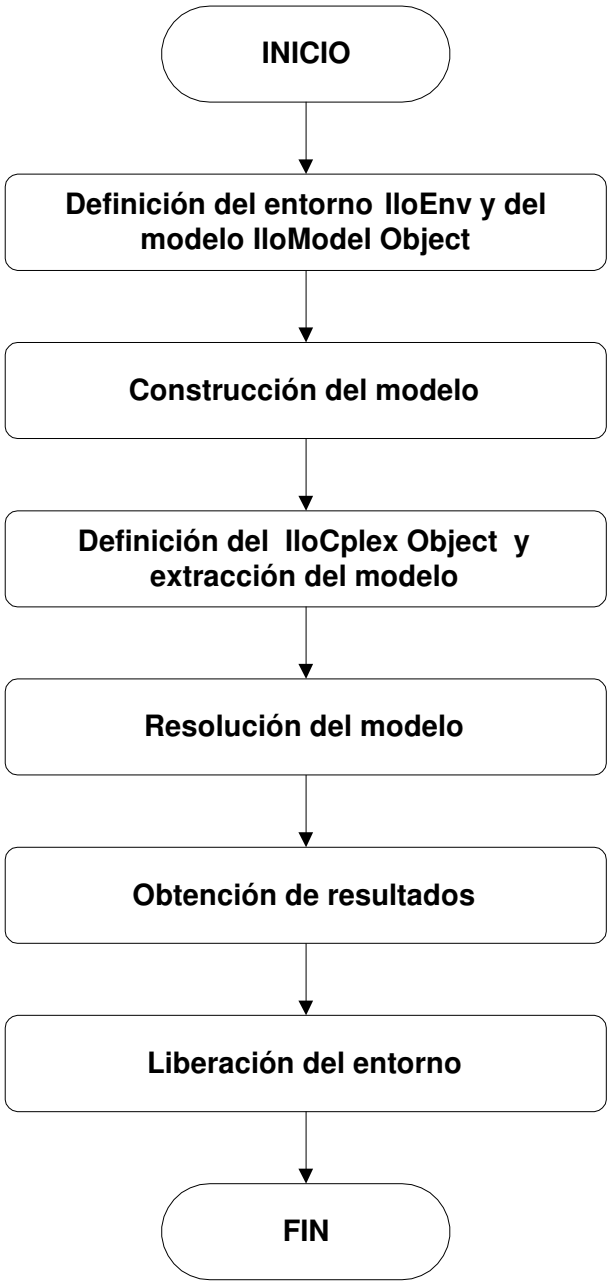

Figura 1.2 Resolución de problemas C++

2.6 Árbol de funciones de la herramienta ILOG CPLEX

Se relacionan a continuación todas las funciones que constituyen la biblioteca ILOG CPLEX y que están a disposición de las distintas tecnologías. Se organizan en 6 categorías diferentes:

1) Funciones de optimización y resultados: estas funciones se usan para la definición de problemas de optimización lineal, realizar su resolución y mostrar los resultados obtenidos. Estas funciones se pueden agrupar a su vez en:

    - Funciones para crear y liberar problemas.

    - Funciones de optimización de problemas.

    - Funciones para acceder a los resultados. Los resultados obtenidos pueden estar asociados a:

        - ✓ Problemas LP o QP.
        - ✓ Problemas MIP.

    - Funciones para analizar las soluciones.

2) Funciones de consulta de problemas: se usan para obtener información sobre un problema que ha sido previamente cargado e implementado. Se pueden usar en cualquier momento, incluso después de que el problema se haya modificado. Se subdividen en:

    - Funciones para acceder a la información de problemas genéricos.

    - Funciones para acceder a la información de problemas concretos. Estas funciones se subdividen en:

        - ✓ De problemas MIP.
        - ✓ De problemas QP.

3) Funciones de lectura y escritura en ficheros: se usan para leer los problemas desde ficheros de sistema y para escribir los problemas o sus soluciones en ficheros. Se clasifican en:

    - Funciones de lectura de ficheros.

    - Funciones de escritura en ficheros.

4) Funciones de modificación de problemas: se usan para modificar problemas después de que hayan sido creados. Entre estas modificaciones se incluyen: añadir o borrar filas y columnas a la matriz de restricciones, modificar el sentido de la función objetivo, modificar el valor de los coeficientes de la matriz de restricciones, modificar los coeficientes de la función objetivo, modificar los límites de las restricciones, modificar los límites de las variables, cambiar el sentido de las restricciones o cambiar los nombres de filas y columnas.

5) Funciones para modificar y mostrar los valores de los parámetros del sistema: se utilizan para controlar el comportamiento de los algoritmos de resolución de ILOG CPLEX. Se clasifican en:

    - Funciones para modificar parámetros.

    - Funciones para obtener los valores de los parámetros.

6) Funciones de utilidades de ILOG CPLEX: se usan con los siguientes propósitos:

    - Funciones de utilidad general: se usan para abrir y cerrar el entorno de ILOG CPLEX.

    - Funciones para manejar mensajes: se utilizan para facilitar al usuario de la aplicación el control sobre los mensajes que se muestran por pantalla o que se mandan a los ficheros de salida. Adicionalmente permiten al usuario la creación de mensajes de salida definidos por el mismo.

    - Funciones de interrupción y llamada: permiten a los usuarios definir aplicaciones para interrumpir y continuar la optimización de un problema y acceder a la información sobre el progreso de la optimización.

En la *tabla 2.4* se exponen a modo de resumen todas las funciones que han sido descritas en este apartado.

Funciones	Optimización y resultados	Crear y liberar problemas	
		Optimización de problemas	
		Acceso a resultados	Problemas LP o QP
			Problemas MIP
		Análisis de soluciones	
	Consulta de problemas	Acceso a información de problemas genéricos	
		Acceso a información de problemas concretos	Problemas MIP
			Problemas QP
	Lectura y escritura de ficheros	Lectura de ficheros	
		Escritura de ficheros	
	Modificación de problemas		
	Modificación y consulta de parámetros	Modificación de parámetros	
		Consulta de parámetros	
	Utilidades de CPLEX	Utilidad general	
		Manejo de mensajes	
		Interrupción y llamada	

Tabla 2.4 Clasificación de funciones ILOG CPLEX

2.7 Descripción de las funciones

2.7.1 IloEnv::end

Descripción: cierra el entorno ILOG CPLEX.

2.7.2 IloModel::Add

Descripción: añade restricciones o la función objetivo al modelo.

2.7.3 IloCplex::extract

Descripción: extrae el modelo que se ha construído en el objeto del tipo IloModel.

Formato: void extract(IloModel model);

Argumentos:

- Model: es el nombre del objeto en el que se ha construído el problema.

2.7.4 IloCplex::setParam

Descripción: fija el valor de los parámetros del sistema que se indiquen como argumento.

Formato: void setParam(IloCplex::BoolParam parameter, IloBool value);
void setParam(IloCplex::IntParam parameter, IloInt value);
void setParam(IloCplex::NumParam parameter, IloNum value);
void setParam(IloCplex::StringParam parameter, const char* value);

Argumentos:

- Parameter: es el nombre del parámetro cuyo valor se quiere modificar. Puede ser de tipo booleano, entero, continuo o una cadena de caracteres.
- Value: es el valor que se quiere dar al parámetro.

2.7.5 IloCplex::importModel

Descripción: importa un modelo previamente construído desde un fichero. Guarda la función objetivo, las variables y las restricciones en las correspondientes variables que se pasan como argumentos.

Formato: void importModel(IloModel& model,
const char* filename,
IloObjective& obj,
IloNumVarArray vars,
IloRangeArray rngs) const;

Argumentos:

- Model: nombre del modelo que se va a leer.
- Filename: nombre del fichero desde el que se va a leer el modelo.
- Obj: variable donde se va a guardar la función objetivo.

- Vars: en cadena en la que se guardan las variables del modelo.
- Rngs: cadena donde se guardan las restricciones.

2.7.6 IloCplex::exportModel

Descripción: escribe el modelo que se ha construído en el fichero que se pasa como argumento. El nombre del fichero debe incluir su extensión. Ésta debe ser una de las reconocidas por ILOG CPLEX. Los formatos que ILOG CPLEX reconoce son: *.sav*, *.lp* o *.mps*.

Formato: void exportModel(const char* filename) const;

Argumentos:

- Filename: nombre del fichero en el que se va a escribir el modelo.

2.7.7 IloCplex::solve

Descripción: resuelve el modelo que se ha construído y extraído previamente. El algoritmo que utiliza es el que se haya fijado previamente o el que ILOG CPLEX seleccione por defecto. Esta función devuelve la constante IloTrue si se encuentra una solución, aunque ésta no sea la óptima.

Formato: IloBool solve() const;

2.7.8 IloCplex::Exception

Descripción: captura errores en tiempo de ejecución.

2.7.9 IloCplex::getParam

Descripción: devuelve el valor del parámetro del sistema que se pase como argumento.

Formato: IloBool getParam(IloCplex::BoolParam parameter) const;
IloInt getParam(IloCplex::IntParam parameter) const;
IloNum getParam(IloCplex::NumParam parameter) const;

const char* getParam(IloCplex::StringParam parameter) const;

Argumentos:

- Parameter: es el nombre del parámetro cuyo valor se quiere conocer. Puede ser de tipo booleano, entero, continuo o una cadena de caracteres.

2.7.10 IloCplex::getNrows

Descripción: devuelve el número de filas de la matriz de restricciones del modelo que se ha construído. No se cuenta la función objetivo ni los límites de las variables.

Formato: IloInt getNrows() const;

2.7.11 IloCplex:: getNcols

Descripción: devuelve el número de columnas que tiene el modelo construído o, lo que es lo mismo, el número de variables del problema.

Formato: IloInt getNcols() const;

2.7.12 IloCplex::getNNZs

Descripción: devuelve el número de elementos distintos de cero de la matriz de restricciones del modelo que se ha construído. No se cuenta la función objetivo ni los límites de las variables.

Formato: IloInt getNNZs() const;

2.7.13 IloCplex::getObjValue

Descripción: devuelve el valor de la función objetivo correspondiente a la solución del problema (no necesariamente tiene que ser factible u óptima). Devuelve un 0 si no se ha extraído en el modelo el objeto del tipo *IloObjective*.

Formato: IloNum getObjValue() const;

2.7.14 IloCplex::getValues

Descripción: devuelve el valor de las soluciones del problema y las guarda en la cadena de caracteres que se pasa como argumento.

Formato: void getValues(IloNumArray vals, const IloNumVarArray vars) const;

Argumentos:

- Vals: cadena de caracteres en las que se guardan los valores de las soluciones.
- Vars: variable en la que se han almacenado las soluciones tras la resolución del modelo.

2.7.15 IloCplex::getValue

Descripción: devuelve el valor de la variable que se pasa como argumento.

Formato: IloNum getValue(const IloNumVar var) const;

Argumentos:

- Var: nombre de la variable cuyo valor se pretende conocer.

2.7.16 IloCplex::getDuals

Descripción: devuelve el valor de las soluciones duales asociadas a las restricciones y los almacena en la cadena de caracteres que se pasa como argumento.

Formato: void getDuals(IloNumArray vals, const IloRangeArray ranges) const;

Argumentos:

- Vals: cadena de caracteres en la que se almacenan los valores de las soluciones duales.

- Ranges: variable en la que se han almacenado las restricciones tras la construcción del modelo.

2.7.17 IloCplex::getDual

Descripción: devuelve el valor de la solución dual que se pasa como argumento.

Formato: IloNum getDual(const IloRange range) const;

Argumentos:

- Range: restricción cuyo valor dual se pretende conocer.

2.7.18 IloCplex::getSlacks

Descripción: devuelve los valores de las variables de holgura asociadas a las restricciones y los almacena en la cadena de caracteres que se pasa como argumento.

Formato: void getSlacks(IloNumArray vals, const IloRangeArray ranges) const;

Argumentos:

- Vals: cadena de caracteres en la que se almacenan los valores de las variables de holgura.
- Ranges: variable en la que se han almacenado las restricciones tras la construcción del modelo.

2.7.19 IloCplex::getStatus

Descripción: devuelve el valor del estado del algoritmo usado en la resolución del problema.

Formato: IloAlgorithm::Status getStatus() const;

Valores devueltos (*tabla 2.5*):

Estado	Significado
Feasible	Se ha encontrado una solución factible
Optimal	Se ha encontrado la solución optima
Infeasible	El problema no es factible
Unbounded	El problema no tiene límites
InfeasibleOrUnbounded	El problema no es factible o no tiene límites
Unknown	No se ha podido terminar la ejecución del programa
Error	Se ha producido un error durante la optimización

Tabla 2.5 Valores del parámetro getStatus

2.7.20 IloCplex::getCplexStatus

Descripción: devuelve el estado ILOG CPLEX tras la llamada al algoritmo de optimización del problema.

Formato: IloCplex::CplexStatus getCplexStatus() const;

Valores devueltos (*tabla 2.6*, *tabla 2.7*, *tabla 2.8*):

Valor	IloCplex::CplexStatus	Significado
Para Simplex y Barrier		
1	IloCplex::Optimal	Se ha encontrado la solución óptima
2	IloCplex::Unbounded	Problema sin limites
3	IloCplex::Infeasible	Problema no factible
4	IloCplex::InfOrUnbd	Problema no factible o sin límites
5	IloCplex::OptimalInfeas	Solución óptima, infactibilidades fuera de escala
6	IloCplex::NumBest	Se ha obtenido una solución pero no se ha probado que sea óptima
10	IloCplex::AbortItLim	Abortado por superarse el límite de iteraciones
11	IloCplex::AbortTimeLim	Abortado por superarse el límite de tiempo
12	IloCplex::AbortObtLim	Abortado por límites de la función objetivo
13	IloCplex::AbortUser	Abortado por el usuario

Tabla 2.6 Valores del parámetro getCplexStatus para Simplex y Barrier

Valor	IloCplex::CplexStatus	Significado
Para Barrier		
20	IloCplex::OptimalFaceUnbounded	El problema tiene un óptimo sin límites
21	IloCplex::AbortPrimObjLim	Abortado por límites de la función objetivo del primal
22	IloCplex::AbortDualObjLim	Abortado por límites de la función objetivo del dual

Tabla 2.7 Valores del parámetro getCplexStatus para Barrier

Valor	IloCplex::CplexStatus	Significado
Para MIP		
101	IloCplex::Optimal	Se ha alcanzado una solución entera óptima
102	IloCplex::OptimalTol	Se ha alcanzado la solución óptima con tolerancia epgap o epagap
103	IloCplex::Infeasible	Solución entera no factible
104	IloCplex::SolLim	Se ha excedido el límite de las soluciones enteras mixtas
105	IloCplex::NodeLimFeas	Se ha excedido el límite del número de nodos, hay solución entera
106	IloCplex::NodeLimInfeas	Se ha excedido el límite del número de nodos, no hay solución entera
107	IloCplex::TimeLimFeas	Se ha excedido el límite del tiempo, hay solución entera
108	IloCplex::TimeLimInfeas	Se ha excedido el límite del tiempo, no hay solución entera
109	IloCplex::FailFeas	Error al finalizar, existe solución entera
110	IlCplex::FailInfeas	Error al finalizar, no existe solución entera
111	IloCplex::MemLimFeas	Límite de memoria, existe solución entera
112	IloCplex::MemLimInfeas	Límite de memoria, no existe solución entera
113	IloCplex::AbortFeas	Abortado, existe solución entera
114	IloCplex::AbortInfeas	Abortado, no existe solución entera
115	IloCplex::OptimalInfeas	Problema óptimo con infactibilidades fuera de escala
116	IloCplex::FailFeasNoTree	No hay memoria, existe solución entera
117	IloCplex::FailInfeasNoTree	No hay memoria, no existe solución entera
118	IloCplex::Unbounded	Problema sin límites
119	IloCplex::InfOrUnbd	Problema no factible o sin límites

Tabla 2.8 Valores del parámetro getCplexStatus para MIP

2.7.21 IloCplex::getNiterations

Descripción: devuelve el número de iteraciones que fueron necesarias para alcanzar la solución del problema en la última llamada al algoritmo de optimización.

Formato: IloInt getNiterations() const;

2.7.22 IloCplex::getNphaseOneiterations

Descripción: devuelve el número de iteraciones empleadas en la fase I para obtener la solución del problema si se usó un algoritmo *simplex*.

Formato: IloInt getNphaseOneIterations() const;

2.7.23 IloCplex::getAlgorithm

Descripción: devuelve el último algoritmo utilizado en la resolución del problema.

Formato: IloCplex::Algorithm getAlgorithm() const;

2.7.24 IloCplex:: getBasisStatuses

Descripción: devuelve los estados de las variables y los almacena en la cadena de caracteres que se pasa como argumento.

Formato: void getBasisStatuses(IloCplex::BasisStatusArray cstat,
const IloNumVarArray vars) const;

Argumentos:

- Cstat: cadena de caracteres en la que se almacenan los valores de los estados de las soluciones.
- Vars: variable en la que se han almacenado las soluciones tras la resolución del modelo.

2.7.25 IloCplex:: getQuality

Descripción: mide la calidad de la solución obtenida.

Formato: IloNum getQuality(IloCplex::MaxPrimalInfeas) const;

Argumentos:

- IloCplex::MaxPrimalInfeas: máximo valor de infactibilidad que se produce tras la resolución del modelo.