Revista Iberoamericana de Automática e Informática Industrial
CEA IFAC comité español de automática
RIAI
ISSN: 1697-7912. Vol. 2, Núm. 4, Octubre 2005, pp. 89-100
http://riai.isa.upv.es

# ENTORNO GRÁFICO DE MODELADO PARA PROBLEMAS DE OPTIMIZACIÓN DE SISTEMAS A GRAN ESCALA

**José Luis Risco Martín[*], Jesús Manuel de la Cruz García[**], Bonifacio de Andrés y Toro[**] y Alberto Herrán González[**]**

**CES Felipe II (UCM)*
*email: jlrisco@cesfelipesegundo.com*
***Departamento de Arquitectura de Computadores y Automática*
*Universidad Complutense de Madrid*
*email: {jmcruz,deandres,aherrang}@fis.ucm.es*

Resumen: En este trabajo se definen los formalismos necesarios para construir un entorno gráfico de modelado de problemas de optimización de sistemas a gran escala. Dado un problema modelado bajo esta orientación se generan automáticamente las bases de datos que albergarán la información del problema, datos ficticios para la realización de pruebas iniciales y los modelos de resolución, así como modelos de depuración en caso de que el resolutor no encuentre solución factible. Además se generan automáticamente editores gráficos que permitan visualizar gráficamente los datos del problema representado y sus soluciones, permitiendo modificar estos de forma cómoda. Finalmente estos formalismos se han utilizado para implementar un editor de problemas de optimización, que contemple las características mencionadas. 

Palabras Clave: Optimización, Meta-modelado, Modelado basado en grafos, Transformaciones, Logística.

## 1. INTRODUCCIÓN

Durante los últimos 30 años los sistemas software han crecido en complejidad considerablemente. Con objeto de administrar grandes proyectos, facilitar análisis y documentación, y llevar a cabo un proceso de mantenimiento y de reusabilidad, han surgido bastantes técnicas en la ingeniería del software durante todo este tiempo. Las dos técnicas más importantes de las surgidas hasta le fecha son el paradigma de la programación orientada a objetos y los lenguajes gráficos de modelado.

En este artículo se expone cómo el lenguaje gráfico de modelado puede ubicarse en el contexto de la *Investigación Operativa*, en particular en el contexto de la *Programación Matemática*. La Programación Matemática es una potente técnica de modelado usada en el proceso de toma de decisiones. Cuando se trata de resolver un problema de este tipo, la primera etapa consiste en identificar las posibles decisiones que pueden tomarse; esto lleva a identificar las variables del problema concreto. La segunda etapa supone determinar qué decisiones resultan admisibles; esto conduce a un conjunto de restricciones que se determinan teniendo presente la naturaleza del problema en cuestión. En la tercera etapa, se calcula el coste/beneficio asociado a cada decisión admisible; esto supone determinar una función objetivo que asigna, a cada conjunto posible de valores para las variables que determinan una decisión, un valor de coste/beneficio. El conjunto de todos estos elementos define el problema de optimización (Castillo *et al.*, 2002).

Nuestra motivación a este problema proviene de dos perspectivas: pragmática y teórica. Desde una perspectiva más pragmática el modelado gráfico presenta una gran expansión y es dominante sobre otros métodos de modelado. De hecho, cabe destacar la importancia que un modelo gráfico adquiere al facilitar la comunicación entre desarrolladores y usuarios. Desde una perspectiva teórica, observamos que la metodología del modelado gráfico ha tenido

gran éxito al abordar problemas del campo de la Investigación Operativa. Por lo que cabe preguntarse qué grado de éxito puede proporcionar en el ámbito de la Programación Matemática, tal y como se expone a continuación.

El contenido de este artículo se divide en las siguientes secciones: En la sección 2 se examinan los resultados previos en las publicaciones realizadas hasta la fecha. En la sección 3 se hace una descripción breve del modelo teórico desarrollado en esta investigación, orientado a modelar problemas de optimización de sistemas a gran escala. En la sección 4 se utiliza la herramienta desarrollada para resolver el problema de la gestión logística y optimización de una red de distribución de gran escala. En la sección 5 se enumeran los problemas más significativos modelados con la herramienta. Finalmente, en la sección 6 se exponen las conclusiones y las posibles extensiones a otro tipo de problemas.

## 2. MODELADO GRÁFICO EN LA OPTIMIZACIÓN

La implementación de los sistemas de apoyo a la gestión de problemas de optimización ha recibido tradicionalmente poca atención en publicaciones académicas. Sin embargo, recientemente ha despertado un mayor interés el tema al constatar que un gran número de estas aplicaciones fracasan al ser implementadas, aún cuando los algoritmos usados son excelentes (Turban, 1995). En (Geoffrion, 1987) se pone de manifiesto el giro de Arthur Geoffrion, cambiando sus temas de investigación de aspectos teóricos a aspectos más prácticos.

En (Gazmuri and Arrate, 1995) presentan un caso donde se analizan algunos aspectos relacionados con la implementación de un sistema de apoyo a la gestión de la planificación de una empresa manufacturera. En (Maturana and Eterovic, 1995) analizan los aspectos relacionados con la interfaz de usuario de este mismo sistema, que es en definitiva el aspecto más relevante de este tipo de herramientas. En (Maturana *et al.*, 1996) describen el proceso de implementación de este sistema de apoyo a la toma de decisiones. En (Zimmermann, 1988) se describe el diseño de un sistema de apoyo a la toma de decisiones para problemas con múltiples funciones objetivo difusas y restricciones difusas que obtiene ciertos datos de toma de decisiones en forma interactiva.

El mayor problema de implementar este tipo de herramientas consiste en que tanto el modelo del problema como los datos del mismo deben vincularse para un correcto mantenimiento del sistema. La gestión de las bases de datos ha sido siempre un aspecto muy importante de los sistemas de Programación Matemática, como se pone de manifiesto en (Palmer, 1984) respecto al sistema PLATFORM. La similitud de los datos de los problemas de optimización y las bases de datos relacionales las han puesto de manifiesto varios autores, incluyendo Choobineh (Choobineh, 1991), Müler-Merbach (Müller-Merbach, 1990), Welch (Welch, 1987), y Mitra (Mitra *et al.*, 1995). También Fourer (Fourer, 1997) ha formulado unos principios generales y los ha aplicado a ejemplos específicos para estructurar las bases de datos ante un problema lineal de logística de distribución.

Un aspecto importante, y frecuentemente olvidado de la implementación de apoyo a la gestión es el análisis de resultados. En (Sharda and Steiger, 1995) presentan INSIGHT, orientada justamente a ayudar a entender cómo influyen los parámetros de un problema sobre los resultados.

Como se puede comprobar, durante un largo periodo de tiempo muchos investigadores han buscado un tratamiento formal al modelado de sistemas de optimización. Todas estas investigaciones se pueden agrupar en dos tendencias: una tendencia más formal, cuyo objetivo es producir entornos de modelado muy precisos y que requieren de un gran conocimiento matemático del problema; y una tendencia más informal, cuyo objetivo es modelar problemas más ligeros y donde la especialización no cobra tanta importancia.

El *Modelado estructurado* de Geoffrion (Geoffrion, 1987), el *Sistema de programación matemática inteligente* de Greenberg (Greenberg, 1996), y el trabajo de Jones en cuanto al *Modelado de sistemas basados en grafos* (Jones, 1990; Jones, 1991), son representativos de esta tendencia más formal. Algunas de las ideas derivadas de estas contribuciones se plasman en implementaciones como MODLER y ANALIZE (Greenberg, 1993), así como entornos gráficos de modelado como MIMI/G (Jones and Baker, 1996), LPForm (Ma *et al.*, 1996), gLPS (Collaud and Pasquier-Boltuck, 1994), y otros.

La tendencia más informal incluye técnicas más conocidas, como los *Métodos de programación estructurada* (MPE) (Rosenhead, 1996) y la *Metodología de sistemas ligeros* (Checkland, 2000).

Nuestra investigación está más cercana a la primera tendencia que a la segunda, si bien contiene características de ambas, ya que se utilizan métodos MPE en un contexto similar a la gramática de grafos propuesta por Jones.

En este trabajo se demuestra el grado en que es posible adaptar las teorías desarrolladas para emplear el modelado gráfico para gestionar y optimizar redes de distribución logística. Tal como se detalla en la Sección 3, se ha creado un metamodelo específico para este tipo de problemas. Los formalismos desarrollados están especialmente orientados a modelar sistemas de gran escala, y teniendo siempre presente la comunicación entre el especialista que diseña el modelo de optimización y el usuario final del mismo. En consecuencia se hace necesario

"ocultar" ciertas características algebraicas inherentes a un modelo de optimización.

Históricamente esto ha supuesto siempre un impedimento en el sentido de que los encargados de la optimización han sido siempre matemáticos, ingenieros especializados, y otros profesionales con una dilatada experiencia técnica. Estos profesionales siempre han estudiado estos sistemas siguiendo una notación algebraica, estable desde hace más de 170 años (Cajori, 1993). Leibniz empleó el símbolo $\int$ para la suma de enteros y la integración. El uso del símbolo $\sum$ para indicar la suma lo introdujo Euler en 1755. Variaciones en el uso de $\sum$ se pueden encontrar en trabajos de Lagrange, Cauchy, Fourier y Jacobi alrededor de 1820. Desde entonces, excepto para situaciones muy concretas, el empleo de estos símbolos ha permanecido estable, como un estándar.

Sin embargo, es razonable especular que la notación matemática tendría actualmente un enfoque muy distinto si las computadoras, proyectores, multimedia, Internet, la propiedad intelectual y otros conceptos más modernos existieran durante la vida de Euler o Lagrange. Como ocurre con otros lenguajes, la notación matemática debe seguir evolucionando influenciada por el avance social.

Sin llevar este aspecto a un contexto muy general, en áreas concretas como el modelado de redes logísticas cierta información algebraica se puede dar por supuesta ocultando por tanto esta información, y mostrando los elementos bajo una perspectiva gráfica y no algebraica. Por ejemplo, se pueden utilizar nodos para representar recursos y arcos para representar los flujos de estos recursos. Evidentemente, el nivel de detalle debe depender directamente de los profesionales implicados en el proyecto.

## 3. MODELO TEÓRICO

### *3.1 Introducción.*

En esta sección se propone el uso del Metamodelado y de Gramáticas de grafos como un medio de especificar formalmente problemas de optimización y responder a un amplio conjunto de preguntas a lanzar sobre el problema. Un modelo de un formalismo se llama *Metamodelo*. Un metamodelo define las primitivas y las propiedades sintácticas y semánticas de un modelo (Jones, 1990). Empleando metamodelos se pueden describir de forma gráfica formalismos de optimización y utilizar estos metamodelos para generar automáticamente herramientas de edición para estos formalismos. Por medio de las gramáticas de grafos (Jones, 1991) se pueden describir de manera formal el tipo de transformaciones permitidas sobre los modelos (generación de herramientas de edición gráfica, validación de los modelos en tiempo de edición y generación de distintas representaciones del modelo, entre otros).

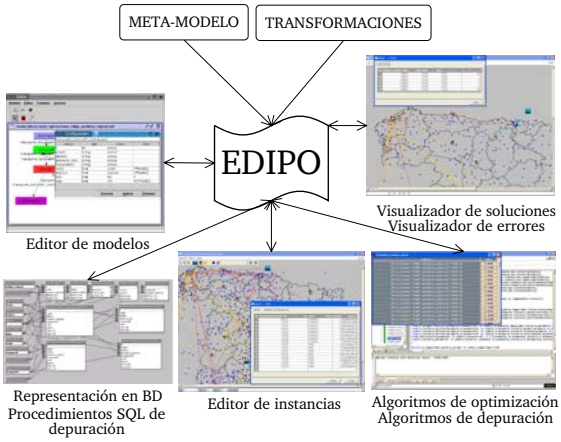


Figura 1. Arquitectura de EDIPO. Dado un Metamodelo y un conjunto de transformaciones, el sistema pone a disposición del usuario un conjunto de herramientas para resolver problemas de optimización.

A continuación se presenta una gramática de grafos que permite la depuración estructural de los modelos definidos mediante este formalismo. Dado que estos formalismos se basan en gramática de grafos, se usará la terminología *Modelado de problemas de optimización basados en grafos*.

En la Figura 1 se representa un esquema de la implementación software de los formalismos que se describen en esta Sección. Se ha implementado un prototipo, EDIPO[1], en el que dado un metamodelo y sus transformaciones asociadas, pone a disposición del usuario un conjunto de herramientas de modelado y optimización. El núcleo del sistema, EDIPO, genera en tiempo de ejecución las herramientas gráficas de modelado, conexión a datos, editores de instancias, modelo de resolución, visores gráficos de soluciones y modelos de depuración en caso de errores en los datos, tal como se aprecia en la Figura 1.

En lo que sigue se describen estos formalismos, necesarios para describir de forma gráfica un problema de optimización, a saber, el Metamodelo, la Gramática de grafos y las Transformaciones.

En ningún caso se detallan las definiciones y teoremas necesarios para la formalización de este método de modelado, sino que se hace una breve mención a las características principales del proceso. Para más detalle, se recomienda ver (Risco-Martín, 2004).

En la Figura 2 se representa este proceso de una forma más esquemática. El metamodelo da lugar a la edición de modelos. El conjunto de transformaciones integradas en la librería basada en conocimiento genera todas las utilidades de gestión y optimización del modelo.

[1] EDItor de Problemas de Optimización

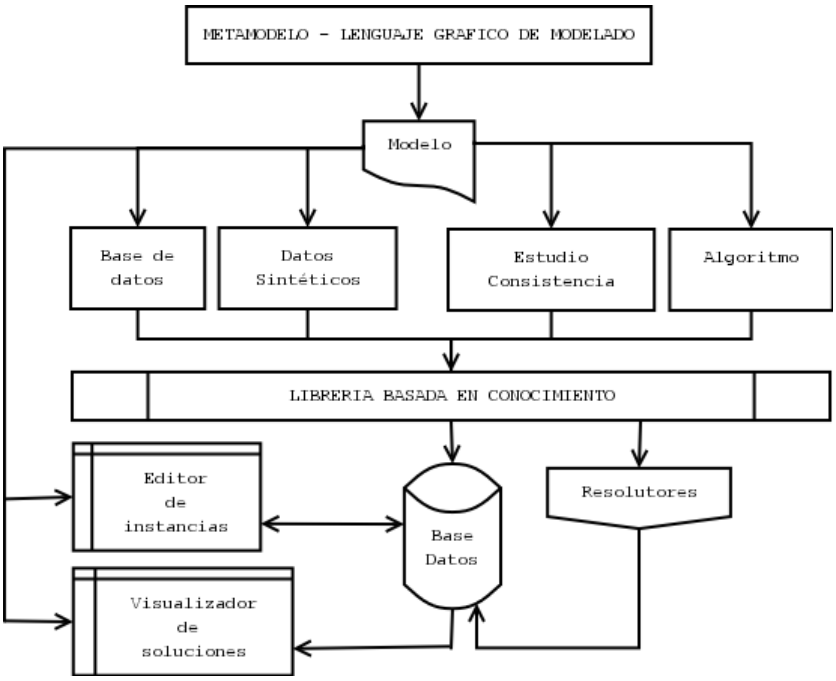


Figura 2. Esquema de desarrollo de las distintas fases del modelado. El metamodelo y el conjunto de transformaciones generan un sistema integrado de gestión logística y optimización.

*3.2 El Metamodelo*.

Como se ha mencionado en la sección anterior, un modelo de un formalismo se llama *Metamodelo*. Un metamodelo define las primitivas y las propiedades sintácticas y semánticas de un modelo (Jones, 1990).

A diferencia de otros enfoques más formales, los metamodelos están orientados a la generación de representaciones visuales de aspectos concretos del sistema de forma incremental y flexible. Los modelos crecen incorporando más detalle gracias a que no es necesario que se instancien absolutamente todos los elementos del metamodelo para tener un modelo. Como ha demostrado el Lenguaje Unificado de Modelado (UML) (Booch *et al.*, 1999) construido también con metamodelos, este tipo de notación facilita enormemente el desarrollo de sistemas.

Los Diagramas Entidad Relación (DER) (Bachman, 1969) también se construyen con metamodelos. Otra ventaja de utilizar metamodelos es que las especificaciones generadas son lo suficientemente estructuradas para ser procesadas de forma automática. Así, se puede plantear generar documentación del sistema en diferentes formatos de diferentes partes de los modelos, e incluso establecer la generación automática de código desde la información recogida en los modelos.

Además se debe cumplir la verificación automática de la construcción de los modelos para ver si cumplen restricciones identificadas por el desarrollador. Para ser capaces de especificar los formalismos de manera completa, los metamodelos deben extenderse con la habilidad de expresar restricciones, e identificar así qué modelos son correctos y cuáles no lo son.

*3.3 Gramática de grafos de optimización.*

En el enfoque que se presenta, los metamodelos se almacenan como grafos, y se usa la gramática de grafos para expresar la manipulación de estos modelos. Algunas manipulaciones que resultan interesantes en el campo del modelado y la optimización incluyen:

- Transformación de un modelo expresado en un formalismo a otro modelo expresado en un formalismo distinto pero equivalente en cuanto al comportamiento. Esto es útil si el formalismo destino permite la resolución del problema de optimización de una forma más sencilla que en el formalismo origen.
- Depuración estructural de modelos, realizando sugerencias de las posibles alternativas para subsanar los errores.
- Generación de código para su procesamiento posterior mediante otras herramientas.

La gramática de grafos se puede usar para describir transformaciones de grafos, o para generar conjuntos de grafos válidos (Ehrig *et al.*, 1999). Por otra parte, el uso de un modelo (en forma de gramática de grafo) para representar manipulaciones de modelos tiene ciertas ventajas sobre una representación implícita (es decir, representando la computación que realiza la transformación en un programa textual) (Blonstein *et al.*, 1996). Por ejemplo, las gramáticas de grafos son una representación gráfica, abstracta, formal, declarativa y de alto nivel de computaciones.

Además, los fundamentos teóricos de los sistemas de reescritura de grafos pueden ayudar a probar propiedades de la transformación, como la corrección y la convergencia (terminación).

En este contexto, el modelado gráfico de problemas de optimización, las ventajas de las gramáticas de grafos son mayores que los inconvenientes. De esta forma se ha implementado un editor de problemas de optimización mediante gramáticas de grafos. Este editor se ha podido implementar de forma totalmente gráfica (además de rigurosa), lo que lo hace fácilmente entendible. La ejecución iterada de las reglas de la gramática del editor permite la corrección de problemas estructurales en el modelo.

El concepto de *Modelado de problemas de optimización basado en grafos* desarrollado, parte del formalismo de la gramática de grafos para describir la construcción de diferentes modelos de optimización y está basado en investigaciones realizadas en sistemas con base en gramática de grafos en campos generales (Jones, 1990). El uso del formalismo de la gramática de grafos facilita la creación de manipulaciones directas, interfaces gráficas de usuario para problemas de optimización que pueden expresarse como grafos atribuidos. En definitiva permite la definición del *Metamodelo* y de las *Transformaciones* de una forma cómoda para el usuario final.

### *3.4 Transformaciones.*

Las transformaciones son un aspecto básico de manipulación de modelos. Las reglas de transformación manipulan un modelo especificado en un formalismo para transformarlo en el mismo modelo especificado en un formalismo distinto. El conocimiento base contiene inicialmente un conjunto de reglas de transformación que trabajan con los modelos definidos en el formalismo descrito. En definitiva, las transformaciones permiten representar un modelo bajo distintos formalismos, ya sean bases de datos, editores gráficos, hojas de cálculo, documentos XML, diagramas UML, modelos de resolución, etc. En este trabajo se han definido tres niveles de transformaciones:

- *Modelo de datos*. En este caso al aplicar las reglas correspondientes a cada uno de los elementos del modelo gráfico se obtiene la representación del mismo en una base de datos. Como se ha mencionado el formalismo descrito está orientado a sistemas de gran escala de datos, y por lo tanto el mejor lugar para almacenar las instancias será una base de datos. La estructura generada en este caso consiste en aplicar una generalización realizada a las teorías desarrolladas por Robert Fourer en el campo de la *Programación Lineal* (Fourer, 1997), extendiéndolas a la *Programación Matemática* en general. En nuestro caso no sólo se generan las tablas donde almacenar los datos que el resolutor necesita, sino las tablas donde almacenar los resultados de la optimización, las tablas donde almacenar resultados del módulo de depuración y las tablas para almacenar el problema en su forma dual, entre otras.
- *Modelo de optimización*. Aplicando las transformaciones de este nivel se genera el modelo del resolutor para a continuación relacionar el modelo con una instancia del mismo y resolver el problema.
- *Modelo de depuración*. Las reglas de transformación en este nivel se dividen a su vez en dos categorías, a saber, *modelo de depuración mediante consultas* en que se generan consultas sobre los datos para analizar la consistencia de la instancia, y *modelo de depuración*, en que se genera un modelo equivalente al modelo de resolución pero aplicando las teorías más conocidas de depuración. Estos modelos se generan en el ámbito de bases de datos o en el ámbito del resolutor a utilizar.

## 4. EJEMPLO. RED LOGÍSTICA DE DISTRIBUCIÓN

### *4.1 Introducción.*

En este caso se va a resolver una red logística de distribución utilizando EDIPO. Es importante señalar que el sistema se puede utilizar con fines pedagógicos para poner de manifiesto la repercusión de ciertas decisiones tomadas sobre escenarios tipo. El entorno de visualización gráfico de las soluciones calculadas por el optimizador es un complemento valioso a los resultados numéricos. Este entorno resulta indispensable para que el usuario pueda apreciar desde una perspectiva global la calidad de las soluciones propuestas por el optimizador. Utiliza el contraste de colores para detectar de forma inmediata la anomalía de ciertos resultados puntuales y poder identificar la causa que los produce. Las restricciones de la red de suministro, producción y distribución pueden ser editadas por el usuario en función de las circunstancias concretas que se prevean para el período de optimización. También se pueden desactivar selectivamente para medir el efecto sobre la solución o bien para relajar la red en ausencia de soluciones factibles. No obstante el sistema dispone de un módulo que determina automáticamente las restricciones que causan la ausencia de solución y el grado de incumplimiento de dichas restricciones.

Los modelos de resolución que se generan utilizan diferentes métodos de optimización en función del perfil computacional y la naturaleza de las restricciones de cada nivel. Fundamentalmente emplean programación lineal, programación entera mixta y programación con restricciones de dominios finitos. Sin embargo, cuando la combinatoria del problema resulta difícil de abordar dentro de un único paradigma de optimización, el sistema utiliza técnicas de hibridación (Risco-Martín *et al.*, 2004).

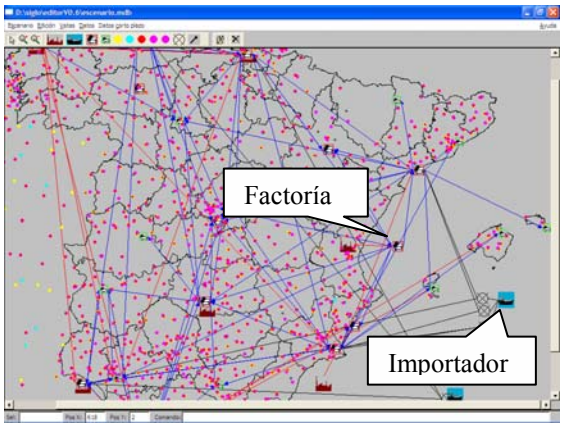


Figura 3. Red logística de distribución. Con el modelo gráfico de la red, EDIPO genera un editor de instancias correspondiente a este modelo. Con este editor se pueden gestionar todos los datos de la instancia concreta a resolver.

### *4.2 Planteamiento del problema y objetivos*.

En la Figura 3 se muestra una representación gráfica de la red logística de distribución. Aunque el proceso natural es crear en primer lugar el modelo gráfico de la red (Figura 4) se comienza ilustrando la instancia para proceder a identificar los distintos elementos que intervienen en la misma. Se han omitido los arcos que representan el transporte secundario debido a la saturación que representa.

La red en estudio se corresponde con una modificación del sistema logístico de una conocida compañía. Esta compañía elabora presupuestos anuales y planifica sus actividades mes a mes. El primer paso para resolver el problema consiste en identificar los *tipos de elementos* que intervienen en la red, para después representar la misma con el editor de modelos. En el ejemplo que se propone se distingue un tipo de fuente (llámese importador), un tipo de intermediario (llámese factoría) y tres tipos de nodos sumidero (llámense distribuidor, granel y cliente). Los arcos representan el transporte de producto por la red, que puede realizarse mediante barco, tren o camión cisterna). La red se corresponde con el siguiente esquema:

- La red logística está discretizada en un intervalo temporal de un año, tomando como periodo un mes.
- Existe una serie de importadores que proporcionan materia prima a las factorías. Los importadores tienen contratado un aprovisionamiento mínimo. El aprovisionamiento se realiza a una sola factoría por periodo y además se considera discreto. Esto es así porque se produce por medio de barcos y las cantidades firmadas con estos son enteras. Este transporte lleva un coste asociado.
- La materia prima se procesa en las factorías para producir producto comercial. Las factorías disponen de una capacidad de almacenamiento de materia prima y producto comercial. En este caso se asignarán producciones discretas, en que la multiplicidad de composición sea un número entero.
- El producto comercial llega a los sumideros, los cuales tienen preestablecida una cierta demanda. De igual forma se va a establecer la demanda como un valor entero, aunque el transporte a todos los sumideros se considerará continuo. En este caso las demandas firmadas con los clientes son cantidades enteras. Sin embargo el transporte se realiza por tren o camión cisterna, y en este caso no es importante que las cantidades sean discretas. En este caso se produce beneficio por la venta final del producto comercial y un coste de transporte del producto.

El modelo de optimización *minimiza costes* y *maximiza beneficios*. Los cotes se generan como resultado del transporte, del aprovisionamiento de los importadores y de la producción en las factorías. Los beneficios se generan tras el abastecimiento a los clientes del producto demandado. El objetivo es el de proporcionar herramientas de modelado, optimización y gestión de los datos que faciliten el análisis de la red logística global para la toma de decisiones.

Los elementos necesarios para analizar el problema son:

- *Nodos de producción*. Modelan el comportamiento tanto de los importadores y factorías como el de los clientes. El nodo importador es un nodo de distribución tipo fuente, el nodo factoría es un nodo de producción en su esquema más general, ya que transforma materia prima en producto comercial; el nodo cliente es un nodo de distribución tipo sumidero, ya que no realiza ninguna producción y fija una demanda.
- *Arcos de transporte*. Modelan el transporte de importador a factoría, el transporte interno entre factorías, y el transporte de factorías a clientes.
- *Arcos de propagación*. Modelan el almacén de materia prima existente tanto en importadores, factorías y clientes, y el almacén de producto comercial en las factorías.

Una vez definido el modelo se podrán generar datos sintéticos para una evaluación inicial de la red. De igual forma se introducirán errores para analizar la consistencia de la misma. Los pasos a seguir son los siguientes:

1. Se han de definir los elementos de la red, qué atributos gráficos poseen y qué atributos de optimización (Metamodelo).
2. Se construye un modelo válido para estas reglas y que se ajuste al problema que se pretende resolver. Este modelo se define mediante EDIPO, en formato gráfico.
3. Se construyen las transformaciones orientadas a generar el modelo bajo algún lenguaje de optimización (OPL, AMPL, ECLIPSE ...), las

sentencias de creación de la base de datos que almacene las instancias con sus correspondientes chequeos de consistencia, los modelos de optimización orientados a resolver conflictos en caso de que la instancia no sea factible, y finalmente la creación de datos sintéticos.

Los puntos 1 y 2 se encapsulan en la herramienta EDIPO, incorporándole como módulo de apoyo el conjunto de transformaciones previamente implementadas.

### *4.3 Elementos que intervienen en la red logística.*

En la Figura 4 se representa el modelo gráfico de la red de distribución construido con EDIPO, el cual contempla todos los elementos de su red logística. Mediante la ventana de configuración se definen los atributos de optimización, i.e., se definen los atributos del Metamodelo definido. Con ello se crea el modelo gráfico que posteriormente se utilizará para generar el editor gráfico de instancias, la base de datos y los datos sintéticos, el modelo de optimización y el visualizador de soluciones. Los elementos que intervienen en el modelo se describen a continuación.

*Importadores.* Con vistas a la instancia conviene considerar que se ha de replicar cada importador de la red física tantas veces como periodos existan en el intervalo de optimización y tipos de materia prima permita importar a la red. Con vistas al modelo, por tanto, existe una clase *importador* que es un nodo de distribución tipo fuente, indexado en tiempo y materia prima; en este sentido también se debe proporcionar el nombre del importador, que identifica el nodo físico de la red real. Además, el importador dispone de una capacidad de almacenamiento.

*Clientes.* Como se ha mencionado, en el problema a tratar existen tres tipos o clases de clientes: distribuidores de granel, agrupaciones de envasado y grandes clientes. Todos ellos reciben producto comercial y por lo tanto se ha de hacer tantas réplicas como producto comercial demanden. Así pues, se modelarán tres clases de cliente, *distribuidor*, *granel* y *cliente*.

*Factorías.* El modelado de las factorías requiere un nodo de producción, pues produce producto comercial a partir de materia prima. Recuérdese que las factorías admiten almacenamiento de materia prima y producto comercial, así como transporte interno entre factorías.

*Transporte importador-factoría.* En este caso el medio de transporte se considerará único (por barco) y discreto. Se definen también límites inferiores y superiores para establecer la capacidad del transporte de cada producto, así como un coste al problema de optimización.

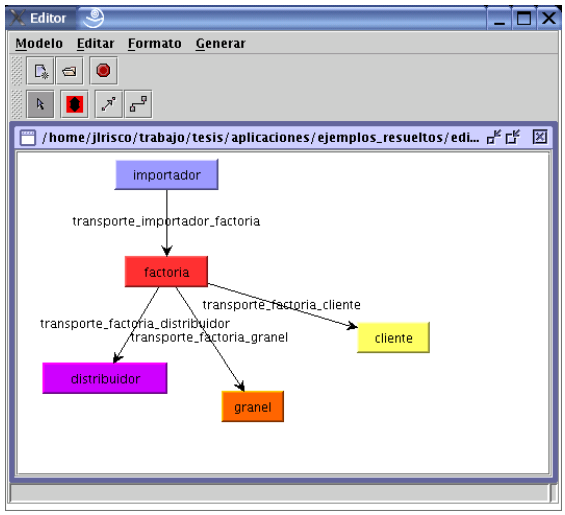


Figura 4. Modelo gráfico de la red. Dado un modelo, EDIPO genera la base de datos, el editor de instancias, el visualizador de soluciones y los modelos de optimización y depuración.

*Transporte factoría-cliente.* Se engloban en este apartado todas las clases cliente, a saber, distribuidores, agrupaciones de granel y grandes clientes. Los medios de transporte disponibles son tren y camión cisterna. En este caso se presentan también los costes de transporte y la capacidad del mismo. Sin embargo aparece un ingreso debido a la venta final, de la que se obtendrán beneficios.

*Transporte interno factoría.* Se produce si se transporta producto de una factoría a otra. Los medios de transporte disponibles para este transporte son barco, tren y camión cisterna. El transporte por este medio lleva consigo un coste implícito en el arco. El transporte interno entre factorías se refleja como un arco que parte de la clase factoría a sí misma. Se dispone de un coste de transporte, de los límites asociados y de los índices, en este caso tiempo, origen, destino, producto y medio de transporte.

*Arcos de propagación.* En EDIPO, los arcos de propagación se definen en la ventana de configuración, y se corresponden exactamente con la capacidad de almacenamiento. Disponen de capacidad de almacenamiento todos los nodos de la red.

### *4.4 Base de datos.*

*Generación de la base de datos de la instancia.* Una vez diseñado el modelo, la primera transformación a aplicar consiste en la generación de la estructura de la base de datos que contendrá la instancia del modelo. En la sección 3.4 se establecieron las bases de generación de la base de datos, la forma y estructura de la misma. La generación de la base de datos se establece para que permita realizar comprobaciones de consistencia, que existan disparadores que alerten de algún dato mal introducido.

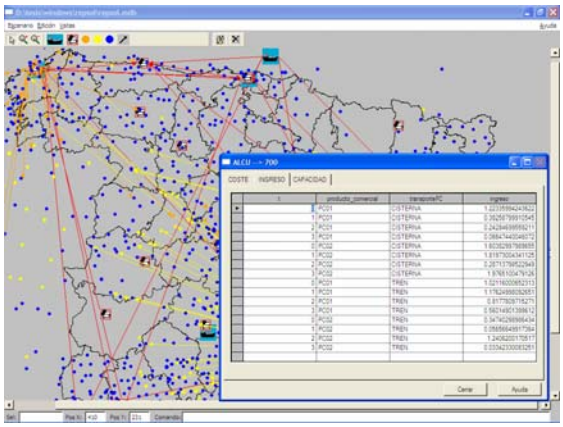

Figura 5. Edición gráfica de la instancia. Pulsando en los distintos elementos de la red el editor hace un filtro en la base de datos recogiendo los que pertenecen al elemento seleccionado.

EDIPO genera directamente la estructura relacional de la base de datos, y la lanza sobre el equipo y la arquitectura seleccionada. En este caso el sistema de base de datos seleccionado es MySQL, y se ubica en un servidor del Departamento de Arquitectura de Computadores y Automática de la Universidad Complutense.

*Generación de los datos sintéticos.* La generación de datos sintéticos requiere de cierto conocimiento de los límites establecidos en el modelo, así como la generación de los índices de los que inicialmente no se cuenta con datos. Estos se generarán aleatoriamente en términos de unos límites previamente establecidos. Esta característica es de gran utilidad para realizar estudios previos del modelo. De esta forma se generan cuatro tipos de materia prima y dos tipos de producto comercial. La relación de producción se ha de introducir manualmente y se corresponde con la Tabla 1. Es una producción discreta, en la que para generar una unidad de producto PC01 se necesitan cuatro unidades de MP01 y tres unidades de MP02, y para generar una unidad de producto PC02 se necesitan cuatro unidades de MP03 y tres unidades de MP04.

Tabla 1. Tabla de producción.

| Componente | Producto | Proporción |
|---|---|---|
| MP01 | PC01 | 4 |
| MP02 | PC01 | 3 |
| MP03 | PC02 | 4 |
| MP04 | PC02 | 3 |

Para generar los parámetros tales como límites en los arcos o límites de almacenamiento se utilizan los recursos de la base de datos. Cada elemento instanciado del modelo generará sus datos en un intervalo previamente definido en el editor de modelos.

Una vez especificadas las reglas de generación de datos se crea la instancia, formada inicialmente por datos sintéticos. Las coordenadas de los nodos se generarían también de forma aleatoria, sin embargo, como se ha mencionado anteriormente se utilizarán las coordenadas de los elementos de la red de distribución logística.

En el caso que nos ocupa, se consideran 10 importadores, 24 factorías, 63 clientes, 823 distribuidores y 813 clientes de granel.

### *4.5 Edición gráfica de la instancia.*

En la Figura 5 se muestra la edición de los datos sintéticos a través del editor de instancias. El editor de instancias no necesita ser reprogramado en ningún caso. Toma el modelo creado y representa la base de datos seleccionada.

Mediante esta herramienta se pueden editar los datos en un contexto gráfico. Basta con pulsar dos veces con el ratón sobre el elemento del que se desea extraer información, realizándose un filtro sobre la base de datos para mostrar los relacionados con el elemento. En la Figura 5 se muestra la edición de uno de los arcos de importador a factoría.

### *4.6 Optimización.*

La optimización de la instancia se realiza desde el entorno en que se haya generado el modelo. En el caso que se está describiendo el modelo generado se corresponde con ILOG OPL, por lo que se optimiza desde esta herramienta. Dada la variedad de elementos, medios de transporte y productos utilizados en la distribución, el problema cuenta con 58000 variables enteras y 125000 restricciones. Además, la optimización se ha realizado en un intervalo de un año dividido en periodos de un mes, lo que explica el alto número de variables y restricciones.

### *4.7 Visualización gráfica de la solución.*

Una vez optimizada la instancia se pueden observar los datos mediante el sistema de base de datos elegido, en este caso MySQL. Sin embargo el sistema implementado permite utilizar el módulo de visualización de datos gráfico, el *visualizador de soluciones*. En la Figura 6 se muestra la solución encontrada en formato gráfico.

La herramienta toma como entrada el modelo gráfico inicial y en consecuencia genera automáticamente el visualizador de soluciones. Este a su vez toma como entrada el modelo de optimización proporcionando una visualización de la solución a través de la base de datos especificada en el modelo. Al igual que el editor de instancias esta herramienta no necesita reprogramarse bajo distintas instancias o modelos, ya que EDIPO la configura automáticamente.

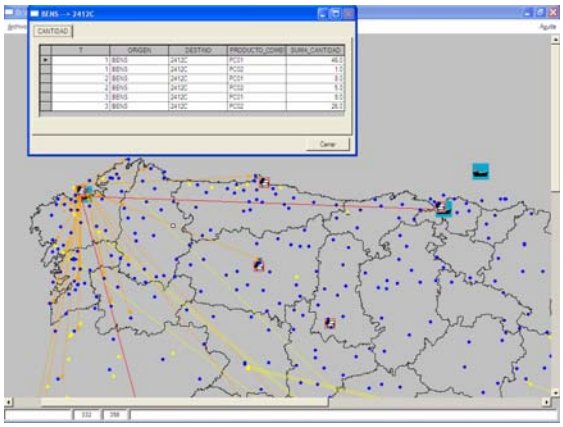

Figura 6. Visualización gráfica de la solución. Examinando un elemento de la red (en este caso una factoría), se visualizan las cantidades que el optimizador ha determinado para el transporte.

### *4.8 Depuración ante fallos de la red.*

La detección de errores en los datos en un sistema de gran escala puede ser un trabajo bastante tedioso. Con objeto de facilitar este trabajo se han definido en EDIPO tres procedimientos de depuración:

1. Depuración gráfica.
2. Depuración en base de datos.
3. Depuración mediante optimización.

En cuanto a los puntos 2 y 3, el conjunto de transformaciones definidas generan modelos de depuración del sistema a resolver. EDIPO dispone de dos procedimientos para generar estos modelos: El primero consiste en crear consultas a realizar en la base de datos. El segundo consiste en generar automáticamente modelos de optimización que introduciendo variables de acoplo o generando nodos o arcos ficticios permita detectar un amplio rango de posibles errores en la red.

*Depuración gráfica.* Este es el procedimiento más rápido. Para ello el editor de instancias recorre la red en busca de nodos desconectados y avisa al usuario con un listado de los mismos.

*Depuración en base datos*. En este caso EDIPO genera automáticamente consultas SQL de depuración de la instancia. Estas consultas comprueban la consistencia de los límites. Por ejemplo, si los límites inferiores de demanda de un cliente son mayores que los límites superiores en el transporte al mismo, las consultas detectan el error y el sistema se lo comunica al usuario.

*Depuración mediante optimización*. Este procedimiento es más costoso en tiempo, ya que implica la ejecución de nuevas optimizaciones. Por el contrario, detecta un mayor rango de posibles errores en la instancia.

- Errores en los límites: Desde un punto de vista gráfico y en el ejemplo en estudio este error se produce cuando existe algún importador desconectado, alguna factoría sin suministro o algún cliente que demanda producto y está desconectado. Para detectar el error se incluyen en el modelo de optimización original variables de acoplo en las restricciones de igualdad y de desigualdad, de esta forma se detectan las restricciones que no se satisfacen y la medida en que lo hacen. En consecuencia EDIPO avisa al usuario indicando el elemento gráfico que induce el error. En términos generales, si el error proviene de una ecuación de balance este se encuentra en un nodo, probablemente en sus límites de almacenamiento. Si por el contrario proviene de una desigualdad, el sistema indica al usuario que por un determinado arco no se puede transportar la cantidad necesaria.
- Errores de indexación: Desde un punto de vista gráfico estos errores se producen cuando existe un arco sin origen y/o destino o cuando se intenta producir un producto comercial con materia prima que no existe. En problemas de transporte este es el error más difícil de detectar, ya que la mayoría de resolutores no pueden proporcionar al usuario un mensaje claro de error. Utilizando el editor de instancias generado por EDIPO *este error no puede darse si el metamodelo es correcto*, debido a que la instancia debe ajustarse perfectamente a las reglas del modelo, y este a su vez a las reglas del metamodelo. Aún así se han desarrollado procedimientos para detectar estos errores. Para ello se genera un modelo de depuración que incluya *elementos virtuales*, que pueden ser nodos, arcos o cualquier tipo de producto o medio de transporte. Las variables de decisión asociadas toman valor no nulo únicamente si la instancia presenta un error de esta naturaleza.

Para mostrar la detección de errores se introducirán algunos y se procederá a su estudio. El modelo de optimización utilizado es el generado mediante EDIPO. En este caso se introduce un error de límites en las variables, haciendo que el límite inferior de un nodo demandante sea mayor que el límite superior. En el cliente 4409C se modifica el límite inferior de la cantidad de producto comercial P01 demandado. Originalmente el valor del límite inferior es 8, mientras que el valor del límite superior es 33. El error introducido es modificar el límite inferior para que pase a valer 34, mayor que el límite superior.

Al realizar la optimización, el resolutor indica que el problema no es factible, por lo que se debe proceder a una depuración. EDIPO genera automáticamente modelos de depuración. Al ejecutar estos modelos se completan tablas auxiliares de salida de datos con el resultado de las modificaciones a realizar en caso de que la red no sea factible. Una vez detectados los errores se deben localizar. Inspeccionar las tablas auxiliares en busca de valores no nulos puede ser complicado. En este caso se utiliza el *visor de soluciones* que tiene introducida la opción de representar únicamente los errores en la red.

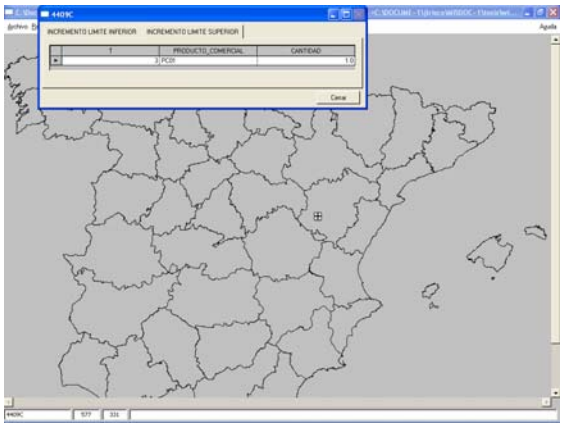

Figura 7. Representación de los errores en la instancia. En el ejemplo en estudio se detecta un nodo cuyos límites de almacén son incorrectos, ya que generan inconsistencias en la optimización.

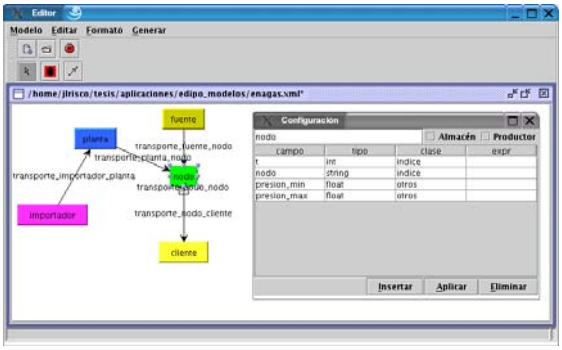


Figura 8. Modelo de la red de transporte de gas.

En la Figura 7 se ilustra el resultado de la visualización gráfica del modelo depurado. Se representa únicamente el cliente 4409C, que es el que contiene el error. Al editar los datos del nodo, se representan los valores sugeridos por el optimizador para que la red sea factible. En este caso se recomienda incrementar el límite superior del nodo en una unidad, y no se sugiere nada para el límite inferior.

## 5. OTROS PROBLEMAS MODELADOS CON EDIPO

La importancia de los formalismos descritos y de la herramienta implementada con estos formalismos reside en que con el editor de modelos se pueden generar las distintas interfaces gráficas necesarias para gestionar y resolver el problema, así como depurar los posibles errores que puedan presentar los datos. Con EDIPO se han resuelto varios problemas de optimización de gran escala de sistemas reales, correspondientes a escenarios de reconocidas empresas de distribución logística. Las restricciones indicadas en los ejemplos se corresponden, de una forma cualitativa, con los escenarios reales mencionados. A continuación se describen brevemente otros dos problemas resueltos con la herramienta implementada. Para más detalle, se recomienda ver (Risco-Martín, 2004).

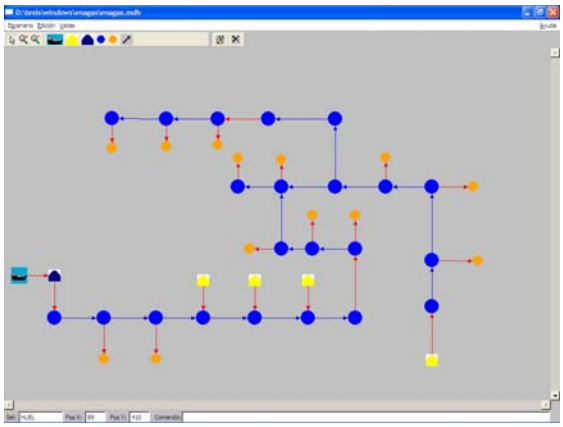

Figura 9. Editor de instancias de la red de transporte de gas. El editor lo genera EDIPO del modelo gráfico correspondiente.

### *5.1 Red de transporte de gas.*

La red en estudio se corresponde con la de una importante empresa de distribución de gas. Se compone de un conjunto de elementos interconectados entre sí. Cada elemento del sistema se corresponde con un componente real de la red y simula su comportamiento estratégico. Las variables asociadas a estos elementos son en general flujos de gas y presiones. Existe un elemento especial, el nodo, que se utiliza para la interconexión entre sí de otros elementos. Las interconexiones son redes de tuberías bidireccionales que representan los flujos de gas.

Los elementos que se distinguen en el modelo de la red son: plantas, fuentes continuas, barcos, nodos, tramos (tuberías), compresores y reguladores, cada uno con sus propiedades físicas y que aportan un conjunto de restricciones y de variables de decisión al modelo de optimización. El Metamodelo creado contempla estos elementos, por lo que se puede utilizar el editor de modelos de EDIPO para crear el modelo de la red, tal como ilustra la Figura 8.

Una vez creado y salvado el modelo gráfico, se puede invocar a EDIPO para que cree el editor de instancias, que con los datos de la misma permite su edición gráfica, tal como se aprecia en la Figura 9.

Una vez creada la instancia se genera el modelo de optimización, y con él se resuelve el problema. Después de este paso se genera el visor de soluciones para poder visualizar de forma gráfica el problema optimizado (Figura 10).

### *5.2 Red de poliductos.*

El problema consiste en optimizar la estrategia de funcionamiento de una red de poliductos en cuanto a tiempo de proceso y cambio de tipos de producto en el transporte se refiere. Los poliductos son redes de tuberías destinadas al transporte de hidrocarburos o productos derivados del petróleo. El transporte de este producto se realiza en paquetes sucesivos.

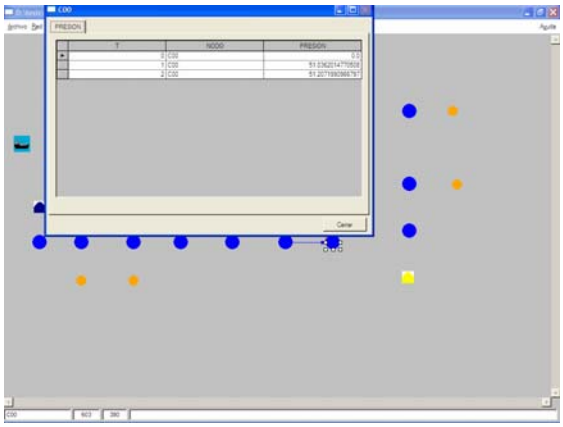

Figura 10. Visualizador de soluciones de la red de transporte de gas. La herramienta la genera EDIPO del modelo gráfico correspondiente.

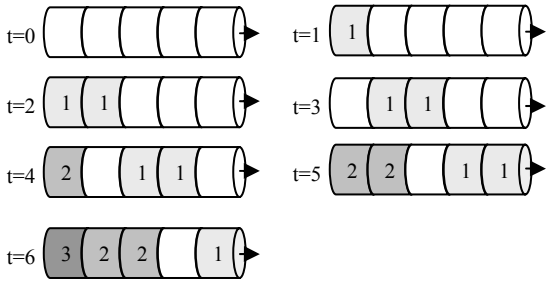


Figura 11. Esquema representativo de la evolución temporal de un poliducto. *t* es el periodo actual y los números indican el tipo de producto. Cada segmento del cilindro representa un tramo.

Un paquete es una cantidad variable del mismo tipo de producto ubicado a lo largo de un poliducto. Desde un punto de vista operativo una red de poliductos estará constituida por una serie de nodos con capacidad de suministro, almacenamiento y recepción de productos, y una serie de arcos que conectan entre sí los nodos. Además, algunos poliductos pueden ser bidireccionales, de forma que mientras se esté transportando producto en un sentido se prohíbe enviar producto en el sentido contrario.

A nivel logístico, el problema que se plantea en las redes de poliductos es el de la planificación temporal del transporte de diferentes productos desde nodos fuentes (oferta) a nodos destino (demanda) pasando por una serie de nodos intermedios. La planificación debe cumplir un conjunto de restricciones temporales, relativas a la fecha mínima y máxima de entrega de los diferentes productos. También deberá cumplir las restricciones relativas a la disponibilidad de productos en las fuentes, y las físicas derivadas de la utilización de los recursos en la red. Se plantea, además, una medida de calidad de la solución en términos de la minimización del intervalo total del tiempo de la planificación, y de la ordenación apropiada de los paquetes sucesivos para conseguir interfases sin mezcla, ver Figura 11. Esta medida de la calidad se suele plantear como función multiobjetivo de un problema de optimización.

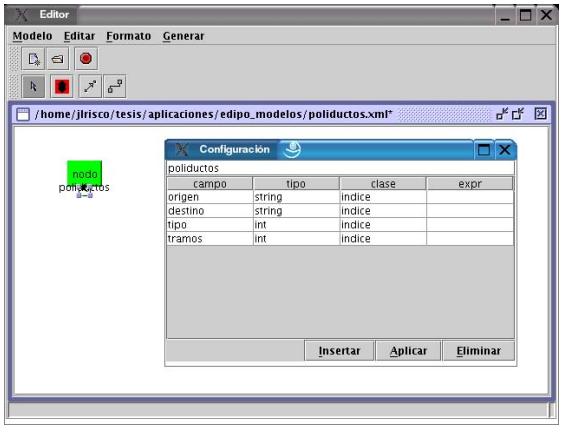


Figura 12. Modelo de la red de poliductos.

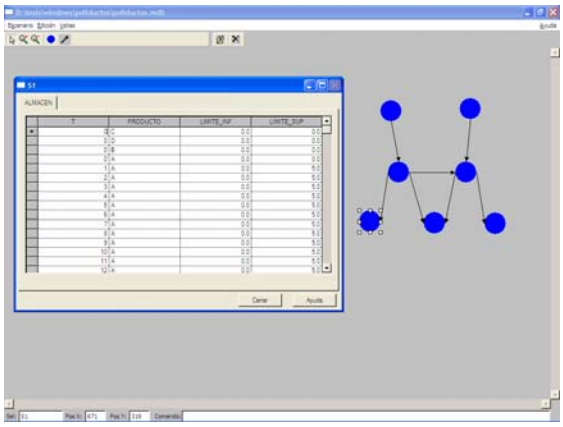

Figura 13. Editor de instancias de la red de poliductos. El editor lo genera EDIPO del modelo gráfico correspondiente.

Los elementos que intervienen en la red son: nodos de distribución con capacidad de almacenamiento y poliductos. En el metamodelo creado en EDIPO están definidos estos elementos, por lo que se puede usar el editor de modelos para diseñar la red, tal como ilustra la Figura 12. En este caso el modelo es simple, ya que la red presenta una única clase de nodo y una única clase de arco, los poliductos.

Con el modelo salvado se genera la base de datos y los distintos entornos gráficos. En la Figura 13 se observa el editor de instancias. Igual que en los ejemplos anteriores, EDIPO genera el modelo de optimización y el visor de soluciones.

## 6. CONCLUSIONES

En este trabajo se han expuesto los formalismos necesarios para modelar gráficamente problemas de optimización. Se han creado un metamodelo y un conjunto de transformaciones válidos para el modelado, gestión y optimización de redes logísticas, generando automáticamente editores gráficos de los datos asociados. Se ha comprobado la validez de este trabajo resolviendo una red de distribución logística., y se han mencionado otros problemas resueltos con el sistema implementado. Aunque las teorías definidas están orientadas a representar problemas que se

puedan modelar mediante grafos, se ha resuelto un Control Predictivo aplicado sobre el sistema híbrido de los tres tanques (Bemporad and Morari, 1999), obteniendo resultados satisfactorios.

## AGRADECIMIENTOS

Los autores quieren agradecer al Ministerio de Ciencia y Tecnología su financiación al proyecto en el que se basa EDIPO (CICYT PI2002-02924).

## REFERENCIAS

Bachman, C.W. (1969). Data Structure Diagrams. *DataBase* **1**(**2**), 4–10.

Bemporad, A. and M. Morari (1999). Control of Systems Integrating Logic, Dynamics, and Constraints. *Automatica* **35**(**3**), 407–427.

Blonstein, D., H. Fahmy and A. Grbavec (1996). Issues in the Practical Use of Graph Rewriting. *Lecture Notes in Computer Science* **1073**, 38–55.

Booch, G., J. Rumbaugh and Jacobson I (1999). *The Unified Modeling Language User Guide*. Addison Wesley.

Cajori, F. (1993). *History of Mathematical Notations*. Dover Publications, Inc.

Castillo, E., A. J. Conejo, P. Pedregal, R. García and N. Alguacil (2002). *Building and Solving Mathematical Programming Models in Engineering and Science*. Pure and Applied Mathematics Series, Wiley, New York.

Checkland, P. (2000). Soft systems methodology: a thirty year retrospective. *Systems Research and Behavioral Science*.

Choobineh, J. (1991). Sqlmp: A data sublanguage for representation and formulation of linear mathematical models. *ORSA Journal on Computing* **3**(**4**), 358–375.

Collaud, G. and J. Pasquier-Boltuck (1994). gLPS: A graphical tool for the definition and manipulation of linear problems. *European Journal of Operations Research* **72**, 277–286.

Ehrig, H., G. Engels, H.J. Kreowski and G. Rozenberg (1999). Handbook of Graph Grammars and Computing by Graph Transformation. *Applications, Languages, and Tools, World Scientific*.

Fourer, R. (1997). Database structures for mathematical programming models. *Decision Support Systems* **20**, 317–344.

Gazmuri, P. and I. Arrate (1995). Modeling and visualization for a production planning. *Decision Support System* **2**, 249–258.

Geoffrion, A. (1987). Introduction to structured modeling. *Management Science* **33**, 547–588.

Greenberg, H.J. (1993). A Computer-Assisted Analysis System for Mathematical Programming Models and Solutions: A User s Guide for ANALYZE. *Kluwer Academic Publishers*.

Greenberg, H.J. (1996). A bibliography for the development of an intelligent mathematical programming system. *ITORMS* **65**, 55–90.

Jones, C.V. (1990). An introduction to graph-based modeling systems, part I: Overview. *ORSA Journal on Computing* **2**(**2**), 136–151.

Jones, C.V. (1991). An introduction to graphbased modeling systems, part II: Graph grammars and the implementation. *ORSA Journal on Computing* **3**(**3**), 180–206.

Jones, C.V. and T.E. Baker (1996). MIMI/G: A graphical environment for mathematical programming and modeling. *Interfaces*.

Ma, P., F.H. Murphy and E.A. Stohr (1996). An Implementation of LPFORM. *INFORMS Journal on Computing*.

Maturana, S. and Y. Eterovic (1995). Vehicle routing and production planning in decision support systems: Designing graphical user interfaces. *International Transactions in Operational Research* **2**(**3**), 233–247.

Maturana, S., P. Gazmuri and C. Villena (1996). Development and implementation of a production planning dss for a chilean manufacturing firm. *Proceedings of the 20th International Conference on Computers & Industrial Engineering*, pp. 757–760.

Mitra, G., B. Kristjansson, C. Lucas and S. Moody (1995). Sets and indices in linear programming modelling and their integration with relational data models. *Computational Optimization and Applications* **4**, 263–292.

Müller-Merbach, H. (1990). Database-oriented design of planning models. *IMA Journal of Mathematics Applied in Business and Industry* **2**, 141–155.

Palmer, K.H. (1984). *A Model-Management framework for Mathematical Programming*. John Wiley & Sons. New York.

Risco-Martín, J.L. (2004). *Sistema Integrado de Gestión Logística Optimizada*. Tesis doctoral. Universidad Complutense de Madrid.

Risco-Martín, J.L., A. Herrán-González, J.M. Cruz-García and B. Andrés-Toro (2004). Hybrid Heuristic and Mathematical Programming in Oil Pipelines Networks. *Journal of Zheijang University*.

Rosenhead, J. (1996). What's the problem? An introduction to problem structuring methods. *Interfaces*.

Sharda, R. and D. Steiger (1995). Using artificial intelligence to enhance model analysis. the impact of emerging technologies. *Computer Science and Operations Research*, pp. 263–279.

Turban, E. (1995). *Decision Support and Expert Systems: Management Support Systems*. Prentice-Hall.

Welch, J.S. (1987). The data management needs of mathematical programming applications. I*MA Journal of Mathematics in Management*, pp. 237–250.

Zimmermann, H.J. (1988). Interactive decision support for semi-structured mathematical programming problems. *Mathematical Models for Decision Support*, pp. 307–319.