X Congreso de Ingeniería de Organización
Valencia, 7 y 8 de septiembre de 2006

Grafos: herramienta informática para el aprendizaje y resolución de problemas reales de teoría de grafos.

Alejandro Rodríguez Villalobos[1]

[1] Dpto. de Organización de Empresas. Escuela Politécnica Superior de Alcoy. Universidad Politécnica de Valencia. Pza. Ferrándiz-Carbonell, 03801 Alcoy. arodriguez@doe.upv.es

Resumen

En este artículo se presenta el estado y resultados alcanzados hasta el momento de un proyecto de investigación que tiene como principal objetivo el desarrollo de una herramienta informática que facilite el aprendizaje y resolución de problemas reales de teoría de grafos. El proyecto comenzó en el año 2003, y actualmente sigue en activo, en proceso de programación de nuevos algoritmos, incorporación de nuevas funciones y generación de documentación actualizada. Este artículo además de difundir el estado y resultados del proyecto, pretende también ser una invitación, para que todo aquel que lo desee pueda sumarse a él y compartir el desarrollo, el conocimiento y otras experiencias relacionadas.

Palabras clave: teoría de grafos, ingeniería de organización industrial, aprendizaje, conocimiento compartido

1. Proyecto *Grafos*

Grafos es una herramienta informática para la construcción, edición y análisis de grafos que pretende ser de utilidad para la docencia, aprendizaje y práctica de la teoría de grafos (*Graph Theory*), Dieter (2004), y otras disciplinas relacionadas como la investigación operativa, diseño de redes, ingeniería de organización industrial, la logística y el transporte, etc. Un grafo puede representar en forma de red un modelo de una realidad empresarial Este modelo podrá ser analizado desde distintos puntos de vista gracias a los algoritmos y funciones incorporados en la herramienta.

La herramienta informática, se puede usar sin restricciones para el modelado, diseño de grafos, análisis y resolución de problemas reales, a partir de los algoritmos y funciones descritas en este artículo. El desarrollo de *Grafos* se ha estructurado en base a dos grandes objetivos:

- Desarrollo de un interfaz para la construcción y edición de grafos en modo tabular o gráfico (Figura 1), que permita la incorporación modular de multitud de funciones.
- Desarrollo de una estructura de clases y librerías .dll con algoritmos de resolución y análisis de problemas de teoría de grafos. Implementación y compilación de diferentes algoritmos.

Independientemente de sus conocimientos actuales sobre la materia, la información recogida en la página web del proyecto puede ser un buen punto de partida para el aprendizaje en mayor profundidad de la teoría de grafos y su aplicación en la realidad empresarial e industrial. Todo el material recopilado en el sitio web del proyecto (software, textos, imágenes, casos, etc.), se distribuye bajo licencia *Creative Commons License*, por lo que se

puede decir que otro de los objetivos de este proyecto es la difusión libre de conocimiento (*http://personales.upv.es/arodrigu/grafos*).

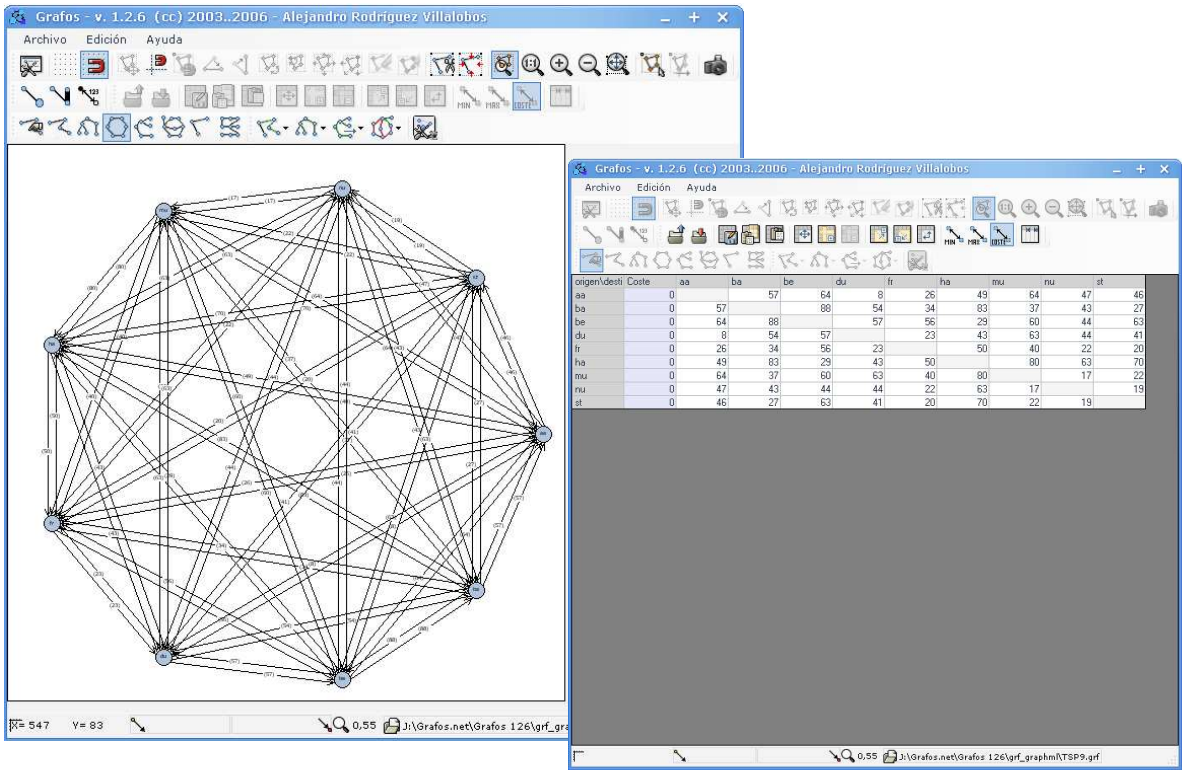

Figura 1. Pantallas de Grafos. Ejemplos de edición gráfica y tabular.

2. Aprendizaje

La filosofía de esta herramienta es la siguiente: *"dibujar, modelar, resolver y analizar"*. Con esto se pretende que el usuario tenga libertad absoluta para tratar y abordar los problemas de grafos. El usuario puede dibujar libremente el grafo sin preocuparse del análisis o algoritmo que utilizará posteriormente. El programa le avisará en caso de no factibilidad o de cualquier otro requerimiento para un análisis en particular. Los estudiantes que usen *Grafos* experimentarán un proceso de aprendizaje basado en su libertad y en etapas de prueba-error (aprendizaje a través del juego), Gallardo (2000). Otros programas existentes, a diferencia de este, guían al usuario paso a paso según el tipo de grafo y problema a resolver, descartando de entrada su libertad de elección y construcción. Esta libertad de cara al usuario ha implicado una mayor complejidad en el desarrollo del código fuente, ya que no sólo hay que contemplar los supuestos aplicables al tipo de análisis o problema a resolver, sino cualquier escenario que pueda generar la interacción con el usuario (incluyendo los de no factibilidad).

3. Desarrollo y estructura

Grafos está desarrollado en Microsoft Visual Studio 2005 (el código fuente actualmente está íntegramente programado en Visual Basic .net 2005, aunque podría integrar fácilmente módulos desarrollados en otros lenguajes de programación .net). Esta plataforma de desarrollo garantiza su funcionamiento en Microsoft Windows y facilita su adaptación a futuras versiones de este sistema operativo (incluyendo Windows Vista, Windows 64-bit, Windows Mobile).

Su estructura es modular, flexible y escalable (Figura 2). La programación orientada a objetos, y su estructura basada en clases y librerías (.dll), permite fácilmente incorporar nuevas funciones y análisis, o mejorar los existentes. Las principales funciones del programa (edición gráfica, edición tabular, dibujar grafo, gestión de la interfaz, etc.) están encapsuladas dentro de él; mientras que los modelos de datos (importación/exportación de datos, meta-datos) y las librerías de análisis de grafos son externas a la estructura. Esto permite poder utilizar estas librerías por otros programas o en otros proyectos.

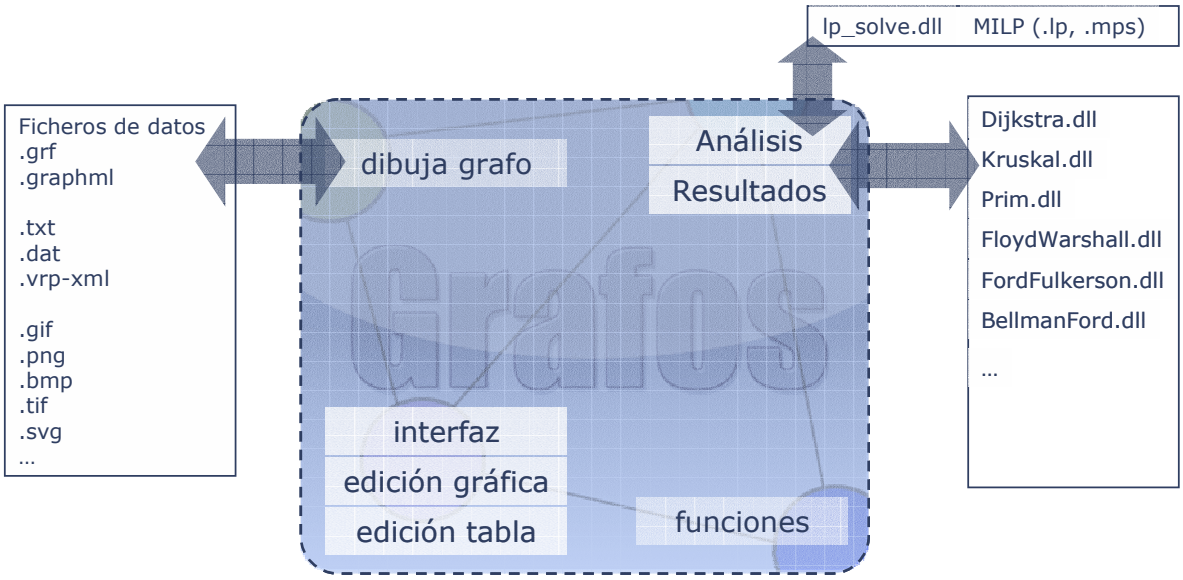

Figura 2. Esquema estructural y funcional.

4. Intercambio de datos

En la figura anterior se puede observar un esquema de los flujos de datos de *Grafos*. Los datos de un grafo se guardarán en dos posibles tipos de archivo: un fichero de texto de tipo secuencial (.grf) y un fichero de tipo estructurado *GRAPHML File Format* (.graphml). Este último tipo de fichero es un estándar que fue iniciado en el comité *"Graph Drawing Steering Committee"* previo al *"Graph Drawing 2000"* de Williamsburg. Se acordó formar un grupo que definiera un nuevo formato de datos para grafos basado en XML (Figura 3), de modo que fuera estándar para la comunidad de dibujo de grafos y sus cooperadores. La propuesta de estructura fue presentada en el siguiente simposio *"Graph Drawing Symposium"* de Viena (ver *http://graphml.graphdrawing.org/*). El principal predecesor de GraphML es GML, que fue el resultado de una iniciativa que comenzó en el *"Graph Drawing 1995"* en Passau y finalizó en el *"Graph Drawing 1996"* de Berkeley. GML es todavía uno de los principales formato de grafos soportado por muchos sistemas de dibujo, aunque no en este proyecto.

```
<?xml version="1.0"?>
<xs:schema targetNamespace="http://graphml.graphdrawing.org/xmlns"
        xmlns="http://graphml.graphdrawing.org/xmlns/1.0rc"
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        elementFormDefault="qualified"
        attributeFormDefault="unqualified"
...
```

Figura 3. Cabecera del fichero XML utilizado en GRAPHML.

Además de estos dos formatos anteriores, el programa es capaz de intercambiar datos con otras aplicaciones (hojas de cálculo, bases de datos, etc.) gracias a su función avanzada de

importación y exportación de datos. El usuario puede configurar fácilmente el formato y estructura de datos que quiere intercambiar (fichero separado por comas, tabulaciones, etc.), así como los datos a tratar (coordenadas de nodos, valor de coste del arco, demanda de los nodos, matriz binaria, etiquetas, etc.). De este modo se incrementan las posibilidades de análisis y sus aplicaciones reales.
Por otro lado, la aplicación permite generar diferentes ficheros de imagen con el grafo resultante: de mapa de bits (.gif, .png, .bmp, .tif), y también de tipo vectorial (.svg). *Scalable Vector Graphics* es un lenguaje de texto que describe imágenes vectoriales, formas, textos y otros gráficos incrustados. Los ficheros SVG son compactos y proporcionan una gran calidad gráfica en la web, en la impresión y en dispositivos de recursos limitados. Además, SVG soporta códigos y animación, lo que lo hace ideal para la interacción, el manejo de datos, y la personalización de los gráficos. SVG está libre de royalties y es un estándar abierto e independiente desarrollado bajo la supervisión de W3C (*World Wide Web Consortium*).
También es importante citar la utilización de ficheros de datos auxiliares que son útiles para la resolución de problemas de rutas. Actualmente se utilizan dos formatos de fichero para la definición de modelos de programación lineal entera mixta (MILP), se trata de estándares (.lp, .mps) que sirven de pasarela entre la aplicación y el solucionador de problemas de optimización (*Solver lp_solve*).
Por último, cabe destacar los ficheros utilizados para manejar los datos que definen problemas de rutas de vehículos VRP (*Vehicle Routing Problems*), Dantzig (1959). En este proyecto se ha propuesto un nuevo formato de fichero (.vrp-xml); diseñado a partir de una estructura de etiquetas VRP-XML que resuelve los problemas que aparecen en el intercambio de datos en el ámbito de la investigación operativa y la logística.

5. Interfaz y funciones de edición

En este proyecto ha sido muy importante el desarrollo de un interfaz de usuario intuitivo (menús y barras de herramientas sensibles al contexto) y que facilitara la construcción y edición de los grafos (rejilla, alineación, centrado, extensión, etc.).
Se trata de un entorno de trabajo visual (modo gráfico y tabular) donde lo que se observa es lo que se obtiene (exportar o imprimir la imagen del grafo). En este entorno, el usuario puede dibujar y editar libremente el grafo gracias a todo el conjunto de funciones y herramientas programadas.
Grafos permite la construcción tanto de grafos dirigidos como no dirigidos. Los arcos pueden tener valores asociados de coste o distancia, flujo mínimo y flujo máximo. Asimismo, los nodos además de tener una etiqueta identificativa pueden tener un valor asociado (peso del nodo, demanda o capacidad de producción). El usuario puede personalizar el grafo con estilos de arco, estilos de nodo, trazos y colores. Por supuesto la distribución del grafo la decide el usuario, aunque el programa le puede ayudar con funciones que dibujan el grafo automáticamente (formato de árbol, radial, orgánico *force directed*, flujo, aleatorio, etc.). En el futuro se espera mejorar e incorporar nuevos algoritmos de *Graph Drawing*, Kamada (1989).
También se pueden importar o exportar las coordenadas de los nodos (por ejemplo para representaciones geográficas GIS). Esta función puede complementarse con la posibilidad de incorporar un mapa como fondo del grafo. El programa puede además calcular la distancia entre nodos e introducir este valor automáticamente (o un coste proporcional) en los arcos del grafo (ver Figura 6).

6. Algoritmos y análisis

Se han implementado algoritmos y funciones que permiten el análisis de grafos (ver Figura 4) desde diferentes puntos de vista: caminos, árboles, flujos y rutas. A continuación se muestra un listado con los posibles algoritmos o análisis implementados actualmente:

Caminos
- Algoritmo de Dijkstra (camino mínimo, camino máximo)
- Algoritmo de Bellman-Ford (camino mínimo, camino máximo)
- Algoritmo de Floyd-Warshall (camino mínimo entre todos los pares de nodos)

Árboles
- Algoritmo de Dijkstra (árbol mínimo/máximo)
- Algoritmo de Kruskal (árbol de coste total mínimo/máximo)
- Algoritmo de Prim (árbol de coste total mínimo/máximo)

Flujos
- Algoritmo de Ford-Fulkerson (flujo máximo)
- Problema del Transbordo-Transporte (coste mínimo)
- Problema de Asignación (coste mínimo)

Rutas
- Problema del Viajante de Comercio (distancia total mínima)
- Problema de los m-Viajantes de Comercio (distancia total mínima)
- Algoritmo de Rutas (paso por nodos seleccionados a coste mínimo)
- Problema de los m-Rutas (distancia total mínima)
- Problema Rutas con Vehículos Capacitados (CVRP)

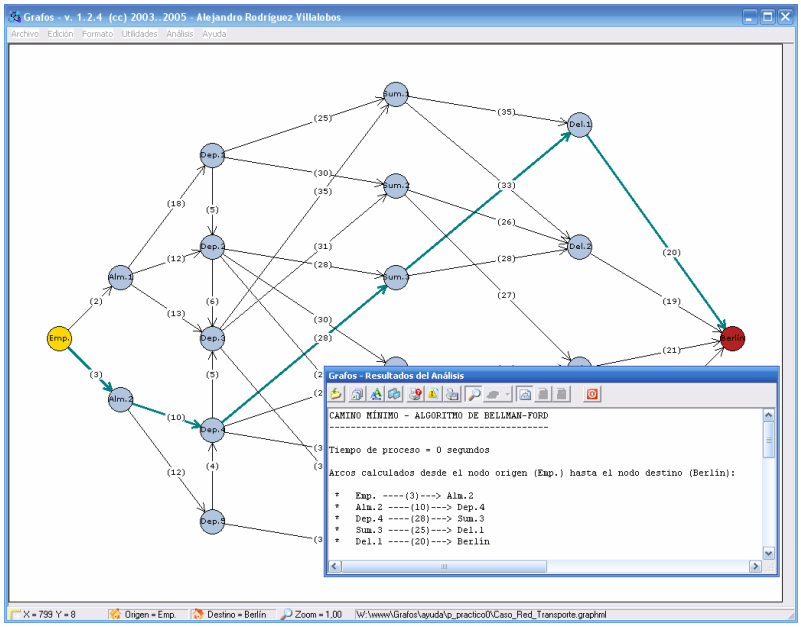

Figura 4. Pantalla de Grafos en un análisis de camino mínimo (Bellman-Ford)

Dentro de la denominación de problemas de rutas o recorridos realmente se engloba todo un amplio conjunto de variantes y personalizaciones de problemas. Desde aquellos más sencillos hasta algunos mucho más complejos que incluso hoy en día son materia de investigación. Todos ellos sin embargo, además del reto computacional que representan, tienen en común su gran importancia en investigación operativa por su aplicación práctica en la realidad. Al igual que el problema del viajante de comercio (*Traveling Salesman Problem - TSP*), la mayoría de los problemas VRP son de complejidad NP-completo, Dantzig (1954). Esto es así, porque el

número de posibles soluciones crece exponencialmente con el número de nodos del grafo (ciudades o puntos de paso), y rápidamente sobrepasa las capacidades de cálculo de los ordenadores más potentes**.** Como se puede observar en el listado anterior, para poder resolver esa variedad de análisis, el programa reúne y combina heurísticas con modelos de optimización. Para resolver éstos últimos, *Grafos* cuenta con la ayuda de *lp_solve*; se trata de un *solver* de programación linear entera mixta de licencia libre (*LGPL - GNU lesser general public license*). Este solucionador resuelve modelos de programación lineal (mixta) puros, con variables enteras/binarias, conjuntos semi-continuos y *special ordered sets* (SOS). No tiene límite en el tamaño de los modelos y acepta ficheros de entrada en formatos .lp y .mps. También se puede usar la librería del *solver* para ser llamada desde lenguajes de programación como: C, VB, .NET, Delphi, Excel, Java, etc. Está escrito en ANSI C y puede ser compilado para distintas plataformas como Linux y Windows. También se puede encontrar LUSOL, un sistema avanzado de factorización LU y resolución de ecuaciones integrado en *lp_solve v5* en el paquete bfp.
Hay que subrayar que una de las aportaciones más importantes de este proyecto ha sido el desarrollo de las rutinas de modelado, y del código fuente necesario para la resolución de los problemas de programación lineal entera mixta. Esto es, el programa es capaz de realizar de manera transparente al usuario y en pocos instantes, el modelo de programación lineal necesario para la resolución y el análisis del problema, que el usuario haya definido en el grafo y en su estructura de meta-datos.

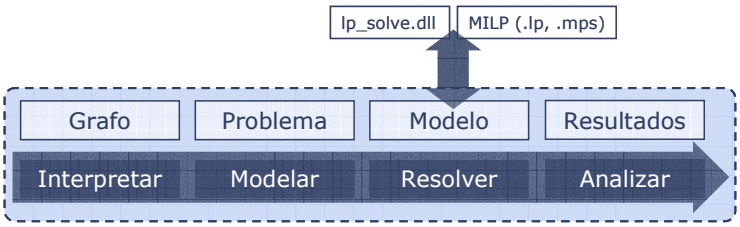

Figura 5. Proceso de interpretación, modelado, resolución y análisis.

En función del tipo de problema, de su contexto, y de la función objetivo a optimizar seleccionada por el usuario, el modelado generará según corresponda todo el conjunto de restricciones y la formulación de la función objetivo. Una vez creado, este modelo será enviado a la librería del *solver* para ser resuelto y analizado posteriormente (ver Figura 5). El modelo de programación lineal puede ser consultado por el usuario (ver Figura 6), o procesado en otro *solver* si se desea. Consultando el modelo, el usuario puede aprender y valorar la complejidad de este tipo de procesos de resolución.

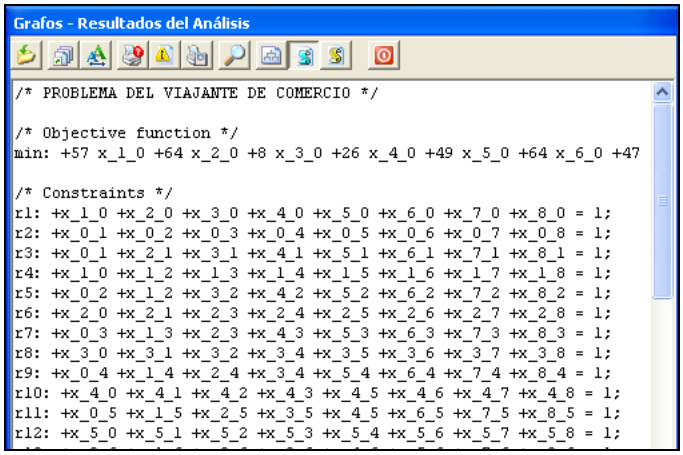

Figura 6. Pantalla de Grafos mostrando el modelo de programación lineal para TSP.

7. Futuro del proyecto

El proyecto sigue en desarrollo, y periódicamente se establecen nuevos objetivos que permitan la incorporación de nuevas funciones y análisis. Por ejemplo, a la fecha de redacción de este artículo, está prevista la programación del algoritmo de Busacker-Gowen (flujo máximo a mínimo coste). También se ampliarán los tipos de análisis para los problemas de rutas de vehículos, concretamente los de la familia CVRP (*Capacitated Vehicle Routing Problem*), Li (2004), teniendo en cuenta nuevas restricciones como: restricción máximo número de clientes, restricción máxima distancia y otras extensiones al problema. Otros análisis previstos son: DVRP (*Distance Vehicle Routing Problem*) y DCVRP (*Distance-Constrained VRP*).
En cuanto a la interfaz de usuario, también se está pensando en mejoras tales como: nuevos algoritmos de dibujado de grafos, mejoras en la edición y personalización del grafo, etc.

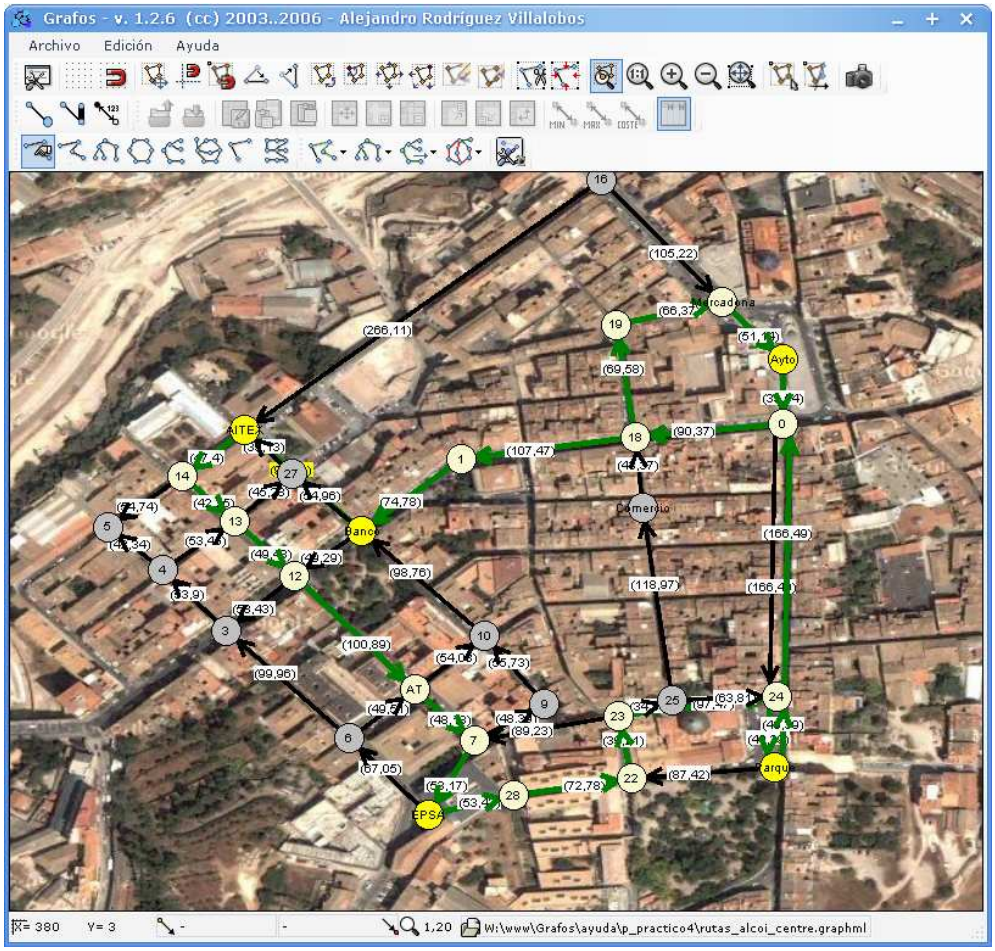

Figura 6. Pantalla de Grafos mostrando un grafo de ruta óptima sobre una imagen de satélite.

Uno de los objetivos más interesantes de este proyecto es el intercambio de conocimiento y experiencias que permitan el desarrollo de nuevos proyectos derivados. En la actualidad ya son varios los proyectos que se han derivado del conocimiento y código fuente compartido.
Una vez establecidas las bases de cálculo y análisis de un grafo, se pueden desarrollar rápidamente otras aplicaciones orientadas más hacia un uso profesional y de aplicación directa a un tipo concreto de problema (cálculo de rutas de preparación de pedidos *picking*, gestión de flota de vehículos, diseño de redes, control de flotas y enlace con sistemas de información geográfica GIS, aplicaciones de emparejamiento o asignación, planificación de

transporte urbano, planificación de recogida de residuos o de aprovisionamiento, problemas de reparto o distribución, sistemas de navegación GPS, planificación de movimientos de robots, vehículos autoguiados, etc.).
Cualquier persona está invitada a sumarse a este proyecto, y si lo desea compartir el desarrollo del mismo, su conocimiento y otras experiencias relacionadas.

Referencias

Aldous, J. et al.; Graphs and Applications:An Introductory Approach. 4ed. Spinger, 2004.
Dantzig G.B., Fulkerson D.R. and Johnson S.M.; Solution of a Large-scale Traveling Salesman Problem. Operations Research 2, 393-410. 1954.
Dantzig G. B. and Ramser R.H.; The Truck Dispatching Problem. Management Science 6, 80–91. 1959.
Gallardo López, B.; Procedimientos. Estrategias de aprendizaje. Su naturaleza, enseñanza y evaluación. Editorial Tirant lo Blanch. Valencia 2000.
Jungnickel, Dieter; Graphs, Networks and Algorithms. 2nd ed. Spinger-Verlag, Berlin 2004.
Kamada T. and Kawai S.; An algorithm for drawing general undirected graphs. Information Processing Letters, 31(1):7-- 15, 1989.
Li F., Golden B., Wasil E.. Very large-scale vehicle routing: new test problems, algorithms, and results. Computers & Operations Research, in press. 2004.