LABORATORIO DE ALGORITMOS PARA LA AUTOMATIZACIÓN DEL DISEÑO ELECTRÓNICO EN JAVA

G. MARRERO CALLICÓ, P. P. CARBALLO Y F. CABALLERO HERNÁNDEZ

Dpto. de Ingeniería Electrónica y Automática. Universidad de Las Palmas de Gran Canaria. Campus Universitario de Tafira. E-35017. Las Palmas de Gran Canaria. España. Email: gustavo@cma.ulpgc.es

En este artículo se presenta un sistema interactivo de simulación de los algoritmos básicos basados en grafos usados en las herramientas de diseño electrónico. El sistema interactivo se ha desarrollado en JAVA y pretende ayudar al estudiante al aprendizaje del diseño y la evaluación de dichos algoritmos para su uso en el campo de la automatización del diseño en microelectrónica.

1. Introducción

En la era de la microelectrónica, el proceso de generación de circuitos integrados se realiza en gran medida usando herramientas de Diseño Asistido por Computador (CAD) [1]. Muchos de los algoritmos usados en la Automatización del Diseño Electrónico (EDA) están basados en grafos por lo que se hace necesario establecer diferentes mecanismos que permitan explicar el funcionamiento de dichos algoritmos.

Con este trabajo se ha perseguido crear un entorno gráfico e interactivo que permita la ejecución de algoritmos EDA usando grafos [2]. Dicho laboratorio virtual permite añadir nuevos algoritmos de forma dinámica y es accesible desde cualquier ordenador o estación de trabajo con un navegador que soporte Java en su versión 1.2 [3]. También se establece una metodología para la implementación de los algoritmos, siguiendo la filosofía de programación orientada a objetos, que permite ir capturando y reutilizando el trabajo realizado con anterioridad. Por último, se ha diseñado una interfaz de usuario amigable, documentando extensivamente cada una de sus posibilidades.

2. Descripción de la aplicación

Las principales características de la herramienta desarrollada son:

1. La aplicación enseña la funcionalidad de un algoritmo mediante la visualización de los cambios generados en el grafo a cada paso del algoritmo. Las transformaciones pueden ser tan simples como cambiar el color de un nodo o lado para indicar que ha sido seleccionado. Adicionalmente, se explicará al alumno la evolución del algoritmo con un texto que varía con cada paso de ejecución.

2. El programa muestra una interfaz de usuario intuitiva y sencilla. Puesto que su uso es principalmente académico, debe atraer la atención del alumno.
3. La estructura de datos subyacente en el programa para la representación de los grafos contiene un variado conjunto de métodos para un acceso eficiente a todos los vértices y flancos o lados, con el fin de poder implementar algoritmos basados en lados o basados en vértices.
4. La ejecución del algoritmo se realiza paso a paso. Cuando el usuario altera o edita propiedades o datos relevantes del grafo mientras ejecuta un algoritmo, el programa detiene su ejecución con el fin de evitar indeterminaciones.
5. El usuario puede ejecutar pasos del algoritmo sobre diferentes grafos de forma alternativa con vistas a un análisis comparativo del procesamiento. Se pueden modificar las propiedades de los nodos y lados del grafo o reinicializar en cualquier momento.
6. Se proporciona una interfaz con otras herramientas de edición y tratamiento de grafos mediante el uso de ficheros ASCII que contiene la descripción del grafo y sus propiedades.
7. La aplicación es independiente de la plataforma y además es transportable a través de Internet. El alumno puede disponer del laboratorio a través de la red de redes, facilitando así su experimentación fuera de horas de clase.

El lenguaje de programación de alto nivel escogido para la implementación de los algoritmos EDA ha sido Java [4] por varias razones. Sus características de simplicidad, orientación a objetos, robustez, dinamismo, multihilo, interactividad, con soporte a aplicaciones Internet, neutral, le convierten en uno de los más apropiados. Java está orientado a objetos, y los paradigmas de esta metodología son adecuados para los programas que requieren interactividad con el operador. Para elaborar la interfaz gráfica de usuario se incluye un paquete denominado *Abstract Window Toolkit (AWT).* Asimismo, Java ofrece la oportunidad de desarrollar programas que pueden ejecutarse sobre múltiples plataformas, característica ideal para una red con diversidad de sistemas operativos y arquitecturas *hardware.* Los *applets* Java, insertados en una página WEB, pueden ser transferidos a través de Internet.

3. Entorno de usuario

En las Figuras 1 y 2 se muestra una descripción del entorno de usuario así como de las de las funciones más importantes a las que pueden accederse desde los menús de la aplicación.

4. Modelado de los grafos

Se ha escogido un modelo orientado a objetos para representar los grafos, con la intención de aprovechar las características intrínsecas del lenguaje. Existen diferentes representaciones para grafos (listas de adyacencias, lista de incidencias, matriz de adyacencias, matriz de incidencias, etc.) que poseen sus ventajas e inconvenientes, dependiendo del tipo de grafo que sea representado y los algoritmos a ejecutar sobre ellos. El tiempo de acceso a un elemento del grafo y el espacio requerido para almacenarlo son algunos de los factores más importantes a tener en cuenta cuando consideremos la representación a elegir. No obstante, nuestro objetivo es ejecutar e ilustrar algoritmos, por lo que los parámetros anteriores no han sido los determinantes a la hora de la elección. El factor decisivo ha sido la influencia que el método nos entrega para representar y manipular los grafos con claridad y sencillez.

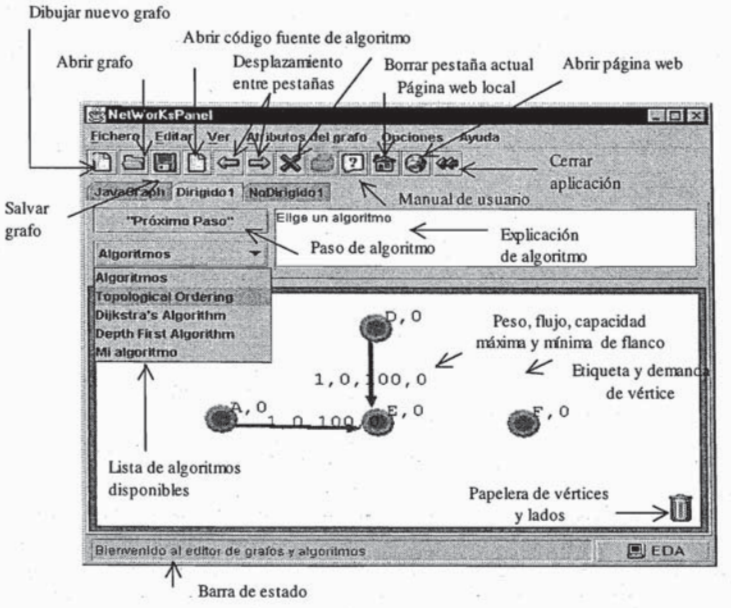


Figura 1. *Entorno de usuario para introducir los grafos.*

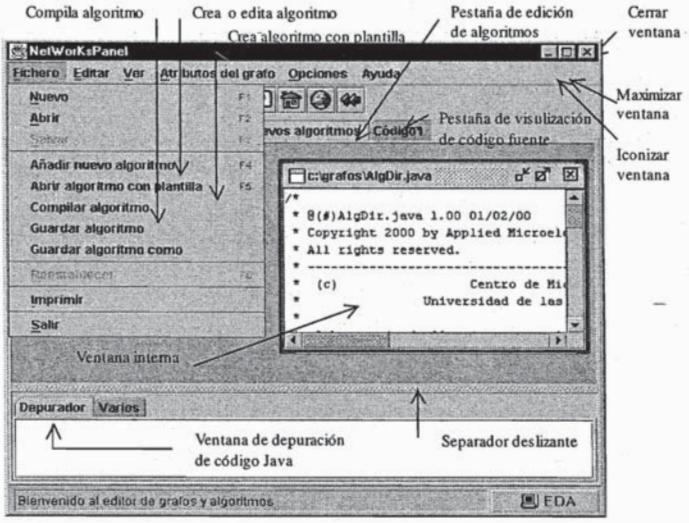
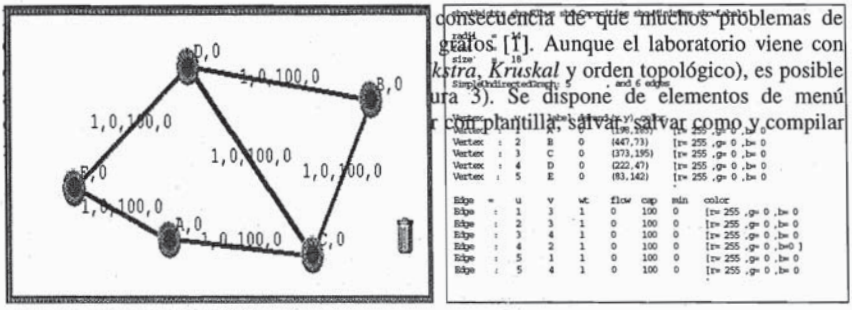


Figura 2. *Entorno de usuario para introducir los algoritmos.*

En nuestro caso, un grafo es una instancia de la clase *Graph*. La clase *Vertex* representa un vértice, y la clase *Edge* es la plantilla de un lado. El grafo debe almacenar información de la colección de nodos y flancos que contiene y, a su vez, cada flanco guarda como datos los vértices que conecta. Además se guarda información de los atributos del grafo. Dichos atributos se pueden mostrar o no según configuración del usuario. Por otra parte, es preciso emplear estructuras de datos dinámicas que permitan incrementar automáticamente su tamaño, pues la herramienta contiene las utilidades de adición y eliminación de nodos y lados.

5. Descripción de los algoritmos



consecuencia de que muchos problemas de grafos. [1]. Aunque el laboratorio viene con *ksira*, *Kruskal* y orden topológico), es posible ura 3). Se dispone de elementos de menú r con plantilla, salvar, salvar como y compilar

Figura 3 *Grafo no dirigido con visualización de atributos y su formato textual.*

6. Conclusiones

Se ha implementado una herramienta en JAVA que permite evaluar con facilidad el desarrollo de algoritmos basados en grafos. Esta herramienta es de utilidad para explicar los algoritmos básicos empleados en las herramientas software de diseño electrónico.

Referencias

[1] N. Sherwani. *Algorithms for VLSI Physical Design Automation*. Kluwer Academic Publishers, 1995. ISBN 0-79239-592-1.

[2] S. Even. *Graph Algorithms*. Computer Science Press, 1979. ISBN 0-914894-21-8.

[3] Java Development Kit 1.2. http://java.sun.com/products/jdk/1.2/index.html

[4] J. Gosling, B. Joy, G. Steele. *The Java Language Specification*. Addison Wesley, agosto 1996. ISBN 0-201-63451.