Tutorial interactivo para la enseñanza y el aprendizaje de los algoritmos de búsqueda en anchura y en profundidad

M. Gloria Sánchez Torrubia, Samuel Gutiérrez Revenga
Departamento de Matemática Aplicada. Facultad de Informática. Universidad Politécnica de Madrid

Resumen

Los grafos y los algoritmos que los manipulan juegan un papel muy importante en la formación, desde sus inicios, de un futuro Ingeniero en Informática.

En este trabajo se presenta una herramienta, desarrollada en Java, implementada como un applet y diseñada como instrumento auxiliar para la enseñanza en el aula y la práctica individual de los algoritmos de búsqueda en profundad y en anchura. Dicha herramienta está incluida dentro de un conjunto de programas creados, como complemento docente, para los estudiantes de primer curso de la asignatura de Matemática Discreta.

Sus características de visualización, sencillez e interactividad, hacen de este tutorial un instrumento de gran valor pedagógico, tanto para ser utilizado por el profesor en el aula, como por el alumno en su aprendizaje individual.

1. Introducción

En la actualidad, las nuevas tecnologías están produciendo un cambio importante en nuestra sociedad. La educación no puede mantenerse al margen de este desarrollo y los avances tecnológicos deberían generar un cambio sustancial en nuestros sistemas de enseñanza.

Por otra parte, la cantidad de estímulos que el alumno está recibiendo de su entorno hacen que, por contraste, la clase y el estudio tradicionales resulten menos motivadores.

En este contexto, se impone la necesidad de introducir nuevos alicientes en el proceso de enseñanza–aprendizaje y las nuevas tecnologías proporcionan el instrumento idóneo para generar el tipo de estímulo que nuestros estudiantes necesitan para involucrarse de lleno en dicho proceso.

Los algoritmos de grafos, estudiados en el programa de Matemática Discreta de la titulación de Ingeniería Informática, son, probablemente, los primeros algoritmos de cierta complejidad a los que el alumno va a enfrentarse en el curso de sus estudios. Esto los hace particularmente importantes, ya que la manera en que el estudiante los afronte va a influir, positiva o negativamente, en su formación posterior.

Los algoritmos de grafos son eminentemente visuales, por lo que, para su comprensión y aprendizaje, es de gran ayuda un entorno gráfico que complemente las explicaciones teóricas. Aún así, los visualizadores gráficos, independientemente de lo bien que estén presentados, tienen poco valor si no consiguen involucrar al estudiante en su propio proceso de aprendizaje. Por esta razón, es esencial que el alumno no permanezca como mero espectador del desarrollo del algoritmo, sino que participe activamente en él. Este es el objetivo principal que inspiró el diseño de nuestras herramientas en general, y de la presentada en este trabajo en particular, ya que es el usuario quien debe decidir, en cada paso, el siguiente nodo que va a ser visitado.

Gracias a su flexibilidad –admiten que el usuario introduzca su propio grafo–, a su interactividad –es el usuario quien va ejecutando los algoritmos mientras el programa comprueba su desarrollo– y a su presentación gráfica –la diferenciación de colores permite saber en qué paso del algoritmo nos encontramos en cada momento–, los tutoriales disponibles en nuestro sitio web, y en particular el dedicado a los algoritmos de búsqueda, resultan muy adecuados, tanto para complementar la exposición del profesor, como para facilitar el aprendizaje del alumno mediante la práctica de los algoritmos.

2. Los algoritmos de búsqueda en anchura y en profundidad

Estos algoritmos tienen su origen en el estudio de los laberintos. El primero en aparecer es el algoritmo de búsqueda en profundidad: Lucas (1882) [4] menciona que Tremaux, en su algoritmo, ya aplicaba esta idea y Tarry (1895)

[16] también la utilizaba en la resolución de laberintos, aunque la versión actual aparece mucho más tarde, en la publicación de Tarjan (1972) [15]. El algoritmo de búsqueda en anchura es utilizado por Moore (1959) [6] en la obtención del camino más corto en un laberinto.

La importancia de estos dos algoritmos radica en sus múltiples aplicaciones: búsqueda de un vértice de características específicas, estudio de conexión o cálculo del número de componentes conexas, obtención del camino más corto en grafos no ponderados, búsqueda de ciclos, búsqueda de puntos de articulación y comprobación de la propiedad de biconexión en grafos, ordenación topológica en grafos dirigidos acíclicos, obtención de caminos hamiltonianos, búsqueda en bases de datos o redes, etc. Otra cualidad, así mismo importante, es el hecho de que ofrecen un ejemplo, dentro de unos algoritmos relativamente sencillos, de dos estrategias de amplia utilización en informática: FIFO (First In First Out) o estructura de almacenamiento en *cola* y LIFO (Last In First Out) o estructura de almacenamiento en *pila*.

Debido a esta última característica, es muy importante, desde el punto de vista pedagógico, que el alumno de primer curso comprenda en profundidad estos algoritmos y se familiarice con el funcionamiento de estas dos estrategias, ya que tendrá que hacer repetido uso de ellas durante su formación y, muy probablemente, en su futuro profesional. También es importante que el alumno sea consciente de cómo esta diferencia de estrategia, que aparece como una pequeña modificación en el algoritmo, influye en el desarrollo de éste y se habitúe a estudiar a fondo todos los algoritmos con los que se encuentre, ya que un pequeño detalle puede tener consecuencias substanciales en el resultado final.

2.1. Descripción de los algoritmos

Dado un grafo, el algoritmo de búsqueda en anchura, recorre sus vértices comenzando por uno de ellos y visitando todos sus adyacentes. A continuación, para cada uno de ellos, se recorren todos sus adyacentes no visitados y así sucesivamente hasta completar todos los vértices del grafo.

El algoritmo de búsqueda en profundidad recorre los vértices del grafo comenzando por uno de ellos, buscando un vértice adyacente no visitado y repitiendo el proceso desde este último vértice. Cuando el vértice no tenga adyacentes, se retrocede al anterior vértice visitado y se repite el proceso hasta completar todos los vértices del grafo.

El orden que se seguirá en la elección del vértice siguiente, a falta de estrategias específicas para aplicaciones concretas, será, en general, aleatoria. En nuestro caso específico y teniendo en cuenta que el applet va dirigido a alumnos de primer curso, hemos decidido que sería de mayor valor pedagógico obligar al alumno a seguir el orden natural en el conjunto de vértices. De ahí que el programa exija al usuario que comience en el vértice 1 y estudie la adyacencia en el conjunto de vértices siempre de menor a mayor.

Sea G=(V,A) un grafo donde V={v_1, v_2, … , v_n} es el conjunto de vértices y A el conjunto de aristas. En el desarrollo de los algoritmos se utilizarán las siguientes estructuras: V', lista de vértices no visitados (se utilizará esta estructura para mantener el orden natural de los vértices), A', conjunto de aristas añadidas al árbol de búsqueda, P lista que almacena el orden de activación de los vértices (según se trate del algoritmo de búsqueda en anchura o en profundidad, esta lista tendrá estructura de *cola* o de *pila*, respectivamente).

/* *inicialización de datos* */
V'←[v_2, v_3, … , v_n]
A'←∅
P←[v_1]

Mientras P no esté vacía hacer
v_a←**primer** elemento de P /* *activación en cola* */
Si existe v_i de V' tal que v_i es adyacente a v_a, hacer
/* *se elegirá el primer vértice v_i de la lista V' que verifique la condición de adyacencia* */:
v←v_i /* *v_i es el nuevo vértice visitado* */
V'←V'-{v} /* *v se elimina del conjunto de vértices no visitados* */
A'←A'∪{v_av} /* *se añade la arista v_av al árbol de búsqueda* */
P←[vert(P),v] /* *se añade v al final de la lista P* */
En caso contrario P←P-[v_a] /* *se elimina el vértice activo v_a de la cola* */

Algoritmo 1. Búsqueda en anchura

/* *inicialización de datos* */
V'←[v_2, v_3, … , v_n]
A'←∅
P←[v_1]
Mientras P no esté vacía hacer
v_a←**último** elemento de P /* *activación en pila* */
Si existe v_i de V' tal que v_i es adyacente a v_a, hacer
/* *se elegirá el primer vértice v_i de la lista V' que verifique la condición de adyacencia* */:
v←v_i /* *v_i es el nuevo vértice visitado* */
V'←V'-{v} /* *v se elimina del conjunto de vértices no visitados* */
A'←A'∪{v_av} /* *se añade la arista v_av al árbol de búsqueda* */
P←[vert(P),v] /* *se añade v al final de la lista P* */
En caso contrario P←P-[v_a] /* *se elimina el vértice activo v_a de la pila* */

Algoritmo 2. Búsqueda en profundidad

Los algoritmos devolverán el árbol de búsqueda (V,A').

3. Tutorial de los algoritmos de búsqueda en anchura y profundidad

A los tutoriales para la enseñanza de grafos se accede desde el sitio web del D.M.A.[1] y los enlaces están situados en la página de applets de java relacionados con matemática discreta[2]

La página de entrada a los algoritmos de búsquedas[3] contiene una pequeña introducción al tema de grafos (lo necesario para la comprensión de los algoritmos), la descripción de dichos algoritmos, las instrucciones de uso de la herramienta y un enlace a la página que contiene el applet propiamente dicho[4].

El programa está diseñado para permitir al usuario dibujar sus propios grafos, dándole la posibilidad de deshacer tantas veces como sea necesario. Una vez introducido, es posible ejecutar ambos algoritmos sobre el mismo grafo, consiguiendo, de esta forma, destacar las diferencias entre los árboles resultantes al aplicar cada uno de ellos (figuras 1 y 2).

[1] http://www.dma.fi.upm.es
[2] http://www.dma.fi.upm.es/java/matematicadiscreta
[3] http://www.dma.fi.upm.es/java/matematicadiscreta/busqueda/
[4] http://www.dma.fi.upm.es/java/matematicadiscreta/busqueda/pag_applet.htm#applet

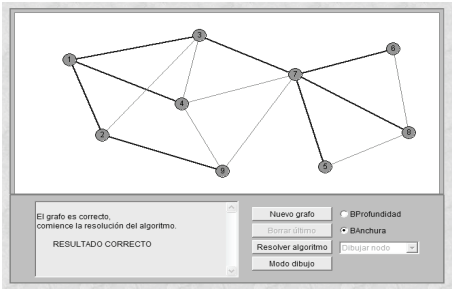

Figura 1. Búsqueda en anchura

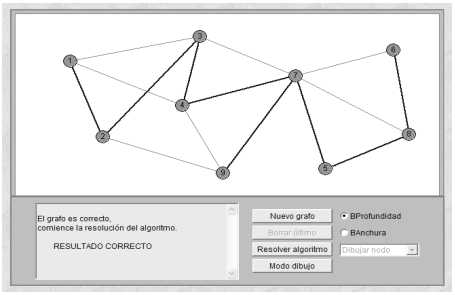

Figura 2. Búsqueda en profundidad

3.1. Introducción del grafo

Los nodos se crean haciendo clic con el botón izquierdo del ratón en la zona de dibujo de la ventana del applet y aparecerán numerados consecutivamente. Para introducir las aristas es necesario seleccionar esta opción en la lista desplegable, pulsar en el vértice origen y arrastrar hasta el vértice extremo.

En cualquier momento, incluso durante la fase de ejecución del algoritmo, es posible volver al modo de dibujo, introducir nuevos vértices y/o aristas seleccionando la opción correspondiente y ejecutar el algoritmo sobre el nuevo grafo.

La opción dibujar nodos, también permite mover los vértices ya dibujados pulsando y arrastrando el vértice que se desee mover.

3.2. Ejecución del algoritmo

Al pulsar el botón de ejecución del algoritmo, el programa estudia si el grafo es conexo (está diseñado para trabajar sólo con este tipo de grafos, ya que esa es la manera en que se enfoca el algoritmo en clase para disminuir su dificultad) y si lo es espera a que el usuario indique el primer

vértice. Según se haya elegido el algoritmo de búsqueda en profundidad o en anchura, se ejecutará uno u otro.

El programa espera, en cada paso, que el usuario pulse el siguiente vértice que será visitado y comprueba la corrección de la respuesta. Si ésta no es correcta aparece un mensaje de error y si lo es marca el vértice visitado en color verde, la arista utilizada en color azul y espera la respuesta correspondiente al paso siguiente indicando, en cada momento, el vértice activo v_a en color azul claro.

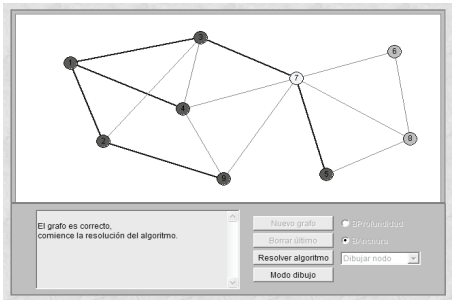

Figura 3. Ejecución del applet

Como hemos señalado anteriormente, el algoritmo, en su versión más general, es capaz de trabajar con grafos no conexos pudiendo, incluso, contar el número de componentes conexas de un grafo, además de permitir una ordenación aleatoria del conjunto de vértices (estas modificaciones del pseudocódigo se proponen como ejercicio avanzado para los alumnos). El applet, sin embargo, ha sido pensado para facilitar una primera aproximación a estos algoritmos. Por esa razón ha sido diseñado para no permitir su aplicación a grafos no conexos. Las razones pedagógicas por las que el programa obliga al alumno a seguir el orden natural de los vértices son fomentar en él el método y la disciplina y evitar el marcado de vértices sin un razonamiento previo, ya que, en grafos pequeños, la probabilidad de que un vértice no visitado sea elegible es relativamente alta.

No se debe olvidar, a la hora de utilizarlo, que este applet está especialmente diseñado para servir de herramienta a estudiantes de primer curso y que las búsquedas están entre los primeros algoritmos a los que el alumno se enfrenta, por lo que hemos considerado oportuno ofrecerlos, al menos en principio, en su forma más simplificada.

4. Valor educativo de los tutoriales interactivos

4.1. Como apoyo a la exposición del profesor

La visualización es un recurso ampliamente utilizado en la enseñanza de la informática. En los últimos años muchos profesores han mostrado interés en hacer uso de herramientas informáticas para mejorar sus exposiciones. Los algoritmos son estructuras dinámicas difíciles de mostrar en una pizarra o una serie de diapositivas. La utilización de visualizaciones dinámicas para estos procesos es muy interesante por razones tanto pedagógicas como de motivación. De este modo, las clases se vuelven más agradables para los alumnos, que también resultan beneficiados por la exposición gráfica.

La interactividad de esta herramienta facilita la tarea del profesor ya que el programa espera a la siguiente entrada del usuario, proporcionándole el tiempo que éste necesite para dar las explicaciones que considere oportunas. Esta opción también permite al profesor presentar, si lo considera pertinente, los errores más habituales del alumnado, de forma que éstos no se produzcan.

Según un estudio realizado por el grupo de trabajo “Improving the Educational Impact on Algorithm Visualization”[7] sobre un grupo de profesores que utilizaban este tipo de apoyo en sus clases, un 90% declaró que la experiencia de enseñar es más agradable, un 86% que aumentaba la participación de los estudiantes, un 83% tenía evidencias de que la clase resultaba más agradable para los alumnos, un 76% opinaba que la visualización proporciona una base para la discusión de los conceptos fundamentales de los algoritmos y un 52% tenía evidencias objetivas de una mejora en el aprendizaje. Algunos de los consultados observaron que los estudiantes dedican mucho más tiempo al aprendizaje cuando disponen de una visualización y tenían la sensación de que los alumnos captan los conceptos principales más rápido, con mayor facilidad y menor esfuerzo por parte del profesor. Por último, un 93% estaba de acuerdo en que el uso de la visualización puede ayudar a los

estudiantes en su aprendizaje y, en el 7% restante, se incluyen los que consideraban que no ayudaba y los NS/NC, pero ninguno opinó que dificultara el aprendizaje.

En nuestra experiencia en la utilización de este applet en el aula, la preparación y el desarrollo de la clase son más cómodos ya que resulta más fácil dibujar el grafo y visualizar el desarrollo de los algoritmos mediante este programa que en la pizarra o en una serie de diapositivas. Además, los alumnos muestran más interés, participan más activamente en la exposición, tanto adelantando la respuesta correcta como planteando los puntos que les resultan oscuros, con lo que la clase resulta más agradable y útil para todos.

Consultados algunos compañeros que han empleado esta herramienta en sus clases, la opinión generalizada ha sido que la herramienta resultaba fácil de utilizar, que tenía una buena presentación gráfica y que aumentaba la motivación y participación de los alumnos.

4.2. Como práctica para el alumno

Como hemos comentado, esta herramienta se puede utilizar para ilustrar gráficamente la presentación de los algoritmos de búsqueda en anchura y en profundidad, pero su mayor potencial reside en la posibilidad que tiene el alumno de practicar directamente con ella sin limitación de tiempo y lugar.

Además, es importante tener en cuenta el efecto que la interactividad tiene en el proceso de aprendizaje. En este sentido se sabe que ésta es un atributo probablemente esencial en cualquier técnica pedagógica que pretenda el éxito. El aprendizaje es, sin duda, mucho más efectivo cuando el estudiante puede controlar el intercambio de información.

Estudios realizados en la Universidad de Deakin (Australia) [14], mostraron que los alumnos que habían utilizado técnicas de aprendizaje interactivo tuvieron un 55% ganado sobre aquellos que recibieron las clases en un entorno tradicional de enseñanza. También se demostró que los alumnos aprendieron el material un 60% más rápido y la retención del conocimiento después de 30 días fue entre un 25 y un 50% superior.

En otro estudio, realizado en la universidad de Arcadia (Canadá), [18] se comprobó que, tanto los resultados como la satisfacción al terminar el curso, fueron claramente superiores en los estudiantes que manejaron herramientas interactivas.

Por otra parte, la experiencia como docentes nos demuestra que es necesaria la práctica de los algoritmos para conseguir su comprensión y aprendizaje. Esta práctica, realizada por el método tradicional de lápiz y papel, conlleva sesiones, a menudo tediosas, de estudio para el alumno, mientras que este otro método suele resultarle mucho más atractivo, lo cual influye muy positivamente en el proceso de aprendizaje. Esta idea se ha visto corroborada por las opiniones de los alumnos, recogidas de forma aleatoria, que afirman haber recurrido en muchos casos a los applets para el estudio de los algoritmos de grafos incluidos en nuestro sitio web y valoran muy positivamente la utilización de estas herramientas.

5. Características deseables en este tipo de herramientas y comparación con otros sistemas

Parece claro que el hecho de ver gráfica y dinámicamente el comportamiento de un algoritmo resulta más atractivo para el alumno que intentar reproducirlo sobre el papel, pero no está demostrado que mejore su comprensión. De hecho, algunos estudios parecen indicar lo contrario, es decir, que el estudiante tiende a adoptar una actitud pasiva que no favorece, sino más bien dificulta, el aprendizaje. Por el contrario, las herramientas interactivas como este tutorial, que exige al alumno la respuesta correcta para continuar, reúnen las mejores características de ambos sistemas, ya que obligan al usuario a reproducir el desarrollo del algoritmo a la vez que le ofrece un entorno más sugestivo. Como además corrigen sus errores y refuerzan sus aciertos, facilitan que el estudiante se involucre en el proceso de aprendizaje, aumentando el tiempo dedicado a esta tarea y mejorando su rendimiento.

Según algunos autores, la técnica más poderosa para lograr la comprensión y el aprendizaje de un algoritmo es su programación directa por parte del alumno. Pero, en el caso que nos ocupa, y dado que este tutorial va dirigido a estudiantes de primero, estos pueden encontrar dificultades para llevar a cabo esta tarea. Además, su aplicación está encuadrada como apoyo a la

asignatura de matemática discreta, cuyo objetivo, dentro del tema de grafos, es la comprensión y manejo de los conceptos y métodos relacionados con este tipo de estructuras, por lo que la programación queda fuera de los objetivos directos de esta asignatura. Una herramienta que permitiera la programación, sería, sin duda, interesante para un curso de grafos a nivel superior, por ejemplo en tercer o cuarto curso, donde el alumno medio está ya preparado para manejar las dificultades que conlleva cualquier lenguaje.

Entre las características deseables en una herramienta dirigida a alumnos de primer curso, se podrían señalar:

- Interactividad: es decir, que favorezcan, o incluso exijan, la participación del usuario, incluyendo indicaciones de error y sus causas o acierto.
- Flexibilidad: que permitan el uso de cualquier grafo (permitiendo al usuario introducirlo) así como la aplicación de varios algoritmos sobre el mismo grafo.
- Simplicidad de uso: de manera que el esfuerzo se dirija exclusivamente al aprendizaje del algoritmo, no de la herramienta.
- Cualidades gráficas, de forma que el proceso sea lo más claro e intuitivo posible.
- Presentación clara y sencilla, de forma que la atención quede centrada en los aspectos esenciales y no se disperse en otros de menor importancia.
- Visualización de la estructura del algoritmo y del paso que se está ejecutando en cada momento.
- Existencia de una librería integrada de ejercicios y posibilidad de almacenamiento de los ejemplos creados por el usuario.
- Disponibilidad para el alumno, de manera que éste pueda utilizarlo en cualquier momento y lugar, que sea independiente de la plataforma y que no sea necesaria la instalación de software específico.
- Peso de descarga reducido, para evitar problemas a aquellos que no disponen de conexión rápida.

La gran cantidad de herramientas de este tipo disponibles para su uso o descarga en páginas web hacen imposible un estudio exhaustivo de todos los recursos desarrollados. Entre los investigados, no hemos encontrado ninguno con las características de interactividad del applet que presentamos.

Gran parte de las herramientas diseñadas para visualizar algoritmos de grafos no disponen de los algoritmos de búsqueda en anchura y en profundidad como es el caso de MaGraDa [1], [20]. Entre las que sí los tienen, muchas de ellas se limitan a presentar una animación continua con grafos no modificables [19], [21]. El sistema JHAVÉ [9] [22] permite al usuario introducir su propio grafo y muestra el proceso paso a paso, intercalando preguntas –que el usuario puede responder o no– encaminadas a clarificar puntos y mejorar la comprensión del algoritmo. Esta herramienta es, entre las estudiadas, la que más se acerca a la interactividad disponible en la nuestra, aunque la de JHAVÉ es claramente menor.

Entre las desventajas de nuestra herramienta, destaca que su aplicación se limita exclusivamente a las búsquedas en anchura y en profundad (compensada, en parte, por el hecho de que en la misma página se pueden encontrar applets para el estudio de otros algoritmos de grafos), así como la imposibilidad de guardar los grafos construidos debido a las restricciones de seguridad inherentes a un applet de Java. Otro defecto radica en la falta de visualización de las estructuras que se van modificando en el algoritmo, especialmente el conjunto de vértices no visitados V' y el vector de orden de activación de vértices P (*pila* o *cola* según se trate del algoritmo de búsqueda en profundidad o en anchura). Para compensar este defecto se recomienda al alumno que vaya reproduciendo en un papel el estado, en cada paso, de estas dos estructuras.

Entre las ventajas del tutorial que presentamos cabe destacar las siguientes:

- No necesita ningún tipo de instalación salvo conexión a Internet.
- Permite la introducción del grafo por parte del usuario.
- El proceso es completamente interactivo, exigiendo al usuario que introduzca cada paso y evitando así una actitud pasiva por parte del alumno.
- Su utilización es extremadamente simple e intuitiva: bastan menos de dos minutos para familiarizarse con la herramienta.

- La visualización, utilizando diferentes colores para el vértice activo, los vértices visitados, los no visitados y las aristas agregadas al árbol de búsqueda, proporciona una idea gráfica e intuitiva del desarrollo del algoritmo.
- El proceso se puede repetir tantas veces como se desee, parando el algoritmo en cualquier momento y volviendo a comenzar desde el principio.
- Los dos algoritmos se pueden aplicar al mismo grafo, permitiendo la comparación entre los árboles obtenidos.

Estas características, que también son extensibles a otros applets de la página, lo hacen muy apropiado, tanto para la presentación en el aula, como para la práctica, por parte de los alumnos, de estos algoritmos.

6. Conclusiones y trabajo futuro

Además de la herramienta aquí presentada, se pueden encontrar en el sitio web del Departamento, tutoriales dedicados a la práctica a este nivel de otros algoritmos de grafos como el de Fleury y el de Dijkstra, y un interfaz que presenta los conceptos básicos de grafos y la matriz de un grafo. Está casi completamente terminado otro applet que implementará los algoritmos de Prim y Kruskal, y recién comenzada una nueva versión del de Dijkstra, con más prestaciones, entre otras la existencia de una librería de ejemplos integrada mantenida por los profesores, la capacidad de guardar los ejercicios realizados (mediante la utilización de Java WebStart o de un applet firmado) y más información del desarrollo del algoritmo (visualizando las modificaciones de los diferentes conjuntos, listas o tablas auxiliares de distancias provisionales que se van produciendo) o/y visualizando el paso que se ejecuta en cada momento. Este trabajo está siendo llevado a cabo por alumnos que realizan así su proyecto fin de carrera, dirigidos y supervisados por profesores que imparten la asignatura.

Como líneas futuras de trabajo nos proponemos completar los algoritmos que se estudian durante el curso y rehacer, mejorándolos, los que van quedándose obsoletos debido a los avances tecnológicos, aumentando sus capacidades e interactividad y tratando de integrarlos todos en un único sistema que permita su utilización para el tema completo de grafos.

En un campo como la informática, donde los cambios tecnológicos se producen a gran velocidad, la formación de los estudiantes debe estar dirigida sobre todo a aprender a aprender. Debe tenderse, por tanto, a utilizar metodologías que favorezcan el autoaprendizaje. En este sentido, es necesario recurrir a los soportes tecnológicos a nuestro alcance para crear materiales pedagógicos que favorezcan la interactividad.

El uso de estas variedades didácticas es un recurso más en la búsqueda de la calidad de enseñanza. Estos medios, utilizados como herramientas de b-learning, no sustituyen la figura del profesor, que sigue siendo el elemento más significativo en el proceso de aprendizaje, pero sí le complementan aumentando su capacidad y poniendo a su disposición otras posibilidades que éste no debería desestimar.

Referencias

[1] Caballero, M., Migallón, V. y Penadés, J.: *MaGraDa: Una herramienta para el tratamiento de grafos en Matemática Discreta*. VII JENUI (2001), pp. 478-481.

[2] Jungnickel, D.: *Graphs, Networks and Algorithms*. Springer 1999.

[3] Kocay, W., Kreher, D. L.: *Graphs Algorithms and Optimization*. Chapman & Hall/CRC 2005.

[4] Lucas, E.: *Recreations Mathematiques*. Paris 1882.

[5] Milkova, E: *Object Teaching of Graph Algorithms*. Proceedings of the 2nd International Conference on the Teaching of Mathematics. Creta 2002.

[6] Moore, E. F.: *The shortest path through a maze*. Proc. Int. Symp. on Theory of Switching Part II (1959). Cambridge, Mass., Harward Univ. Press, pp. 285-292.

[7] Naps, T. L., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., Korhonen, A., Malmi, L., McNally, M., Rodger, S., Velazquez-Iturbide, J. A.: *Exploring the Role of Visualization and Engagement in Computer Science Education*. inroads - Paving the Way Towards

Excellence in Computing Education. pp. 131-152, ACM Press, 2003.

[8] Naps, T. L., Lucas, J., Rößling, G.: *VisualGraph – A Graph Class Designed for Both Undergraduate Students and Educators*. Proc. of the 34th ACM SIGCSE Technical Symposium on Computer Science Education (2003) pp. 176-171.

[9] Naps, T. L., Eagan, J. R., Norton, L. L.: *JHAVÉ An Enviroment to Actively Engage Students in Web-based Algorithm Visualizations*. Proc. of the 31th ACM SIGCSE Technical Session (2000).

[10] Rosen, K. H.: *Matemática Discreta y sus aplicaciones*. McGraw Hill 2003.

[11] Rößling, G., Freisleben, B.: *Experiences in Using Animations in Introductory Computer Science Lectures*. Proc. of the 31th ACM SIGCSE Technical Symposium on Computer Science Education (2000) pp 134-138.

[12] Sánchez Torrubia M. Gloria, Giménez Martínez, Víctor. *Java Tutorials: a good tool for teaching and learning Graph Algorithms*. 2006. 3rd International Conference in the Teaching of Mathematics (comunicación aceptada).

[13] Sánchez Torrubia M. Gloria, Lozano-Terrazas Víctor M. *Algoritmo de Dijkstra: Un tutorial interactivo*. 2001. Proc. VII JENUI. pp 254-258.

[14] Street, S., Goodman, A.: *Some experimental Evidence on the Educational Value of Interactive Java Applets in Web-based Tutorials*. 3rd Australasian Conference on Computer Science Education. Association for Computing Machinery (1998) pp. 94-100.

[15] Tarjan, R. E.: *Depth first search and linear graph algorithms*. SIAM J Comp. 1, pp 146-160 (1972).

[16] Tarry, G.: *Le problème des labyrinths*. Nouv. Ann. de Math. 14 (1895), 147.

[17] Waldock, J., Gretton, H., Challis, N.: *Using the web to enhance student learning*. Proc of the 2nd International Conference on the Teaching of Mathematics (2002).

[18] Retson, D., Williams, P. J., Simmons, S.: *The Effectiveness of Computer-Based Studio Teaching of Physics*. http://aitt.acadiau.ca/research/science/PhysicsEdStudy5.PDF.

[19] *Depth first search, Breadth first search* http://javafaq.nu/java/free-online-books/artificial_inteligence_book.shtml.

[20] M. Caballero, V. Migallón y J. Penadés: *Prácticas con MaGraDa* http://www.dccia.ua.es/~jpenades/MaGraDa/MaGraDa.html.

[21] *Example of Graph Algorithm's Animation: Depth First Search* http://sziami.cs.bme.hu/~gsala/alg_anims/3/graph-e.html.

[22] *JHAVÉ Java-Hosted Algorithm Visualization Environment* http://jhave.org/.