View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by UPCommons. Portal del coneixement obert de la UPC

XV JENUI. Barcelona, 8-10 de julio de 2009
ISBN: 978-84-692-2758-9
http://jenui2009.fib.upc.edu/

Aplicaciones Web para prácticas de esquemas algorítmicos

Francisco Bermúdez, Murilo Boratto, Domingo Giménez, José Víctor Jiménez, Andrés Palazón
Departamento de Informática y Sistemas
Universidad de Murcia
Facultad de Informática, Campus de Espinardo
30071 Murcia
{fjavier,domingo}@um.es, {jvictorjimenez,muriloboratto,andrespalazon,}@gmail.com

Resumen

Debido a las dificuldades del aprendizaje inicial de las asignaturas de algoritmos en las carreras de Informática, se trabaja en el desarrollo de una aplicación Web que guíe y mejore el aprendizaje y la resolución de problemas mediante esquemas algorítmicos. La idea es dar soporte a un ambiente de desarrollo común para alumnos y profesores, en el que se seguirá una estructura definida previamente en la utilización de los esquemas algorítmicos. El alumno deberá utilizar en el desarrollo de las prácticas los esquemas proporcionados, completando los huecos correspondientes a las funciones que aparecen en ellos, con lo que se pretende corregir la tendencia en una aproximación inicial a los algoritmos de alejarse de los esquemas, lo que produce un gasto de tiempo importante hasta que se comprende la importancia de los esquemas algorítmicos básicos.

1. Introducción

Tradicionalmente las prácticas de los primeros cursos de enseñanza de los algoritmos se han realizado confiando en el cumplimiento por parte de los alumnos de los esquemas algorítmicos con que se trabaja. Ésto, en la práctica, se encamina a que cada alumno realice la implementación de los problemas reestructurando el esquema, lo que conlleva una carencia en la enseñanza de los mismos en el sentido de la pérdida de tiempo que acarrea por parte del docente la corrección de ejercicios cuya solución se desvía con mucha frecuencia de los esquemas con que se está trabajando, y la pérdida de tiempo y de conceptos por parte del alumno, que aporta muchas veces soluciones que difícilmente se ajustan al esquema y que son corregidas y redirigidas por el profesor a posteriori. Además, el resultado no siempre es el deseado, ya que el cumplimiento de las tareas propuestas no siempre es exhaustivo.

Debido a las dificultades que entraña el aprendizaje inicial de las asignaturas de algoritmos en las carreras de Informática se ha pensado en una aplicación que guíe y mejore el aprendizaje y la resolución de problemas mediante el uso de esquemas algorítmicos. Ésto dará soporte a un lugar de desarrollo común, en el que se seguirá una estructura definida previamente en la utilización de los algoritmos correspondientes y en la estructura propia de la aplicación desarrollada que contiene la solución de implementación al problema.

La herramienta deberá contener soporte a la enseñanza de esquemas algorítmicos secuenciales (divide y vencerás, programación dinámica, *backtracking*, ramificación y poda...), pero debido al auge de los sistemas paralelos también se ha pensado incluir esquemas algorítmicos paralelos, ya sean esquemas puramente paralelos (*pipeline*, maestro-esclavo, bolsa de tareas...) como esquemas secuenciales paralelizados.

El esfuerzo en la búsqueda de nuevas formas de enseñar algoritmos se está llevando a cabo en la mayoría de los casos orientado a la animación de algoritmos [3, 11, 13]. Sin embargo, estas herramientas suelen orientarse a un tipo concreto de algoritmos (por ejemplo: algunas permiten la visualización de esquemas de ordenación únicamente), enfocando su radio de ac-

tuación a esquemas básicos de programación, no estando los esquemas más complejos recogidos. Asimismo, la animación permite la visualización únicamente de estructuras básicas, como arrays y en algún caso matrices, árboles o listas. Hay otras herramientas que facilitan el trabajo con algoritmos de una forma más general. Por ejemplo, DFD [7] es un editor e intérprete de algoritmos representados en diagramas de flujo, permite crear diagramas de flujo y a partir de ellos ejecutarlos o realizar otras funciones como ejecutar paso a paso, ejecutar hasta un punto de ruptura, etc. También Mooshak [16] ayuda a la realización de prácticas de programación, permitiendo evaluar de forma automática si el código fuente es una solución correcta para un problema dado. No hemos encontrado ningún desarrollo orientado a la creación de un lugar común de trabajo en el entorno de la programación. Las herramientas desarrolladas en ese sentido son muy específicas, muy ligadas a la asignatura relacionada, ya que tienden a crear redes sociales en torno a una materia muy concreta con necesidades diferentes.

El auge que las aplicaciones web están teniendo en los últimos años, así como las ventajas que proporcionan este tipo de aplicaciones frente a los desarrollos de escritorio ha impulsado la creación de la aplicación que aquí se presenta para su utilización en un entorno web. Esto se enmarca en la tendencia actual de Web 2.0, transición que se ha dado de aplicaciones tradicionales hacia aplicaciones que funcionan a través del web enfocadas al usuario final [2]. No obstante, también interesa que se pueda tener una versión para utilizar desconectado de la red global, para usuarios que no puedan disponer de conexión permanente al servidor web que aloja la herramienta.

Así, el artículo presenta la herramienta para la programación de esquemas algorítmicos con C++ WEA (Web de Esquemas Algorítmicos) [12], como realización de las ideas propuestas, y se muestra también la propuesta de extensión de la herramienta para incluir esquemas paralelos. El objetivo principal ha sido la creación de una aplicación en un entorno web para que los alumnos de las asignaturas de iniciación a los algoritmos en las carreras de informática dispongan de herramienta de ayuda didáctica. Se trata de una herramienta desarrollada con la tecnología Java (J2EE). Permite la edición básica, compilación y análisis de una serie de algoritmos programados en C++. Se ha utilizado tecnología Java por la facilidad que ofrece para desarrollar apliaciones web, y los esquemas se han desarrollado en C++ por ser este el lenguaje que se utiliza en la asignatura de Algoritmos y Estructuras de Datos de la Universidad de Murcia [10], pero la herramienta es fácilmente modificable para desarrollar los esquemas en otros lenguajes. Los esquemas algorítmicos dados no son modificables por el alumno y están ya implementados. En este sentido, la etapa de diseño de la aplicación ya ha sido resuelta gracias a la herramienta desarrollada. El alumno únicamente se tiene que centrar en rellenar las funciones propias para su problema. Obligatoriamente siempre se cumple la correcta y exhaustiva utilización del esquema algorítmico, con lo que se pretende aliviar el problema mencionado de la tendencia de los alumnos de alejarse del esquema algorítmico. Adicionalmente, en el caso paralelo el alumno podrá practicar con esquemas paralelos, pero también con esqueletos paralelos que le permitirán obtener programas paralelos a partir del esqueleto y programando trozos de código secuencial. El profesor es el encargado de guiar el aprendizaje de los alumnos, para lo que podría modificar los esquemas o incluir nuevos ejemplos. Para esto, en el estado actual de la herramienta, es necesario que tenga acceso al código y lo pueda modificar.

La herramienta se basa en la premisa de que los usuarios tendrán un nivel de comprensión de la algorítmica fundamental, y que por tanto han seguido un primer curso de iniciación a la programación. En el caso paralelo se supone que tienen conceptos de arquitecturas paralelas y conocen algo de programación concurrente. Los desarrollos podrán ir desde ejemplos básicos hasta el nivel máximo de expresión que ofrece C++. Los esquemas algorítmicos básicos se desarrollan por medio de clases que el alumno debe completar con los métodos apro-

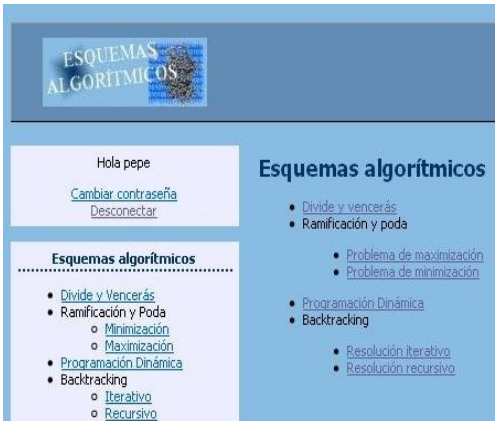

Figura 1: Imagen de la página principal

piados para cada problema. Adicionalmente, la herramienta incluye ejemplos guiados y tutoriales de utilización.

2. Descripción de la aplicación

Esta sección explica el funcionamiento de WEA, comentando los aspectos más destacables divididos en las áreas y funciones más importantes de la aplicación. La aplicación, inicialmente, incluye cuatro esquemas secuenciales implementados (figura 1): divide y vencerás, backtracking, ramificación y poda y programación dinámica; pero ha sido diseñada de forma que sea fácil la inclusión de nuevos esquemas o la modificación y mejora de los que inicialmente contiene. En particular se ha planificado la inclusión de esquemas de avance rápido y de algoritmos probabilistas, y en esquemas paralelos se está trabajando en la inclusión de un esquema maestro-esclavo para divide y vencerás que se había desarrollado como una herramienta separada [4], y se están desarrollando versiones paralelas de backtracking usando esquemas maestro-esclavo y bolsa de tareas [14].

2.1. Visión general

Los usuarios deben ingresar en la web con su cuenta, y a continuación elegir el algoritmo sobre el que se desee trabajar. Tras eso se pasa a la implementación del problema. El usuario tiene que programar los algoritmos en C++. Así, para cada uno de ellos se ha creado las estructura de clases (orientación a objetos) correspondiente. En cada una de las clases se dejan las cabeceras de las funciones necesarias a rellenar, y se da la opción de añadir nuevas funciones auxiliares para la resolución del problema. En cualquier caso, como mínimo el usuario debe implementar las funciones especificadas por la aplicación. Una vez rellenas podrá compilar su aplicación y comprobar si el resultado es el que esperaba.

La aplicación web se puede dividir en 3 áreas que agrupan varias páginas y funciones relacionadas:

- Edición de código.

  Zona para la introducción del código por parte del usuario y la creación de la estructura de aplicación necesaria para la compilación.

- Compilación y ejecución.

  Zona para la compilación y ejecución del código creado en la zona de edición.

- Análisis de algoritmos.

  Zona para el análisis de algoritmos, incluyendo visualización de gráficas y pruebas.

En todas las zonas se dispone de un menú principal para facilitar la navegabilidad y funcionalidad ofrecida a los usuarios. En este menú se dispondrá de un área de gestión de usuario, un árbol de navegación por los algoritmos y una serie de enlaces de interés.

2.2. Edicion de código

La edición puede realizarse de dos maneras, introduciendo el código a través de teclado o cargando un fichero con el código correspondiente a cada función de la clase correspondiente. La pantalla de edición es similar sea cual sea el esquema escogido. Las pantallas cambian según sea el fichero de la estructura elegido para editar en ese momento. La página principal de cada esquema muestra la plantilla del propio

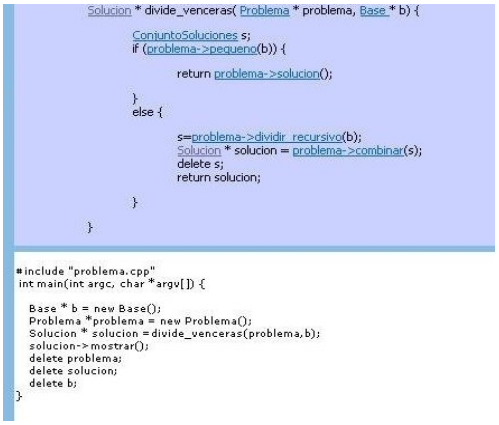

Figura 2: Zona de la página principal de un algoritmo, con el esquema divide y vencerás

Figura 3: Gráficas de resultados

esquema y una zona de edición donde se puede introducir el código correspondiente al `main` (la figura 2 muestra la zona de la pantalla donde aparece un esquema divide y vencerás y el programa para realizar una implementación).

El esquema dispone de enlaces a las funciones que deben ser implementadas de las clases auxiliares. Estos enlaces llevan a otras páginas que contienen áreas de edición de código para cada clase auxiliar. En éstas páginas se debe completar la declaración de la clase correspondiente, introducir la implementación de los métodos obligatorios y las posibles funciones auxiliares, métodos auxiliares y sobrecarga de operadores, en caso de que los haya.

El menú de edición contiene botones para:

- Cargar el código introducido en la web en el fichero correspondiente. Genera el fichero de la estructura de compilación correspondiente a la página en la que se encuentra.
- Volver a la página principal del algoritmo, en caso de que no se encuentre en ella el usuario.
- Cargar en la página web el código proveniente de un fichero. En este caso, el archivo debe estar formateado correctamente, tal como se indica en la web.

Además, el menú de edición de la página principal del algoritmo incluirá botones para compilar el problema implementado y para cargar el fichero de entrada por defecto para la ejecución del algoritmo.

2.3. Compilación y ejecución

Para la correcta compilación del algoritmo, se debe cargar cada uno de los ficheros necesarios de la estructura de la aplicación a crear. La página de compilación muestra los errores generados por el compilador para la implementación del problema. Si no hay errores indica que la compilación es correcta. A partir de una compilación correcta se puede ejecutar el resultado, o realizar pruebas sobre los análisis. La ejecución siempre se realiza redireccionando el fichero de entrada por defecto a la entrada estándar. La entrada a los programas de usuario se hace siempre por entrada estándar. Se mostrará el tiempo de ejecución y los valores que el usuario desee que aparezcan por la salida estándar.

2.4. Análisis de algoritmos

A partir de una compilación correcta, se puede hacer análisis de algoritmos (un ejemplo se muestra en la figura 3) sobre el soporte de la plataforma web gracias a varias funciones:

- Pruebas sobre funciones.

  Funcionalidad y páginas relacionadas donde se pueden realizar pruebas sobre funciones interesantes a modificar para el estudio del algoritmo. Para cada prueba se modifica la función correspondiente del esquema escogido. La entrada estándar para la ejecución será el fichero de entrada por defecto. Se muestran los resultados en una tabla y, opcionalmente si la salida estándar para cada ejecución es un número únicamente, una gráfica de barras con los valores correspondientes. También muestra una tabla con los tiempos de ejecución para cada prueba y una gráfica de barras con los valores de tiempo de ejecución frente a número de prueba.

- Pruebas sobre entradas.

  Se realiza una prueba por cada fichero de entrada cargado. Esta funcionalidad recoge a la anterior, pero aquella se ha incluido porque hace más sencillo el análisis para usuarios noveles. Las pruebas se numeran con un índice. La ejecución muestra lo mismo que la prueba anterior.

El alumno puede realizar experimentos con funciones distintas y entradas distintas, pero debe ser el profesor el que guíe los experimentos que quiere que realice el alumno dependiendo de los objetivos docentes que persiga. Por ejemplo, en una ordenación por mezcla se puede pedir que obtenga el tamaño del caso base y el método directo de ordenación con que se tiene el menor tiempo de ejecución variando el tamaño del problema, o en un método de ramificación y poda se puede pedir que evalúe el tiempo de ejecución y el número de nodos generados con distintas funciones de obtención de las cotas y de la estimación del beneficio en cada nodo.

3. Implementación

Esta sección recoge alguno de los aspectos técnicos más significativos de la aplicación.

La aplicación se basa en la tecnología Java (`J2EE`) con `JSP` [15]. Se ha implementado un sistema basado en el patrón de diseño modelo-vista-controlador.

La aplicación está dividida en tres partes claramente diferenciadas. Por un lado se tiene la gestión de usuarios, para tener un control de acceso a la aplicación. Por otro tenemos la parte de programación de los distintos algoritmos. Y por último la vistas, la interfaz gráfica de la aplicación. Entre las librerías Java más destacadas que se han utilizados están la de la generación de gráficos (`JfreeCharts`) y la de la subida de ficheros al servidor (`commonFileUpload`).

Para favorecer la creación de un paquete de escritorio para aquellos usuarios que deseen instalarse la aplicación en su ordenador, se ha realizado una gestión de usuarios mediante ficheros encriptados con `MD5`, eliminándose así toda dependencia de una base de datos.

3.1. Aspectos didácticos

El diseño de la herramienta se ha orientado desde el principio para su utilización en un ámbito didáctico, más concretamente para la realización de las prácticas de algorítmica en las carreras de informática. Por ello se apoya en ayudas y ejemplos que guían su uso desde el principio.

El objetivo principal es obligar a los usuarios a seguir la estructura de los esquemas algorítmicos sin perder el orden de un buen diseño de clases a la hora de generar una aplicación para resolver un problema. Además, el alumno tendrá la ventaja de no tener que adquirir conocimientos extras (con la pérdida de tiempo que eso conlleva) sobre la configuración de entornos de programación, compiladores, herramientas de análisis, etc.

3.2. Utilización de la aplicación

Como ya se ha comentado, la aplicación básicamente se trata de un editor y compilador de código C++. Como aplicación web, tiene una gestión de usuarios, de modo que para utilizarla hay que autenticarse. Una vez realizado este paso el usuario tiene la opción de escoger el esquema algorítmico que quiere utilizar para resolver su problema. Una vez seleccionado el

algoritmo a utilizar el usuario pasa a una pantalla donde se muestra el esquema y las partes de él que puede editar. Además tiene acceso tanto a la documentación relacionada como a ejemplos didácticos del esquema en cuestión.

Una vez generado el código, el usuario tiene que compilarlo pulsando el botón compilar. Ésto le lleva a una pantalla donde, o bien se le muestran los errores de compilación y se le da la opción de corregir, o bien se le permite ejecutar su algoritmo y analizar la ejecución. Las pruebas son de dos tipos: modificando funciones puntuales del esquema, y modificando entradas.

Hay que tener en cuenta que en todo momento el usuario puede subir un fichero de valores de entrada al algoritmo y que, si lo carga en el servidor, la aplicación automáticamente se lo pasará por la entrada estándar a los ejecutables en cada ejecución. Como es lógico, para el caso de realizar pruebas modificando entradas los ficheros de entrada utilizados serán los subidos en la pantalla de dichas pruebas, y se harán tantas pruebas como ficheros se hayan cargado en el servidor.

Para las pruebas modificando funciones, según el algoritmo escogido, la aplicación ofrece unas funciones u otras para modificar. El usuario tiene que rellenar las distintas implementaciones para una misma función y compilarlo todo. Automáticamente la aplicación ejecuta las aplicaciones generadas, tantas como implementaciones haya realizado.

4. Esquemas y Ejemplos

En esta sección se muestran los esquemas algorítmicos secuenciales y los ejemplos de uso incluidos en la aplicación, así como los esquemas paralelos en que se está trabajando para su inclusión.

4.1. Esquemas Secuenciales

Como ya hemos comentado, los esquemas incluidos son divide y vencerás, programación dinámica, backtracking y ramificación y poda [8, 5, 10]. Para cada uno de ellos se ha incluido un ejemplo básico que muestra el funcionamiento de la aplicación paso a paso, pero se incluye la posibilidad de que el profesor añada nuevos ejemplos. Los ejemplos incluidos inicialmente son: ordenación rápida (divide y vencerás), números de Fibonacci (programación dinámica), mochila 0/1 (backtracking y ramificación y poda) y un problema de asignación de tareas a procesadores con el que los alumnos habían trabajado en las prácticas de una asignatura [6] (ramificación y poda, minimización).

En la figura 2 se muestra un esquema divide y vencerás en el que aparecen los métodos y clases que el alumno debe completar para realizar una implementación de este esquema para el problema con el que esté trabajando. Aparecen las clases `Problema`, `Base`, `Solucion` y `ConjuntoSoluciones`, y los métodos `pequeno`, `solucion`, `dividir_recursivo` y `combinar` de la clase `Problema`. El alumno los ha implementado en el fichero `problema.cpp` que incluye dentro de su programa `main.cpp`. Más ejemplos pueden verse en [12].

La inclusión de nuevos esquemas secuenciales en la aplicación se puede hacer de forma sencilla, y se tiene planificado incluir avance rápido y alguna versión de algoritmos probabilistas y de esquemas metaheurísticos.

4.2. Inclusión de Esquemas Paralelos

La inclusión de esquemas paralelos es más compleja debido a la mayor dificultad de la programación paralela y a que es necesario un conocimiento de la tecnología de esqueletos paralelos [9].

Para cada esquema paralelo a incluir se realiza primero un análisis detallado y se desarrolla una aplicación específica para él, y una vez evaluada se podría incorporar a la herramienta secuencial, pero parece más apropiado desarrollar una herramienta con el mismo formato y funcionalidad pero diferenciada para esquemas paralelos.

Así, el trabajo desarrollado hasta ahora sobre esquemas paralelos ha consistido en el desarrollo y evaluación de una aplicación para un esquema maestro-esclavo, incluyendo como ejemplo un divide y vencerás para ordenación

por mezcla y ordenación rápida. Esta aplicación, una vez evaluada, se está transformando a la estructura de la herramienta secuencial. Además, como otro ejemplo de maestro-esclavo se ha estudiado una versión de backtracking para el problema de la mochila 0/1, y un esquema de bolsa de tareas para el que también se usa como ejemplo la mochila 0/1. Las versiones paralelas incluyen esquemas con programación en memoria compartida y con paso de mensajes. El usuario de la herramienta verá el esquema algorítmico paralelo independientemente de la implementación, pero podrá experimentar con las dos implementaciones siempre que el sistema sobre el que trabaje las soporte.

Para cada esquema paralelo debe crearse la estructura de clases correspondiente, igual que en el caso secuencial. La diferencia en este caso es que se ofrece al usuario esquemas paralelos con los que puede experimentar pero sólo necesita programar métodos secuenciales que se incluyen en el esquema.

Como ejemplo de esquema paralelo el algoritmo 1 muestra el maestro-esclavo para divide y vencerás [1]. Aparecen remarcadas las funciones que tiene que programar el usuario. **DivideVencerás** es un esquema secuencial, que se programaría siguiendo la filosofía de la herramienta WEA (hay que programar las funciones básicas del esquema). Las otras dos funciones (**dividir** y **combinar**) las programa también el usuario, pero serán distintas de las correspondientes secuenciales, pues el número de subproblemas no será dos, sino que coincidirá con el número de procesos con que el usuario quiera experimentar. Así, aunque sólo hay que programar funciones secuenciales, el usuario debe tener un cierto conocimiento del esquema paralelo que utiliza pues el formato de los subproblemas que genera y de las subsoluciones que se combinan debe coincidir con el usado por las funciones de distribución y recepción. Además del esquema básico con el paralelismo oculto se pueden preparar versiones que incluyan la posibilidad de programar parte de las funciones paralelas (distribuir y recibir), o habría que especificar el formato de los datos con los que se puede trabajar.

```
DVParaleloME (p: problema, n: tamaño,
          i: id de proceso): solucion
     if MAESTRO then
          dividir p en bloques de
               problemas mas pequeños
               p1, p2,..., pk
          distribuir bloques a los
               procesos ESCLAVOS
          DivideVencer (p1, n1)
          recibir resultados parciales
          combinar los resultados
               parciales
     else
          recibir del MAESTRO bloque
               en pi
          si = DivideVencer (pi, ni)
          enviar resultado al MAESTRO
     endif end DVParaleloME
```

Algoritmo 1: Divide y Vencerás Maestro-Esclavo.

5. Conclusiones y Trabajos Futuros

Este trabajo presenta una herramienta para la enseñanza de esquemas algorítmicos. La herramienta se ha desarrollado en el curso 2007-2008 dentro de un Proyecto Fin de Carrera de Ingeniería Informática. Se ha diseñado de una manera flexible, de forma que permita la fácil inclusión de nuevos esquemas y ejemplos. En particular se está aplicando la misma estructura a esquemas algorítmicos paralelos.

Se está considerando su utilización para el próximo curso de manera experimental en algunos grupos de prácticas de la asignatura de Algoritmos y Estructuras de Datos de segundo curso. Además, puede ser usada de manera autónoma, tanto conectándose a la página de la aplicación como descargándola de ella.

El abanico de posibilidades de desarrollo futuro que presenta esta aplicación es muy amplio. Se pueden incluir nuevos esquemas algorítmicos secuenciales o paralelos.

En su estado actual no está preparada para facilitar la gestión del curso permitiendo a los alumnos registrar sus trabajos para que los

revise el profesor o para que se evalúen directamente. El profesor, para incluir nuevos ejemplos y esquemas debe modificar directamente el código de la aplicación, por lo que otra mejora sería desarrollar un entorno de edición e inclusión de esquemas, ejemplos y prácticas guiadas. Una vez mejorados estos aspectos se debería integrar la aplicación en algún entorno de e-learning.

La herramienta podría adaptarse para ser usada en entornos científicos (no sólo didácticos) para resolver de forma cercana a la óptima problemas enviados por usuarios y que siguieran uno de los esquemas en la aplicación.

6. Agradecimientos

Este trabajo ha sido financiado en parte por la Fundación Séneca, proyecto 08763/PI/08, y por el Ministerio de Educación, proyecto TIN2008-06570-C04-02.

Referencias

[1] F. Almeida, D. Giménez, J. M. Mantas, and A. M. Vidal. *Introducción a la programación paralela*. Paraninfo Cengage Learning, 2008.

[2] F. Almeida A. Santos and V. Blanco. Lightweight web services for high performace computing. In *ECSA*, pages 225–236, 2007.

[3] K. Biliana and D. Thiébaut. Sorting Algorithms. `http://maven.smith.edu/thiebaut/`, 1997.

[4] M. Boratto, D. Giménez, and A. M. Vidal. Automatic parametrization on Divide-and-Conquer Algorithms. In *ICM2006*, pages 495–496, 2006.

[5] Brassard and Bratley. *Fundamentos de algoritmia*. Prentice-Hall, 1997.

[6] A. L. Calvo, A. Cortés, D. Giménez, and C. Pozuelo. Using metaheuristics in a parallel computing course. In *ICCS (2)*, pages 659–668, 2008.

[7] F. Cárdenas, N. Castillo, and E. Daza. Editor e Intérprete de Algoritmos Representados en Diagramas de Flujo. `http://mx.geocities.com/ikky_fenix/`. Universidad de Magdalena Santa Marta.

[8] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. The MIT Press, 1990.

[9] I. Dorta, C. León, C. Rodríguez, and A. Rojas. Parallel skeletons for Divide-and-Conquer and Branch-and-Bound techniques. In *PDP*, pages 292–298, 2003.

[10] D. Giménez, G. García, J. Cervera, and N. Marín. *Algoritmos y estructuras de datos. Volumen II: Algoritmos*. Texto guía, Univ. de Murcia, Diego Marín, 2003.

[11] J. Gosling, J. Harrison, and J. Boritz. Sorting Algorithms Demo. `http://www.cs.ubc.ca/~harrison/`, 2007.

[12] J. V. Jiménez and A. Palazón. WEB de Esquemas Algorítmicos. Proyecto fin de carrera, Departamento de Informática y Sistemas, Universidad de Murcia, `http://www.um.es/pcgum/PFCs_y_TMs/index.html`, 2008.

[13] R. Laza, D. González-Peña, J. R. Méndez, F. Fernández-Riverola, J. Baltasar García, and M. Reboiro. ALT: Algorithm Learning Tool. `http://trevinca.ei.uvigo.es/rlaza/`, 2007.

[14] M. Quesada. Técnicas de autooptimización en recorridos de árboles por medio de backtracking. Proyecto fin de carrera, Facultad de Informática, Universidad de Murcia, `http://www.um.es/pcgum/PFCs_y_TMs/index.html`, February 2009.

[15] Sun Microsystems, Inc. Developer Resources for Java Technology. `http://java.sun.com`.

[16] System for managing programming contests on the Web. `http://mooshak.dcc.fc.up.pt/~zp/mooshak/`.