Un asistente extensible para la experimentación interactiva con problemas combinatorios

J. Ángel Velázquez Iturbide, Ouafae Debdi, Daniel Gómez García, Jesús del Fresno Ramírez, Manuel Rubio Sánchez, Maximiliano Paredes Velasco
Departamento de Lenguajes y Sistemas Informáticos I
Universidad Rey Juan Carlos
28933 Móstoles, Madrid
{angel.velazquez, ouafae.debdi. manuel.rubio, maximiliano.paredes} @urjc.es

Resumen

Los algoritmos voraces tienen un problema desde el punto de vista docente: no son muy adecuados para su ejercitación por los alumnos. Para remediar esta situación, hemos diseñado un enfoque didáctico basado en experimentación y soportado por varios asistentes interactivos para problemas concretos (mochila, selección de actividades, árbol de recubrimiento de coste mínimo). A partir de nuestra experiencia, la comunicación analiza los problemas encontrados y presenta mejoras en varios aspectos. Primero, hemos ampliado ligeramente el método experimental. Segundo, hemos modificado los asistentes interactivos para soportar mejor el método experimental e incluso poder usarlo con otras técnicas de diseño (programación dinámica, algoritmos aproximados). Tercero, hemos integrado dos asistentes interactivos en uno solo, llamado GreedEx, que puede ampliarse para dar soporte a otros problemas combinatorios. Por último, hemos ampliado GreedEx con un tercer problema, el de la mochila 0/1. En resumen, estas mejoras han reducido las limitaciones del método experimental, han ampliado el dominio de aplicación de los asistentes, y facilitan su extensión a otros problemas y su mantenimiento.

1. Introducción

El trabajo aquí presentado es una contribución dentro de una línea de investigación sobre el desarrollo de aplicaciones educativas de animación de programas para técnicas de diseño de algoritmos. Dicha línea toma la taxonomía de Bloom [1] como marco para la especificación de aplicaciones educativas. Se trata de un marco, muy conocido, para medir el grado de conocimiento de una materia por el alumno. En orden creciente de dificultad, pueden alcanzarse los niveles de: conocimiento, comprensión, aplicación, análisis, síntesis y evaluación.

Hemos abordado el aprendizaje de los algoritmos voraces mediante un método experimental [8]. También hemos diseñado tres asistentes interactivos (es decir, aplicaciones educativas interactivas) para dar soporte a dicha experimentación [4][6][8]: AMO (para el problema de la mochila), SEDA (para el problema de la selección de actividades [2]) y TuMiST (para el problema del árbol de recubrimiento de coste mínimo). Los dos primeros son aplicaciones maduras, que se han sometido a numerosas evaluaciones de experto y a 4 evaluaciones de usabilidad [6]; TuMiST es un prototipo.

Conviene resaltar que apenas existen trabajos realizados para la mejora del aprendizaje de los algoritmos voraces. Aparte de numerosas animaciones dispersas de algoritmos voraces, disponibles en la web (p.ej. [9]), apenas podemos citar el sistema PathFinder [5]. Soporta un algoritmo concreto (el algoritmo de Dijkstra) pero, a diferencia de nuestros asistentes, se centra en la mera comprensión del algoritmo final.

La estructura de la comunicación es la siguiente. En la segunda sección se presentan los antecedentes de este trabajo. La sección tercera describe nuestra experiencia e identifica las principales limitaciones y problemas encontrados. En la sección cuarta describimos las mejoras realizadas, agrupadas en tres subsecciones. Por último, incluimos nuestras conclusiones y algunas líneas de trabajo futuro.

2. Antecedentes

Describimos nuestro trabajo anterior, presentando primero el método experimental y después los asistentes desarrollados para soportarlo.

2.1. Aprendizaje activo de algoritmos voraces

El método experimental que describimos pretende situar el aprendizaje de los algoritmos voraces en los niveles de comprensión, análisis y evaluación de la taxonomía de Bloom. Lo ilustramos con un ejemplo.

Sea el problema de la selección de actividades [2]. Dado un conjunto de n actividades, cada una con un instante de inicio c_i y un instante de fin f_i, se desea seleccionar un subconjunto de tamaño máximo de actividades que no se solapen.

Por ejemplo, dado el conjunto de actividades de la Tabla 1, el subconjunto formado por las actividades {5,0,8} es una solución válida, pero el subconjunto {0,3,4,5} es una solución de tamaño máximo.

i	0	1	2	3	4	5	6	7	8	9
s_i	24	13	8	1	12	0	0	14	4	24
f_i	28	14	27	14	15	7	13	30	16	30

Tabla 1. Un ejemplo del problema de selección de actividades

Podemos concebir distintas funciones de selección de las actividades que permitan resolver este problema mediante un algoritmo voraz:

- Orden creciente/decreciente de índice.
- Orden creciente/decreciente de inicio.
- Orden creciente/decreciente de fin.
- Orden creciente/decreciente de duración.

Si probamos a resolver el problema con estas funciones de selección, se obtienen los resultados de la Tabla 2.

Función de selección	Actividades seleccionadas	Nº actividades
Índice ↑	{0, 1, 5}	3
Índice ↓	{9, 8}	2
Inicio ↑	{5, 2}	2
Inicio ↓	{9, 1, 6}	3
Fin ↑	{5, 1, 0}	3
Fin ↓	{9, 8}	2
Durac. ↑	{1, 0, 5}	3
Durac. ↓	{2, 5}	2

Tabla 2. Resultado de aplicar distintas funciones de selección

Esta ejecución nos permite descartar las tres funciones de selección que no han producido un resultado óptimo, quedando otras cuatro funciones como candidatas. (Parece obvio que la primera función de selección no es una candidata seria, pero tenemos que descartarla experimentalmente.)

Si repetimos este proceso sobre nuevos datos de entrada, podemos ir descartando más funciones de selección:

Función de selección	1ª ejec.	2ª ejec.	3ª ejec.	4ª ejec.	5ª ejec.
Índice ↑	3	2	2	5	2
Índice ↓	2	–	–	–	–
Inicio ↑	2	–	–	–	–
Inicio ↓	3	2	2	5	3
Fin ↑	3	2	2	5	3
Fin ↓	2	–	–	–	–
Durac. ↑	3	2	2	5	2
Durac. ↓	2	–	–	–	–

Tabla 3. Resultados sobre varios datos de entrada

(Aunque en la comunicación expliquemos el método con independencia de su implementación mediante los asistentes interactivos, todos los ejemplos se han generado aleatoriamente en los asistentes.)

En la Tabla 3 puede observarse que, tras la quinta ejecución, las funciones de selección se han reducido a dos. En general, si probamos con un número suficiente de datos de entrada, las funciones de selección candidatas se reducen a las verdaderamente óptimas, que en este problema son las dos restantes: orden decreciente de inicio y orden creciente de fin.

2.2. Asistentes interactivos

Los tres asistentes interactivos tienen una interfaz similar. La Figura 1 muestra la interfaz de usuario de SEDA, el asistente desarrollado para el problema de la selección de actividades [2]. Se muestra un estado intermedio de la ejecución del algoritmo, teniendo activa la función de selección de orden creciente de duración. Se utiliza el conjunto de datos de la Tabla 1, habiéndose considerado ya 5 actividades candidatas (en orden, las numeradas 1, 4, 0, 9 y 5).

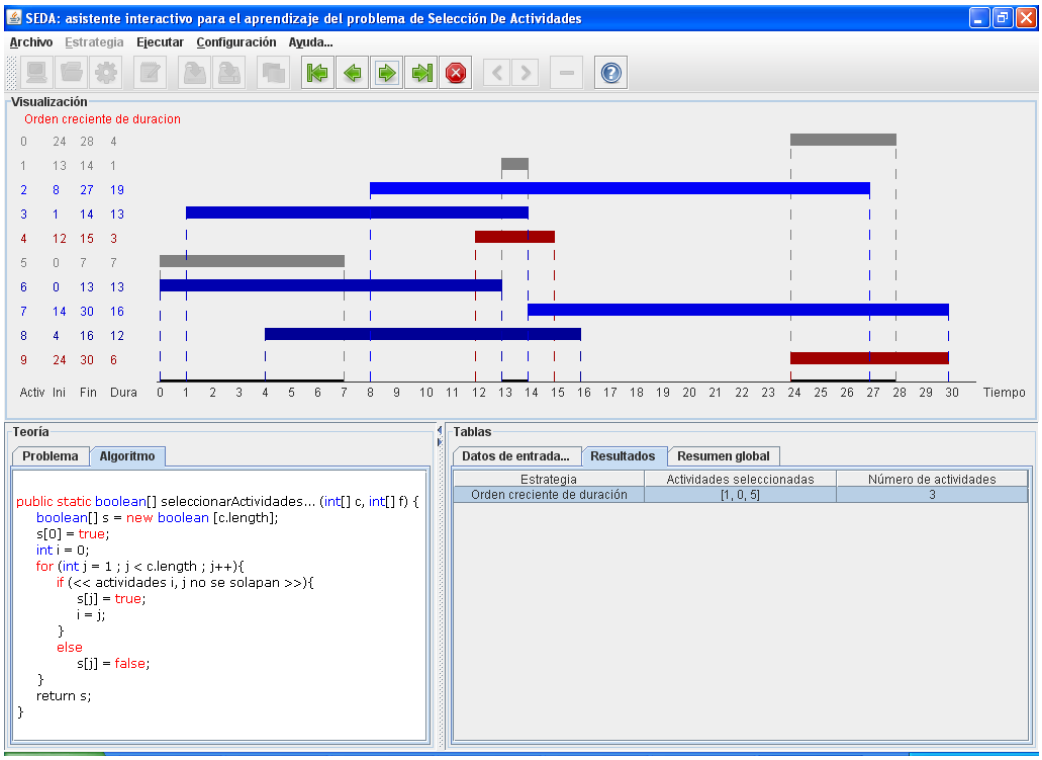

Figura 1. Interfaz de usuario del asistente del problema de selección de actividades

Se distinguen claramente tres zonas en la interfaz de usuario, aparte del menú principal y la barra de iconos. En la parte superior se encuentra el panel de visualización, que muestra gráficamente los datos de entrada y salida. En la parte inferior izquierda se encuentra el panel de teoría. Consta de dos pestañas: la pestaña del problema contiene su enunciado, y la del algoritmo (visible en la figura), su codificación en Java. Finalmente, la parte inferior derecha contiene el panel de tablas, que a su vez contiene tres pestañas, cada una con una tabla: datos de entrada, resultados (visible) y resumen.

Al arrancar la aplicación, el usuario sólo encuentra contenido en el panel de teoría. Como puede verse en la Figura 1, consta de dos pestañas (problema y algoritmo), que el usuario debe leer para comprender el problema planteado. Después, el usuario debe producir datos de entrada para ejecutar el algoritmo voraz (con distintas funciones de selección). Una vez producidos los datos, se muestran gráficamente en el panel de visualización.

SEDA visualiza las actividades en formato bidimensional sobre un eje temporal horizontal. La visualización muestra tantas filas como actividades hay. Cada actividad se representa como una barra comprendida entre los instantes de comienzo y fin.

Las actividades se colorean con tonos (en la Figura 1, en azul) según la función de selección activa, correspondiendo los tonos más oscuros a las actividades que antes se seleccionarán con dicha función. En el mismo panel se van coloreando las actividades ya seleccionadas por el algoritmo, tanto las elegidas como las descartadas (en la figura, en colores gris y rojo, respectivamente).

Una vez escogida una función de selección, podemos examinar su ejecución de forma más o menos detallada. SEDA proporciona cuatro acciones de ejecución:

- Avanzar un paso. Selecciona o descarta la siguiente actividad, según la función de selección activa, y actualiza su visualización.
- Realizar la ejecución completa. Ejecuta el algoritmo en un solo paso, utilizando la función de selección activa, y muestra gráficamente el resultado.
- Retroceder un paso. Es la acción complementaria de la primera.
- Retroceder al comienzo. Es la acción complementaria de la segunda.

Obsérvese que estas cuatro acciones permiten examinar el efecto de la función de selección activa sobre los datos de entrada, con el grado de detalle que el usuario desee. Incluso si hay pasos que se quieren volver a examinar, puede volverse atrás en la ejecución. De esta forma, se facilita su comprensión y análisis (en términos de Bloom).

El usuario puede desorientarse ya que va a manejar muchos datos y a realizar numerosas ejecuciones. Para evitarlo, se incluyen tres tablas, que se corresponden con las presentadas en la subsección 2.1:

- Tabla de datos de entrada. Es similar a la Tabla 1. Contiene los datos de entrada del problema. Es redundante con el panel de visualización, pero permite reordenar las columnas para facilitar su examen.
- Tabla de resultados. Es similar a la Tabla 2. Contiene el resultado de ejecutar los datos de entrada actuales con las distintas funciones de selección. Como puede verse en la Figura 1, contiene tres columnas, correspondientes a: función de selección, índices de las actividades seleccionadas con dicha función de selección, y cuántas actividades son.
- Tabla de resumen. Es similar a la Tabla 3. Contiene el número de actividades que ha seleccionado cada función de selección sobre cada conjunto de datos de entrada.

Las tablas soportan actividades propias de los niveles de análisis y evaluación de Bloom.

3. Experiencia y análisis

Los asistentes se han utilizado durante los cursos académicos 2007-2008, 2008-2009 y 2009-2010 con la técnica de diseño de algoritmos voraces en la asignatura “Diseño y Análisis de Algoritmos”, troncal de tercer curso de Ingeniería Informática en la Universidad Rey Juan Carlos. En el primer curso se usó AMO para la práctica del tema. En los dos cursos siguientes, el profesor usó AMO y TuMiST en el aula y después se usó SEDA para la práctica.

Cada sesión de prácticas se aprovechó para realizar una evaluación de usabilidad de SEDA [6]. También se realizó una evaluación de usabilidad de AMO [6] en la asignatura “Estructuras de Datos y Algoritmos Avanzados”, optativa de tercer curso de la misma titulación.

En la penúltima sesión de laboratorio con SEDA también se realizó un análisis de los informes entregados por los alumnos, para identificar las características de sus soluciones. Como resultado, hemos identificado aspectos susceptibles de mejora en: el método experimental, los asistentes interactivos y su didáctica [7].

Los problemas identificados en la evaluación anterior pueden clasificarse en varias categorías. Junto a detalles de la interfaz de usuario o de las funciones de los asistentes interactivos, podemos destacar:

- Falta de “garantía” de que el uso del asistente sobre sucesivos datos de entrada converge en las estrategias óptimas. Este problema no se da con AMO ni TuMiST, ya que sus problemas asociados producen datos numéricos muy variados (beneficios, longitudes de caminos) y la convergencia sucede en muy pocas iteraciones; a veces, en una sola. Sin embargo, el problema de SEDA produce números enteros pequeños, por lo que pueden realizarse 10 ejecuciones y que la función de selección en orden creciente de duración (no óptima) se mantenga. Como puede verse en la Tabla 3, incluso estrategias claramente no óptimas (como orden creciente de índice) pueden tardar varias iteraciones en ser descartadas.
- Dificultad de mantenimiento de la coherencia de los distintos asistentes. La existencia de varios asistentes, uno para un problema distinto, fue una decisión de diseño fundada. En un análisis realizado sobre las figuras de las distintas técnicas de diseño existentes en libros de texto, se encontró que los algoritmos

voraces no disponían de una visualización común [3]. Analizando más detenidamente este fenómeno, concluimos nuestra imposibilidad de diseñar una herramienta genérica para su aprendizaje como técnica de diseño. La alternativa consistió en desarrollar varios asistentes específicos, cada uno para un problema combinatorio resoluble mediante algún algoritmo voraz.

Aunque el enfoque ha sido adecuado, presenta el problema del mantenimiento de la coherencia entre los asistentes interactivos. Las evaluaciones de experto y de usabilidad nos han permitido identificar numerosas oportunidades de mejora. Sin embargo, resulta muy difícil garantizar que se realizan las mismas modificaciones en todos los asistentes.

- Dominio de aplicación muy limitado. Los asistentes desarrollados son muy útiles para el aprendizaje de los algoritmos voraces. Sin embargo, hay problemas combinatorios resolubles con otras técnicas de diseño con los que podrían usarse. Esto tendría la virtud de ampliar su dominio de aplicación y, en definitiva, hacerlos más útiles para los profesores de asignaturas de algoritmos.

En la sección siguiente presentamos las mejoras realizadas para abordar estos problemas.

4. Avances

Comentamos en esta sección los avances y desarrollos realizados en respuesta a los problemas señalados en la sección anterior.

4.1. Método experimental

La Tabla 3 mostró, tras probar con cinco juegos de datos de entrada, las dos funciones de selección que son óptimas para el problema de la selección de actividades. Sin embargo, también permite observar que, si los datos se generan aleatoriamente, pueden transcurrir bastantes iteraciones hasta que se revele que algunas funciones de selección no son óptimas. De hecho, es muy corriente que la función de selección de orden creciente de duración se mantenga durante más de 10 iteraciones.

Conviene llamar la atención sobre las limitaciones de nuestro método experimental [8]. En efecto, el método solamente garantiza que las funciones de selección que no son óptimas en alguna ejecución no permiten resolver de forma óptima el problema. Sin embargo, no tenemos ninguna garantía de que las funciones de selección que se mantienen en un momento dado sean realmente óptimas. Para garantizar su optimidad, deberíamos hacer una demostración formal.

La situación es análoga a la que da con pruebas y verificación de programas. Las pruebas son un método experimental que permite identificar errores, pero no permiten asegurar que el programa es correcto siempre. Para demostrar su corrección, deberíamos verificar formalmente el programa.

Sin embargo, si realizáramos un número lo suficientemente alto de ejecuciones, esta limitación desaparecería. Veamos en la Tabla 4 los resultados de ejecutar todas las estrategias 101 veces. (Serían preferible 100 ejecuciones, pero hemos escogido este "número mágico" para dar un resultado real de la aplicación, que actualmente presenta una restricción sobre el número de ejecuciones que estamos corrigiendo.)

Tabla 4. Resultados sobre 101 datos de entrada

Función de selección	Nº de ejecuciones óptimas
Índice ↑	51
Índice ↓	56
Inicio ↑	37
Inicio ↓	101
Fin ↑	101
Fin ↓	46
Durac. ↑	97
Durac. ↓	7

Puede verse que, efectivamente, la función de selección en orden creciente de duración produce el valor óptimo en una inmensa mayoría de las veces (97 sobre 101), pero no en todas. Por tanto, no es óptima.

4.2. Mejoras de los asistentes

Hemos desarrollado diversas funciones en los asistentes interactivos para dar mayor apoyo bien

a la docencia bien al método experimental. En lo que se refiere a funciones docentes, destacamos:

- Exportación. Los asistentes se han concebido para ser usados en sesiones de experimentación, similares a los laboratorios de las ciencias. Conviene que los resultados se entreguen en forma de informe. Para facilitar su elaboración, se permite exportar (en formatos gráficos estándar) una visualización individual o una secuencia de visualizaciones en un directorio.

  Se han añadido dos funciones más. La primera es generar una animación en formato GIF. La segunda consiste en exportar en un solo paso las visualizaciones de todas las funciones de selección ejecutadas sobre los datos de entrada actuales. Se crea un conjunto de directorios, uno por función de selección, donde cada uno contiene una secuencia de visualizaciones.
- Configuración de las visualizaciones. Se permite configurar los colores de la visualización y el efecto de transición de la animación (parpadeo).

En cuanto al método experimental, se han añadido varias funciones, algunas relacionadas con la ejecución de las funciones de selección y otras con las tablas.

En lo que se refiere a la ejecución de las funciones de selección, se han añadido otras tres acciones:

- Ejecutar todas las funciones de selección. Una vez que el alumno ya entiende las distintas funciones de selección, suele interesarle ejecutarlas en un solo paso sobre los nuevos datos de entrada.
- Ejecutar un subconjunto de funciones de selección. Es similar, pero limitando las funciones de selección a las que son óptimas hasta el momento. Produce un efecto similar al mostrado en la Tabla 3.
- Ejecutar un número muy alto de datos de entrada ("ejecución intensiva"). Da soporte a la ampliación del método experimental antes descrito. El usuario puede elegir entre ejecutar sobre un número dado de datos de entrada o durante un cierto tiempo. Como la ejecución intensiva puede durar bastante, el usuario puede realizar algunas funciones en paralelo sin que el asistente se quede bloqueado.

Según se interacciona con la aplicación, ésta va cargando los resultados en las distintas tablas. Sin embargo, puede suceder que en algún momento haya excesiva información. En estos casos, el usuario puede realizar varias acciones de borrado sobre las tablas:

- Borrar una fila de la tabla de resultados. Permite borrar el resultado de una estrategia, normalmente porque se trata de una estrategia que ya se sabe que no es óptima.
- Borrar una fila de la tabla de resumen. Permite borrar un juego de datos de entrada completo y sus resultados, normalmente cuando su ejecución no aporta nada nuevo con respecto a otras ejecuciones anteriores.
- Borrar los datos de toda la sesión. Permite comenzar desde cero sin tener que salirse de la aplicación.

Por último, se ha añadido una tabla adicional, similar a la Tabla 4, llamada tabla abreviada. Es accesible mediante una nueva pestaña, situada a la derecha de las otras pestañas de tablas (véase la parte inferior derecha de la Figura 1).

4.3. Asistente único GreedEx

Una forma obvia de evitar incoherencias entre los distintos asistentes es integrarlos en uno único. No sería todavía un sistema genérico que soporte a todos los algoritmos voraces, pero daría soporte a varios algoritmos.

Hemos adoptado este enfoque desarrollando un nuevo sistema, que hemos denominado GreedEx (de "GREEDy algorithms" y "EXperimentation"). En un primer paso, GreedEx soporta los algoritmos de la mochila y de la selección de actividades.

GreedEx presenta una interfaz de usuario y unas funciones comunes a todos los problemas, pero adapta algunos al problema e idioma seleccionado. Los cambios de la interfaz de usuario se realizan dinámica y rápidamente, sin que el usuario lo aprecie.

El usuario puede elegir el problema a estudiar en un menú fijo. Los elementos que varían con cada problema son:

- Diálogos de entrada o modificación de datos.
- Panel del problema (pestañas de explicación y de algoritmo).
- Panel de visualización.
- Menús de las funciones de selección a ejecutar (y su implementación).
- Formato de las tablas de datos de entrada y de resultados, así como leyendas de las funciones de selección en las tablas de resultados, de resumen y abreviada.
- Configuración de la visualización.

Actualmente estamos integrando el problema de la mochila 0/1 [2]. Afortunadamente, pueden aprovecharse muchos de los elementos del problema de la mochila, limitándose los cambios necesarios a algunos cambios menores en:

- Panel del problema.
- Tablas de resultados y de resumen.
- Ejecución de las funciones de selección.

La Tabla 5 ilustra el ejemplo con el resultado de realizar cinco ejecuciones sobre datos generados aleatoriamente. Hemos resaltado en negrita las funciones de selección que son óptimas con cada juego de datos de entrada.

Función de selección	**1ª ejec.**	**2ª ejec.**	**3ª ejec.**	**4ª ejec.**	**5ª ejec.**
Índice ↑	296	**0**	299	331	92
Índice ↓	332	**0**	251	365	164
Peso ↑	381	**0**	170	290	**256**
Peso ↓	223	**0**	174	364	225
Beneficio ↑	158	**0**	170	288	125
Beneficio ↓	**404**	**0**	**316**	**371**	225
Peso/Ben. ↑	381	**0**	**316**	365	248
Peso/Ben. ↓	158	**0**	171	328	125
Ben./Peso ↑	158	**0**	171	328	125
Ben./Peso ↓	381	**0**	**316**	365	248

Tabla 5. Resultados del problema de la mochila 0/1 sobre varios datos de entrada

Puede observarse que en la segunda ejecución el resultado ha sido cero con todas las funciones de selección porque la mochila era más pequeña que cualquiera de los objetos. Más interesante es el hecho de que ninguna función de selección es óptima siempre. Por tanto, este problema sólo puede resolverse de forma óptima con otras técnicas de diseño de algoritmos (normalmente, con programación dinámica).

5. Conclusiones y trabajos futuros

Hemos descrito en la sección anterior diversas aportaciones realizadas. Desde un punto de vista tecnológico, el cambio mayor ha sido la integración en un único asistente, que permitirá integrar múltiples problemas combinatorios, sin inconvenientes de incoherencia en el soporte dado a cada uno.

También cabe destacar el alto número de evaluaciones de experto realizadas (por un profesor con experiencia en docencia de la algoritmia) más las 4 evaluaciones de usabilidad. Como resultado, se han añadido o mejorado diversas funciones que han dado lugar a una aplicación muy madura.

Desde el punto de vista docente, podemos destacar tres hechos. En primer lugar, el alto apoyo dado por GreedEx al método experimental. Sin embargo, es interesante llamar la atención sobre el hecho de que no siempre es fácil determinar el punto de equilibrio justo entre dar libertad al usuario (el alumno) o guiarle. Por ejemplo, una función nueva que incluimos fue la exportación del informe completo. A partir de toda la información necesaria (a partir de tablas, visualizaciones almacenadas en disco, etc.), GreedEx la almacenaba en formato XML y posteriormente la traducía a PDF. Sin embargo, al final hemos quitado esta función porque encauzaba excesivamente al alumno y podría eliminar parte de los beneficios de nuestro enfoque, orientado a la experimentación y descubrimiento activo por parte del alumno.

En segundo lugar, hemos podido extender nuestro método experimental a problemas resolubles con otras técnicas. El caso más evidente es el problema de la mochila 0/1. Con muy poco esfuerzo hemos incluido este problema, que no puede resolverse de forma óptima con algoritmos voraces, sino con otras técnicas (normalmente, con programación dinámica). A pesar de que GreedEx no se podrá usar para desarrollar los algoritmos de programación dinámica, permitirá que el alumno descubra que ninguna función de selección voraz es óptima para este problema (véase Tabla 5).

La inclusión de la ejecución intensiva también permite comprobar qué porcentaje aproximado de los datos de entrada son resueltos de forma óptima por funciones de selección que no lo son en general. Esta posibilidad nos abre una vía para utilizar el método experimental para el estudio de los algoritmos aproximados (o heurísticas voraces). Un ejemplo claro lo tenemos en el problema de selección de actividades, en el que la selección en orden creciente de duración produce la solución óptima más del 95% de las veces (véase Tabla 4).

Por último, podemos dar por concluido el diseño y desarrollo informático de GreedEx, salvo en la posibilidad de integrar más problemas (p.ej. el árbol de recubrimiento con coste mínimo, ya implementado en TuMiST).

Como trabajos futuros, tenemos pendiente un análisis detallado de los resultados de las dos últimas sesiones de SEDA (sólo está analizada la primera sesión [7]). También prevemos realizar una evaluación de la mejora del aprendizaje de los algoritmos voraces. Tenemos planificada una evaluación más larga en el tiempo, de la que nuestro método experimental es una parte destacada.

Agradecimientos

Este trabajo se ha financiado con el proyecto TIN2008-04103/TSI del MICINN.

Referencias

[1] Bloom, B., Furst, E., Hill, W., y Krathwohl, D.R. *Taxonomy of Educational Objectives: Handbook I, The Cognitive Domain*. Addison-Wesley, 1956.

[2] Cormen, T.H., Leiserson, C.E., y Rivest. R.L. *Introduction to Algorithms*. The MIT Press, 2ª ed., 2001.

[3] Fernández Muñoz, L., y Velázquez Iturbide, J.Á. "Estudio sobre la visualización de las técnicas de diseño de algoritmos". En *Actas del VII Congreso Internacional de Interacción Persona-Ordenador, Interacción 2006*, pp. 315-324.

[4] Debdi, O., Granada, J.D. y Velázquez Iturbide, J.Á. "Ayudante interactivo para los algoritmos de Prim y Kruskal". En *XVI Jornadas de Enseñanza Universitaria de la Informática*, *JENUI 2010*, aceptado.

[5] Sánchez Torrubia, M.G., Torres Blanc, C., y López Martínez, M.A. "PathFinder: A visualization eMathTeacher for actively learning Dijkstra's algorithm", *Electronic Notes in Theoretical Computer Science*, 224(151-158), enero 2009.

[6] Velázquez Iturbide, J.Á., Lázaro Carrascosa, C.A., y Hernán Losada, I. "Asistentes interactivos para el aprendizaje de algoritmos voraces". En *IEEE Revista Iberoamericana de Tecnologías del Aprendizaje, IEEE-RITA*, 4(3):213-220, agosto 2009.

[7] Velázquez Iturbide, J.Á.; Pérez Carrasco, A. "Experimental inquiry into greedy algorithms". En *Proc. 2nd Workshop on Methods and Cases in Computing Education, MCCE 2009*, pp. 1-6, 2009.

[8] Velázquez Iturbide, J.Á.; Pérez Carrasco, A. "Active learning of greedy algorithms by means of interactive experimentation". En *Proc. 14th Annual Conf. Innovation and Technology in Computer Science Education, ITiCSE 2009*, pp. 119-123, 2009.

[9] "AlgoViz.org – The Algorithm Visualization Portal", http://algoviz.org/, accedido 17 junio 2010